Package 'DMAR'

November 15, 2026

Type Package

Title Design, Measurement, and Analysis in R (DMAR)

Version 1.1.0

Date 2026-11-14

Imports grDevices, MASS, generics, parallel, stats, utils, withr

Suggests boot, car, ggplot2 (>= 3.4.0), ggrain, knitr, lavaan (>= 0.7-3), lme4, lmerTest, mice (>= 3.18.0), nlme, OpenMx, patchwork, R.rsp, reformulas, rmarkdown, testthat (>= 3.2.0)

Description Methods for design, measurement, and analysis, with the aim of being user friendly yet methodologically sound. 'DMAR' (pronounced ``Dee-Mar'') implements them with interfaces, defaults, and documentation that are consistent across the package and grounded in the methodological literature. The package emphasizes effect size estimation with confidence intervals; sample size planning through accuracy in parameter estimation (AIPE) and power analysis (including composite power for designs whose conclusions require several results to hold at once), with minimum risk, sequential, and equivalence frameworks; reliability, agreement, and measurement more broadly, from coefficient omega with confidence intervals to measurement invariance; factor analysis and structural equation modeling, in which constructs, latent variables measured by multiple indicators, are modeled directly, with confirmatory factor analysis, convergent and discriminant validity, and sample size planning for structural equation models; mediation analysis, from the simple mediation model with bootstrap intervals to likelihood ratio tests of arbitrary indirect effects by model-based constrained optimization (MBCO), with multiple groups and the probing of moderated mediation; equivalence and noninferiority testing; meta-analysis; repeated measures, multivariate, ANOVA, and ANCOVA designs; and inference grounded in model comparison throughout. Measurement is approached from a psychometric perspective, and although many of the methods grew up in human-centered research, they apply broadly across the empirical sciences. Much of what is implemented traces to the author's methodological work, interests,

and collaborations. 'DMAR' is a more general and greatly expanded reimagining of the 'MBESS' package (Kelley, 2007a, <doi:10.18637/jss.v020.i08>; 2007b, <doi:10.3758/BF03192993>), which has been on CRAN for more than two decades and remains available there. Most functions accept either raw data or the summary statistics typically reported in published articles, so an analysis can be reproduced from a paper without the original data, which is useful both for extending a published analysis and for meta-analytic work. Most estimation, inference, and planning functions return one consistently formatted data frame that composes with the broader R ecosystem; the regression functions for incomplete data and a few others return a list suited to their task instead. Most effect sizes are reported with a confidence interval, either by the function that estimates them or by a companion interval function. The interfaces and documentation are written for researchers who have data and a question but who are not R experts; for methodologists the package also provides specialized procedures, such as critical values of the studentized maximum modulus, Dunnett, and Bryant-Paulson distributions computed from the distributions themselves rather than read from printed tables.

Depends R (>= 4.0.0)

URL https://kenkelley.org, https://yelleknek.github.io/DMAR/, https://github.com/yelleKneK/DMAR

BugReports https://github.com/yelleKneK/DMAR/issues

License GPL (>= 3)

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

VignetteBuilder knitr, R.rsp

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first ss_aipe_crd_es, reliability_omega_categorical, cv_bryant_paulson, ci_dunnett, reliability_omega, ss_power_composite_sem, ss_aipe_composite_sem, mediation_mbco, ss_aipe_new_sensitivities

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Ken Kelley [aut, cre] (ORCID: <https://orcid.org/0000-0002-4756-8360>)

Maintainer Ken Kelley <kkelley@nd.edu>

Repository CRAN

Date/Publication 2026-11-15 21:40:24 UTC

Contents

DMAR-package . . . 10
adjusted_means . . . 14
analysis_of_change . . . 17
ancova . . . 20
anova.mlmr . . . 22
anova.mlmr_mv . . . 24
anova_within . . . 25
anova_within_two_way . . . 26
average_variance_extracted . . . 29
bayes_independent_t . . . 33
bayes_one_sample_t . . . 36
bayes_paired_t . . . 38
bessel_errors . . . 41
bifactor_indices . . . 44
bryant_paulson . . . 49
cfa_1 . . . 53
cfa_2 . . . 58
ci_bifactor_indices . . . 62
ci_c . . . 71
ci_correlation . . . 74
ci_crossed . . . 78
ci_cv . . . 80
ci_c_ancova . . . 83
ci_c_ancova_bp . . . 86
ci_dunnett . . . 89
ci_dunnett_t3 . . . 91
ci_eigenvalue . . . 94
ci_eta_squared . . . 95
ci_eta_squared_generalized . . . 101
ci_games_howell . . . 107
ci_mahalanobis . . . 110
ci_nc_chisq . . . 113
ci_nc_F . . . 115
ci_nc_t . . . 118
ci_omega_squared . . . 121
ci_proportion . . . 125
ci_proportion_paired . . . 126
ci_pvaf . . . 129
ci_R2 . . . 132
ci_ratio_of_means . . . 135
ci_ratio_of_means_paired . . . 141
ci_rc . . . 147
ci_reg_coef . . . 150
ci_rmsea . . . 153
ci_sc . . . 156
ci_scheffe . . . 158

ci_sc_ancova . . . 160
ci_sm . . . 163
ci_smd . . . 165
ci_smd_c . . . 168
ci_snr . . . 170
ci_src . . . 172
ci_srsnr . . . 176
ci_tukey_kramer . . . 179
ci_two_phase . . . 181
cles . . . 184
cliff_delta . . . 187
cohen_f . . . 192
cohen_h . . . 194
cohen_kappa . . . 196
combine_p . . . 201
common_method_adjusted_r . . . 203
common_method_marker . . . 206
common_method_single_factor . . . 208
compare_cov_structures . . . 210
compare_sem . . . 212
content_validity_index . . . 216
contrast_adjusted . . . 219
contrast_test . . . 221
convert_cor_cov . . . 225
convert_d_or . . . 226
convert_d_r . . . 227
convert_F_chisq . . . 229
convert_R2 . . . 231
convert_r_Z . . . 233
convert_t_smd . . . 234
convert_z_normal . . . 235
convert_Z_r . . . 236
correction_for_attenuation . . . 237
correlations_test . . . 240
covmat_from_cfa . . . 244
cov_fiml . . . 246
cov_sem . . . 250
cv . . . 252
cv_bonferroni_f . . . 254
cv_bryant_paulson . . . 256
cv_chisq . . . 259
cv_dunnett . . . 261
cv_f . . . 263
cv_scheffe . . . 265
cv_smm . . . 266
cv_t . . . 269
cv_tukey_hsd . . . 270
cv_z . . . 273

depression_bdi . . . 274
descriptives . . . 276
design_consequences . . . 278
design_effect . . . 281
df_barnard_rubin . . . 283
df_missing . . . 286
diagnosis_agreement . . . 290
dmacs . . . 292
dmar_bootstrap . . . 296
dmar_metadata . . . 301
dmar_output_helpers . . . 310
drinks_trial . . . 314
dunn_test . . . 318
ecvi . . . 320
effects_coding . . . 322
epsilon_corrections . . . 324
equivalence_c . . . 325
equivalence_proportion_paired . . . 329
equivalence_r . . . 333
equivalence_rmsea . . . 335
equivalence_smd . . . 338
eta_squared . . . 341
eta_squared_generalized . . . 345
expected_partial_r . . . 350
expected_r . . . 352
expected_R2 . . . 356
expected_smd . . . 357
factorial_anova . . . 360
factor_congruence . . . 362
factor_count . . . 367
factor_scores . . . 371
fa_bifactor . . . 382
fa_confirmatory . . . 390
fa_exploratory . . . 406
fa_second_order . . . 415
fleiss_kappa . . . 424
format_p . . . 428
fraction_missing_information . . . 430
glance.dmar_fa_confirmatory . . . 439
glance.dmar_mediation_mbco . . . 440
glance.dmar_reliability . . . 440
glance.mlmr . . . 441
gwet_ac . . . 442
helmert_coding . . . 444
holzinger_swineford . . . 445
htmt . . . 448
icc . . . 451
icc_lmer . . . 453

irt_grm . . . 455
irt_information . . . 460
is_orthogonal_set . . . 464
krippendorff_alpha . . . 465
kurtosis . . . 468
limits_of_agreement . . . 470
lin_ccc . . . 473
manova_split_plot . . . 474
mauchly_test . . . 478
measurement_alignment . . . 480
measurement_invariance . . . 485
mediate . . . 496
mediation_mbco . . . 500
meta_contrast . . . 512
meta_es . . . 514
meta_r . . . 518
meta_smd . . . 521
missing_coverage . . . 522
missing_patterns . . . 524
mixed_anova . . . 525
mlmr . . . 528
mlmr_mv . . . 537
moments_nc_chisq . . . 543
moments_nc_F . . . 545
moments_nc_t . . . 547
mr_cv . . . 549
mr_smd . . . 554
nnt_from_smd . . . 556
now . . . 562
obrien_test . . . 564
omega_squared . . . 567
orthogonal_polynomial . . . 571
pairwise_within . . . 574
parallel_analysis . . . 576
plot_ci . . . 579
plot_equivalence . . . 581
plot_fa_confirmatory . . . 584
plot_forest . . . 586
plot_irt_information . . . 588
plot_mediation_mbco . . . 590
plot_R2 . . . 593
plot_randomization_test . . . 595
plot_regions_of_significance . . . 597
plot_smd . . . 599
plot_trajectories . . . 601
plot_trajectories_fitted . . . 604
power_density_equivalence_md . . . 606
power_equivalence_c . . . 608

power_equivalence_md . . . 611
power_equivalence_md_plot . . . 614
power_fisher_exact . . . 616
prime_time_achievement . . . 619
print_anova . . . 627
print_summary . . . 628
probability_of_superiority_paired . . . 629
procrustes_phi . . . 633
proportion_of_superiority . . . 635
pygmalion . . . 637
R2_mixed_effects . . . 639
R2_mixed_effects_decomposition . . . 642
randomization_test . . . 644
randomization_test_paired . . . 650
regions_of_significance . . . 653
reliability . . . 657
reliability_alpha . . . 662
reliability_H . . . 676
reliability_kr20 . . . 685
reliability_omega . . . 690
reliability_omega_categorical . . . 707
responder_analysis . . . 723
sd_unbiased . . . 727
sem_exact_fit_bootstrap . . . 729
sem_fit_measures . . . 734
sem_information_criteria . . . 747
sem_residual_fit . . . 754
sem_sam . . . 758
signal_to_noise_R2 . . . 759
simple_effects_AB . . . 761
simple_structure . . . 765
simulate_ancova_data . . . 767
simulate_ancova_factorial_data . . . 769
simulate_anova_data . . . 773
simulate_longitudinal_gompertz . . . 774
simulate_longitudinal_logistic . . . 778
simulate_longitudinal_negative_exponential . . . 782
simulate_longitudinal_polynomial . . . 785
simulate_longitudinal_richards . . . 792
simulate_regression_data . . . 796
skewness . . . 798
smd . . . 799
smd_c . . . 802
smd_trimmed . . . 804
ss_aipe_c . . . 806
ss_aipe_cliff_delta . . . 809
ss_aipe_cliff_delta_sensitivity . . . 811
ss_aipe_composite_sem . . . 814

ss_aipe_crd . . . 819
ss_aipe_crd_es . . . 826
ss_aipe_crossed . . . 837
ss_aipe_crossed_sensitivity . . . 842
ss_aipe_cv . . . 846
ss_aipe_cv_sensitivity . . . 850
ss_aipe_c_ancova . . . 854
ss_aipe_c_ancova_sensitivity . . . 856
ss_aipe_c_sensitivity . . . 859
ss_aipe_equivalence_r . . . 863
ss_aipe_equivalence_r_sensitivity . . . 865
ss_aipe_equivalence_smd . . . 868
ss_aipe_equivalence_smd_sensitivity . . . 872
ss_aipe_icc . . . 875
ss_aipe_icc_sensitivity . . . 878
ss_aipe_indirect_effect . . . 882
ss_aipe_indirect_effect_sensitivity . . . 887
ss_aipe_mean . . . 890
ss_aipe_mean_sensitivity . . . 894
ss_aipe_mixed_effects . . . 898
ss_aipe_mixed_effects_sensitivity . . . 902
ss_aipe_omega_squared . . . 906
ss_aipe_omega_squared_sensitivity . . . 908
ss_aipe_partial_r . . . 911
ss_aipe_partial_r_sensitivity . . . 916
ss_aipe_pcm . . . 920
ss_aipe_pcm_sensitivity . . . 923
ss_aipe_r . . . 926
ss_aipe_R2 . . . 929
ss_aipe_R2_sensitivity . . . 933
ss_aipe_rc . . . 937
ss_aipe_rc_sensitivity . . . 940
ss_aipe_reg_coef . . . 944
ss_aipe_reg_coef_sensitivity . . . 951
ss_aipe_reliability . . . 956
ss_aipe_reliability_sensitivity . . . 967
ss_aipe_rmsea . . . 974
ss_aipe_rmsea_sensitivity . . . 975
ss_aipe_r_sensitivity . . . 979
ss_aipe_sc . . . 982
ss_aipe_sc_ancova . . . 986
ss_aipe_sc_ancova_sensitivity . . . 991
ss_aipe_sc_sensitivity . . . 994
ss_aipe_semipartial_r . . . 998
ss_aipe_semipartial_r_sensitivity . . . 1003
ss_aipe_sem_path . . . 1007
ss_aipe_sem_path_sensitivity . . . 1009
ss_aipe_sm . . . 1014

ss_aipe_smd 1016
ss_aipe_smd_sensitivity 1020
ss_aipe_sm_sensitivity 1023
ss_aipe_src 1027
ss_aipe_src_sensitivity 1032
ss_aipe_two_phase 1036
ss_aipe_two_phase_cost 1039
ss_aipe_two_phase_sensitivity 1043
ss_power_c 1047
ss_power_composite_ancova 1049
ss_power_composite_ancova_2group 1053
ss_power_composite_anova 1060
ss_power_composite_factorial_ancova 1063
ss_power_composite_factorial_ancova_het 1067
ss_power_composite_factorial_anova 1072
ss_power_composite_sem 1075
ss_power_contrast 1079
ss_power_c_ancova 1083
ss_power_equivalence_c 1085
ss_power_factorial_ancova 1088
ss_power_factorial_anova 1090
ss_power_indirect_effect 1093
ss_power_mixed_effects 1096
ss_power_one_way_anova 1098
ss_power_pcm 1100
ss_power_r 1104
ss_power_R2 1107
ss_power_R2_change 1111
ss_power_R2_sensitivity 1117
ss_power_rc 1120
ss_power_reg_coef 1124
ss_power_reg_coef_sensitivity 1129
ss_power_rm_anova 1133
ss_power_sc 1135
ss_power_sem 1137
ss_power_smd 1139
ss_power_split_plot_anova 1141
ss_seq_c 1144
ss_seq_c_sensitivity 1147
ss_seq_two_phase 1150
summary_t_test 1152
summary_t_test_broom 1154
teacher_expectancy 1155
test_market 1157
tidy.dmar_ci_R2 1159
tidy.dmar_ci_smd 1160
tidy.dmar_cliff_delta 1161
tidy.dmar_fa_confirmatory 1169

tidy.dmar_fmi . . . 1170
tidy.dmar_mediation_mbco . . . 1171
tidy.dmar_R2_mixed_effects . . . 1171
tidy.dmar_reliability . . . 1172
tidy.mlmr . . . 1173
tidy.mlmr_mv . . . 1174
unbiased_R2 . . . 1175
vargha_delaney_A . . . 1177
variance_components_mls . . . 1184
var_alpha . . . 1186
var_cv . . . 1188
var_ete . . . 1191
var_icc . . . 1194
var_indirect_effect . . . 1196
var_omega_squared . . . 1199
var_partial_r . . . 1201
var_r . . . 1203
var_R2 . . . 1205
var_semipartial_r . . . 1206
var_smd . . . 1210
var_smd_trimmed . . . 1212
welch_t . . . 1214
welch_t_broom . . . 1216

Index **1218**

`DMAR-package` *Design, Measurement, and Analysis in R*

Description

An R package for design, measurement, and analysis, covering effect sizes, confidence intervals, sample size planning, reliability and agreement, mediation analysis, equivalence testing, meta-analysis, experimental and quasi-experimental designs, repeated measures, and model comparison-based inference. DMAR (pronounced "Dee-Mar") is methodological in nature, drawing on the psychometric and statistical traditions, and follows the methodological and applied research program of the author. Many aspects of the package trace to the author's methodological work and collaborations, including sample size planning via accuracy in parameter estimation (AIPE; Kelley & Maxwell, 2003; Kelley & Rausch, 2006; Maxwell, Kelley, & Rausch, 2008), the definition and communication of effect sizes (Kelley & Preacher, 2012; Preacher & Kelley, 2011), and the model comparison perspective of Maxwell, Delaney, and Kelley (2027). Its methods and examples are chosen for research in which the independent or dependent variables involve the person, across psychology, sociology, education, behavioral economics, management, marketing, and information systems.

Details

The package covers effect size estimation, confidence intervals for effect sizes, sample size planning, multivariate methods, factor analysis, and certain latent variable models, and many of its functions accept the sample estimates or results reported in published articles as well as raw data. Sample size planning is offered under several complementary frameworks: accuracy in parameter estimation (AIPE), power analysis, minimum risk, and equivalence. Most exported functions return a tidy `data.frame` with a `term` column and a numeric `value` column (some carry additional typed columns per term), and **ggplot2** is used for the plotting functions. A few functions return the shape their task calls for instead: model fits such as `mlmr` return a richer list-like object with `coef` / `vcov` / `confint` methods, `descriptives` returns a list of summary tables, and a small number of scalar utilities such as `skewness` and `kurtosis` return a bare numeric.

DMAR builds heavily on the **MBESS** package (Kelley, 2007a, 2007b), which has been on CRAN for more than two decades. **MBESS** was originally framed for the behavioral, educational, and social sciences; the methods of both packages are not specific to those fields.

Function families. Every exported function is listed once below, in the family the reference index of the package site also uses (`ci_bifactor_indices` appears in two).

Effect sizes `smd`, `smd_c`, `smd_trimmed`, `eta_squared`, `eta_squared_generalized`, `omega_squared`, `cohen_f`, `cohen_h`, `cles`, `cliff_delta`, `vargha_delaney_A`, `proportion_of_superiority`, `probability_of_superiority_paired`, `nnt_from_smd`, `responder_analysis`, `correction_for_attenuation`, `signal_to_noise_R2`, `expected_partial_r`, `expected_r`, `expected_R2`, `expected_smd`, `unbiased_R2`, `sd_unbiased`, `descriptives`, `skewness`, `kurtosis`.

Confidence intervals `ci_bifactor_indices`, `ci_c`, `ci_c_ancova`, `ci_c_ancova_bp`, `ci_crossed`, `ci_cv`, `ci_dunnett`, `ci_dunnett_t3`, `ci_eigenvalue`, `ci_eta_squared`, `ci_eta_squared_generalized`, `ci_games_howell`, `ci_mahalanobis`, `ci_omega_squared`, `ci_proportion`, `ci_proportion_paired`, `ci_pvaf`, `ci_r`, `ci_R`, `ci_R2`, `ci_ratio_of_means`, `ci_ratio_of_means_paired`, `ci_rc`, `ci_reg_coef`, `ci_rmsea`, `ci_sc`, `ci_sc_ancova`, `ci_scheffe`, `ci_sm`, `ci_smd`, `ci_smd_c`, `ci_snr`, `ci_src`, `ci_srsnr`, `ci_tukey_kramer`, `ci_two_phase`.

Sample size planning, accuracy in parameter estimation (AIPE) `ss_aipe_c`, `ss_aipe_c_ancova`, `ss_aipe_c_ancova_sensitivity`, `ss_aipe_c_sensitivity`, `ss_aipe_cliff_delta`, `ss_aipe_cliff_delta_sens`
`ss_aipe_composite_sem`, `ss_aipe_crd_both_fixed_budget`, `ss_aipe_crd_both_fixed_width`, `ss_aipe_crd_es_both_fixed_budget`, `ss_aipe_crd_es_both_fixed_width`, `ss_aipe_crd_es_n_clusters_fixe`
`ss_aipe_crd_es_n_clusters_fixed_width`, `ss_aipe_crd_es_n_individuals_fixed_budget`, `ss_aipe_crd_es_n_individuals_fixed_width`, `ss_aipe_crd_n_clusters_fixed_budget`, `ss_aipe_crd_n_clusters_fixed_width`, `ss_aipe_crd_n_individuals_fixed_budget`, `ss_aipe_crd_n_individ`
`ss_aipe_crossed`, `ss_aipe_crossed_sensitivity`, `ss_aipe_cv`, `ss_aipe_cv_sensitivity`, `ss_aipe_equivalence_r`, `ss_aipe_equivalence_r_sensitivity`, `ss_aipe_equivalence_smd`, `ss_aipe_equivalence_smd_sensitivity`, `ss_aipe_icc`, `ss_aipe_icc_sensitivity`, `ss_aipe_indirect_effect`
`ss_aipe_indirect_effect_sensitivity`, `ss_aipe_mean`, `ss_aipe_mean_sensitivity`, `ss_aipe_mixed_effects`
`ss_aipe_mixed_effects_sensitivity`, `ss_aipe_omega_squared`, `ss_aipe_omega_squared_sensitivity`, `ss_aipe_partial_r`, `ss_aipe_partial_r_sensitivity`, `ss_aipe_pcm`, `ss_aipe_pcm_sensitivity`, `ss_aipe_r`, `ss_aipe_r_sensitivity`, `ss_aipe_R2`, `ss_aipe_R2_sensitivity`, `ss_aipe_rc`, `ss_aipe_rc_sensitivity`, `ss_aipe_reg_coef`, `ss_aipe_reg_coef_sensitivity`, `ss_aipe_reliability`, `ss_aipe_reliability_sensitivity`, `ss_aipe_rmsea`, `ss_aipe_rmsea_sensitivity`, `ss_aipe_sc`, `ss_aipe_sc_ancova`, `ss_aipe_sc_ancova_sensitivity`, `ss_aipe_sc_sensitivity`, `ss_aipe_sem_path`, `ss_aipe_sem_path_sensitivity`, `ss_aipe_semipartial_r`, `ss_aipe_semipartial_r_sensitivity`,

ss_aipe_sm, ss_aipe_sm_sensitivity, ss_aipe_smd, ss_aipe_smd_sensitivity, ss_aipe_src, ss_aipe_src_sensitivity, ss_aipe_two_phase, ss_aipe_two_phase_cost, ss_aipe_two_phase_sensitivity.

Sample size planning, power ss_power_c, ss_power_c_ancova, ss_power_composite_ancova, ss_power_composite_ancova_2group, ss_power_composite_anova, ss_power_composite_factorial_ancova, ss_power_composite_factorial_ancova_het, ss_power_composite_factorial_anova, ss_power_composite_sem, ss_power_contrast, ss_power_equivalence_c, ss_power_factorial_ancova, ss_power_factorial_anova, ss_power_indirect_effect, ss_power_mixed_effects, ss_power_one_way_anova ss_power_pcm, ss_power_r, ss_power_R2, ss_power_R2_change, ss_power_R2_sensitivity, ss_power_rc, ss_power_reg_coef, ss_power_reg_coef_sensitivity, ss_power_rm_anova, ss_power_sc, ss_power_sem, ss_power_smd, ss_power_split_plot_anova, power_equivalence_c, power_equivalence_md, power_equivalence_md_plot, power_density_equivalence_md, power_fisher_exact.

Design analysis and design utilities design_consequences, design_effect, effects_coding, helmert_coding, orthogonal_polynomial, is_orthogonal_set.

Minimum risk and sequential estimation mr_smd, mr_cv, ss_seq_c, ss_seq_c_sensitivity, ss_seq_two_phase.

Equivalence testing equivalence_smd, equivalence_r, equivalence_c, equivalence_proportion_paired, equivalence_rmsea.

Reliability and agreement reliability, reliability_alpha, reliability_H, reliability_kr20, reliability_omega, reliability_omega_categorical, icc, icc_lmer, R2_mixed_effects, R2_mixed_effects_decomposition, regions_of_significance, cohen_kappa, fleiss_kappa, krippendorff_alpha, gwet_ac, lin_ccc, limits_of_agreement, loa, content_validity_index.

ANOVA, ANCOVA, and contrasts ancova, factorial_anova, anova_within, anova_within_two_way, mixed_anova, manova_split_plot, pairwise_within, simple_effects_AB, contrast_test, contrast_adjusted, adjusted_means, dunn_test, obrien_test, mauchly_test, epsilon_corrections, welch_t, summary_t_test, correlations_test, randomization_test, randomization_test_paired.

Meta-analysis and evidence synthesis meta_contrast, meta_es, meta_r, meta_smd, combine_p.

Bayesian analyses bayes_independent_t, bayes_one_sample_t, bayes_paired_t.

Mediation mediate, mediation_mbco.

Longitudinal change analysis_of_change.

Maximum likelihood regression (FIML) mlmr, mlmr_mv.

Missing data missing_patterns, missing_coverage, cov_fiml, fraction_missing_information, df_missing, df_barnard_rubin.

Multivariate and latent variable methods cfa_1, cfa_2, fa_confirmatory, cfa_k, fa_bifactor, fa_second_order, fa_exploratory, efa_k, factor_count, factor_congruence, parallel_analysis, sem_sam, compare_cov_structures, cov_sem, covmat_from_cfa, procrustes_phi, variance_components_mls, measurement_invariance, htmt, average_variance_extracted, bifactor_indices, ci_bifactor_indices, factor_scores, common_method_marker, common_method_adjusted_r, common_method_single_factor, sem_fit_measures, sem_exact_fit_bootstrap, sem_residual_fit, sem_information_criteria, compare_sem, ecvi, simple_structure, measurement_alignment, dmacs, irt_grm, irt_information.

Critical values cv_bonferroni_f, cv_bryant_paulson, cv_chisq, cv_dunnett, cv_f, cv_scheffe, cv_smm, cv_t, cv_tukey_hsd, cv_z, cv, dbryant_paulson, pbryant_paulson, qbryant_paulson.

Parameterization conversions convert_chisq_F, convert_cor_cov, convert_d_or, convert_d_r, convert_delta_lambda, convert_F_chisq, convert_f_R2, convert_lambda_delta, convert_lambda_R2,

convert_or_d, convert_r_d, convert_r_Z, convert_R2_f, convert_R2_lambda, convert_z_normal, convert_Z_r.

Variance utilities var_alpha, var_cv, var_ete, var_icc, var_indirect_effect, var_omega_squared, var_partial_r, var_r, var_R2, var_semipartial_r, var_smd, var_smd_trimmed.

Noncentral distributions moments_nc_chisq, moments_nc_F, moments_nc_t, ci_nc_t, ci_nc_F, ci_nc_chisq.

Data simulation simulate_ancova_data, simulate_ancova_factorial_data, simulate_anova_data, simulate_longitudinal_gompertz, simulate_longitudinal_logistic, simulate_longitudinal_negative_ex simulate_longitudinal_polynomial, simulate_longitudinal_richards, simulate_regression_data.

Plots plot_cfa_k, plot_ci, plot_equivalence, plot_fa_confirmatory, plot_forest, plot_irt_information, plot_mediation_mbco, plot_R2, plot_randomization_test, plot_regions_of_significance, plot_smd, plot_trajectories, plot_trajectories_fitted.

Display helpers dmar_tbl, dmar_metadata, dmar_bootstrap, format_p, as_kable, results_sentence, print_anova, print_summary, now.

Broom tidiers and model methods `tidy()` and `glance()` methods for the tables above (see dmar_tidiers), and `anova()`, `coef()`, `vcov()`, and `confint()` methods for `mlmr` and `mlmr_mv` fits.

Data sets bessel_errors (Bessel's grouped distribution of Bradley's astronomical observation errors), depression_bdi (the three-group depression study with a pretest covariate of Maxwell, Delaney, and Kelley), diagnosis_agreement (Cohen's weighted kappa illustration), drinks_trial (Smith, Meyers, and Delaney's Community Reinforcement Approach drinking trial), holzinger_swineford (the 1939 factor analysis study), prime_time_achievement (the Indiana Prime Time third grade achievement evaluation), pygmalion (Rosenthal and Jacobson's teacher-expectancy data), teacher_expectancy (Raudenbush's meta-analysis of 18 teacher-expectancy experiments), and test_market (Bryant and Bruvold's controlled test-market experiment for ANCOVA with a random covariate). Each data set's page gives its source.

Feedback. Bug reports, feature requests, and suggestions for new methods are welcomed by email to Ken Kelley <kkelley@nd.edu> (please put "DMAR" in the subject line). See https://kenkelley.org for Ken Kelley's web site, https://kenkelley.org/publications/ for related publications, and https://github.com/yelleKneK/DMAR for the project's GitHub page.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*(2), 137–152. doi:10.1037/a0028086

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Preacher, K. J., & Kelley, K. (2011). Effect size measures for mediation models: Quantitative strategies for communicating indirect effects. *Psychological Methods, 16*(2), 93–115. doi:10.1037/a0022658

See Also

Useful links:

- https://kenkelley.org
- https://yelleknek.github.io/DMAR/
- https://github.com/yelleKneK/DMAR
- Report bugs at https://github.com/yelleKneK/DMAR/issues

`adjusted_means` *Adjusted Cell and Marginal Means From a Fitted Linear Model*

Description

Given a fitted `lm` or `aov` object with one or more factors among its predictors, `adjusted_means()` returns the means the model actually compares, sometimes called least-squares means or estimated marginal means. By default the table has one row per cell of the crossed factor design, each cell's mean being the model's predicted response at that combination of factor levels with every covariate held at its sample mean (the adjusted cell means of an ANCOVA; for a model without covariates, the model-based cell means). Naming one or more factors in `by` instead returns the marginal means of those factors, formed by averaging the cell predictions over the remaining factors with either equal or frequency-proportional weights. Every mean is accompanied by its standard error and a t confidence interval on the model's residual degrees of freedom.

Usage

```
adjusted_means(
  model,
  by = NULL,
  weights = c("equal", "proportional"),
  conf_level = 0.95
)
```

Arguments

model	A fitted lm or aov object with one or more factors (and optionally covariates) on the right-hand side of the formula.
by	NULL (default) for the cell means table, or a character vector naming one or more of the model's factors for their marginal means. The output rows cross the named factors in the order given, first factor varying fastest.
weights	Weighting used to average cell predictions into marginal means, so it matters only when by is supplied and the data are unbalanced. "equal" (default) weights every combination of the averaged-over factors equally; "proportional" weights each combination by its observed frequency.
conf_level	The confidence level for the intervals (default 0.95).

Details

The reference grid and adjusted cell means. The reference grid is the crossing of the model's factor levels, enumerated in the order the factors appear in the model formula with the first factor varying fastest (the order expand.grid produces). This is the same cell order contrast_adjusted expects, so contrast weights can be read off this table row by row. Every covariate enters the grid at its sample mean, and a transformed covariate is evaluated by applying the transformation to the mean of the raw variable: with log(x) in the formula the grid carries mean(x) and the model matrix applies log(), and a poly(x, 2) basis is evaluated at $\bar{x}$, matching predict on new data at the covariate mean. Writing L for the matrix whose rows are the design-matrix rows of the grid cells, the cell means are $L\hat{\beta}$, each standard error is the square root of the corresponding diagonal element of $L\,\mathrm{vcov}(\hat{\beta})\,L'$, and each interval is the t interval on the model's residual degrees of freedom.

Marginal means and the two weightings. With by, the cell predictions are averaged over the factors not named there, and the averaging happens in the coefficient map itself: the marginal mean's L row is the weighted average of its cells' rows, so the estimate and the standard error both follow from one linear function of the coefficients. weights = "equal" weights every combination of the averaged-over factors equally; this is the population marginal mean of Searle, Speed, and Milliken (1980), the mean for a population in which every cell is equally represented regardless of the sample's cell sizes. weights = "proportional" weights each averaged-over combination by its observed frequency (in a weighted fit, by its total prior weight), so the marginal mean targets a population whose margins are shaped like the sample's. With balanced data the two weightings coincide; with unbalanced data they generally differ, and the choice between them is a substantive question about the population of interest, not a technical one (Maxwell, Delaney, and Kelley, 2027, Chapter 7).

Nonestimable means. When the fitted design is rank deficient (for example an empty factorial cell), the model has no predicted value for the affected cell, and a marginal mean that averages over such a cell does not exist either. adjusted_means() refuses with an error naming the affected rows rather than reporting a value contaminated by lm's arbitrary zero for the aliased coefficient.

Scope. The function covers single-stratum lm and aov fits with a single response. Multi-stratum aovlist fits (within-subjects designs fit with an Error() term) are refused, because a within-subjects marginal mean takes its standard error from the matching error stratum, which this function does not compute. Factors must enter the model as variables in the data, not as conversions inside the formula: y ~ factor(g) + x is refused, so convert g in the data first.

Value

A `data.frame` (class `dmar_tbl`) with one row per cell of the reference grid or, with `by`, one row per combination of the named factors. The leading columns give the factor levels; the numeric columns are `estimate` (the adjusted mean), `se` (its standard error), and `ci_lower` / `ci_upper` (the *t* confidence limits). The residual degrees of freedom of the intervals are attached as the `df_residual` attribute and, when `by` is supplied, the weighting as the `weights` attribute. The stored values keep full precision; only the display rounds (see `dmar_tbl`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 7 on nonorthogonal factorial designs and Chapter 9 on designs with covariates.)

Searle, S. R., Speed, F. M., & Milliken, G. A. (1980). Population marginal means in the linear model: An alternative to least squares means. *The American Statistician, 34*(4), 216–221.

See Also

`contrast_adjusted` for a confidence interval on a single contrast of the adjusted cell means; `ancova` for the one-way ANCOVA table; `ci_dunnett` for simultaneous many-to-one comparisons.

Other hypothesis tests: `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# 1. Cell means of a 2 x 3 factorial (no covariate): end-of-study IQ in
#    the pygmalion expectancy experiment, grades 1 through 3. The grade
#    factor is created in the data, not inside the formula.
pyg <- subset(pygmalion, grade <= 3)
pyg$grade <- factor(pyg$grade)
fit <- lm(iq_8 ~ treatment * grade, data = pyg)
adjusted_means(fit)

# 2. Marginal means of grade, averaging the cell means over treatment.
adjusted_means(fit, by = "grade")

# 3. An ANCOVA: each adjusted mean holds the covariate, here the pretest
#    depression score, at its sample mean.
fit_ancova <- lm(bdi_post ~ condition + bdi_pre, data = depression_bdi)
adjusted_means(fit_ancova)

# 4. With unbalanced cells (few bloomers in every grade) the two
```

```
#    weightings answer different questions.
adjusted_means(fit, by = "treatment")
adjusted_means(fit, by = "treatment", weights = "proportional")
```

`analysis_of_change` *Analysis of Change: Fit Change Models to One or Many Trajectories*

Description

Fits a change model to longitudinal data: any of the package's four nonlinear change models (negative exponential, logistic, Gompertz, Richards; the parameterizations of Kelley, 2005, 2008, exactly as generated by the `simulate_longitudinal_*()` simulators) or a polynomial of any order (the linear change model of `simulate_longitudinal_polynomial`). Two estimation methods are offered. The default, `method = "two_stage"`, fits each unit's curve separately, using only that unit's data, and then summarizes the unit-level parameters: their mean, their standard deviation and variance across units (the individual differences), and the standard error of the mean. `method = "mixed"` fits the proper random-coefficients mixed-effects model simultaneously, with every parameter carrying a random effect, so the reported between-unit spread is a variance component purged of estimation noise. With a single trajectory ($N = 1$, or `id = NULL`) the two-stage method reduces to one least squares fit of that unit's change, reported with its standard errors.

Usage

```
analysis_of_change(
  data,
  id,
  time,
  outcome,
 model = c("negative_exponential", "logistic", "gompertz", "richards", "polynomial"),
  method = c("two_stage", "mixed"),
  order = 1L,
  start = NULL,
  maxiter = 500L
)
```

Arguments

`data`	A `data.frame` in long format: one row per observation, as returned by the `simulate_longitudinal_*()` simulators.
`id`	Name of the column identifying units (persons, animals, trees, classrooms). `NULL` treats all rows as a single trajectory (`method = "two_stage"` only).
`time`	Name of the time column.
`outcome`	Name of the outcome column.

`model`	Which change model to fit: `"negative_exponential"` (parameters alpha, zeta, gamma), `"logistic"` (alpha, beta, gamma, zeta), `"gompertz"` (alpha, beta, gamma, zeta), `"richards"` (alpha, beta, gamma, delta, zeta), or `"polynomial"` (coefficients `b0` through `bP`, with the order P set by `order`). See the corresponding simulator's help page for what each parameter means.
`method`	How the model is estimated. `"two_stage"` (the default) fits one curve per unit, each from that unit's data alone, and summarizes across units; it is transparent, works at $N = 1$, and never lets one unit's data influence another's fit. `"mixed"` estimates the random-coefficients model simultaneously: `lme4::lmer()` for the polynomial (a linear mixed model) and `nlme::nlme()` for the nonlinear curves (started at the two-stage estimates), with uncorrelated random effects on every parameter. See Details for how to choose.
`order`	Polynomial order P when `model = "polynomial"` (1 is straight-line change, 2 quadratic, and so on). Default `1`. Ignored for the nonlinear models.
`start`	Optional named numeric vector of starting values for a nonlinear model's parameters, used for every unit (and, under `method = "mixed"`, for the fixed effects). `NULL` (default) derives data-driven starting values per unit (see Details). Ignored for the polynomial, whose fit is closed form.
`maxiter`	Maximum number of iterations passed to `nls` (and, under `method = "mixed"`, to the nonlinear mixed-effects optimizer). Default `500`.

Details

Choosing between the methods. The two-stage (curve-by-curve) route is the transparent classic: every unit's curve is inspectable, no unit's data influence another's fit, and it is the only method available for a single trajectory. Its known cost is that `sd_units` reflects the spread of *estimates*, which adds each fit's estimation noise to the true individual differences; with short or noisy trajectories it therefore overstates the population standard deviation. The mixed method estimates that between-unit variation as a variance component, separating it from level-one error, and borrows strength across units, at the price of a harder estimation problem (and, for the nonlinear curves, occasional convergence trouble; the fit is started at the two-stage estimates, and a failure suggests simplifying the model or falling back to two-stage). Under `method = "mixed"` the random effects are uncorrelated across parameters, matching the simulators' default.

Relation to existing tools. The two-stage method is the idea behind `nlme::lmList()` and `nlme::nlsList()`, and the mixed method wraps `lme4::lmer()` and `nlme::nlme()`; base R also ships self-starting curves (`SSasymp`, `SSfpl`, `SSgompertz`) in other parameterizations. What this function adds is the package's landmark parameterizations (the intercept-shifting ζ floor, the Richards δ; in `SSgompertz`'s $a\exp(-b_2 b_3^x)$ the asymptote a is a landmark of the curve, but b_2 and b_3 are not), the exact match to the `simulate_longitudinal_*()` simulators so design studies close the loop, one interface across linear and nonlinear change, and the package's tidy summary with failed fits dropped under a single counted warning.

Units whose two-stage fit does not converge are dropped with a single warning reporting how many, and the effective count is the `"n_used"` attribute. If no unit's fit converges the function stops. The polynomial fit is closed form and does not fail on any trajectory with at least $P + 1$ occasions.

Starting values. Unless `start` is supplied, each unit's nonlinear starting values are derived from that unit's data: the floor and the span from early and late observations, the inflection time from where the trajectory crosses the middle of its range, the curvature from the time the trajectory

needs to travel the central half of its range, and, for the Richards model, a logistic start (delta = 1). Cleanly measured trajectories rarely need more; hard cases (very short series, strong decreasing curves, near-flat change) may need an explicit `start`.

Value

A `data.frame` with one row per model parameter and columns

`term` The parameter name.

`estimate` The mean of the unit-level estimates (two-stage; with one trajectory, that unit's estimate) or the fixed effect (mixed).

`se` The standard error of `estimate`: the between-unit standard deviation over the square root of the number of fitted units (two-stage), the asymptotic standard error of the single fit (one trajectory), or the fixed-effect standard error (mixed).

`sd_units` The between-unit standard deviation of the parameter: the spread of the unit-level estimates (two-stage; `NA` for a single trajectory) or the random-effect standard deviation (mixed).

`var_units` The corresponding variance.

Attributes: `"model"`, `"method"`, `"n_units"` (units supplied), `"n_used"` (units whose fit converged; equal to `"n_units"` under `method = "mixed"`), `"sigma"` (the level-one residual standard deviation), and `"per_unit_estimates"` (a units-by-parameters matrix: the separate unit-level estimates under two-stage, or the unit-level predictions `coef()` under mixed, which are shrunken toward the fixed effects).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2005). *Estimating nonlinear change models in heterogeneous populations when class membership is unknown: Defining and developing the latent classification differential change model* (Doctoral dissertation). University of Notre Dame.

Kelley, K. (2008). Nonlinear change models in populations with unobserved heterogeneity. *Methodology, 4*(3), 97–112.

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-effects models in S and S-PLUS*. Springer.

Richards, F. J. (1959). A flexible growth function for empirical use. *Journal of Experimental Botany, 10*(2), 290–301.

See Also

The generators `simulate_longitudinal_negative_exponential`, `simulate_longitudinal_logistic`, `simulate_longitudinal_gompertz`, `simulate_longitudinal_richards`, `simulate_longitudinal_polynomial`; `plot_trajectories` for plotting the data being fit; `nlme::lmList()`, `nlme::nlsList()`, `lme4::lmer()`, and `nlme::nlme()` for the engines and their generalizations.

Examples

```
# Simulate a Gompertz population with individual differences, then
# recover both the mean curve and the spread of its parameters.
set.seed(113)
d <- simulate_longitudinal_gompertz(
  n_per_group = 40, target_times = 0:10,
  fixed_parameters = c(alpha = 75, beta = 3, gamma = 0.55, zeta = 10),
  random_variances = c(alpha = 25, beta = 0.4, gamma = 0.005, zeta = 4),
  error_variance = 4
)
analysis_of_change(d, id = "id", time = "time", outcome = "y",
                   model = "gompertz")

# The same data as a straight-line (order 1) polynomial: the linear
# model has nothing to say about floors, ceilings, or timing.
analysis_of_change(d, id = "id", time = "time", outcome = "y",
                   model = "polynomial", order = 1)

# A single unit's trajectory: N = 1 is one fit, with asymptotic
# standard errors in place of between-unit spread.
one <- d[d$id == levels(d$id)[1], ]
analysis_of_change(one, id = NULL, time = "time", outcome = "y",
                   model = "gompertz")

# The proper mixed-effects model, fit simultaneously across units:
# nlme::nlme() for a nonlinear curve, lme4::lmer() for the polynomial
# (the latter requires lme4 to be installed).
analysis_of_change(d, id = "id", time = "time", outcome = "y",
                   model = "gompertz", method = "mixed")
analysis_of_change(d, id = "id", time = "time", outcome = "y",
                   model = "polynomial", order = 1,
                   method = "mixed")
```

---

ancova *Analysis of Covariance (ANCOVA)*

Description

Fits a one-way analysis of covariance so the covariate-adjusted group comparison is available from a single call, without assembling the adjusted means, the omnibus test, and the effect sizes by hand. It returns the adjusted (covariate-corrected) group means with standard errors, the covariate-adjusted omnibus F for the group effect (Type III sums of squares), partial η^2 and partial ω^2 with noncentral F confidence intervals, and a homogeneity-of-regression check, returned in one `data.frame`.

Usage

```
ancova(data, outcome, treatment, covariates, conf_level = 0.95)
```

Arguments

`data`	A `data.frame` containing the response, the treatment factor, and the covariate(s).
`outcome`	Character name of the response column.
`treatment`	Character name of the grouping factor column (the groups being compared, for example treatment arms); a factor or character column.
`covariates`	Character vector of one or more covariate column names.
`conf_level`	Confidence level for the effect size CIs. Default `0.95`.

Details

Covariate-adjusted test (Type III sums of squares). The omnibus F tests the group effect after adjusting for the covariate(s), that is, the Type III sum of squares for the grouping factor. For a one-way ANCOVA (one grouping factor, with the covariate slopes held constant) the Type II and Type III sums of squares for the group effect coincide, and both equal the sequential sum of squares obtained with the covariate(s) entered first and the grouping factor last, which is how it is computed here; the value matches `car::Anova(fit, type = 3)`. The choice of sum-of-squares type changes the result only in designs with more than one factor or with interactions among factors (Maxwell, Delaney, and Kelley, 2027, Chapter 7); for those, the two-way and mixed analyses report their sum-of-squares type and allow Type I, II, or III.

Adjusted means. The adjusted mean for treatment level j is the model-predicted response at $X = \bar{X}$ (the covariate grand mean):

$$\hat{\mu}_j^{\mathrm{adj}} \;=\; \hat{\mu}_j - \sum_k \hat{\beta}_k (\bar{X}_{kj} - \bar{X}_k),$$

where $\hat{\beta}_k$ is the within-cell slope on covariate k and $\bar{X}_{kj}, \bar{X}_k$ are the per-cell and grand means of covariate k.

Homogeneity of regression. The model fit here holds the within-group covariate slopes β_k constant across groups. This is a property of the particular model being fit, not an assumption of analysis of covariance in general: it is a testable claim. Adding all group-by-covariate interactions gives an expanded model, and a model comparison F-test of the additive model against the expanded one (`stats::anova`) assesses whether the slopes differ across groups (Maxwell, Delaney, and Kelley, 2027, Chapter 9). A large F indicates the slopes are not constant, in which case the single adjusted comparison is not the whole story and the interaction model should be entertained directly. The check is reported in the `F_homogeneity_of_regression` rows.

Effect size CIs. Partial η^2 and partial ω^2 use the noncentral F framework (`ci_eta_squared` and `ci_omega_squared` with `estimand = "partial"`).

Value

A `data.frame` (class `dmar_tbl`) with rows for the omnibus test (`F_value`, `df_1`, `df_2`, `p_value`), the sum-of-squares type used (`sum_of_squares_type`; 3 for Type III), the point estimates and confidence intervals of partial η^2 and partial ω^2 (`eta_squared_partial`, `eta_squared_partial_ci_lower`, `eta_squared_partial_ci_upper`, and the same three rows for `omega_squared_partial`), the adjusted group means and their standard errors (one row per level of `treatment`), and the homogeneity-of-regression F-test. The result carries the `dmar_tbl` class, so it prints to 3 decimal places with

whole numbers (such as the degrees of freedom) shown without a decimal part and p-values to 4 decimal places (a p-value below 0.0001 prints as "< 0.0001"); the stored values keep full precision. Control the display with `print(x, digits = )` or globally with `options(dmar.digits = )` (see `dmar_tbl`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Huitema, B. E. (2011). *The analysis of covariance and alternatives* (2nd ed.). Wiley.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on analysis of covariance and Chapter 7 on higher-order designs.)

See Also

`ci_c_ancova`, `ci_sc_ancova`, `ss_aipe_sc_ancova`, `omega_squared`

Other hypothesis tests: `adjusted_means()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# 1. Compare the two groups in the Pygmalion data on eighth-grade IQ,
#    adjusting for pre-test IQ.
ancova(outcome = "iq_8", treatment = "treatment",
       covariates = "iq_pre", data = pygmalion)

# 2. The same comparison adjusting for two covariates (pre-test IQ and
#    grade).
ancova(outcome = "iq_8", treatment = "treatment",
       covariates = c("iq_pre", "grade"), data = pygmalion)
```

`anova.mlmr` *Likelihood Ratio Test for Nested Mlmr Fits*

Description

Compares two or more nested `mlmr` fits with the likelihood ratio test, delegating the chi square computation to `lavTestLRT`. The models must be nested (every parameter in the more restricted model is also in the more general one) and must be fit to the same data with the same missing data handling.

Usage

```
## S3 method for class 'mlmr'
anova(object, ...)
```

Arguments

`object`	An `mlmr` fit.
`...`	Additional `mlmr` fits, nested with respect to `object`.

Details

"The same data" means the same observations, with the variables the fits share holding the same values. It does not mean the same number of complete cases. Under `missing = "fiml"` a legitimate nested comparison routinely has different complete-case counts: adding a predictor that is itself incompletely observed lowers the number of rows complete on every modeled variable, yet the observations, and the information each of them contributes to the likelihood, are unchanged. Comparing `y ~ x1` with `y ~ x1 + x2` when `x2` has missing values is exactly the comparison full information maximum likelihood exists to support, and it is accepted. Fits run on genuinely different data are refused.

Each smaller model is refit as a constrained version of the largest one on the largest fit's data, with the slopes of its absent predictors fixed at zero. That keeps the comparison on a single joint observed variable set, which is what makes the chi square difference interpretable when the predictors are modeled as random (`fixed_x = FALSE`, the `mlmr` default).

Value

A `data.frame` of class `anova` reporting the degrees of freedom, AIC, BIC, log-likelihood, chi square test statistic, and *p*-value for each consecutive pairwise comparison.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
# Both fits ask for the Wald interval and skip the effect size block,
# since the likelihood ratio test needs neither and each costs refits.
fit1 <- mlmr(t6_paragraph_comprehension ~ t5_general_information,
             data = holzinger_swineford, ci_method = "wald",
             effect_sizes = FALSE)
fit2 <- mlmr(t6_paragraph_comprehension ~ t5_general_information +
               t9_word_meaning,
             data = holzinger_swineford, ci_method = "wald",
             effect_sizes = FALSE)
anova(fit1, fit2)
```

`anova.mlmr_mv` *Compare Nested Multivariate FIML Regression Fits*

Description

Compares two or more nested `mlmr_mv` fits with the likelihood ratio test, delegating the chi square computation to `lavTestLRT`. The models must be nested (every predictor in the more restricted model is also in the more general one) and must be fit to the same outcomes and the same data with the same missing data handling. This is the multivariate counterpart of `anova.mlmr`: because every outcome is regressed on the shared predictor set, dropping a predictor drops its slope on every outcome, so the nesting constraint sets that slope to zero across all outcomes at once.

Usage

```
## S3 method for class 'mlmr_mv'
anova(object, ...)
```

Arguments

`object`	An `mlmr_mv` fit.
`...`	Additional `mlmr_mv` fits, nested with respect to `object`.

Details

"The same data" means the same observations, with the variables the fits share holding the same values. It does not mean the same number of complete cases. Under `missing = "fiml"` a legitimate nested comparison routinely has different complete-case counts: adding a predictor that is itself incompletely observed lowers the number of rows complete on every modeled variable, yet the observations, and the information each of them contributes to the likelihood, are unchanged. That comparison is accepted. Fits run on genuinely different data are refused.

Value

A `data.frame` of class `anova` reporting the degrees of freedom, AIC, BIC, log-likelihood, chi square test statistic, and *p*-value for each consecutive pairwise comparison.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
# Both fits ask for the Wald interval and skip the effect size block,
# since the likelihood ratio test needs neither and each costs refits.
fit1 <- mlmr_mv(cbind(t6_paragraph_comprehension, t9_word_meaning) ~
                  t5_general_information,
                data = holzinger_swineford,
                ci_method = "wald", effect_sizes = FALSE)
```

```
fit2 <- mlmr_mv(cbind(t6_paragraph_comprehension, t9_word_meaning) ~
                  t5_general_information + t7_sentence,
                data = holzinger_swineford,
                ci_method = "wald", effect_sizes = FALSE)
anova(fit1, fit2)
```

`anova_within` *One Way Within-Subjects ANOVA With Sphericity Diagnostics and Corrections*

Description

Performs the univariate one-way within-subjects F test together with Mauchly's test of sphericity and the three standard ε-corrected p-values (Greenhouse-Geisser, Huynh-Feldt, and lower-bound). Returns everything in a single tidy `data.frame` so the user can decide which adjustment to report.

Usage

```
anova_within(x, id = NULL, time = NULL, outcome = NULL)
```

Arguments

`x`	Either an $n \times k$ numeric matrix or `data.frame` (rows = subjects, columns = repeated measurements); *or* a long-format `data.frame` together with `id`, `time`, and `outcome` column names.
`id`	Column name in `x` identifying the subject when `x` is in long format (`NULL` otherwise).
`time`	Column name in `x` identifying the within-subjects factor level when `x` is in long format (`NULL` otherwise).
`outcome`	Column name in `x` identifying the dependent variable when `x` is in long format (`NULL` otherwise).

Details

The unadjusted within-subjects F statistic is the same regardless of sphericity; corrections shrink the numerator and denominator degrees of freedom by a factor of $\hat{\varepsilon} \in [1/(k-1),\ 1]$, and the p-value is recomputed against the adjusted reference F distribution. When Mauchly's test rejects, prefer the Huynh-Feldt-corrected p-value (less conservative than Greenhouse-Geisser).

For multi-factor within-subjects designs or mixed designs, fit the model with `stats::aov(... + Error(id/within))` or with `lme4::lmer()` directly.

Value

A `data.frame` with one row per reported F test: `adjustment` (`"none"`, `"Greenhouse-Geisser"`, `"Huynh-Feldt"`, `"lower_bound"`), `F_value`, `df_1`, `df_2`, `p_value`, and `epsilon` (the correction factor used; `NA` for the unadjusted row). `attr(<output>, "mauchly")` contains the row from `mauchly_test`, and the partial η^2 is attached as `attr(<output>, "eta_squared_partial")`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 11.)

See Also

mauchly_test, epsilon_corrections, aov

Other within-subjects analysis: anova_within_two_way(), epsilon_corrections(), mauchly_test(), pairwise_within(), plot_trajectories_fitted()

Other hypothesis tests: adjusted_means(), ancova(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# Simulated within-subjects data with no real effect.
set.seed(113)
Y <- matrix(rnorm(20 * 4), nrow = 20)
anova_within(Y)

# Built-in within-subjects example: nlme::Orthodont (distance ~ age).
res <- anova_within(nlme::Orthodont,
                    id = "Subject", time = "age", outcome = "distance")
res
attr(res, "mauchly")
attr(res, "eta_squared_partial")
```

---

anova_within_two_way *Two-Factor Within-Subjects ANOVA With Sphericity Adjustments*

Description

Computes the full two-factor within-subjects ANOVA table, main effects A, B, and the $A \times B$ interaction with each effect tested against its own residual stratum and with Greenhouse-Geisser, Huynh-Feldt, and lower-bound sphericity adjustments applied per effect. Returns a tidy long-form `data.frame` that composes with the rest of DMAR.

Usage

```
anova_within_two_way(data, outcome, factor_A, factor_B, subject)
```

Arguments

`data`	A `data.frame` in long format, with one row per subject-by-cell observation.
`outcome`	Character name of the response column in `data`.
`factor_A`	Character name of the first within-subjects factor.
`factor_B`	Character name of the second within-subjects factor.
`subject`	Character name of the subject-id column.

Details

Design. The two within-subjects factors *A* (with a levels) and *B* (with b levels) are fully crossed; every subject contributes $a \cdot b$ observations. Each of the three fixed effects is tested against its own subject-by-effect residual stratum:

- $F_A = MS_A / MS_{A:S}$
- $F_B = MS_B / MS_{B:S}$
- $F_{A:B} = MS_{A:B} / MS_{A:B:S}$

Sphericity. Each effect's univariate *F*-ratio assumes sphericity of its corresponding subject-by-effect residual covariance matrix. Three adjustments are reported per effect: Greenhouse-Geisser (Greenhouse & Geisser, 1959), Huynh-Feldt (Huynh & Feldt, 1976), and the lower bound $\epsilon = 1/df$, where *df* is the effect's numerator degrees of freedom; this is the smallest value ϵ can attain, reached under maximal departure from sphericity.

Subjects needed to estimate epsilon. The Greenhouse-Geisser epsilon for an effect with q numerator degrees of freedom is estimated from the sample covariance matrix of q orthonormal contrasts among the effect's cell means, a different matrix for each effect (Maxwell, Delaney, & Kelley, 2027, Chapters 11 and 12). That matrix has rank at most $n - 1$, so when $n - 1 < q$ it is necessarily singular; the same rank deficiency makes the multivariate approach to a within-subjects design mathematically impossible when $n < a$ (Maxwell, Delaney, & Kelley, 2027, Chapter 13). The Greenhouse-Geisser formula still returns a number in that case, but the number is an artifact of the rank deficiency rather than an estimate: it cannot exceed $(n - 1)/q$ no matter what the population epsilon is, even under exact sphericity, where the population value is 1. Rather than report a value the design cannot support, the function reports `NA` for the Greenhouse-Geisser and Huynh-Feldt rows of any effect with $n - 1 < q$ and issues a single warning naming the condition (`car::Anova` likewise declines to report the corrections for an effect whose error matrix is singular). The unadjusted row and the lower-bound row remain: the lower bound $1/q$ is Geisser and Greenhouse's a priori bound on epsilon, valid no matter how badly sphericity is violated, and it requires no estimate of the covariance matrix (Maxwell, Delaney, & Kelley, 2027, Chapter 11).

Per-effect partial η^2. Computed as $SS_{\text{effect}} / (SS_{\text{effect}} + SS_{\text{effect, error}})$ using the appropriate subject-by-effect residual sum of squares.

Balanced data assumed. The implementation assumes a fully balanced design (every subject observed once in every cell). When the design is unbalanced, the function errors and recommends a mixed-effects fit via `lmer`.

Sums of squares are unambiguous here. Because the design is balanced, the within-subjects factors are orthogonal and the Type I, Type II, and Type III sums of squares for each effect coincide. A Type toggle is therefore not meaningful, and the reported decomposition is unambiguous: the sum of squares attributed to each effect does not depend on the order in which terms enter the model (Maxwell, Delaney, & Kelley, 2027, Chapter 12).

Value

A `data.frame` (class `dmar_tbl`) with rows for each of the three effects (`A`, `B`, `A:B`) crossed with each sphericity adjustment (`none`, `Greenhouse-Geisser`, `Huynh-Feldt`, `lower_bound`). Columns: `effect`, `adjustment`, `F_value`, `df_1`, `df_2`, `p_value`, `epsilon`, `eta_squared_partial`. When an effect has too few subjects for its epsilon to be estimable ($n - 1$ smaller than the effect's numerator degrees of freedom; see Details), the Greenhouse-Geisser and Huynh-Feldt rows for that effect carry `NA` in `epsilon`, `df_1`, `df_2`, and `p_value`, and a single warning names the condition; the unadjusted and lower-bound rows are unaffected. The table prints through the `dmar_tbl` display layer: whole numbers such as the unadjusted degrees of freedom without a decimal part, other values to 3 decimal places, and *p*-values to 4 decimal places (one below 0.0001 prints as "< 0.0001"); the stored values keep full precision. `tidy()` gives one row per row of the table, its `term` joining `effect` and `adjustment` and its `estimate` the `F_value`, and `glance()` gives one row, the number of rows (`n_terms`) with `conf_level` and `B_used` `NA`, since no interval or bootstrap is formed (see `dmar_tidiers`). `as.data.frame()` returns the table as a plain `data.frame` with the same columns and values.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Greenhouse, S. W., & Geisser, S. (1959). On methods in the analysis of profile data. *Psychometrika, 24*(2), 95–112.

Huynh, H., & Feldt, L. S. (1976). Estimation of the Box correction for degrees of freedom from sample data in randomized block and split-plot designs. *Journal of Educational Statistics, 1*(1), 69–82.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapters 11–13.)

See Also

`anova_within`, `mauchly_test`, `epsilon_corrections`

Other within-subjects analysis: `anova_within()`, `epsilon_corrections()`, `mauchly_test()`, `pairwise_within()`, `plot_trajectories_fitted()`

Examples

```
# 1. Balanced 3x2 within-subjects design simulated for illustration.
set.seed(113)
n_sub <- 12
grid  <- expand.grid(subject = factor(1:n_sub),
                     A = factor(c("a1", "a2", "a3")),
                     B = factor(c("b1", "b2")))
grid$y <- with(grid,
  3 * (A == "a2") + 1 * (A == "a3") +
  2 * (B == "b2") + 1 * ((A == "a3") & (B == "b2")) +
  rnorm(nrow(grid), 0, 1) +
  rep(rnorm(n_sub, 0, 1.5), times = 6))
```

```
anova_within_two_way(grid, outcome = "y", factor_A = "A",
                     factor_B = "B", subject = "subject")
```

`average_variance_extracted` *Average Variance Extracted (AVE)*

Description

The average variance extracted (AVE) is the mean proportion of indicator variance a factor accounts for, a standard convergent validity summary for a reflective measurement block in confirmatory factor analysis and structural equation modeling. With standardized loadings ℓ_j,

$$\mathrm{AVE} = \frac{1}{J}\sum_j \ell_j^2.$$

The quantity itself is elementary. In a model with cross-loadings, an item contributes its loading to the AVE of every factor it loads on; the same shared variance then counts toward each factor's summary, so compare AVE values across factors of such a model with that overlap in mind. A loading the model holds at zero, fixed (`0*item`) or constrained (`a*item` with `a == 0`), is not a loading: the model written with it is the model written without it, and the item is not one of the factor's indicators. Fornell and Larcker (1981) are credited for establishing it as a validity criterion: a construct shows convergent validity when its AVE reaches the conventional 0.50 (the construct explains at least half its indicators' variance), and the Fornell-Larcker discriminant criterion compares each construct's AVE with its squared correlations with the other constructs. AVE is closely related to composite reliability (omega); for discriminant validity, `htmt` gives the heterotrait-monotrait ratio of correlations, a criterion that does not use the AVE.

Usage

```
average_variance_extracted(
  fit = NULL,
  loadings = NULL,
  conf_level = 0.95,
  ci_method = c("none", "percentile"),
  B = 1000L,
  seed = NULL
)
```

Arguments

`fit`	Optional **lavaan** fit of one group and one level (for example from `cfa_1` or `lavaan::cfa`); the standardized loadings of every factor are extracted and one AVE is reported per factor. A fit whose optimizer did not converge, and a multiple-group or multilevel fit, are refused (see *The fit* under Details).

`loadings` Optional numeric vector of standardized loadings for a single block, as an alternative to `fit`. Supply exactly one of `fit` and `loadings`. No interval can be constructed from loadings alone (their sampling variability is not carried by the numbers), so `ci_method = "percentile"` requires `fit`.

`conf_level` Confidence level for the bootstrap interval (default `0.95`); used when `ci_method = "percentile"`.

`ci_method` Interval method: `"none"` (the default) or `"percentile"`. No closed-form interval for the AVE is in common use; the percentile bootstrap is the standard route in the validity literature. The cases in the fitted data are resampled with replacement `B` times, the model is refit to each resample, and each factor's interval is the pair of empirical quantiles of its `B` AVE values (Efron & Tibshirani, 1993). Replications whose refit fails or does not converge are set aside, and the interval is computed from those that return a value; a single warning reports how many were set aside, and gives the caution described under *Details* when it holds. A replication whose solution is improper for a factor is kept and counted (see *Details*). An interval requires that no more than half of the attempted replications were set aside and that at least the number the level requires returned an estimate (400 at the 95 percent level; see `B`). Otherwise the point estimates and counts are retained, with `NA` limits and a warning of class `dmar_bootstrap_insufficient`. These floors limit the error of drawing a finite number of resamples; they are not coverage guarantees.

`B` Number of bootstrap replications when `ci_method = "percentile"`, a single whole number. Default `1000`, smaller than the package's usual `10000` because every replication refits the model; raise it for a reported analysis when time allows. The fewest accepted is 400 at the default `conf_level` (200 at 0.90, and 2000 at 0.99, which the default does not meet); a smaller or fractional `B` is refused before anything is computed (class `dmar_count_invalid`). See *Details* and `dmar_bootstrap`.

`seed` Optional integer seed for the bootstrap. The default `NULL` uses the current state of the random number generator; a supplied seed is set internally and the prior state restored on exit.

Details

The fit. The AVE is read from a solution, so a fit whose optimizer did not converge (`lavaan::lavInspect(fit,"converged"` is `FALSE`, as when it reached its iteration limit) is refused with an error of class `dmar_not_converged`: its estimates mark where the optimizer stopped. A fit that converged to an improper solution is not refused; it is described under *Improper solutions* below.

A factor's standardized loadings are those of one group at one level. A factor of a multiple-group fit therefore has an AVE in each group, and a factor of a two-level fit has one at each level, even when it has the same name at both. Neither the unweighted mean of those values nor a mean weighted by group size is the AVE of any group, so the function describes a fit of one group and one level and refuses a multiple-group or multilevel fit with an error of class `dmar_argument_invalid` that names the groups (as `groups`) or the levels (as `levels`). The AVE of one group or level of such a model is `average_variance_extracted(loadings = )` given the factor's standardized loadings in that group or at that level: the factor's column of that group's or level's `lambda` matrix in `lavaan::lavInspect(fit, "std")`, without the zeros of the items that do not load on it. For

a group, a fit of the model to that group's data alone also gives it, without any equality constraints across groups.

Improper solutions. When the error variance of an indicator is estimated below zero (a Heywood case; for an ordered item, the residual variance of its latent response), its communality, one minus its standardized error variance, exceeds one, and so does its squared standardized loading in a model without cross-loadings. The AVE averages those communalities, so for a factor with such an indicator it is not a proportion of variance: its `ave` (and its interval) is `NA`, with a warning naming the indicators and their communalities, as in the measurement table of `fa_confirmatory`. On `holzinger_swineford`, a factor measured by visual perception and lozenges alone, beside one measured by general information, paragraph comprehension, and sentence completion, gives visual perception an error variance below zero and a communality of 1.30, so the first factor's AVE is `NA` (the mean of its squared standardized loadings would be .726), while the verbal factor's, .705, is reported. A larger sample, or reconsidering the model for those indicators, are the usual remedies.

No interval is formed for such a factor, so when every factor of the fit is improper the bootstrap is not run, and the table records no confidence level.

The bootstrap meets the same problem one resample at a time. A replication whose solution is improper for a factor that is proper in the observed fit is kept, as the reliability family keeps one under its default `improper = "keep"` (see `reliability_omega`), and counted. Its AVE for that factor averages a communality above one and can itself exceed one, so such replications gather in the upper tail of the factor's bootstrap distribution. The line printed under the table says how many replications were attempted, how many returned a value, and how many of those came from an improper solution, with, for two or more factors, the count for each; when the replications set aside and the improper ones kept for a factor reach the tail area beyond a limit, $(1 - C)/2$ of `B` for confidence level C, it adds a caution, since the limits then depend on how those replications were treated; with two or more factors the caution gives the share of each factor it holds for, counted for that factor alone. The warning the call gives says the same, as the reliability family's does, naming each factor the caution holds for with its count. On 80 pupils drawn from `holzinger_swineford` (`set.seed(113)`), a factor of visual perception, cubes, and lozenges beside a verbal factor is proper in the sample, with an AVE of .449, but 101 of the 198 replications that returned a value were improper for it (`B = 200`, `seed = 113`), and its upper limit, 1.19, is not a proportion of variance; the verbal factor had none. With the 2 replications whose refit failed or did not converge, 103 of the 200 are in question for the spatial factor, against the 5 that lie beyond each limit of a .95 interval, and the warning names it. The counts are those the console and a knitted report print, and they are kept in the `bootstrap` attribute.

Bootstrap replications. The percentile interval reads its two limits from the quantiles of the `B` replicated AVE values of each factor, so it is run only when `B` puts at least 10 replications beyond each limit (400 at the 95 percent level). A replication in which an AVE is undefined (the refit failed or did not converge) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, beside the fuller `bootstrap` list, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the limits of every factor are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. `options(dmar.bootstrap_exploratory =`

TRUE) lets a smaller B (2 or more) run for a first look or a demonstration; every such call gives a warning of class dmar_bootstrap_exploratory and the result is not one to report. See dmar_bootstrap for the rule and its reasons.

The percentile bootstrap interval resamples the cases behind fit and refits the model once per replication, so its cost is B model fits. That refitting is why the examples below stop at the point estimates: the fewest replications a 95 percent interval is reported from, B = 400, run well beyond the time an example is given on the two-factor model there. To obtain the interval, pass ci_method = "percentile" together with a seed, as in average_variance_extracted(fit, ci_method = "percentile", seed = 113), and keep the default B = 1000 or more for a reported analysis.

A supplied two-stage fit requires eligible unrestricted first-stage moments. Bootstrap intervals currently refuse separately stored two-stage auxiliary variables because the refits do not carry their resampled values. Resample and refit items and auxiliaries together explicitly when that analysis is needed.

Value

A data.frame (classes dmar_ave and dmar_tbl) with one row per factor: factor (label), ave, and ci_lower / ci_upper (the percentile bootstrap limits; NA when ci_method = "none"). All three are NA for a factor whose solution is improper (see Details). With ci_method = "percentile" and at least one factor whose solution is proper, so that an interval is formed, the table carries the attributes conf_level, B (requested), B_used (the replications the intervals rest on), n_dropped, B_required, and bootstrap_status (see *Bootstrap replications* under Details), and bootstrap, the counts in the form the reliability family records them (see reliability_omega): attempted (B), returned, failed (the replications whose refit failed or did not converge, which **lavaan**'s bootstrap does not tell apart, counted in reasons as non-finite), reasons, error_messages, undetermined, improper, and no_solution (all zero here), improper_kept (the replications used whose solution was improper for at least one factor, including when its interval is withheld), not_converged (zero), refit_failed (the same count as failed, under the reason that the line under the table and the warning give for both causes, that the refit failed or did not converge), improper_rule ("keep"), caution, and improper_kept_by_factor, the improper replications kept for each factor, with factors_with_intervals recording which intervals were eligible.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Fornell, C., & Larcker, D. F. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of Marketing Research, 18*(1), 39–50.

See Also

htmt for discriminant validity; reliability_omega for the composite reliability of the same block (coefficient omega is what the composite-reliability literature computes); cfa_1 to obtain the fit.

Other multivariate and latent variable methods: bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
# Directly from the standardized loadings a paper reports:
average_variance_extracted(loadings = c(.8, .7, .6))

# From a fitted model, one AVE per factor (requires lavaan).
data(holzinger_swineford)
fit <- lavaan::cfa(
  "verbal    =~ t6_paragraph_comprehension + t7_sentence +
                t9_word_meaning
   deduction =~ t20_deduction + t22_problem_reasoning +
                t23_series_completion",
  data = holzinger_swineford)
ave_tbl <- average_variance_extracted(fit)
ave_tbl

# The Fornell and Larcker (1981) discriminant criterion compares each
# factor's AVE with the squared correlation between the factors: a
# factor should account for more of its own indicators' variance than
# it shares with the other factor. Here the comparison favors verbal
# and goes against deduction, whose AVE falls below the shared
# variance. Fitting with fa_confirmatory(..., output = "measurement") puts the
# AVE values and the latent correlations in one table.
lavaan::lavInspect(fit, "cor.lv")["verbal", "deduction"]^2

# The broom verbs: one row per factor.
generics::tidy(ave_tbl)
generics::glance(ave_tbl)
```

---

bayes_independent_t *Bayesian Independent-Samples* t *Analysis*

Description

The Bayesian counterpart of the two-sample (pooled-variance) t test. It reports the posterior of the standardized mean difference $\delta = (\mu_1 - \mu_2)/\sigma$ under the default Jeffreys-Zellner-Siow (JZS) prior, a Cauchy prior on δ with Jeffreys priors on the nuisance parameters (the variance), summarized by its median, mean, a credible interval, and the probability statement $P(\delta > 0 \mid \text{data})$. The JZS default Bayes factor (Rouder, Speckman, Sun, Morey, & Iverson, 2009) is also reported. See bayes_one_sample_t for the model, the package's interpretive stance, and the computational details (exact quadrature, no Monte Carlo error).

Usage

```
bayes_independent_t(
  x = NULL,
  y = NULL,
  mean_1 = NULL,
  sd_1 = NULL,
  n_1 = NULL,
  mean_2 = NULL,
  sd_2 = NULL,
  n_2 = NULL,
  prior_location = 0,
  prior_scale = sqrt(2)/2,
  prior_mean = NULL,
  prior_sd = NULL,
  conf_level = 0.95
)
```

Arguments

`x`, `y`	Numeric vectors: the observations of the two independent groups (δ is positive when `x` runs higher). Omit both to supply summary statistics instead.
`mean_1`, `sd_1`, `n_1`	Summary statistics of the first group: mean, standard deviation, and sample size. The Bayes factor depends on the data only through the *t* statistic and the sample sizes, so the summary form is exact, not an approximation. Supply either raw data or all six summary values, never both.
`mean_2`, `sd_2`, `n_2`	Summary statistics of the second group.
`prior_location`	Location of the Cauchy prior on δ. Defaults to 0, the JZS prior; a nonzero value centers the prior on an expected effect (Gronau, Ly, & Wagenmakers, 2020).
`prior_scale`	The way to adjust the prior. It is the scale (width) of the Cauchy prior on the standardized effect δ (the JZS prior), so a user who wants a more or less informative prior sets `prior_scale`: larger values say larger effects are plausible a priori, smaller values concentrate the prior near zero. The default $\sqrt{2}/2 \approx 0.707$ is the JZS "medium" prior. Fully custom or subjective priors beyond the Cauchy family are not supported by the **BayesFactor** engine.
`prior_mean`, `prior_sd`	Mean and standard deviation of a normal prior on δ, for prior beliefs stated as moments. Supplying them selects the normal prior; they cannot be combined with the Cauchy arguments. See the prior section of Details.
`conf_level`	Probability mass of the credible interval.

Details

The likelihood of the pooled-variance *t* statistic given δ is noncentral *t* with $df = n_1 + n_2 - 2$ and noncentrality $\delta\sqrt{n_1 n_2/(n_1 + n_2)}$; equal variances are assumed, as in the standard JZS development. The raw-scale rows transform the δ summaries through the pooled standard deviation. The noncentral *t* density is computed exactly, as described in `bayes_one_sample_t`.

Value

A `data.frame` (class `dmar_tbl`) with the same rows as `bayes_one_sample_t`, plus `n_1` and `n_2`.

Specifying the prior. The default prior on the standardized effect δ is the JZS Cauchy centered at zero. Its `prior_scale` r is not a standard deviation: a Cauchy has no mean and no variance (those integrals diverge), so beliefs stated as prior moments cannot be expressed through it. What the scale does fix is the quartiles: half the prior mass lies within $\pm r$ of the location, so the default $r = \sqrt{2}/2$ says a 50 percent prior bet that $|\delta| < 0.71$. A directional prior keeps the Cauchy and moves `prior_location` (Gronau, Ly, & Wagenmakers, 2020). A researcher who thinks in prior moments instead sets `prior_mean` and `prior_sd`, which use a normal prior with exactly those moments; the two families are exclusive.

The families are also linked by an exact identity: a Cauchy with location μ and scale r is a normal prior $N(\mu, r^2/z^2)$ whose z is standard normal, that is, a normal prior whose variance you are not sure of. Choosing the Cauchy is therefore choosing a normal prior with built-in doubt about its own width, which is why its tails are heavier and its Bayes factors more conservative. A normal matched to the Cauchy's interquartile range has `prior_sd = 1.4826 * prior_scale`. The full posterior of δ is returned in the `"posterior"` attribute as a data frame of `delta` and `density`, so any posterior probability, not only the reported ones, can be computed from it.

A standardized effect size enters through the summary form directly: an observed Cohen's d with group sizes n_1 and n_2 is `mean_1 = d, mean_2 = 0, sd_1 = 1, sd_2 = 1`, since d is the mean difference in pooled standard deviation units.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Gronau, Q. F., Ly, A., & Wagenmakers, E.-J. (2020). Informed Bayesian t-tests. *The American Statistician, 74*(2), 137–143. doi:10.1080/00031305.2018.1562983

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review, 16*(2), 225–237. doi:10.3758/PBR.16.2.225

Jeffreys, H. (1961). *Theory of probability* (3rd ed.). Oxford University Press.

Zellner, A., & Siow, A. (1980). Posterior odds ratios for selected regression hypotheses. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, & A. F. M. Smith (Eds.), *Bayesian statistics: Proceedings of the First International Meeting* (pp. 585–603). University of Valencia Press.

See Also

`bayes_one_sample_t` and `bayes_paired_t` for the other designs; `ci_smd` and `welch_t` for the frequentist analyses of the same comparison.

Other Bayesian t analyses: `bayes_one_sample_t()`, `bayes_paired_t()`

Examples

```
set.seed(113)
g1 <- rnorm(35, 105, 15)
```

```
g2 <- rnorm(35, 100, 15)
bayes_independent_t(g1, g2)

# The probability that the effect is positive is read directly off the
# p_delta_positive row: a probability statement, not a p-value.
```

`bayes_one_sample_t` *Bayesian One-Sample* t *Analysis*

Description

The Bayesian counterpart of the one-sample *t* test. It reports the posterior distribution of the standardized effect $\delta = (\mu - \mu_0)/\sigma$, where μ_0 is `null_mean`, under the default Jeffreys-Zellner-Siow (JZS) prior, a Cauchy prior on δ with Jeffreys priors on the nuisance parameters (the variance), summarized by its median, mean, a credible interval, and the direct probability statement $P(\delta > 0 \mid \text{data})$. The JZS default Bayes factor (Rouder, Speckman, Sun, Morey, & Iverson, 2009) is also reported.

Usage

```
bayes_one_sample_t(
  x = NULL,
  null_mean = 0,
  mean = NULL,
  sd = NULL,
  N = NULL,
  prior_location = 0,
  prior_scale = sqrt(2)/2,
  prior_mean = NULL,
  prior_sd = NULL,
  conf_level = 0.95
)
```

Arguments

`x`	Numeric vector of observations. Omit to supply summary statistics instead.
`null_mean`	The comparison value for the mean under the point null, μ_0 in the formulas (and the centering value for δ). Defaults to 0.
`mean`, `sd`, `N`	Summary statistics: the sample mean, standard deviation, and total sample size. The Bayes factor depends on the data only through the t statistic and N, so the summary form is exact, not an approximation. Supply either `x` or all three summary values, never both. An argument `n` is refused by name, with a condition of class `dmar_argument_invalid`, since R would otherwise match it to `null_mean` by its leading character and take it as the comparison value.
`prior_location`	Location of the Cauchy prior on δ. Defaults to 0, the JZS prior; a nonzero value centers the prior on an expected effect (Gronau, Ly, & Wagenmakers, 2020).

`prior_scale` The way to adjust the prior. It is the scale (width) r of the Cauchy prior on the standardized effect δ (the JZS prior), so a user who wants a more or less informative prior sets `prior_scale`: larger values say larger effects are plausible a priori, smaller values concentrate the prior near zero. The default $\sqrt{2}/2 \approx 0.707$ is the JZS "medium" prior. Fully custom or subjective priors beyond the Cauchy family are not supported by the **BayesFactor** engine.

`prior_mean`, `prior_sd` Mean and standard deviation of a normal prior on δ, for prior beliefs stated as moments. Supplying them selects the normal prior; they cannot be combined with the Cauchy arguments. See the prior section of Details.

`conf_level` Probability mass of the (central) credible interval. Defaults to 0.95.

Details

The posterior is a *probability statement* about the parameter given the model, the prior, and the data, and that is how these functions are meant to be read. A Bayes factor is a different kind of claim, a comparison of how well two models predicted the data, and it leans harder on the prior; it is reported because it may be helpful for some questions. Neither replaces the estimation-first habits of the rest of the package; `ci_sm` and `ci_smd` remain the frequentist complements.

With a Jeffreys prior on (μ_0, σ^2) and $\delta \sim \text{Cauchy}(0, r)$, all inference flows through the observed *t* statistic, whose likelihood given δ is noncentral *t* with noncentrality $\delta\sqrt{N}$. The posterior of δ is computed by quadrature (no Monte Carlo error) and the Bayes factor by the one-dimensional integral of that likelihood against the Cauchy prior, the exact JZS form. The noncentral *t* density is exact: R's `dt()` is used where it is accurate, and elsewhere (a noncentrality above 37.62, where it rests on a normal approximation, more than 1,000 degrees of freedom, or $|t| < .1$, where it loses accuracy to cancellation) the density is integrated over the sampling distribution of the standard deviation. The raw-scale rows transform the δ summaries through the sample standard deviation (a plug-in, as is conventional for reporting).

Value

A `data.frame` (class `dmar_tbl`) with the posterior summaries of δ (`delta_posterior_median`, `delta_posterior_mean`, `delta_lower`, `delta_upper`, `p_delta_positive`), the same summaries mapped to the raw mean difference scale (`raw_*`), the Bayes factors (`bf_10`, `bf_01`), the observed `t` and `df`, the `prior_scale`, and `N`.

Specifying the prior. The default prior on the standardized effect δ is the JZS Cauchy centered at zero. Its `prior_scale` r is not a standard deviation: a Cauchy has no mean and no variance (those integrals diverge), so beliefs stated as prior moments cannot be expressed through it. What the scale does fix is the quartiles: half the prior mass lies within $\pm r$ of the location, so the default $r = \sqrt{2}/2$ says a 50 percent prior bet that $|\delta| < 0.71$. A directional prior keeps the Cauchy and moves `prior_location` (Gronau, Ly, & Wagenmakers, 2020). A researcher who thinks in prior moments instead sets `prior_mean` and `prior_sd`, which use a normal prior with exactly those moments; the two families are exclusive.

The families are also linked by an exact identity: a Cauchy with location μ and scale r is a normal prior $N(\mu, r^2/z^2)$ whose z is standard normal, that is, a normal prior whose variance you are not sure of. Choosing the Cauchy is therefore choosing a normal prior with built-in doubt about its own width, which is why its tails are heavier and its Bayes factors more conservative. A normal matched

to the Cauchy's interquartile range has `prior_sd = 1.4826 * prior_scale`. The full posterior of δ is returned in the `"posterior"` attribute as a data frame of `delta` and `density`, so any posterior probability, not only the reported ones, can be computed from it.

A standardized effect size enters through the summary form directly: an observed d relative to `null_mean = 0` is `mean = d, sd = 1`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Gronau, Q. F., Ly, A., & Wagenmakers, E.-J. (2020). Informed Bayesian t-tests. *The American Statistician, 74*(2), 137–143. doi:10.1080/00031305.2018.1562983

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review, 16*(2), 225–237. doi:10.3758/PBR.16.2.225

Jeffreys, H. (1961). *Theory of probability* (3rd ed.). Oxford University Press.

Zellner, A., & Siow, A. (1980). Posterior odds ratios for selected regression hypotheses. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, & A. F. M. Smith (Eds.), *Bayesian statistics: Proceedings of the First International Meeting* (pp. 585–603). University of Valencia Press.

See Also

`bayes_paired_t` and `bayes_independent_t` for the two-sample designs; `ci_sm` for the frequentist standardized mean.

Other Bayesian t analyses: `bayes_independent_t()`, `bayes_paired_t()`

Examples

```
set.seed(113)
x <- rnorm(40, mean = 0.4, sd = 1)
bayes_one_sample_t(x)

# Against a nonzero comparison value, with a wider prior.
bayes_one_sample_t(x, null_mean = 0.1, prior_scale = 1)
```

`bayes_paired_t` *Bayesian Paired-Samples* t *Analysis*

Description

The Bayesian counterpart of the paired *t* test, the analysis of `bayes_one_sample_t` applied to the within-pair differences. It reports the posterior of the standardized difference $\delta = \mu_D/\sigma_D$ under the default Jeffreys-Zellner-Siow (JZS) prior, a Cauchy prior on δ with Jeffreys priors on the nuisance parameters (the variance), summarized by its median, mean, a credible interval, and the probability statement $P(\delta > 0 \mid \text{data})$. The JZS default Bayes factor (Rouder, Speckman, Sun, Morey, & Iverson, 2009) is also reported. See `bayes_one_sample_t` for the model, the package's interpretive stance, and the computational details (exact quadrature, no Monte Carlo error).

Usage

```
bayes_paired_t(
  x = NULL,
  y = NULL,
  mean_diff = NULL,
  sd_diff = NULL,
  N = NULL,
  prior_location = 0,
  prior_scale = sqrt(2)/2,
  prior_mean = NULL,
  prior_sd = NULL,
  conf_level = 0.95
)
```

Arguments

`x`, `y`	Numeric vectors of paired observations, the same length, in matching order. The analysis is of `x - y`. Omit both to supply summary statistics of the differences instead.
`mean_diff`, `sd_diff`, `N`	Summary statistics of the paired differences: their mean, their standard deviation, and the number of pairs. These are the quantities a paper's paired *t* test reports. Supply either raw data or all three summary values, never both.
`prior_location`	Location of the Cauchy prior on δ. Defaults to 0, the JZS prior; a nonzero value centers the prior on an expected effect (Gronau, Ly, & Wagenmakers, 2020).
`prior_scale`	The way to adjust the prior. It is the scale (width) of the Cauchy prior on the standardized effect δ (the JZS prior), so a user who wants a more or less informative prior sets `prior_scale`: larger values say larger effects are plausible a priori, smaller values concentrate the prior near zero. The default $\sqrt{2}/2 \approx 0.707$ is the JZS "medium" prior. Fully custom or subjective priors beyond the Cauchy family are not supported by the **BayesFactor** engine.
`prior_mean`, `prior_sd`	Mean and standard deviation of a normal prior on δ, for prior beliefs stated as moments. Supplying them selects the normal prior; they cannot be combined with the Cauchy arguments. See the prior section of Details.
`conf_level`	Probability mass of the credible interval.

Value

A `data.frame` (class `dmar_tbl`) with the same rows as `bayes_one_sample_t`, where δ is the standardized within-pair difference and the raw rows are on the difference scale; `N` is the number of pairs.

Specifying the prior. The default prior on the standardized effect δ is the JZS Cauchy centered at zero. Its `prior_scale` r is not a standard deviation: a Cauchy has no mean and no variance (those integrals diverge), so beliefs stated as prior moments cannot be expressed through it. What the scale does fix is the quartiles: half the prior mass lies within $\pm r$ of the location, so the default $r = \sqrt{2}/2$ says a 50 percent prior bet that $|\delta| < 0.71$. A directional prior keeps the Cauchy and moves `prior_location` (Gronau, Ly, & Wagenmakers, 2020). A researcher who thinks in prior moments instead sets `prior_mean` and `prior_sd`, which use a normal prior with exactly those moments; the two families are exclusive.

The families are also linked by an exact identity: a Cauchy with location μ and scale r is a normal prior $N(\mu, r^2/z^2)$ whose z is standard normal, that is, a normal prior whose variance you are not sure of. Choosing the Cauchy is therefore choosing a normal prior with built-in doubt about its own width, which is why its tails are heavier and its Bayes factors more conservative. A normal matched to the Cauchy's interquartile range has `prior_sd = 1.4826 * prior_scale`. The full posterior of δ is returned in the `"posterior"` attribute as a data frame of `delta` and `density`, so any posterior probability, not only the reported ones, can be computed from it.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Gronau, Q. F., Ly, A., & Wagenmakers, E.-J. (2020). Informed Bayesian t-tests. *The American Statistician, 74*(2), 137–143. doi:10.1080/00031305.2018.1562983

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review, 16*(2), 225–237. doi:10.3758/PBR.16.2.225

Jeffreys, H. (1961). *Theory of probability* (3rd ed.). Oxford University Press.

Zellner, A., & Siow, A. (1980). Posterior odds ratios for selected regression hypotheses. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, & A. F. M. Smith (Eds.), *Bayesian statistics: Proceedings of the First International Meeting* (pp. 585–603). University of Valencia Press.

See Also

`bayes_one_sample_t` for the model and stance; `bayes_independent_t` for unpaired groups; `probability_of_superiorit` and `randomization_test_paired` for other paired analyses.

Other Bayesian t analyses: `bayes_independent_t()`, `bayes_one_sample_t()`

Examples

```
set.seed(113)
before <- rnorm(30, 100, 12)
after  <- before + rnorm(30, 3, 6)
```

```
bayes_paired_t(after, before)
```

`bessel_errors` *Bessel's (1818) Grouped Frequency Distribution of Bradley's Astronomical Observation Errors*

Description

The nine-bin grouped frequency distribution that Friedrich Wilhelm Bessel published in 1818 for the absolute errors of 300 stellar position observations made by British Astronomer Royal James Bradley at the Greenwich Observatory between 1750 and 1762. Bessel compared the empirical distribution of these errors to the normal distribution, providing one of the early empirical demonstrations that observational errors are approximately normally distributed, a position Gauss had developed on theoretical grounds a decade earlier. The data are reproduced from Maxwell, Delaney, and Kelley (2027, *Designing Experiments and Analyzing Data: A Model Comparison Perspective*, 4th ed., Routledge), Table 1.4.

Usage

```
bessel_errors
```

Format

A data frame with 9 observations on 6 variables, one row per bin of the grouped frequency distribution. The error magnitudes are in seconds of arc.

`bin` Integer bin index, 1 through 9.

`lower` Lower edge of the bin (inclusive), in seconds of arc.

`upper` Upper edge of the bin (exclusive), in seconds of arc.

`midpoint` Bin midpoint, `(lower + upper) / 2`, in seconds of arc. The conventional plug-in value when approximating moments from a grouped frequency distribution.

`observed` Empirical frequency: the number of Bradley's 300 absolute errors that fell in the bin.

`expected` Expected frequency under a normal distribution with mean 0 and standard deviation approximately 0.22 seconds of arc (a least squares fit to the expected counts gives 0.216). These are Bessel's own normal-model expectations as reproduced in Maxwell, Delaney, and Kelley (2027). Both the observed and expected columns sum to 300.

Details

This data set ships in the original grouped form Bessel reported. The 300 individual error values are not available; what Bessel published, and what is reproduced here, is the 9-bin frequency distribution. Computations that require the underlying continuous values must either be approximated from the bin midpoints (the usual weighted-moments approach, illustrated in the examples) or estimated parametrically by assuming a distributional form within each bin.

Historical context. Friedrich Wilhelm Bessel (1784–1846) was a German astronomer and mathematician best known to statisticians for the Bessel correction (`n - 1` in the unbiased variance estimator) and the Bessel functions. The 1818 monograph that contains this frequency distribution is part of a much larger effort to produce a reference catalog of stellar positions, the *Fundamenta astronomiae*, derived from the observations of James Bradley (1693–1762), the third Astronomer Royal of Britain. The observational errors are small (most under 0.5 seconds of arc) and approximately normally distributed.

Why this data set matters for measurement and analysis. Bessel's 1818 comparison is one of the earliest empirical demonstrations that observational error is approximately normal, complementing the theoretical case Gauss had made on independent grounds. It is also a clean worked example for the approximation of moments from grouped frequency data when only binned counts (rather than individual observations) are available, a common situation in published reports.

Approximating moments from grouped data. When the underlying continuous values are unavailable, the standard approach is to plug the bin midpoints in for the unknown individual values and form a weighted mean and weighted variance using the bin frequencies as weights. The frequency-weighted mean and variance are

$$\bar{x}_w = \frac{\sum_k f_k m_k}{\sum_k f_k}, \qquad s_w^2 = \frac{\sum_k f_k (m_k - \bar{x}_w)^2}{(\sum_k f_k) - 1}$$

where f_k is the bin frequency and m_k is the bin midpoint. The examples below compute both, on the observed frequencies and on Bessel's normal-model expected frequencies. The two are close, consistent with Bessel's conclusion that the empirical and theoretical distributions agree.

Author(s)

Ken Kelley

Source

Bessel, F. W. (1818). *Fundamenta astronomiae pro anno MDCCLV deducta ex observationibus viri incomparabilis James Bradley in specula astronomica Grenovicensi per annos 1750–1762 institutis* [Foundations of astronomy for the year 1755, deduced from the observations of the incomparable man James Bradley at the Greenwich astronomical observatory during 1750–1762]. Friedrich Nicolovius.

Reproduced in Maxwell, Delaney, and Kelley (2027), Table 1.4.

References

Bessel, F. W. (1818). *Fundamenta astronomiae pro anno MDCCLV deducta ex observationibus viri incomparabilis James Bradley in specula astronomica Grenovicensi per annos 1750–1762 institutis*. Friedrich Nicolovius.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 1, Section 1.4.5.2, on the historical and empirical basis for the normal distribution.)

Kelley, K. (2026). *DMAR: Methods for the design, measurement, and analysis of human-centered outcomes in R* [R package]. https://github.com/yelleKneK/DMAR

Stigler, S. M. (1986). *The history of statistics: The measurement of uncertainty before 1900*. Belknap Press of Harvard University Press. (See Chapter 5 on the development of the normal distribution and the role of astronomical errors.)

Examples

```
data(bessel_errors)
bessel_errors

# Total frequencies (each column should sum to 300).
colSums(bessel_errors[, c("observed", "expected")])

# Weighted mean of the absolute error, approximating the
# individual observations by the bin midpoints. Because every
# bin lower edge is at or above zero, this is the mean
# absolute error rather than the mean error itself.
wmean_obs <- with(bessel_errors,
                  sum(observed * midpoint) / sum(observed))
wmean_exp <- with(bessel_errors,
                  sum(expected * midpoint) / sum(expected))
c(observed = wmean_obs, expected_under_normal = wmean_exp)

# Weighted variance and standard deviation of the absolute
# error, using bin midpoints as plug-in values for the
# individual observations.
wvar_obs <- with(bessel_errors,
                 sum(observed * (midpoint - wmean_obs)^2) /
                 (sum(observed) - 1))
c(weighted_variance = wvar_obs,
  weighted_sd       = sqrt(wvar_obs))

# Side-by-side bar plot of observed and expected counts.
# Bessel's normal-model expectation tracks the empirical
# distribution closely except in the long right tail, where
# Bradley's three largest errors (counts 3, 1, 1) exceed what
# the normal model predicts.
op <- par(mar = c(5, 4, 4, 2))
barplot(rbind(bessel_errors$observed, bessel_errors$expected),
        beside    = TRUE,
        names.arg = sprintf("%.1f-%.1f",
                            bessel_errors$lower,
                            bessel_errors$upper),
        legend.text = c("Observed (Bradley)",
                        "Expected under normal model"),
        args.legend = list(x = "topright", bty = "n"),
        xlab = "Absolute error (seconds of arc)",
        ylab = "Frequency",
        main = "Bessel (1818) on Bradley's 300 stellar positions")
par(op)
```

`bifactor_indices` *Bifactor Model Dimensionality and Reliability Indices*

Description

Computes the indices that describe how much of the common variance of a scale's items, and of the variance of its total and subscale scores, a general factor accounts for beside the group factors of its subscales (Rodriguez, Reise, and Haviland, 2016): the explained common variance (ECV), coefficient omega and omega hierarchical (omega_H) for the general factor, omega hierarchical subscale (omega_HS) for each group factor, the percentage of uncontaminated correlations (PUC), and the index those authors call coefficient H, the closed form of Hancock and Mueller (2001) applied to one factor's loadings, which counts the other factors' common variance as uncorrelated error and so is not the maximal reliability of the factor under the fitted model; that maximal reliability is returned beside the table. The input is a fitted bifactor model in which one general factor loads on every item and each item loads on at most one of two or more orthogonal group factors. `fa_bifactor` fits such a model and reports these indices with delta method intervals, with omega, omega_H, and omega_HS for the sum of the items as scored rather than of the standardized items (see *Details*).

Usage

```
bifactor_indices(
  fit,
  general = NULL,
  denominator = c("model_implied", "observed")
)
```

Arguments

`fit`	A fitted bifactor **lavaan** model: one general factor on all items plus two or more orthogonal group factors, each item on one group factor at most. An item on no group factor is on the general factor alone: it counts in the general factor's indices and in PUC, and in no subscale. An item listed under two or more group factors is refused unless all but one of those loadings are held at zero, fixed (`0*item`) or constrained (`a*item` with `a == 0`); such a loading describes the same model as leaving the item off that factor. The factors must be orthogonal (the bifactor specification), and a fit that correlates any two of them is refused. Residual covariances among the items are allowed and enter the model implied denominators. A fit with fewer than two group factors, a multiple-group fit, a multilevel fit, and a fit whose optimizer did not converge are refused (see *Details*).
`general`	Optional name of the general factor. When `NULL` (default) the general factor is detected as the one that loads on every item.
`denominator`	How the total-score variance in the denominators of omega, omega_H, and omega_HS is estimated: `"model_implied"` (the default; the variance the fitted bifactor model reproduces, as in Rodriguez, Reise, and Haviland, 2016, with any residual covariances the model specifies) or `"observed"` (the variance

of the composite estimated from the data, the denominator of robust omega in `reliability_omega`). For a fit to ordered items both describe the sum of the latent responses, not of the summed categories. See *Details*.

Details

Let λ_i^g be item i's standardized loading on the general factor, λ_i^s its loading on its group factor (0 for an item on the general factor alone), and θ_i its standardized residual variance. With orthogonal factors the overall ECV is $\sum_i(\lambda_i^g)^2 / \sum_i[(\lambda_i^g)^2 + (\lambda_i^s)^2]$; omega and omega_H share the total-score variance $(\sum_i \lambda_i^g)^2 + \sum_g(\sum_{i\in g} \lambda_i^s)^2 + \sum_i \theta_i$ as denominator, with the general part $(\sum_i \lambda_i^g)^2$ in the numerator of omega_H. Each group factor's omega_HS uses the analogous numerator and that subscale's own total variance. PUC is one minus the share of item pairs that fall within the same group factor. Omega_H against omega says how much of the total score's reliable variance is due to the general factor; omega_HS against a subscale's omega says how much of the subscale score's reliable variance is its own, beyond the total; ECV and PUC describe how close the items come to measuring one thing. The table reports the numbers; whether a total or subscale score is worth reporting is the researcher's judgment, and the package draws no line, as on the page of `fa_bifactor`.

The metric of omega. The loadings and residual variances above are standardized, so omega, omega_H, and omega_HS describe the sum of the standardized items, as Rodriguez, Reise, and Haviland (2016) write them. `fa_bifactor` reports them for the sum of the items as scored, the coefficients `fa_confirmatory` and `semTools::compRelSEM()` compute. The two agree when the items have equal variances and differ otherwise: in the first example below, whose items have nearly equal variances, omega_H is 0.67348 here and 0.67305 for the sum of the items as scored, while on the 15 tests of the example of `fa_bifactor`, whose variances range from 8.0 to 153.3, it is 0.577 here and 0.552 there. ECV, *H*, the maximal reliability, and PUC do not differ.

A group factor's row. The row of group factor s describes its items, the subscale s, and its ECV and omega are not the general row's quantities restricted to a group factor. Its ECV is the general factor's share of the subscale's common variance,

$$\frac{\sum_{i\in s}(\lambda_i^g)^2}{\sum_{i\in s}[(\lambda_i^g)^2 + (\lambda_i^s)^2]},$$

so one minus it is the group factor's own share of that variance, and a high value says the subscale's common variance is mostly general (it goes with a low omega_HS). It is not the group factor's share of the common variance of all the items, which is $\sum_{i\in s}(\lambda_i^s)^2$ over the denominator of the overall ECV. Its omega is the omega of the subscale's total score, $(\sum_{i\in s} \lambda_i^g)^2 + (\sum_{i\in s} \lambda_i^s)^2$ over that score's variance, the reliability of the subscale score with both factors counted as true score; omega_HS keeps only the group factor's part of the numerator. Its H is the index described next, computed from the group factor's loadings. The printed table states what ECV and omega mean on these rows.

Coefficient H. The column `H` is the index Rodriguez, Reise, and Haviland (2016) report: the closed form of Hancock and Mueller (2001),

$$H = \frac{\sum_i \lambda_i^2/(1-\lambda_i^2)}{1 + \sum_i \lambda_i^2/(1-\lambda_i^2)},$$

over the general loadings for the general factor and over a group factor's loadings for that factor. It counts $1 - \lambda_i^2$ as item i's error variance, and so treats the variance an item shares with the other

factors as error that is uncorrelated across items. In a bifactor model that variance is correlated (the items of a subscale share their group factor, and every item shares the general factor), so the index is not the maximal reliability of the factor under the fitted model, which is what `reliability_H` computes for a block of items whose errors are correlated (Gabler & Raykov, 2017). That maximal reliability is $\lambda_f' R^{-1} \lambda_f$, where λ_f holds the factor's standardized loadings (zero for an item off the factor) and R is the item correlation matrix the model reproduces: the squared multiple correlation of the factor with all of the items, the largest reliability any weighted sum of them reaches as a measure of that factor, and the square of the determinacy of its regression factor score in `factor_scores`. It is returned as the `"maximal_reliability"` attribute and printed under the table, and `ci_bifactor_indices` gives it a confidence interval, as it does every index of the table but PUC, which depends on the model's structure alone.

The two differ in either direction, and what sets the direction is not the same on the two kinds of row. On the general factor's row the index counts the group factors' variance as uncorrelated error, but the items of a subscale share it. When the loadings on each group factor are positive and similar, no weighting of the items removes that shared variance without removing general variance with it, and the index is the larger: in the first example below it is 0.769 for the general factor, whose maximal reliability is 0.678, barely above its omega_H of 0.673, so no weighting of the items reaches 0.769. When an item is nearly a pure indicator of its group factor, or a group factor's loadings differ in sign, a weighting can cancel group variance that the index counts as error, and the maximal reliability is the larger. On a group factor's row the index counts the general factor's variance, with the unique variance, as uncorrelated error. When the subscale's own items have proportional loadings on the two factors (equal loadings, for instance), no weighting of them alone removes the general factor without removing the group factor with it, but the optimal weights use every item, and the items of the other subscales measure the general factor and nothing of this group factor, so the weights use them to remove it. The better those items measure the general factor, the closer the maximal reliability comes to the closed form above with each item's unique variance θ_i in place of $1 - (\lambda_i^s)^2$, which is above the index. Which of the two prevails depends on the general loadings, the group loadings, and the number of items outside the subscale together, not on the sign and similarity of the group loadings alone. In a population of nine items in three subscales of three, with every general loading .7 and every group loading .3 (an ECV of .84), the index is 0.229 for each group factor and its maximal reliability 0.279, while for the general factor they are 0.896 and 0.865; with general and group loadings of .5, as in the first example below, the index is the larger on every row (0.500 against 0.471 for a group factor in the population); and with two subscales of six items, general loadings .8, and group loadings .4, the index is the larger for a group factor (0.533 against 0.468), although the general factor is stronger than in the first population.

The composite that reaches a group factor's maximal reliability is the factor's regression score from all of the items, and it weights the items of the other subscales negatively: in the first population the optimal weights are proportional to 1 for each of the subscale's three items and -0.405 for each of the other six. No score of the subscale's own items reaches that reliability. The best weighting of them alone reaches $\lambda_s' R_{ss}^{-1} \lambda_s$, with λ_s their group loadings and R_{ss} their correlations, which is omega_HS when the items have equal loadings on both factors, and then below the index as well: 0.125 in the first population, against an index of 0.229 and a maximal reliability of 0.279. So a maximal reliability above the index on a group factor's row, which is common when the items outside the subscale measure a strong general factor well (a profile of high ECV, in which the subscale rows are commonly consulted), describes a factor score computed from all of the items, not the subscale score; how much of a subscale score's variance is due to its group factor is omega_HS. Read the column as the published index, comparable with values reported beside it, and the attribute as the reliability that optimal weights of all of the items could reach in the population. The sample

values of both overstate their population values on average, the attribute most of all for a weakly determined group factor, and a composite weighted with sample estimates reaches less, as Aguirre-Urreta, Rönkkö, and McIntosh (2019) found for H of a single factor; `ci_bifactor_indices` gives the bootstrap estimate of that bias with the intervals.

The H of a factor any of whose standardized loadings is at or beyond one in absolute value is `NA`, since the index then gives that item an error variance that is not positive.

Each denominator is the sum of the corresponding block of an item correlation matrix on the standardized metric of the loadings: all items for omega and omega_H, a group factor's items for its omega_HS. The default (`denominator = "model_implied"`) takes the correlation matrix the fitted model reproduces. For an orthogonal bifactor model without residual covariances that block sum is the total-score variance written above, the form given by Rodriguez, Reise, and Haviland (2016); when the model also specifies residual covariances among the items, the block sum carries each of them as well, since they are part of the variance of the composite the model reproduces, although not of its common variance, so the numerators leave them out. With `denominator = "observed"` the matrix is instead the item correlation matrix estimated from the data (**lavaan**'s sample statistics, which under `missing = "ml"` are the full information estimates). This is the denominator of robust omega (`reliability_omega`) carried to the bifactor coefficients: it estimates the variance of the composite consistently whether or not the bifactor model reproduces it, while the numerators remain model based. When the model reproduces the item correlations the two settings agree, and a gap between them shows that it does not reproduce the composite variance. The printed table names the denominator in a line under the table.

The indices assume uncorrelated factors: a correlation between the general factor and a group factor, or between two group factors, is common variance that belongs to neither, and the partition behind ECV, omega_H, and omega_HS has no place for it. A fit that estimates such a correlation is refused with an error naming the pair.

A fit with one group factor. A general factor beside a single group factor is not a bifactor model with subscales: its indices would divide the common variance between those two factors alone, every item would belong to the one subscale, and PUC would be 0. Such a fit is refused, as `fa_bifactor` and `ci_bifactor_indices` refuse it, with a message that names the group factor. A general factor with one further factor can be fitted with `lavaan::cfa()` and read from its own parameter estimates.

Ordered-categorical items. For a fit with ordered indicators (WLSMV or DWLS on polychoric correlations), **lavaan**'s sample statistics are the polychoric correlations, and the standardized loadings are on the metric of the continuous latent responses behind the categories. Both settings then give the reliability of the sum of those latent responses, not of the summed category scores a user computes, and categorization lowers reliability, often substantially (Green & Yang, 2009, Table 1). The printed line says which metric the denominator is on, and the `"metric"` attribute records it. For a single factor, `reliability_omega_categorical` gives the reliability of the summed categories; **DMAR** does not compute the bifactor indices on that metric.

A bifactor model that is over-parameterized for the data frequently yields an improper (Heywood) solution, a standardized loading outside $[-1, 1]$ or a negative residual variance, whose indices are not trustworthy. When that happens the indices are still returned but a warning is issued and the `"improper"` attribute is set to `TRUE`; H is `NA` for a factor with a loading at or beyond one, as described above, and the maximal reliability is `NA` for every factor, since it is not defined for an improper solution.

A fit whose optimizer did not converge (`lavaan::lavInspect(fit,"converged")` is `FALSE`, as when it reached its iteration limit) is not a solution, proper or improper: its estimates mark where the

optimizer stopped. It is refused with an error of class `dmar_not_converged`, as `ci_bifactor_indices` refuses it.

One group and one level. The standardized loadings and residual variances of a fit are those of one group at one level: each group of a multiple-group fit has its own, standardized by that group's variances, so the factors have indices in each group, and neither an unweighted mean nor a mean weighted by group size is the index of any group. A multiple-group fit is refused, as `average_variance_extracted` refuses it, with an error of class `dmar_argument_invalid` that names the groups (as `groups`); the indices of one group are those of the model fitted to that group's data alone (which drops any equality constraints across groups). A multilevel fit, whose loadings and residual variances differ by level, is refused with an error of the same class; the standardized solution of each level is in `lavaan::lavInspect(fit, "std")`.

Value

A `data.frame` (class `dmar_tbl`) with one row per factor (the general factor first, then each group factor) and columns `factor`, `ECV`, `omega`, `omega_H`, `omega_HS`, `PUC`, and `H`. Quantities that do not apply to a row are `NA` (for example omega_H and PUC on a group factor). On a group factor's row, ECV is the general factor's share of the common variance of that factor's items and omega is the reliability of their total score (see *Details*); H is the index of Rodriguez, Reise, and Haviland (2016) on every row. The `"improper"` attribute flags a Heywood solution, the `"maximal_reliability"` attribute holds the maximal reliability the fitted model implies for each factor (a vector named by factor, in the order of the rows; `NA` for an improper solution), the `"denominator"` attribute records the `denominator` setting, and the `"metric"` attribute records the metric of the items the indices describe: `"observed"` for continuous items, `"latent_response"` for ordered items, whose coefficients describe the sum of the latent responses.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Aguirre-Urreta, M. I., Rönkkö, M., & McIntosh, C. N. (2019). A cautionary note on the finite sample behavior of maximal reliability. *Psychological Methods, 24*(2), 236–252. doi:10.1037/met0000176

Gabler, S., & Raykov, T. (2017). Evaluation of maximal reliability for unidimensional measuring instruments with correlated errors. *Structural Equation Modeling, 24*(1), 104–111. doi:10.1080/10705511.2016.1159916

Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika, 74*, 155–167. doi:10.1007/s11336-00890993

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), *Structural equation modeling: Present and future* (pp. 195–216). Scientific Software International.

Reise, S. P. (2012). The rediscovery of bifactor measurement models. *Multivariate Behavioral Research, 47*(5), 667–696. doi:10.1080/00273171.2012.715555

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016). Evaluating bifactor models: Calculating and interpreting statistical indices. *Psychological Methods, 21*(2), 137–150. doi:10.1037/met0000045

See Also

fa_bifactor, which fits the bifactor model and gives these indices delta method intervals; ci_bifactor_indices for bootstrap confidence intervals for these indices and for the maximal reliability of each factor; reliability_omega, reliability_H, and factor_scores, whose regression score determinacy is the square root of the maximal reliability.

Other multivariate and latent variable methods: average_variance_extracted(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
# Nine items: one general factor and three orthogonal group factors.
set.seed(113)
n <- 600
g <- rnorm(n); grp <- list(rnorm(n), rnorm(n), rnorm(n))
X <- vapply(1:9, function(i)
  0.5 * g + 0.5 * grp[[ceiling(i / 3)]] + sqrt(0.5) * rnorm(n), numeric(n))
colnames(X) <- paste0("x", 1:9)
model <- "g  =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
          f1 =~ x1 + x2 + x3
          f2 =~ x4 + x5 + x6
          f3 =~ x7 + x8 + x9"
fit <- lavaan::cfa(model, data = as.data.frame(X),
                   orthogonal = TRUE, std.lv = TRUE)
bifactor_indices(fit)

# The same indices with the observed composite variances in the
# denominators; when the model reproduces the item correlations the
# two settings agree.
bifactor_indices(fit, denominator = "observed")
```

---

bryant_paulson *The Bryant-Paulson Generalized Studentized Range Distribution*

Description

Distribution function (pbryant_paulson), quantile/critical-value function (qbryant_paulson), and density (dbryant_paulson) for the Bryant-Paulson generalized studentized range, the sampling distribution of the studentized range of covariate-*adjusted* means in the analysis of covariance (ANCOVA) when the covariate(s) are *random*. These are the analysis-of-covariance analogs of ptukey / qtukey and supply the critical values needed for Tukey-Kramer-type simultaneous confidence intervals on (and tests of) contrasts of adjusted means.

Usage

```
pbryant_paulson(q, n_covariates, n_groups, df, lower_tail = TRUE)

qbryant_paulson(prob, n_covariates, n_groups, df, lower_tail = TRUE)

dbryant_paulson(q, n_covariates, n_groups, df)
```

Arguments

`q`	Vector of quantiles (values of the generalized studentized range statistic).
`n_covariates`	The number of random covariates, p ($p \geq 0$). With `n_covariates = 0` the distribution is exactly the ordinary studentized range, computed as in `cv_tukey_hsd` (`ptukey` and `qtukey` where those are accurate, the exact distribution where they are not).
`n_groups`	The number of groups (treatments) being compared, k ($k \geq 2$); this is the "sample size" of the range.
`df`	The error degrees of freedom of the ANCOVA model, ν. In a one-way ANCOVA with N total observations, k groups, and p covariates, $\nu = N - k - p$. `Inf` gives the limiting distribution (see Details).
`lower_tail`	Logical; if `TRUE` (default) probabilities are $P(Q \leq q)$, otherwise $P(Q > q)$.
`prob`	Vector of probabilities. For `qbryant_paulson` this is the cumulative probability (e.g., `0.95` returns the upper 5% critical value).

Details

The statistic. In a balanced ANCOVA with k groups and p random covariates, let $\hat{\theta}_i$ be the adjusted group means and $\hat{\sigma}_{y|x}$ the square root of the ANCOVA error mean square (on ν degrees of freedom). The Bryant-Paulson statistic is the studentized range of the adjusted means,

$$Q \;=\; \frac{\max_i \hat{\theta}_i - \min_i \hat{\theta}_i}{\hat{\sigma}_{y|x}\sqrt{K_1 - K_2}},$$

where $K_1 - K_2$ is the design constant that scales the variance of a single adjusted mean (for a one-way design with n per group, $K_1 - K_2 = 1/n$). The studentizer uses only this "between-only" standard error: the extra sampling variability induced by having to *estimate* the covariate adjustment from random covariates is carried by the distribution of Q itself, not by a per-comparison standard-error correction. This is what distinguishes the procedure from naively applying Tukey's method to adjusted means.

The distribution. Bryant and Paulson (1976) give the exact CDF of Q_p in their Equation (17), a single integral over a variable that combines the χ^2_ν error estimate with a random covariate-shrinkage factor δ that, by their Equations (11)–(12), has a $\mathrm{Beta}((\nu+1)/2,\, p/2)$ distribution. Carrying out the error integral with the studentized-range routine `ptukey` reduces Equation (17) to the equivalent one-dimensional form

$$P(Q_p \leq q) \;=\; \int_0^1 \mathrm{ptukey}\Big(q\sqrt{\delta};\, k,\, \nu\Big)\; f_{\mathrm{Beta}}\big(\delta;\, \tfrac{\nu+1}{2},\, \tfrac{p}{2}\big)\, d\delta,$$

which this package evaluates (the reduction is exact; see the source-code comments in 'R/bryant_paulson.R' for the one-line derivation from Bryant and Paulson's p. 634 conditioning argument). When $p = 0$ the factor δ degenerates at 1 and Q_p is exactly the ordinary studentized range (Bryant and Paulson, 1976, Sec. 1), so the code short-circuits to the studentized range distribution function. The integral is evaluated with `integrate` after the change of variables $\delta = 1 - u^2$, which removes the endpoint singularity of the Beta weight at $\delta = 1$ when $p = 1$ and makes the quadrature converge in a few subdivisions; `qbryant_paulson` inverts it with `uniroot`. Bryant and Bruvold (1980) later showed the same distribution and critical values remain valid when the covariates are *not* identically distributed across groups (their grouped-covariate model, Eq. 1.3), and added the Duncan multiple-range extension.

Accuracy at small df. The studentized range distribution function inside the integral is `ptukey` where that is accurate and is evaluated directly elsewhere, without `ptukey`, by integrating the probability integral of the range against the χ^2_ν error density. `ptukey` loses accuracy at small ν, increasingly so as k grows (at $\nu = 3$, $k = 20$ its probability error reaches $\approx 3 \times 10^{-4}$, enough to move the critical value by about 0.2, and it is larger at $\nu = 2$), and again above 10000 df; the direct evaluation is used for fewer than 9 error df with up to 6 groups, fewer than 11 with up to 10, fewer than 15 with up to 20, fewer than 21 with up to 50, fewer than 31 with up to 100, fewer than 41 with up to 250, and, with more groups, below a bound that rises by steps to 111 with more than 3000 groups; above 10000 and up to 10^8 df; and, with more than 50 groups, wherever the probability is below .75 (.80 with more than 1000 groups; `cv_tukey_hsd` gives the whole rule and how it was measured). At probabilities above .999 or below .001 the quantile also comes from the exact distribution, with or without covariates (with covariates every node of the integral over the covariate-shrinkage weight then uses the exact studentized range: through `ptukey` the quantile at .9999 with one covariate, 10 groups, and 11 df was 12.977072 instead of 12.977142), and with no covariates also wherever `qtukey` stops short of its root. The direct path costs a fraction of a second per probability.

Large df. As ν grows, δ tends to 1 and Q_p to the studentized range with infinite df, whatever the number of covariates; with `df = Inf` the functions return that limit, computed as for `n_covariates = 0`. At large ν the weight of the integral lies within a few multiples of $\sqrt{p/\nu}$ of $u = 0$, where $\delta = 1 - u^2$, so the integral over u stops at the square root of the upper 10^{-15} quantile of the $\mathrm{Beta}(p/2,\, (\nu+1)/2)$ distribution of u^2 rather than at 1, which leaves out weight 10^{-15}. Over the whole unit interval the quadrature missed the weight from about 4×10^7 df with one covariate (5.6×10^7 with six), and the distribution function came out near 0. The weight is computed on the log scale, since with many covariates its factors overflow and underflow separately. At 10^8 df the distribution function is still below its infinite-df limit by about 10^{-8} (7×10^{-9} at 3.7 with one covariate and four groups).

Validation. The implementation reproduces Bryant and Paulson's (1976) Table 1 exactly, to the two decimal places tabled, over the whole of its range: both tail areas ($\alpha = .05$ and $\alpha = .01$), all three covariate counts ($p = 1, 2, 3$), every tabled number of groups ($k = 2, \ldots, 8, 10, 12, 16, 20$), and every tabled error degrees of freedom ($\nu = 2, \ldots, 8, 10, 12, 14, 16, 18, 20, 24, 30, 40, 60, 120$), which is 1188 critical values in all. The corners of the table are included: $q_{.01;\,1,2,2} = 19.09$, $q_{.01;\,3,20,2} = 73.01$, $q_{.01;\,3,20,3} = 33.13$, and $q_{.05;\,1,6,14} = 4.83$ (the value used in the Bryant and Bruvold, 1980 worked example). Two of the 1188 entries, $q_{.01;\,2,8,3}$ and $q_{.01;\,2,20,4}$, have exact values of 23.165013 and 19.745008, each roughly 10^{-5} above the 23.165 and 19.745 half-way points, so they round to 23.17 and 19.75; the 1976 table rounds them down, to 23.16 and 19.74. Both values were confirmed to fourteen significant figures by two independent high-order quadrature engines that share no code with the package implementation. A large-scale simulation of the Bryant and Paulson statistic confirms the computed values independently, and the Bryant and Bruvold (1980) Table 2 Duncan ranges are reproduced as well. See the package tests.

Value

Numeric vectors. `pbryant_paulson` returns cumulative (or upper-tail) probabilities, `dbryant_paulson` returns density values, and `qbryant_paulson` returns critical values (quantiles) of the generalized studentized range. Results are recycled to the length of the longest of `q`/`prob` and the parameter arguments.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bryant, J. L., & Paulson, A. S. (1976). An extension of Tukey's method of multiple comparisons to experimental designs with random concomitant variables. *Biometrika, 63*, 631–638. doi:10.1093/biomet/63.3.631

Bryant, J. L., & Bruvold, N. T. (1980). Multiple comparison procedures in the analysis of covariance. *Journal of the American Statistical Association, 75*(372), 874–880. doi:10.2307/2287175

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

See Also

`ci_c_ancova_bp` for the simultaneous confidence intervals these critical values produce; `ptukey` and `qtukey` for the ordinary (fixed-covariate or no-covariate) studentized range.

Examples

```
# Critical value from the worked example of Bryant and Bruvold (1980):
# k = 6 panels, p = 1 covariate, nu = 14 error df, alpha = .05. The quantile
# is found by inverting the distribution function with uniroot; it is 4.83,
# the entry in Table 1 of Bryant and Paulson (1976).
qbryant_paulson(0.95, n_covariates = 1, n_groups = 6, df = 14)

# The ordinary Tukey value (ignoring that the covariate is random and
# estimated) is smaller, 4.64, so it yields intervals that are too narrow:
qtukey(0.95, nmeans = 6, df = 14)

# How much too narrow: the Bryant-Paulson area beyond the Tukey value is the
# familywise error rate Tukey's method actually delivers in this design.
# With no covariate the two distributions coincide, so the area is .0499,
# the nominal .05 up to the rounding of 4.64 itself. Each additional random
# covariate carries more estimation uncertainty, stretches the distribution
# to the right, and pushes the rate up, to .063, .076, and .091.
pbryant_paulson(4.64, n_covariates = 0:3, n_groups = 6, df = 14,
                lower_tail = FALSE)

# The p = 0 entry above is exactly the ordinary studentized range:
ptukey(4.64, nmeans = 6, df = 14, lower.tail = FALSE)
```

cfa_1 *One Factor Confirmatory Factor Analysis Model*

Description

Fits a single factor (congeneric by default) confirmatory factor analysis model to raw item data or a sample covariance matrix. This is the one factor special case of `fa_confirmatory`: the function is a convenience wrapper that only requires the data (and, when the data hold more than the items, a vector of item names), builds the one factor specification, and forwards everything else to `fa_confirmatory()`. The factor is named `f1`, so the rows of the returned table are `lambda_f1_1`, `lambda_f1_2`, ..., `psi_f1_1`, ..., and `omega_f1`, exactly as a one factor `fa_confirmatory()` call would report them (the `syntax` column names the item behind each number).

Usage

```
cfa_1(
  data = NULL,
  items = NULL,
  S = NULL,
  N = NULL,
  equal_loading = FALSE,
  equal_error = FALSE,
  estimator = "ML",
  missing = "fiml",
  se = NULL,
  conf_level = 0.95,
  output = c("verbose", "measurement", "summary", "standardized", "fit"),
  ...,
  wald_df = Inf,
  rmsea_conf_level = 0.9
)
```

Arguments

`data`	A raw data matrix or data frame, rows are respondents and columns include the items. A matrix without column names is given the names `y1`, `y2`, ... Supply exactly one of `data` or `S`.
`items`	Character vector naming the items of the factor (three or more; two are accepted with `equal_loading = TRUE`, the just identified tau-equivalent case). The default `NULL` uses every column of `data` (or of `S`), so a data set that holds only the items needs no `items` at all. An item that is not a column of `data` or an item of `S` is refused with an error of class `dmar_items_invalid` (see `dmar_metadata`).
`S`	A symmetric covariance matrix of the items; `N` is then required. Dimnames are optional here: a matrix without them is given the item names `y1`, `y2`, ... Supply exactly one of `data` or `S`.
`N`	Total sample size. Required with `S`; ignored (inferred from the rows) with `data`.

`equal_loading` Logical, or a named logical vector with one element per factor. `TRUE` constrains the loadings within a factor to a single value (across factors nothing is equated). Defaults to `FALSE`.

`equal_error` Logical, or a named logical vector with one element per factor. `TRUE` constrains the error variances within a factor to a single value. Defaults to `FALSE`.

`estimator` Character; estimator passed to **lavaan**. Must be one of `"ML"` (default; maximum likelihood, fully efficient under multivariate normality), `"MLR"` (robust maximum likelihood, raw data required: maximum likelihood estimates with the test statistic corrected for nonnormality, the scaled statistic of Satorra and Bentler, 1994, in the form Yuan and Bentler, 2000, gave it for incomplete data; its matching Huber-White sandwich standard errors are used when `se` is omitted), `"WLS"` (the asymptotic distribution free estimator of Browne, 1984; raw data and a large sample required), `"WLSMV"` (diagonally weighted least squares with mean- and variance-adjusted test statistic; Muthén, 1984; Muthén, du Toit, & Spisic, 1997; the standard choice for ordered categorical items, and what `ordered` switches to), or `"GLS"` (generalized least squares; Browne, 1974). With a robust estimator the reported fit indices are the robust versions.

`missing` Character; missing-data handling when raw data are supplied: `"fiml"` (default; full information maximum likelihood, which uses every case with an observed item; **lavaan**'s name `"ml"` is accepted), `"listwise"` (listwise deletion), `"pairwise"` (passed to **lavaan**, for ordered items), `"two_stage"` (two-stage maximum likelihood with normal theory standard errors and test), or `"robust_two_stage"` (two-stage maximum likelihood with standard errors and test robust to nonnormality); lavaan's spellings `"two.stage"` and `"robust.two.stage"` are accepted for the last two. See *Details* for two-stage estimation. Ignored with `S`, except that a two-stage request with `S` is an error, since a covariance matrix has no incomplete cases. With `ordered` items, `"ml"`/`"fiml"` and the two-stage options are not available; use `"pairwise"` or `"listwise"`. FIML needs a maximum likelihood estimator of continuous items with normal theory or Huber-White standard errors. When the default meets ordered items, a least squares estimator (`"WLS"`, `"WLSMV"`, `"GLS"`), `se = "robust.sem"`, or a Satorra-Bentler kind of test passed through `...`, the items are fitted as they stand if no item value is missing (listwise deletion then deletes nothing), and otherwise the call stops and names the option to choose, rather than deleting cases unasked. With complete data FIML and listwise deletion give the same estimates and the same chi square, and for `estimator = "ML"` the same fit indices; the standard errors differ slightly, since FIML takes them from the observed information, as **lavaan** does and as Savalei (2010) recommends with incomplete data, and listwise deletion from the expected information.

`se` Standard error method passed to **lavaan**. The default, `NULL`, follows the estimator: model-based `"standard"` errors for ML, GLS and WLS; Huber-White sandwich `"robust.huber.white"` errors for MLR; and `"robust.sem"` errors for WLSMV. Explicit compatible choices are retained. The ML and MLR point estimates use the same likelihood; their standard errors and tests use different reference assumptions. This difference does not depend on whether optimization is iterative or an equivalent solution is computed analytically. `"robust.sem"` requires complete data and is refused with FIML; use `"robust.huber.white"` to retain incomplete respondents. No standard-error request silently changes the

missing-data treatment. "none" requests point estimates only. Under two-stage estimation the matching corrected two-stage errors are used; se must be NULL, "standard", or "none". MLR and Huber-White errors require raw data. Robust SEM errors from covariance input require ML and a matching NACOV in ....

conf_level — Confidence level for the parameter confidence intervals, including the delta method intervals for omega, AVE, and *H*. Defaults to 0.95. It does not reach the RMSEA interval, whose level is rmsea_conf_level.

output — Format of the returned object:

"verbose" (default) Parameter estimates with confidence intervals, the per-factor defined parameters (loading_sum, error_sum, omega, ave, H), and fit information.

"measurement" The measurement-property rows only: per factor omega, ave, and H (with delta method standard errors and confidence intervals), the latent correlation phi for every factor pair (with its confidence interval), and, for raw data, the heterotrait-monotrait ratio htmt for every factor pair via htmt.

"summary" The raw summary() output from **lavaan** (not a data frame).

"standardized" The standardized parameter estimates from lavaan::standardizedSolution(), whose defined rows describe the standardized items: omega there is the reliability of the unit-weighted sum of the standardized items, while ave and H do not depend on the metric of the items and equal the rows of the other tables. For a factor with equal_loading or equal_error these rows are computed from each item's own standardized loading and error variance (see *The standardized solution* under *Details* of fa_confirmatory). Under se = "none" the se, z, pvalue, ci.lower, and ci.upper columns are NA.

"fit" The raw **lavaan** fit object. Escape hatch for direct lavaan access, including likelihood ratio tests between two fa_confirmatory() fits via lavaan::lavTestLRT().

... — Additional arguments forwarded to fa_confirmatory and, through it, to lavaan.

wald_df — The reference distribution of the Wald intervals and *p*-values of the parameter rows: Inf (the default), the normal distribution, which leaves **lavaan**'s intervals and *p*-values as they are; a single positive number, the degrees of freedom of a *t* distribution for every parameter row, the intervals and *p*-values recomputed from the same estimates and standard errors; or "missing_information", von Hippel's (2016) degrees of freedom $\max\{3, N(1-\gamma_j)-1\}$ for each loading, from its fraction of missing information γ_j ($N-1$ with complete data), with every other row left on the normal reference. A number applies to the tables of output = "verbose", "measurement", and "standardized", the keyword to that of "verbose" only (the "measurement" table has no loading), and either needs standard errors other than "none" and "bootstrap"; the keyword also needs the fits its calibration covers, maximum likelihood (estimator = "ML" or "MLR") of continuous items by FIML or with complete data, with normal theory or Huber-White standard errors. Anything else is an error that says why. It follows ..., so it must be named in full. See *Details*.

rmsea_conf_level — Confidence level of the RMSEA interval of the fit rows, which the rmsea_ci_level row records. Defaults to 0.90, the conventional level, whose limits decide the

tests of close and not close fit at .05 (see sem_fit_measures); rmsea_conf_level = 0.95 gives a 95 percent interval. It sets the level in the table DMAR returns and in its glance(); output = "summary" returns the **lavaan** summary, which shows lavaan's 90 percent interval whatever this argument is.

Details

The model is identified by fixing the factor variance to 1 and estimating every loading. equal_loading and equal_error impose the classical measurement structures on the single factor, and the header of the printed table names the structure implied by the constraints. For the composite reliability coefficient with the observed total variance in the denominator, use reliability_omega with denominator = "observed"; for the model implied omega, the omega_f1 row of this function's output and reliability_omega agree.

Ordered categorical items are not supported here; use fa_confirmatory, whose ordered argument fits WLSMV with the theta parameterization and reports categorical omega on the sum score metric, or reliability_omega_categorical.

A fit whose likelihood has no maximum (an error variance that runs off below zero while the likelihood is flat or still rising, as it can on three items whose correlations have a negative product) stops with an error of class dmar_no_maximum that prints the **lavaan** code of the fit, as fa_confirmatory describes; there, bounds = "pos.var" (which ... also passes on from here) or a different model are the choices.

Value

The value of the corresponding fa_confirmatory call: a data.frame (classes dmar_fa_confirmatory, dmar_tbl) with one row per parameter (estimate, se, test_statistic, p_value, ci_lower, ci_upper) followed by the fit rows, or the alternative shapes selected by output (see ?fa_confirmatory).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Browne, M. W. (1974). Generalized least squares estimators in the analysis of covariance structures. *South African Statistical Journal, 8*, 1–24.

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*(1), 62–83. doi:10.1111/j.20448317.1984.tb00789.x

Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika, 74*(1), 155–167. doi:10.1007/s11336008-90993

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*(1), 69–92. doi:10.1037/a0040086

McDonald, R. P. (1999). *Test theory: A unified treatment*. Lawrence Erlbaum Associates.

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika, 49*(1), 115–132.

Muthén, B., du Toit, S. H. C., & Spisic, D. (1997). *Robust inference using weighted least squares and quadratic estimating equations in latent variable modeling with categorical and continuous outcomes*. Unpublished technical report.

Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis. In A. von Eye & C. C. Clogg (Eds.), *Latent variables analysis: Applications for developmental research* (pp. 399–419). Thousand Oaks, CA: Sage.

Savalei, V. (2010). Expected versus observed information in SEM with incomplete normal and nonnormal data. *Psychological Methods, 15*(4), 352–367. doi:10.1037/a0020143

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*, 165–200.

See Also

`fa_confirmatory` for the general function this wraps (factor analysis with any number of factors, intercept constraints, ordered categorical items, and the measurement output); `cfa_2` for the two factor wrapper; `reliability_omega` for coefficient omega with confidence intervals.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
set.seed(113)
f <- rnorm(200)
loadings <- c(0.5, 0.6, 0.65, 0.7, 0.8)
X <- sapply(loadings, function(l) l * f + rnorm(200, sd = sqrt(1 - l^2)))
colnames(X) <- paste0("y", 1:5)

# All columns are items, so the data are all that is needed.
cfa_1(X)

# Equal loadings (essentially tau-equivalent), named in the header.
cfa_1(X, equal_loading = TRUE)

# From a covariance matrix and sample size, as a paper reports them,
# here with a subset of its variables named in \'items\' (which selects
# columns of raw data in the same way).
cfa_1(S = cov(X), N = 200, items = c("y1", "y2", "y3", "y4"))
```

cfa_2 *Two Factor Confirmatory Factor Analysis Model*

Description

Fits a two factor confirmatory factor analysis model to raw item data or a sample covariance matrix. This is the two factor special case of `fa_confirmatory`: the function is a convenience wrapper that only requires the items of each factor, builds the two factor specification, and forwards everything else to `fa_confirmatory()`. The factors are named `f1` and `f2`, so the rows of the returned table are `lambda_f1_1`, `lambda_f2_1`, `phi_f1_f2` (the factor correlation), `omega_f1`, `omega_f2`, and so on, exactly as a two factor `fa_confirmatory()` call would report them (the `syntax` column names the item behind each number). To name the factors substantively, or for three or more factors, call `fa_confirmatory` directly.

Usage

```
cfa_2(
  data = NULL,
  factor_1,
  factor_2,
  S = NULL,
  N = NULL,
  equal_loading = FALSE,
  equal_error = FALSE,
  correlated_factors = TRUE,
  estimator = "ML",
  missing = "fiml",
  se = NULL,
  conf_level = 0.95,
  output = c("verbose", "measurement", "summary", "standardized", "fit"),
  ...,
  wald_df = Inf,
  rmsea_conf_level = 0.9
)
```

Arguments

`data`	A raw data matrix or data frame, rows are respondents and columns include the items named in `factor_1` and `factor_2`. Supply exactly one of `data` or `S`.
`factor_1`	Character vector naming the items of the first factor (two or more).
`factor_2`	Character vector naming the items of the second factor (two or more). No item may appear in both factors. An item of either factor that is not a column of `data` or an item of `S` is refused with an error of class `dmar_items_invalid` (see `dmar_metadata`).
`S`	A symmetric covariance matrix of the items, with dimnames naming the items; `N` is then required. Supply exactly one of `data` or `S`.

`N` Total sample size. Required with `S`; ignored (inferred from the rows) with `data`.

`equal_loading` Logical, or a named logical vector with one element per factor. `TRUE` constrains the loadings within a factor to a single value (across factors nothing is equated). Defaults to `FALSE`.

`equal_error` Logical, or a named logical vector with one element per factor. `TRUE` constrains the error variances within a factor to a single value. Defaults to `FALSE`.

`correlated_factors`
Logical. If `TRUE` (default) the factors covary freely; because each factor variance is fixed to 1, the `phi` terms for factor pairs are the latent correlations. If `FALSE` the factor covariances are fixed to zero.

`estimator` Character; estimator passed to **lavaan**. Must be one of `"ML"` (default; maximum likelihood, fully efficient under multivariate normality), `"MLR"` (robust maximum likelihood, raw data required: maximum likelihood estimates with the test statistic corrected for nonnormality, the scaled statistic of Satorra and Bentler, 1994, in the form Yuan and Bentler, 2000, gave it for incomplete data; its matching Huber-White sandwich standard errors are used when `se` is omitted), `"WLS"` (the asymptotic distribution free estimator of Browne, 1984; raw data and a large sample required), `"WLSMV"` (diagonally weighted least squares with mean- and variance-adjusted test statistic; Muthén, 1984; Muthén, du Toit, & Spisic, 1997; the standard choice for ordered categorical items, and what `ordered` switches to), or `"GLS"` (generalized least squares; Browne, 1974). With a robust estimator the reported fit indices are the robust versions.

`missing` Character; missing-data handling when raw data are supplied: `"fiml"` (default; full information maximum likelihood, which uses every case with an observed item; **lavaan**'s name `"ml"` is accepted), `"listwise"` (listwise deletion), `"pairwise"` (passed to **lavaan**, for ordered items), `"two_stage"` (two-stage maximum likelihood with normal theory standard errors and test), or `"robust_two_stage"` (two-stage maximum likelihood with standard errors and test robust to nonnormality); lavaan's spellings `"two.stage"` and `"robust.two.stage"` are accepted for the last two. See *Details* for two-stage estimation. Ignored with `S`, except that a two-stage request with `S` is an error, since a covariance matrix has no incomplete cases. With `ordered` items, `"ml"`/`"fiml"` and the two-stage options are not available; use `"pairwise"` or `"listwise"`. FIML needs a maximum likelihood estimator of continuous items with normal theory or Huber-White standard errors. When the default meets ordered items, a least squares estimator (`"WLS"`, `"WLSMV"`, `"GLS"`), `se = "robust.sem"`, or a Satorra-Bentler kind of test passed through `...`, the items are fitted as they stand if no item value is missing (listwise deletion then deletes nothing), and otherwise the call stops and names the option to choose, rather than deleting cases unasked. With complete data FIML and listwise deletion give the same estimates and the same chi square, and for `estimator = "ML"` the same fit indices; the standard errors differ slightly, since FIML takes them from the observed information, as **lavaan** does and as Savalei (2010) recommends with incomplete data, and listwise deletion from the expected information.

`se` Standard error method passed to **lavaan**. The default, `NULL`, follows the estimator: model-based `"standard"` errors for ML, GLS and WLS; Huber-White sandwich `"robust.huber.white"` errors for MLR; and `"robust.sem"` errors

for WLSMV. Explicit compatible choices are retained. The ML and MLR point estimates use the same likelihood; their standard errors and tests use different reference assumptions. This difference does not depend on whether optimization is iterative or an equivalent solution is computed analytically. `"robust.sem"` requires complete data and is refused with FIML; use `"robust.huber.white"` to retain incomplete respondents. No standard-error request silently changes the missing-data treatment. `"none"` requests point estimates only. Under two-stage estimation the matching corrected two-stage errors are used; `se` must be `NULL`, `"standard"`, or `"none"`. MLR and Huber-White errors require raw data. Robust SEM errors from covariance input require ML and a matching `NACOV` in `...`.

`conf_level` Confidence level for the parameter confidence intervals, including the delta method intervals for omega, AVE, and *H*. Defaults to 0.95. It does not reach the RMSEA interval, whose level is `rmsea_conf_level`.

`output` Format of the returned object:

`"verbose"` (default) Parameter estimates with confidence intervals, the per-factor defined parameters (`loading_sum`, `error_sum`, `omega`, `ave`, `H`), and fit information.

`"measurement"` The measurement-property rows only: per factor `omega`, `ave`, and `H` (with delta method standard errors and confidence intervals), the latent correlation `phi` for every factor pair (with its confidence interval), and, for raw data, the heterotrait-monotrait ratio `htmt` for every factor pair via `htmt`.

`"summary"` The raw `summary()` output from **lavaan** (not a data frame).

`"standardized"` The standardized parameter estimates from `lavaan::standardizedSolution()`, whose defined rows describe the standardized items: `omega` there is the reliability of the unit-weighted sum of the standardized items, while `ave` and `H` do not depend on the metric of the items and equal the rows of the other tables. For a factor with `equal_loading` or `equal_error` these rows are computed from each item's own standardized loading and error variance (see *The standardized solution* under *Details* of `fa_confirmatory`). Under `se = "none"` the `se`, `z`, `pvalue`, `ci.lower`, and `ci.upper` columns are `NA`.

`"fit"` The raw **lavaan** fit object. Escape hatch for direct lavaan access, including likelihood ratio tests between two `fa_confirmatory()` fits via `lavaan::lavTestLRT()`.

`...` Additional arguments forwarded to `fa_confirmatory` and, through it, to `lavaan` (for example `ordered`, `equal_intercept`, or `M`).

`wald_df` The reference distribution of the Wald intervals and *p*-values of the parameter rows: `Inf` (the default), the normal distribution, which leaves **lavaan**'s intervals and *p*-values as they are; a single positive number, the degrees of freedom of a *t* distribution for every parameter row, the intervals and *p*-values recomputed from the same estimates and standard errors; or `"missing_information"`, von Hippel's (2016) degrees of freedom $\max\{3, N(1-\gamma_j)-1\}$ for each loading, from its fraction of missing information γ_j ($N-1$ with complete data), with every other row left on the normal reference. A number applies to the tables of `output = "verbose"`, `"measurement"`, and `"standardized"`, the keyword to that of `"verbose"` only (the `"measurement"` table has no loading), and either needs standard errors other than `"none"` and `"bootstrap"`; the keyword also

needs the fits its calibration covers, maximum likelihood (`estimator = "ML"` or `"MLR"`) of continuous items by FIML or with complete data, with normal theory or Huber-White standard errors. Anything else is an error that says why. It follows `...`, so it must be named in full. See *Details*.

`rmsea_conf_level`
: Confidence level of the RMSEA interval of the fit rows, which the `rmsea_ci_level` row records. Defaults to 0.90, the conventional level, whose limits decide the tests of close and not close fit at .05 (see `sem_fit_measures`); `rmsea_conf_level = 0.95` gives a 95 percent interval. It sets the level in the table DMAR returns and in its `glance()`; `output = "summary"` returns the **lavaan** summary, which shows lavaan's 90 percent interval whatever this argument is.

Details

Each factor is identified by fixing its variance to 1 and estimating every loading; each item loads on exactly one factor (simple structure). The factor correlation is estimated by default and `correlated_factors = FALSE` fixes it to zero. Per-factor constraint vectors use the factor names, for example `equal_loading = c(f1 = TRUE, f2 = FALSE)`. A fit whose likelihood has no maximum (an error variance that runs off below zero while the likelihood is flat or still rising) stops with an error of class `dmar_no_maximum` that prints the **lavaan** code of the fit, as `fa_confirmatory` describes; there, `bounds = "pos.var"` (which `...` also passes on from here) or a different model are the choices.

Value

The value of the corresponding `fa_confirmatory` call: a `data.frame` (classes `dmar_fa_confirmatory`, `dmar_tbl`) with one row per parameter (`estimate`, `se`, `test_statistic`, `p_value`, `ci_lower`, `ci_upper`) followed by the fit rows, or the alternative shapes selected by `output` (see `?fa_confirmatory`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Browne, M. W. (1974). Generalized least squares estimators in the analysis of covariance structures. *South African Statistical Journal, 8*, 1–24.

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*(1), 62–83. doi:10.1111/j.20448317.1984.tb00789.x

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika, 49*(1), 115–132.

Muthén, B., du Toit, S. H. C., & Spisic, D. (1997). *Robust inference using weighted least squares and quadratic estimating equations in latent variable modeling with categorical and continuous outcomes*. Unpublished technical report.

Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis. In A. von Eye & C. C. Clogg (Eds.), *Latent variables analysis: Applications for developmental research* (pp. 399–419). Thousand Oaks, CA: Sage.

Savalei, V. (2010). Expected versus observed information in SEM with incomplete normal and nonnormal data. *Psychological Methods, 15*(4), 352–367. doi:10.1037/a0020143

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*, 165–200.

See Also

`fa_confirmatory` for the general function this wraps; `cfa_1` for the one factor wrapper; `htmt` and `average_variance_extracted` for the discriminant and convergent validity summaries the `output = "measurement"` table reports alongside omega.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
data(holzinger_swineford)

# Two factors, each named by its items.
cfa_2(holzinger_swineford,
      factor_1 = c("t6_paragraph_comprehension", "t7_sentence",
                   "t9_word_meaning"),
      factor_2 = c("t20_deduction", "t22_problem_reasoning",
                   "t23_series_completion"))

# The measurement properties: omega, ave, and H per factor, the
# factor correlation, and the htmt ratio.
cfa_2(holzinger_swineford,
      factor_1 = c("t6_paragraph_comprehension", "t7_sentence",
                   "t9_word_meaning"),
      factor_2 = c("t20_deduction", "t22_problem_reasoning",
                   "t23_series_completion"),
      output = "measurement")
```

`ci_bifactor_indices` *Bootstrap Confidence Intervals for the Bifactor Indices*

Description

Forms nonparametric bootstrap confidence intervals for the indices bifactor_indices reports for a fitted bifactor model: the explained common variance (ECV), coefficient omega, omega hierarchical (omega_H) for the general factor, omega hierarchical subscale (omega_HS) for each group factor, the index Rodriguez, Reise, and Haviland (2016) call coefficient H, the closed form of Hancock and Mueller (2001) applied to one factor's loadings, which counts the other factors' common variance as uncorrelated error, and the maximal reliability of each factor under the fitted model, which the index is not (see bifactor_indices). Each replication resamples the cases the fit used, refits the same model with the same estimator, and recomputes every index, so the intervals rest on the same replications (those of the maximal reliability, under `improper = "keep"`, on the proper ones among them). Replications whose solution is improper are set aside by default and counted, and the table says how many.

Usage

```
ci_bifactor_indices(
  fit,
  general = NULL,
  denominator = c("model_implied", "observed"),
  ci_method = c("percentile", "bca"),
  conf_level = 0.95,
  B = 1000L,
  seed = NULL,
  ncpus = 1L,
  improper = c("drop", "keep"),
  progress = NULL
)

## S3 method for class 'dmar_ci_bifactor'
tidy(x, ...)

## S3 method for class 'dmar_ci_bifactor'
glance(x, ...)
```

Arguments

`fit`	A fitted bifactor **lavaan** model to raw data, as for bifactor_indices: one general factor on all items plus two or more orthogonal group factors, each item on one group factor at most (an item on none is on the general factor alone). A fit that `bifactor_indices()` refuses is refused here, and so are a multiple-group fit, a fit with a cluster variable or sampling weights, a fit to a covariance matrix, and a fit that did not converge (see *Details*).
`general`	Optional name of the general factor, as for bifactor_indices; by default the factor that loads on every item.
`denominator`	How the total-score variance in the denominators of omega, omega_H, and omega_HS is estimated, as for bifactor_indices: `"model_implied"` (the default) or `"observed"`. Each replication uses the same setting.

`ci_method` The bootstrap interval: `"percentile"` (the default) or `"bca"`, the bias-corrected and accelerated interval. See *Details*.

`conf_level` Confidence level of the intervals. Defaults to `0.95`.

`B` Number of bootstrap replications, a single whole number. Defaults to `1000`, the package's count for a bootstrap whose every replication refits a model; see *Details* for the cost. The fewest accepted at the default `conf_level` is 400 for `"percentile"` (10 replications beyond each limit; 200 at 0.90, and 2000 at 0.99, which the default does not meet) and 2000 for `"bca"` (50 beyond each nominal limit), which the default does not meet either: pass `B = 2000` or more with `ci_method = "bca"`. A smaller `B` is refused before anything is computed (class `dmar_count_invalid`). The count is held to the number requested: replications set aside afterward are counted and reported in a warning of class `dmar_bootstrap_replications_dropp` and an index for which fewer than half of the `B` replications return a value has its interval withheld (`NA`) under a warning of class `dmar_bootstrap_insufficient`. Where refits can fail, request more than the minimum. See `dmar_bootstrap`, which also describes `options(dmar.bootstrap_exploratory = TRUE)` for a first look with a smaller `B`.

`seed` Random number seed for the bootstrap. The default `NULL` uses the current state of the random number generator; a supplied seed is set internally and the caller's state is restored on exit.

`ncpus` The number of processes over which the replications are spread. The default `1` runs them in this R session; a larger number uses forked workers (**boot**'s `parallel = "multicore"`) or, on Windows, a socket cluster (`"snow"`), whose workers load the installed **DMAR**. **boot** draws every resample before any replication runs, so a given `seed` returns the same intervals, counts, and replicates whatever `ncpus` is. A session that forks workers and also starts background processes through **callr** or **processx** needs the environment variable `PROCESSX_NOTIFY_OLD_SIGCHLD` set before **processx** is loaded (see `sem_exact_fit_bootstrap`).

`improper` How a replication whose solution is improper (a standardized loading beyond one or a negative residual variance, a Heywood case) is treated: `"drop"` (the default) sets it aside, and `"keep"` uses its indices when every one of them but the maximal reliability, which is not defined for an improper solution, is a proportion. Either way such replications are counted. With `"drop"` an improper point estimate stops the call. See *Details*.

`progress` `NULL` (the default), or a function of two arguments that the bootstrap calls as it runs, so that an application can show how far a long bootstrap has gone and let its user stop it (a Shiny app, for example). It is called as `progress(done, total)`, with the number of replications attempted so far and `B`: after each replication when `ncpus = 1`, and with more processes after each block of replications they complete together, about a fiftieth of `B` (`ncpus * ceiling(B / (50 * ncpus))` replications, the last block possibly fewer), so that `done` advances by the block. A BCa interval takes its acceleration from a regression on the resampling frequencies when at least as many replications return a value as there are cases, and from the jackknife, one leave-one-out refit per case, when fewer do or when that regression fails or gives influence values that are not all finite; the jackknife runs at most once in a call, and not at all when no interval is formed or every returned replicate has the same value. While it runs, the callback is

called again after each refit (with more processes, after each block of `ncpus * ceiling(n / (50 * ncpus))` refits, `n` the number of cases resampled), with `done` and `total` both `B`, so it can be called more than `B` times, and `done` equal to `total` does not mean that the bootstrap has finished. If the callback returns `FALSE`, the bootstrap stops before the next replication or block with an error of class `dmar_bootstrap_cancelled`, whose elements `done` and `total` say how far it got (see `dmar_metadata`); any other value is ignored. An error inside the callback stops the call with an error of class `dmar_progress_error`, whose message names the callback and which carries `done`, `total`, and, as `parent`, the callback's own condition with its class intact, so that an application can tell a failing callback from a failing analysis. The callback runs outside the random number stream, so a given `seed` returns the same result with or without it, and a callback that draws random numbers leaves the caller's stream as it found it. With `NULL` nothing is called. Ignored unless a bootstrap method is selected.

`x` A `dmar_ci_bifactor` table returned by `ci_bifactor_indices()`.

`...` Unused.

Details

Why intervals. The bifactor indices are estimates, and the decisions they are read for (whether a total score is close enough to unidimensional to report alone, whether a subscale carries reliable variance beyond the general factor) are decisions about the population values. A confidence interval says how precisely the sample has located each of them. In the bifactor model of 15 Holzinger and Swineford tests described under *Improper replications* below, omega_H is 0.577 with a 95 percent interval from 0.468 to 0.672 (`B = 1000`, `seed = 113`): 301 cases locate the share of total score variance due to the general factor only to within an interval about 0.2 wide. `bifactor_indices()` reports the point estimates alone. The intervals are a separate function, rather than an argument of `bifactor_indices()`, because they need columns that its one-row-per-factor table cannot gain without its shape depending on an argument.

What is resampled. The cases are the rows of data **lavaan** stored with `fit`, which are the rows the fit used: under listwise deletion (the default, and **lavaan**'s default for ordered items) the complete rows, and under `missing = "ml"` (full information maximum likelihood) every row with at least one observed item, each with its own pattern of observed and missing values, so that a resample has the missing data the sample has and each replication is again a full information fit. A row with nothing observed, which **lavaan** keeps in its stored data but leaves out of the sample, is left out here too, and the number of rows resampled is checked against the number of cases **lavaan** reports. Rows are drawn with replacement, `B` times (Efron & Tibshirani, 1993).

How each replication is fit. Each resample is fit with the parameter table and the options of `fit`: the same model, the same estimator, the same missing data treatment, the same ordered items, and default starting values, as for `fit` itself. Standard errors and test statistics are not computed in a replication, since no index uses them; robust maximum likelihood (MLR) and WLSMV have the estimates of maximum likelihood and diagonally weighted least squares and differ from them only in those, so a replication is the cheaper fit with the same estimates. Before any resample is drawn, the model is refit this way to the rows of `fit` itself, and the call stops unless that refit reproduces the point estimates, so the replications are refits of the model the point estimates describe. The indices of each replication are `bifactor_indices()` of its refit, with the general factor named, so every index is computed as the point estimate is.

Improper replications. A bifactor model is prone to improper solutions, a standardized loading past one or a residual variance below zero, which can come from sampling error around a small residual variance, from a model that does not describe the items, or from a group factor the data barely separate from the general factor, and a resample can give such a solution where the sample does not. A replication is improper when `bifactor_indices()` flags its solution or **lavaan**'s own check after estimation fails, and the point estimate is judged the same way. For a fit with auxiliary variables (**lavaan**'s `aux` under full information maximum likelihood) that check is made on the model's own parameters, as `sem_fit_measures` explains: **lavaan**'s own check also tests the residual covariances of the auxiliary variables, which are not positive definite once one of them is related to the general factor strongly enough, however proper the bifactor solution. A bifactor model of 15 of the Holzinger and Swineford tests (`holzinger_swineford`: a general factor on all of them, with spatial, verbal, and memory group factors on tests 1–4, 5–9, and 14–19) gave an improper solution in 212 of 1,000 resamples (`seed = 113`) and did not converge in 17 more. The omega of the improper solutions ranged from 0.84 to 26.5, 132 of them above one; kept, all of them put the upper limit of the 95 percent interval for omega at 13.7 and the lower limit for ECV at 0.004, where the proper replications give [0.855, 0.902] and [0.309, 0.509]. Such values are not sampling variability around a proper structure: a solution in which a residual variance has run away, or in which the general factor has all but vanished (an ECV near zero), is not a perturbation of the sample's solution but a different one, and its indices do not estimate the quantities the point estimate does. The default, `improper = "drop"`, therefore sets them aside. This differs from the default of the reliability family (`reliability_omega`), where an improper single-factor replication that still yields a coefficient is typically a loading that ran just past one, and keeping it preserves the upper tail of the bootstrap distribution. Setting replications aside conditions the intervals on admissible solutions: they describe the sampling distribution of the indices among resamples that the model fits properly. Yuan and Hayashi (2003, Section 6) argue that a replication that fails because its resample lies far from the model is informative and should be counted rather than discarded; for a test statistic they count it as an extreme value, but an index has no side on which such a replication belongs, so here it is counted and reported rather than placed. Every replication set aside, for any reason, is counted in the `bootstrap` attribute, named in a warning, and stated under the printed table, and when the replications set aside (or the improper ones kept) reach the tail area of an interval, $(1-C)/2$ of `B` for a confidence level C, the warning and the printed line add a caution that the limits depend on how they were treated. When fewer than half of the `B` replications, or fewer than the number the interval method and the level require (400 for the percentile interval and 2000 for BCa at the 95 percent level; see `dmar_bootstrap`), return a value for an index, its interval is withheld, and the warning (class `dmar_bootstrap_insufficient`) and the printed line say so, as in `reliability_omega`. A run at exactly the required `B` therefore loses its intervals to a single replication set aside; the default `B = 1000` leaves room for them at the 95 percent level. With `improper = "keep"` an improper replication is used when every index it gives but the maximal reliability, which is not defined for an improper solution (see *The maximal reliability* below), is a proportion; one whose solution gives an index outside [0, 1] (an omega above one belongs to a composite whose implied error variance is negative) or an H that is not defined (a standardized loading at or past one leaves its item no error variance in H's formula) yields no coefficient and is set aside under either setting, as in the reliability family. So is a replication whose likelihood has no maximum, a residual variance running off below zero while the likelihood is flat or still rising (see `fa_bifactor`): it is counted among the replications that raised an error, under a message that names the item, and the run goes on (of the 22 improper replications in 100 of the 15-test model above, `seed = 113`, 8 were of this kind), while a refit of the data stored with `fit` that has no maximum stops the call with an error of class `dmar_no_maximum`. Keeping the admissible indices of improper replications can change their inter-

val limits; the maximal reliability uses only proper replications under either setting. With "drop" an improper point estimate stops the call, since an interval built from proper replications would not describe it; with "keep" it is reported with a warning, unless one of its indices other than the maximal reliability is not defined or lies outside [0, 1]; its maximal reliability, which is not defined, is reported as NA.

The maximal reliability. The rows maximal_reliability give each factor's maximal reliability under the fitted model, $\lambda_f' R^{-1} \lambda_f$ with R the item correlations the model reproduces (the "maximal_reliability" attribute of bifactor_indices()): the reliability the optimally weighted items reach as a measure of the factor, and the square of the determinacy of its regression factor score. Like every other index, it is read from bifactor_indices() on each refit, so its interval costs nothing beyond the replications the other indices need. It is a reliability estimate in its own right, as much in need of an interval as omega (Raykov & Penev, 2006, give one for a composite of congeneric measures), and it is the quantity to read beside the index H, which can lie above or below it. It is not defined for an improper solution. With improper = "drop" every replication used is proper, and its intervals rest on the same replications as the others; with "keep" the improper replications kept for the other indices have no maximal reliability, so its intervals rest on the proper replications among them, which are the replications "drop" uses, and the printed table says how many. Its interval is therefore the same under either rule. An improper point estimate, reported under "keep", has no maximal reliability, and these rows then carry no estimate and no interval.

The bias of H and the maximal reliability. The sample H, the closed form the index applies to a factor's loadings, is a positively biased estimator of its population value, and a composite weighted with sample estimates is on average less reliable than the population value: Aguirre-Urreta, Rönkkö, and McIntosh (2019) found both effects pronounced in samples of about 100 and slight beyond about 1,000, for H of a single factor. The maximal reliability is the reliability of weights that are optimal for the sample's estimates and is biased upward in the same way, most of all for a group factor that the data determine weakly. Run with B = 200 in place of its 10, the example below gives a mean of the replicates that exceeds the estimate by 0.031 for the first group factor's maximal reliability (whose replicates have a standard deviation of 0.078) and by 0.019 for its H, against 0.005 for its omega and 0.003 for its omega_HS. The percentile interval takes its limits from the replicates as they fall, and so carries the bias; the BCa interval (ci_method = "bca") moves its limits to correct for the median bias of the replicates. The bias of each index with an interval, the mean of the replicates its interval uses minus the estimate, is the bias element of the bootstrap attribute (PUC, which has no interval, has none, and neither has the maximal reliability of an improper point estimate under improper = "keep"), and the printed table gives it for H and the maximal reliability. Read the maximal reliability as a ceiling that weights estimated from these data will not quite reach.

Other replications set aside. A refit that does not converge is set aside and counted as such, and so is one that raises an error; the warning quotes the most frequent error. For ordered items, a resample that draws no response in some category of an item cannot refit that item's thresholds, and is set aside as an error that says so. The intervals are formed from the replications that remain, and the effective count is the B_used attribute.

The two intervals. The percentile interval takes its limits from the empirical quantiles of the bootstrap estimates; it respects the [0, 1] range of the indices and is not forced to be symmetric. Kelley and Pornprasertmanit (2016) found it the most consistently accurate of the intervals they compared for the single-factor omega with an observed denominator (their hierarchical omega, which is not the omega_H here), which is the reason it is the default; the sources cited here do not evaluate the coverage of bootstrap intervals for the bifactor indices themselves. The bias-corrected

and accelerated (BCa) interval moves the two quantile positions to correct for median bias and for the rate at which the variance of the estimator changes with the parameter, which makes it second-order accurate where the percentile interval is first-order accurate (DiCiccio & Efron, 1996). Its acceleration is estimated as **boot** estimates it, by regression on the resampling frequencies when at least as many replications as cases were used and otherwise, or when that regression fails, by the jackknife, one leave-one-out refit per case, which is run once for all of the indices; an index whose BCa interval cannot be formed gets the percentile interval, with a warning, and the `ci_method` attribute and the printed line say which. PUC, the proportion of item pairs whose items lie on different group factors, depends only on the model's structure, so it is the same in every replication and has no interval; its row carries the point value with `NA` limits.

Cost. Each replication is a model fit, so the call costs `B` fits and, for `"bca"` with fewer replications used than cases, as many more as there are cases. A proper fit of the 15-test Holzinger and Swineford model takes a few hundredths of a second, but a solution that runs away took seconds before the optimizer stopped, so a model that gives many improper replications is slow to bootstrap as well as hard to interpret. `ncpus` spreads the replications over several processes, and `seed` makes a call reproducible. Requires the **lavaan** and **boot** packages.

What is refused. A fit that `bifactor_indices()` refuses (correlated factors, no factor that loads on every item, fewer than two group factors, an item on more than one) is refused, with the same message. So is a multiple-group fit, since resampling the pooled rows would let the group sizes vary from replication to replication (fit each group separately); a fit with a `cluster` variable, whose cases are not independent and cannot be resampled one at a time; a fit with sampling weights, which a resample of rows does not carry; a fit to a covariance matrix, which has no cases to resample; and a fit that did not converge.

Bootstrap intervals from supplied two-stage fits currently refuse separately stored auxiliary variables: each refit must resample and carry those values with the items. Use an eligible delta interval where available, or explicitly resample and refit them together.

Value

A `data.frame` (class `dmar_ci_bifactor`, `dmar_tbl`) with one row for each index that applies to a factor, the general factor first (ECV, omega, omega_H, PUC, H, maximal_reliability) and then each group factor (ECV, omega, omega_HS, H, maximal_reliability; its ECV is the general factor's share of the common variance of that factor's items, and its omega the reliability of their total score, as defined under *Details* of `bifactor_indices`, which the printed table repeats), and columns `factor`, `index`, `estimate` (the value `bifactor_indices()` reports, in its table or, for `maximal_reliability`, in its `"maximal_reliability"` attribute), `se` (the standard deviation of the replications used), `ci_lower`, and `ci_upper`. PUC has `NA` in `se` and the limits, and so does every column of the maximal_reliability rows when the point estimate is improper, and so do the `se` and limits of any index whose interval is withheld (see `require_half` under the `bootstrap` attribute below). The rows depend on the model alone, and the columns never change. Attributes:

`bootstrap` The counts and draws: `attempted` (`B`), `returned` (the replications used), `failed` (those set aside), `reasons`, `error_messages`, `undetermined` (always zero here: it counts the replications that left a polychoric correlation undetermined, which only `reliability_omega_categorical` meets), `improper` (improper replications set aside), `improper_kept`, `no_solution` (always zero here: it counts the resamples of three items whose correlations admit no single-factor solution, which only the single-factor coefficients of `reliability_omega` and `reliability_H` meet), `not_converged`, `too_few_cases` (always zero here: it counts the replications the

single-factor coefficients set aside for too few distinct cases), `improper_rule`, `caution` (`TRUE` when the replications set aside and the improper ones kept number at least $(1 - C)/2$ of `B`, with C the confidence level, so that how they were treated can decide where a limit falls), `require_half` (`TRUE`: an index's interval requires at least half of all attempted replications to return a value for it; otherwise its `se`, `ci_lower`, and `ci_upper` are `NA`, and the warning and the printed line say so; this is an estimability safeguard, not a coverage guarantee), `required` (the fewest replications a request may ask for, 400 for the percentile interval and 2000 for BCa at the 95 percent level; the count is held to `B`, not to the replications that return a value) and `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`, the last two when some index has its interval withheld; see `dmar_bootstrap`), `target_returned` and `target_withheld` (for each bootstrapped index, that is, each column of `t` and named as those columns, the replications that returned a value for it and whether its interval was withheld), `acceleration` and `acceleration_method` (one per index), `jackknife`, `ncpus`, `t` (a `B` by index matrix of the estimates of every replication, one column for each row with an interval, named `"factor:index"`, and `NA` where a replication gave none; the indices of an improper replication set aside under `"drop"` are kept here when they are proportions, and its maximal reliabilities are `NA`), `status` (the fate of every replication, a factor with levels `"proper"`, `"improper"`, `"error"`, `"non_finite"`, `"undetermined"`, and `"not_converged"`), and `bias` (one per column of `t`, named as its columns: the mean of the replicates the index's interval uses minus its estimate, the bootstrap estimate of its bias; see *Details*).

`B`, `B_used` The replications requested and the smallest number any reported interval rests on (under `improper = "keep"`, the intervals of the maximal reliability rest on the proper replications, the count of finite values in a maximal reliability column of `t`, which can be fewer than `bootstrap$returned`).

`n_dropped`, `B_required`, `bootstrap_status` `B` minus `B_used`, the fewest replications a request may ask for, and the verdict, in the names every bootstrap result of the package carries.

`ci_method` The interval of each index, named `"factor:index"`: the method requested, or the percentile label with the reason when a BCa interval could not be formed.

`ci_method_requested`, `conf_level`, `improper_rule`, `denominator` The settings of the call.

`metric` As for `bifactor_indices()`: `"observed"` for continuous items and `"latent_response"` for ordered items.

`general` The name of the general factor.

`improper` Whether the point estimate is improper (possible only with `improper = "keep"`).

`N`, `N_incomplete` The rows resampled and how many of them have a missing value.

`generics::tidy()` returns one row per index with `term` (`"factor:index"`), `estimate`, `se`, `ci_lower`, `ci_upper`, and `conf_level`; `generics::glance()` returns one row with the counts and the settings.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Aguirre-Urreta, M. I., Rönkkö, M., & McIntosh, C. N. (2019). A cautionary note on the finite sample behavior of maximal reliability. *Psychological Methods, 24*(2), 236–252. doi:10.1037/met0000176

DiCiccio, T. J., & Efron, B. (1996). Bootstrap confidence intervals. *Statistical Science, 11*(3), 189–228.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), *Structural equation modeling: Present and future* (pp. 195–216). Scientific Software International.

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*(1), 69–92. doi:10.1037/a0040086

Raykov, T., & Penev, S. (2006). A direct method for obtaining approximate standard error and confidence interval of maximal reliability for composites with congeneric measures. *Multivariate Behavioral Research, 41*(1), 15–28. doi:10.1207/s15327906mbr4101_2

Reise, S. P. (2012). The rediscovery of bifactor measurement models. *Multivariate Behavioral Research, 47*(5), 667–696. doi:10.1080/00273171.2012.715555

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016). Evaluating bifactor models: Calculating and interpreting statistical indices. *Psychological Methods, 21*(2), 137–150. doi:10.1037/met0000045

Yuan, K.-H., & Hayashi, K. (2003). Bootstrap approach to inference and power analysis based on three test statistics for covariance structure models. *British Journal of Mathematical and Statistical Psychology, 56*, 93–110.

See Also

`bifactor_indices` for the point estimates; `reliability_omega` for a single-factor omega and its intervals; `reliability_H` for coefficient H.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
# The nine-item bifactor model of ?bifactor_indices: one general factor
# and three orthogonal group factors, simulated so that resamples give
# proper solutions and the example runs quickly. (The 15-test
# Holzinger and Swineford model of Details gives an improper solution
# in a fifth of its resamples, each of which took seconds to fit.)
set.seed(113)
n <- 600
```

```
g <- rnorm(n); grp <- list(rnorm(n), rnorm(n), rnorm(n))
X <- vapply(1:9, function(i)
  0.5 * g + 0.5 * grp[[ceiling(i / 3)]] + sqrt(0.5) * rnorm(n), numeric(n))
colnames(X) <- paste0("x", 1:9)
model <- "g  =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
          f1 =~ x1 + x2 + x3
          f2 =~ x4 + x5 + x6
          f3 =~ x7 + x8 + x9"
fit <- lavaan::cfa(model, data = as.data.frame(X),
                   orthogonal = TRUE, std.lv = TRUE)

# Every replication refits the bifactor model. B = 10 keeps the
# example quick and is below the 400 a reported interval needs; the
# option lets it run and the warning says so. A reported interval
# deserves the default B = 1000. A resample whose solution is improper
# is set aside, and the warning and the printed lines count it.
old <- options(dmar.bootstrap_exploratory = TRUE)
ci <- ci_bifactor_indices(fit, B = 10, seed = 113)
options(old)
ci

# Every replication's indices, and what happened to it.
head(attr(ci, "bootstrap")$t[, c("g:omega", "g:omega_H", "g:H",
                                 "g:maximal_reliability")])
table(attr(ci, "bootstrap")$status)

# The broom verbs: one row per index, and the counts in one row.
generics::tidy(ci)
generics::glance(ci)
```

---

ci_c *Confidence Interval for a Contrast in a Fixed Effects ANOVA*

Description

Computes the confidence interval for an unstandardized contrast of means in a fixed effects analysis of variance, so a focused comparison among groups (a pairwise difference or any weighted combination of the means) is reported with its precision and in the units of the response. Homogeneity of variance is assumed, as in the ANOVA on which `s_anova` is based.

Usage

```
ci_c(
  means = NULL,
  s_anova = NULL,
  c_weights = NULL,
  n_per_group = NULL,
  N = NULL,
```

```
  contrast = NULL,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  df_error = NULL
)
```

Arguments

means	A vector of the group means or the means of the particular level of the effect (for fixed effect designs)
s_anova	The standard deviation of the errors from the ANOVA model (i.e., the square root of the mean square error); required, as a single positive number, since the standard error of the contrast is formed from it and the means and sample sizes cannot supply it
c_weights	The contrast weights (choose weights so that the positive *c*-weights sum to 1 and the negative *c*-weights sum to -1; i.e., use fractional values not integers)
n_per_group	Sample sizes *per group* or level of the particular factor (if length 1 it is assumed that the per group/level sample sizes are equal)
N	Total sample size
contrast	The contrast effect, obtained by multiplying the *j*th mean by the *j*th contrast weight.
conf_level	Confidence interval coverage (i.e., 1- Type I error rate); default is .95. A level above .999999998, whose tail areas are below 1e-9, is refused
alpha_lower	Type I error for the lower confidence limit, given together with alpha_upper; one without the other is refused.
alpha_upper	Type I error for the upper confidence limit. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in ci_nc_t).
df_error	The degrees of freedom for the error. In one-way designs it is *N* minus the number of groups, N - length(means), and need not be specified; it must be specified if the design has multiple factors.

Value

A 3-row data.frame with columns term and value. The term values are "ci_lower" (the lower confidence limit on the population contrast), "contrast" (the estimated unstandardized contrast), and "ci_upper" (the upper limit).

Note

Be sure to use the standard deviation and not the error variance for s_anova, not the square of this value (the error variance) which would come from the source table (i.e., use the root mean square error, not the mean square error).

Be sure to use fractional *c*-weights when doing complex contrasts (not integers) to specify `c_weights`. For example, in an ANCOVA of four groups, if the user wants to compare the mean of group 1 and 2 with the mean of group 3 and 4, `c_weights` should be specified as c(0.5, 0.5, -0.5, -0.5) rather than c(1, 1, -1, -1). Make sure the sum of the contrast weights is zero.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means.)

Steiger, J. H. (2004). Beyond the *F* Test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

See Also

`ci_sc`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
# Here is a four group example. Suppose that the means of groups 1--4 are 2, 4, 9,
# and 13, respectively. Further, let the error variance be .64 and thus the standard
# deviation would be .80 (note we use the standard deviation in the function, not the
# variance). The contrast of interest here is the average of groups 1 and 4 versus the
# average of groups 2 and 3.
ci_c(means = c(2, 4, 9, 13), s_anova = .80, c_weights = c(.5, -.5, -.5, .5),
n_per_group = c(3, 3, 3, 3), N = 12, conf_level = .95)

# Here is an example with two groups.
ci_c(means = c(1.6, 0), s_anova = .80, c_weights = c(1, -1),
n_per_group = c(10, 10), N = 20, conf_level = .95)

# An example given by Maxwell, Delaney, & Kelley (2027) :
# 20 subjects of mild hypertensives are assigned to one of four treatments: drug
# therapy, biofeedback, dietary modification, and a treatment combining all the
```

```
# three previous treatments. Subjects' blood pressure is measured two weeks
# after the termination of treatment. Now we want to form a 95% level
# confidence interval for the difference in blood pressure between subjects
# who received drug treatment and those who received biofeedback treatment

## Drug group's mean = 94; group size=4
## Biofeedback group's mean = 91; group size=6
## Diet group's mean = 92; group size=5
## Combination group's mean = 83; group size=5
## Mean Square Within (i.e., 'error_variance') = 67.375

ci_c(means = c(94, 91, 92, 83), s_anova = sqrt(67.375), c_weights = c(1, -1, 0, 0),
n_per_group = c(4, 6, 5, 5), N = 20, conf_level = .95)
```

`ci_correlation` *Confidence Intervals for the Population Correlation and Multiple Correlation*

Description

Two confidence intervals on the correlation scale share this page, named by the convention that lowercase *r* is the Pearson product-moment correlation between two variables and capital *R* is the multiple correlation between an outcome and a set of predictors.

`ci_r()` forms a confidence interval for the population correlation coefficient ρ. The confidence interval is for the population value ρ; the required input is the corresponding sample value, the observed sample correlation coefficient *r*. This approach assumes that the two variables on which the correlation is based are bivariate normally distributed (e.g., Hays, 1994, Chapter 14).

`ci_R()` constructs a confidence interval for the population multiple correlation coefficient $\rho = \sqrt{\rho^2}$ from the sample multiple correlation coefficient (or, equivalently, from the observed F-statistic and degrees of freedom). The interval is obtained by inverting the sampling distribution of the sample R^2 and propagating the limits through the monotone (square root) transform.

The two estimands meet at a single predictor: the multiple correlation from a regression on one predictor is the absolute value of the Pearson correlation between the outcome and that predictor.

Usage

```
ci_r(r, N, conf_level = 0.95, alpha_lower = NULL, alpha_upper = NULL)

ci_R(
  R = NULL,
  df_1 = NULL,
  df_2 = NULL,
  conf_level = 0.95,
  random_predictors = TRUE,
  F_value = NULL,
  N = NULL,
```

```
  p = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  ...
)
```

Arguments

r	Observed value of the sample correlation coefficient (specifically the zero-order Pearson product-moment correlation coefficient), for ci_r()
N	Total sample size. For ci_r() it must be at least 4 (see Details).
conf_level	Confidence interval coverage (i.e., 1 - Type I error rate); default is .95. For ci_r() a level above .999999998, whose tail areas are below 1e-9, is refused
alpha_lower	The Type I error rate for the lower confidence interval limit
alpha_upper	The Type I error rate for the upper confidence interval limit. If the confidence level is omitted, supplied tail areas determine it. An explicitly supplied level must agree with their implied coverage. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound. For ci_r() a nonzero tail area below 1e-9 is refused (see "Small tail areas" in ci_nc_t).
R	Observed value of the sample multiple correlation coefficient, for ci_R()
df_1	Numerator degrees of freedom
df_2	Denominator degrees of freedom
random_predictors	Whether or not the predictor variables are random or fixed (random is default)
F_value	Obtained *F*-value
p	Number of predictors
...	Allows one to potentially include parameter values for inner functions

Details

The Pearson correlation interval (ci_r). This approach will not generally lead to a symmetric confidence interval. The function first transforms r into *Z*', forms a confidence interval for the population value (i.e., ζ), and then transforms the confidence limits for ζ into the scale of the correlation coefficient. The interval requires a sample size of at least 4. The variance of *Z*' is $1/(N-3)$, which is infinite at $N = 3$; there the interval would be vacuous, covering $[-1, 1]$ regardless of *r*, and for smaller *N* the variance is undefined. The function therefore stops with an error when $N < 4$.

Fixed vs. random predictors (ci_R). The two regression models give *different* sampling distributions for the sample R^2, and so different confidence intervals on ρ. Under fixed predictors the design matrix is treated as constant in hypothetical replications of the study, and the omnibus F-statistic follows a noncentral *F* with p and $N-p-1$ degrees of freedom and noncentrality $\lambda = N\rho^2/(1-\rho^2)$ (Cohen, 1988); the CI on ρ^2 is obtained by inverting that distribution and then taking the square root

(see ci_nc_F). Under random predictors the design matrix is itself a draw from a joint multivariate normal distribution and the unconditional sampling distribution of the sample R^2 is due to Fisher (1928); the bisection on the three-moment noncentral F approximation of Lee (1971, Section 5.1) that ci_R2 uses for the random-predictor CI on ρ^2 is applied here and the limits are mapped to ρ. Gatsonis and Sampson (1989) document the comparison; in the behavioral, educational, and social sciences predictor variables are almost always random, so the default is random_predictors = TRUE. Pass random_predictors = FALSE for designs in which the predictor variables are fixed by design.

Value

ci_r() returns a 3-row data.frame with columns term and value. The term values are "ci_lower" (the lower confidence limit on the population correlation ρ), "r" (the observed sample correlation coefficient), and "ci_upper" (the upper limit on ρ).

ci_R() returns a 3-row data.frame with columns term, value, prob_less, and prob_greater. The rows are ordered "ci_lower", "R" (the sample multiple correlation coefficient supplied by the user, the point estimate), and "ci_upper", so the point estimate sits between its confidence limits. The lower and upper limits are the confidence limits on the population multiple correlation coefficient ρ (square roots of the corresponding limits on ρ^2). The prob_less and prob_greater columns are those of ci_R2 for the same sample, and its help page says what they hold (they are NA for the "R" estimate row).

Note

The ci_r() confidence interval assumes that the two variables the correlation is based on are bivariate normal. See Hays (1994, Chapter 14) for details.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J. & Olejnik, S. (2000). Determining sample size for accurate estimation of the squared multiple correlation coefficient. *Multivariate Behavioral Research, 35*, 119–137. doi:10.1207/s15327906mbr3501_5

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Gatsonis, C., & Sampson, A. R. (1989). Multiple correlation: Exact power and sample size calculations. *Psychological Bulletin, 106*(3), 516–524.

Fisher, R. A. (1928). The general sampling distribution of the multiple correlation coefficient. *Proceedings of the Royal Society of London, Series A, 121*(788), 654–673. doi:10.1098/rspa.1928.0224

Hays, W. L. (1994). *Statistics* (5th ed.). Fort Worth, TX: Harcourt Brace College Publishers.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*, 524–555. doi:10.1080/00273170802490632

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Lee, Y. S. (1971). Some results on the sampling distribution of the multiple correlation coefficient. *Journal of the Royal Statistical Society, Series B, 33*(1), 117–130.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Smithson, M. (2003). *Confidence intervals*. Thousand Oaks, CA: Sage Publications.

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Steiger, J. H., & Fouladi, R. T. (1992). R2: A computer program for interval estimation, power calculations, sample size estimation, and hypothesis testing in multiple regression. *Behavior Research Methods, Instruments, & Computers, 24*(4), 581–582. doi:10.3758/BF03203611

See Also

ci_R2, ss_aipe_r, ss_power_r, var_r, ss_aipe_R2, convert_r_Z, convert_Z_r, ci_nc_t

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# Pearson correlation, from Hays. Suppose N = 100 and r = .35.
ci_r(r = .35, N = 100, conf_level = .95)

# Here is another way to enter the above example.
ci_r(r = .35, N = 100, conf_level = NULL,
     alpha_lower = .025, alpha_upper = .025)

# Here are examples of one-sided confidence intervals.
ci_r(r = .35, N = 100, conf_level = NULL, alpha_lower = 0, alpha_upper = .05)
ci_r(r = .35, N = 100, conf_level = NULL, alpha_lower = .05, alpha_upper = 0)

# Multiple correlation from a five-predictor regression.
ci_R(R = .7071, df_1 = 5, df_2 = 50, conf_level = .95,
     random_predictors = TRUE)
```

`ci_crossed` *Confidence Interval for the Mean of a Crossed Design of Units and Raters*

Description

The confidence interval for the population mean score of a fully crossed design, in which every rater scores every unit once, when the raters, like the units, are a sample from a population. The units are the objects of measurement (persons, essays, recordings, or the topics of an information retrieval evaluation) and the raters are whoever or whatever produces the scores (judges, coders, assessors, or automated scoring models). Because a different panel of raters would have given a different mean, the standard error carries the variability of the raters as well as of the units; treating the raters as fixed, as an interval computed from the unit means alone does, understates it whenever raters differ. This is the interval `ss_aipe_crossed` plans for, and its variance component rows are the planning values that function takes.

Usage

```
ci_crossed(ratings, conf_level = 0.95)
```

Arguments

`ratings`	A numeric matrix or `data.frame` with one row per unit and one column per rater, holding each rater's score of each unit. At least two units and two raters are needed, and every cell must hold a finite score: the design is complete, with no missing values. A cell may also hold a difference of two scores, as when every rater scores every unit under two conditions; the interval is then for the mean difference.
`conf_level`	Confidence level of the interval (default `0.95`).

Details

Model. The score of unit u by rater r is $y_{ur} = \mu + a_u + b_r + e_{ur}$, with independent unit effects, rater effects, and residuals of mean zero and variances σ_u^2, σ_r^2, and σ_e^2; the residual holds the unit by rater interaction, which one score per cell cannot separate from error. This is the design Shrout and Fleiss (1979) call Case 2, a random sample of judges each of whom rates every target, and it is the crossed design with one random facet of generalizability theory (Brennan, 2001). The grand mean $\bar{y}$ of the $n_u \times n_r$ array estimates μ with variance

$$V = \frac{\sigma_u^2}{n_u} + \frac{\sigma_r^2}{n_r} + \frac{\sigma_e^2}{n_u n_r}.$$

Estimate and interval. With the mean squares MS_u, MS_r, and MS_e of the two-way analysis of variance for units, raters, and residual, the ANOVA estimates of the components are $(MS_u - MS_e)/n_r$, $(MS_r - MS_e)/n_u$, and MS_e, each of the first two set to zero when it is negative. Plugged into V they give

$$\hat{V} = \frac{k_u MS_u + k_r MS_r + (1 - k_u - k_r) MS_e}{n_u n_r},$$

with k_u equal to 1 when $MS_u > MS_e$ and 0 otherwise, and k_r likewise for raters. The interval is $\bar{y} \pm t_\nu \sqrt{\hat{V}}$, where the degrees of freedom ν are those Satterthwaite (1946) gives for a linear combination of independent mean squares,

$$\nu = \frac{\left(\sum_k a_k MS_k\right)^2}{\sum_k (a_k MS_k)^2 / \nu_k},$$

with the coefficients a_k of the numerator above and ν_k the degrees of freedom of each mean square: $n_u - 1$, $n_r - 1$, and $(n_u - 1)(n_r - 1)$. The truncation keeps $\hat{V}$ at or above $MS_e/(n_u n_r)$, so it is never negative. When few raters carry a large rater component, ν is near $n_r - 1$ and the interval is wide: the data then hold little information about how far the mean of another panel of raters would lie from this one's.

An information retrieval evaluation. To compare two retrieval systems over topics judged by several relevance assessors (people, or automated judges such as language models), let each row be a topic, each column an assessor, and each cell the difference between the two systems' effectiveness scores on that topic computed from that assessor's judgments. The interval is then for the mean difference between the systems over the populations of topics and assessors.

Designs chosen from the data. The interval assumes the numbers of units and raters were fixed in advance, as `ss_aipe_crossed` fixes them, and not chosen from the data, as a two-stage sequential procedure chooses them from a pilot array.

Value

A `data.frame` (class `dmar_tbl`) with rows `ci_lower`, `grand_mean`, and `ci_upper` (the estimate between its limits), `se` (the estimated standard error of the grand mean), `df` (its Satterthwaite degrees of freedom), the estimated variance components `var_unit`, `var_rater`, and `var_residual` (the first two truncated at zero), and the counts `n_units` and `n_raters`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Brennan, R. L. (2001). *Generalizability theory*. Springer. doi:10.1007/9781475734560

Satterthwaite, F. E. (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin, 2*(6), 110–114. doi:10.2307/3002019

Shrout, P. E., & Fleiss, J. L. (1979). Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin, 86*(2), 420–428. doi:10.1037/00332909.86.2.420

See Also

`ss_aipe_crossed` to plan the numbers of units and raters, `icc` for the intraclass correlations of the same array, and `variance_components_mls` for intervals on variance components.

Other confidence intervals: `ci_proportion()`, `ci_proportion_paired()`, `ci_two_phase()`

Examples

```
# Shrout and Fleiss (1979, Table 2): four judges each rated the same
# six targets.
ratings <- matrix(c( 9, 2, 5, 8,
                     6, 1, 3, 2,
                     8, 4, 6, 8,
                     7, 1, 2, 6,
                    10, 5, 6, 9,
                     6, 2, 4, 7),
                  nrow = 6, byrow = TRUE)
fit <- ci_crossed(ratings)
fit

# The judges differ a great deal in leniency, and with only four of
# them the interval for the mean over the population of judges is wide.
# Treating these four judges as the only ones of interest ignores that
# variability; the interval from the six target means is much narrower.
t.test(rowMeans(ratings))$conf.int

# The variance components are planning values for a larger study, here
# one whose 95% interval for the mean is no wider than 2 points.
v <- setNames(fit$value, fit$term)
ss_aipe_crossed(true_var_unit = v[["var_unit"]], true_var_rater = v[["var_rater"]],
                true_var_residual = v[["var_residual"]], width = 2)
```

`ci_cv` *Confidence Interval for the Coefficient of Variation*

Description

Computes the confidence interval for the population coefficient of variation, the standard deviation relative to the mean, from the noncentral *t* distribution, so that variability can be reported on a scale that is free of the measurement units. The interval is exact under normality and is not robust to departures from it (see Details).

Usage

```
ci_cv(
  cv = NULL,
  mean = NULL,
  sd = NULL,
  N = NULL,
  data = NULL,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL
)
```

Arguments

cv	Sample coefficient of variation, the sample standard deviation divided by the sample mean, given with N. Must be positive.
mean	Sample mean; must be positive. It is given with sd and N.
sd	Sample standard deviation (square root of the unbiased estimate of the variance). Must be positive.
N	Total sample size, given with cv or with mean and sd. Must exceed 1; it need not be a whole number.
data	Numeric vector of at least two finite scores, given in place of the summary statistics: the coefficient of variation and the sample size are taken from it.
conf_level	Desired confidence level (1-Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused
alpha_lower	The lower tail area. An omitted confidence level yields to supplied tail areas; an explicit level must agree with them.
alpha_upper	The upper tail area, supplied together with alpha_lower. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in ci_nc_t).

Details

The interval. For a random sample of N scores from a normal population with a positive mean, $\sqrt{N}/k$, where $k = s/M$ is the sample coefficient of variation, has a noncentral t distribution on $N - 1$ degrees of freedom whose noncentrality parameter is $\sqrt{N}/\kappa$, with $\kappa = \sigma/\mu$ the population coefficient of variation (Johnson & Welch, 1940; Kelley, 2007, Equations 5 to 8; McKay, 1932). The function finds the confidence limits for the noncentrality parameter (ci_nc_t) and divides $\sqrt{N}$ by each: the lower limit for κ comes from the upper limit for the noncentrality parameter, and the upper limit from the lower.

Normality. The interval is exact when the scores are a random sample from a normal distribution with a positive mean: its coverage is then the stated level at every sample size. It is not robust to departures from normality, and a larger sample does not repair it, because the distribution that is inverted is that of a normal sample whatever its size. In simulations of a nominal 95 percent interval, with 10,000 samples in each condition (population coefficients of variation of .3 and .5, sample sizes of 30 and 200), coverage was .81 to .85 for Laplace data, .90 to .94 for lognormal data, and .96 to .97 for gamma data, where it was .945 to .951 for normal data, the stated level within the error of the simulation. For the Laplace and the lognormal data it was lower at 200 than at 30. A large-sample interval that does not assume normality follows from the asymptotic normal distribution of k (Kelley, Darku, & Chattopadhyay, 2018, Equations 29 to 33). It needs a large sample: with 400,000 samples in each of the same conditions it covered .806 to .905 at $N = 30$ and .894 to .942 at $N = 200$, below the stated level in every condition, normal data included. The function does not compute it.

An infinite upper limit. The upper limit is infinite when $\sqrt{N}/k$ is at most the 1 - alpha_upper quantile of the central t distribution on $N - 1$ degrees of freedom, which happens when the sample

coefficient is large for the sample size. The lower confidence limit of the noncentrality parameter is then not positive: at that tail area the data do not rule out a population mean of zero, and no finite coefficient of variation bounds the interval above. The infinite limit is part of the exact interval. Over repeated samples from a normal population with a positive mean, the interval covers the population coefficient with the stated probability, counting the intervals that are bounded above and those that are not. In 20,000 simulated samples with a population coefficient of .8 and $N = 6$, the upper limit of the 95 percent interval was infinite in 31 percent and coverage was .952; with a coefficient of .5 and $N = 5$ it was infinite in 9 percent and coverage was .948. The stated level describes all of the intervals together and neither kind alone (computed from the noncentral *t* distribution, coverage is .92 and .73 among the intervals with an infinite limit and .96 and .97 among the others). Nothing is signaled for an infinite limit, and the `prob_less` and `prob_greater` columns hold the requested tail areas there as at a finite limit. A sample whose mean is not positive has no interval and is refused; with a population coefficient of .5 or less and five or more scores the probability of such a sample is below .00001. Where such samples are not rare, coverage among the samples that have an interval is somewhat above the stated level: it is .962 for a 95 percent interval with a coefficient of 1 and $N = 5$, where 1.3 percent of samples have a mean that is not positive. Of the 20,000 samples with a coefficient of .8 above, 25 had such a mean, and the figures given describe the other 19,975.

Value

A 4-row `data.frame` with columns `term`, `value`, `prob_less`, and `prob_greater`. The rows are ordered so the two point estimates sit between the confidence limits: `"ci_lower"` (lower confidence limit on the coefficient of variation), `"cv"` (the sample coefficient of variation), `"cv_unbiased"` (the nearly unbiased estimate, $(1 + 1/(4N))$ times the sample value; see `cv`), and `"ci_upper"` (upper confidence limit). The `prob_less` and `prob_greater` columns hold the requested tail areas: `alpha_lower` and `1 - alpha_lower` at the lower limit, and `1 - alpha_upper` and `alpha_upper` at the upper limit, also when the upper limit is infinite. They are `NA` for the point-estimate rows.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Johnson, N. L., & Welch, B. L. (1940). Applications of the non-central *t*-distribution. *Biometrika, 31*(3–4), 362–389. doi:10.1093/biomet/31.34.362

Kelley, K. (2007). Sample size planning for the coefficient of variation from the accuracy in parameter estimation approach. *Behavior Research Methods, 39*(4), 755–766. doi:10.3758/BF03192966

Kelley, K., Darku, F. B., & Chattopadhyay, B. (2018). Accuracy in parameter estimation for a general class of effect sizes: A sequential approach. *Psychological Methods, 23*(2), 226–243. doi:10.1037/met0000127

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on confidence intervals for effect sizes.)

McKay, A. T. (1932). Distribution of the coefficient of variation and the extended *t* distribution. *Journal of the Royal Statistical Society, 95*(4), 695–698.

See Also

cv

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# Fifteen scores from a normal population with a mean of 5 and a
# standard deviation of 1, whose coefficient of variation is .20.
set.seed(113)
x <- rnorm(15, mean = 5, sd = 1)

# The interval from the scores, from their mean and standard deviation,
# and from the sample coefficient itself: three routes to one interval.
ci_cv(data = x)
ci_cv(mean = mean(x), sd = sd(x), N = 15)
ci_cv(cv = sd(x) / mean(x), N = 15)

# Unequal tail areas, with .01 below the interval and .04 above it.
ci_cv(data = x, alpha_lower = .01, alpha_upper = .04)
```

---

ci_c_ancova *Confidence Interval for an (Unstandardized) Contrast in ANCOVA With One Covariate*

Description

Calculates the confidence interval for an unstandardized contrast in the one-covariate ANCOVA. Two procedures are available through the `procedure` argument. The default (`"t"`) returns a single *per-comparison* interval based on the *t* distribution: it gives the correct $(1-$ `conf_level`$)$ coverage for one contrast chosen in advance, and its standard error includes the $(\sum c_i \bar{X}_i)^2 / SS_{\text{within}(x)}$ term that accounts for the covariate separation between the groups in that one contrast. The `"bryant_paulson"` procedure instead returns Bryant–Paulson *simultaneous* (familywise) intervals over a whole family of contrasts of adjusted means; when selected, `ci_c_ancova` simply forwards its arguments to ci_c_ancova_bp and returns that result. See Details for which to use when.

Usage

```
ci_c_ancova(
  contrast = NULL,
  adj_means = NULL,
  s_ancova = NULL,
  c_weights,
  n_per_group,
```

```
  cov_means,
  SSwithin_x,
  conf_level = 0.95,
  procedure = c("t", "bryant_paulson"),
  ...
)
```

Arguments

`contrast`	The unstandardized contrast of adjusted means
`adj_means`	The vector that contains the adjusted mean of each group on the dependent variable
`s_ancova`	The standard deviation of the errors from the ANCOVA model (i.e., the square root of the mean square error from ANCOVA); required, as a single positive number, with either procedure, since the standard error of the contrast is formed from it and the adjusted means and sample sizes cannot supply it
`c_weights`	The contrast weights
`n_per_group`	Either a single number that indicates the sample size *per group* or a vector that contains the sample size of each group
`cov_means`	A vector that contains the group means of the covariate
`SSwithin_x`	The sum of squares within groups obtained from the summary table for ANOVA on the covariate
`conf_level`	The desired confidence interval coverage, (i.e., 1 - Type I error rate)
`procedure`	The interval procedure, one of `"t"` (the default, a single per-comparison interval based on the *t* distribution) or `"bryant_paulson"` (Bryant–Paulson simultaneous intervals for a family of contrasts of adjusted means). When `"bryant_paulson"` is chosen the call is forwarded to `ci_c_ancova_bp`; see Details.
`...`	Allows one to potentially include parameter values for inner functions. When `procedure = "bryant_paulson"`, these are passed on to `ci_c_ancova_bp` (for example `n_covariates`, `df`, or `contrast_type`).

Details

Per-comparison versus simultaneous. The two procedures answer different questions and are not interchangeable. Use the default `procedure = "t"` when a single contrast was planned in advance: the interval has exact per-comparison coverage and its width reflects the covariate adjustment for that specific contrast through the $(\sum c_i \bar{X}_i)^2 / SS_{\text{within}(x)}$ term. Use `procedure = "bryant_paulson"` when several contrasts (for example all pairwise comparisons of adjusted means) are examined together and the coverage statement must hold simultaneously across the family: the Bryant–Paulson generalized studentized range supplies a larger critical value that controls the familywise error rate and, because the covariates are random, correctly absorbs the extra sampling uncertainty from estimating the covariate adjustment (which holds on average over the covariate distribution, so the per-contrast separation term is not added again). A per-comparison interval used as if it were simultaneous understates the family error rate; a simultaneous interval used for one planned contrast is wider than necessary. See `ci_c_ancova_bp` for the full description of the simultaneous procedure and its arguments.

Value

A 3-row `data.frame` with columns `term` and `value` (numeric). The `term` values are `"ci_lower"` (the lower confidence limit on the unstandardized ANCOVA contrast), `"contrast"` (the unstandardized contrast point estimate), and `"ci_upper"` (the upper limit).

When `procedure = "bryant_paulson"`, the return value is whatever `ci_c_ancova_bp` returns (a table with one row per contrast and columns `contrast`, `estimate`, `ci_lower`, and `ci_upper`).

Note

Be sure to use the standard deviation and not the error variance for `s_ancova`, not the square of this value which would come from the source table (i.e., do not use the variance of the error but rather use the square root).

If `n_per_group` receives a single number, that number is considered as the sample size *per group*. If `n_per_group` receives a vector, the vector is considered as the sample size of each group.

Be sure to use fractions not the integers to specify `c_weights`. For example, in an ANCOVA of four groups, if the user wants to compare the mean of group 1 and 2 with the mean of group 3 and 4, `c_weights` should be specified as c(0.5, 0.5, -0.5, -0.5) rather than c(1, 1, -1, -1). Make sure the sum of the contrast weights are zero.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

See Also

`ci_c`,`ci_sc_ancova`, `ci_c_ancova_bp` for the Bryant–Paulson simultaneous procedure

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
# Maxwell, Delaney, & Kelley (2027) offer an example that 30 depressive
# individuals are randomly assigned to three groups, 10 in each, and ANCOVA
# is performed on the posttest scores using the participants' pretest
```

```
# scores as the covariate. The means of pretest scores of group 1 to 3 are
# 17, 17.7, and 17.4, respectively, and the adjusted means of groups 1 to 3
# are 7.5, 12, and 14, respectively. The error variance in ANCOVA is 29,
# and the sum of squares within groups from ANOVA on the covariate is 752.5.

# To obtain the confidence interval for adjusted mean of group 1 versus group 2:
ci_c_ancova(adj_means = c(7.5, 12, 14), s_ancova = sqrt(29),
            c_weights = c(1, -1, 0), n_per_group = 10,
            cov_means = c(17, 17.7, 17.4), SSwithin_x = 752.5)

# That interval is the right one for a single contrast planned in advance.
# For the family of all three pairwise comparisons of the adjusted means,
# with coverage that holds simultaneously across the family, select
# procedure = "bryant_paulson"; the call is forwarded to ci_c_ancova_bp().
# The simultaneous limits are wider, which is the price of the family
# statement: group 1 against group 2 runs from -10.61 to 1.61, where the
# single planned contrast above ran from -9.46 to 0.46.
ci_c_ancova(adj_means = c(7.5, 12, 14), s_ancova = sqrt(29), n_per_group = 10,
            procedure = "bryant_paulson")
```

`ci_c_ancova_bp` *Bryant–Paulson Simultaneous Confidence Intervals for Contrasts of Adjusted Means in ANCOVA*

Description

Constructs Tukey–Kramer-type *simultaneous* confidence intervals on one or more contrasts of covariate-adjusted means in the analysis of covariance (ANCOVA) when the covariate(s) are *random*, using the Bryant–Paulson generalized studentized range (see `bryant_paulson`). Unlike the per-comparison interval of `ci_c_ancova`, these intervals control the *familywise* error rate over the whole family of comparisons and, through the Bryant–Paulson critical value, correctly account for the extra sampling uncertainty that comes from estimating the covariate adjustment from random covariates. Naively applying Tukey's method to adjusted means ignores that uncertainty and produces intervals that are too narrow (below-nominal coverage). The Bryant–Paulson critical value exceeds the studentized range critical value for the same α, number of means, and error degrees of freedom, so these intervals are wider than Tukey's (Maxwell, Delaney, & Kelley, 2027, Chapter 9). Over the range of Bryant and Paulson's (1976) Table 1, the excess is under 5% for every number of means and both tabled α levels (.05 and .01) when the error degrees of freedom ν are at least 14 with one covariate, 30 with two, or 40 with three. It grows as ν falls and as the numbers of covariates and means rise; at $\nu = 2$ and $\alpha = .05$ it reaches 34% with one covariate and 84% with three (92% at $\alpha = .01$).

Usage

```
ci_c_ancova_bp(
  adj_means,
  s_ancova,
  c_weights = NULL,
```

```
  n_per_group,
  n_covariates = 1,
  df = NULL,
  conf_level = 0.95,
  contrast_type = c("pairwise", "allowance")
)
```

Arguments

adj_means	A numeric vector of the covariate-adjusted group means (one per group). The number of groups k is inferred from its length.
s_ancova	The standard deviation of the errors from the ANCOVA model, i.e., the square root of the ANCOVA error mean square (use the standard deviation, *not* the variance from the source table).
c_weights	Optional contrast weights. May be (i) a numeric vector of length k giving a single contrast, or (ii) a matrix/data.frame with k columns, each row a contrast. If NULL (the default), all $k(k-1)/2$ pairwise comparisons are returned. For each contrast the weights should sum to zero.
n_per_group	Either a single number giving the common per-group sample size or a numeric vector of per-group sample sizes. The Bryant–Paulson distribution is exact for balanced designs; for unequal n a Tukey–Kramer harmonic adjustment is used (see Details).
n_covariates	The number of random covariates, p. Default 1.
df	Optional error degrees of freedom ν. If NULL (default) it is computed for a one-way ANCOVA as $\nu = \sum n - k - p$. Supply it directly for other designs (e.g., a randomized-block ANCOVA, where ν differs).
conf_level	The simultaneous (familywise) confidence level. Default 0.95.
contrast_type	One of "pairwise" (default) or "allowance". "pairwise" uses the Tukey–Kramer quadratic standard error, exact for pairwise comparisons. "allowance" uses Tukey's allowance, which yields intervals that hold simultaneously over *all* contrasts, including complex ones (this is the form in Eq. (2.4) of Bryant and Bruvold, 1980). The width factor each choice applies is given in Details. The two coincide for pairwise comparisons.

Details

The interval. For a contrast $\psi = \sum_i c_i \theta_i$ of adjusted means, the simultaneous interval is

$$\hat{\psi} \;\pm\; q_{\alpha;\,p,k,\nu}\;\hat{\sigma}_{y|x}\;w(c),$$

where $q_{\alpha;\,p,k,\nu}$ is the upper-α Bryant–Paulson critical value (qbryant_paulson) and the width factor is $w(c) = \frac{1}{\sqrt{2}}\sqrt{\sum_i c_i^2/n_i}$ for contrast_type = "pairwise" or $w(c) = \frac{1}{2}\sum_i |c_i|\sqrt{1/n}$ for contrast_type = "allowance" (balanced n). For a pairwise difference with common n both reduce to $q_{\alpha;\,p,k,\nu}\,\hat{\sigma}_{y|x}\sqrt{1/n}$, reproducing the critical difference of Bryant and Bruvold (1980).

No per-comparison covariate term. By design the standard error here does *not* include the $(\bar{X}_i - \bar{X}_j)^2/SS_{\text{within}(x)}$ term that appears in a single-comparison ANCOVA interval (ci_c_ancova). In the Bryant–Paulson framework the random-covariate uncertainty is carried by the (larger) critical

value, which holds *on average* over the covariate distribution; adding the per-pair term as well would double-count it.

Unequal sample sizes. For unbalanced designs the `"pairwise"` standard error uses $\sqrt{c_i^2/n_i}$ directly (the Tukey–Kramer generalization); coverage is then approximate but typically very close to nominal and slightly conservative.

Value

A `data.frame` (class `dmar_tbl`) with one row per contrast and columns `contrast` (a label), `estimate` (the contrast of adjusted means $\hat{\psi}$), `ci_lower`, and `ci_upper`. The Bryant–Paulson critical value used is stored in the `"critical_value"` attribute and the confidence level in the `"conf_level"` attribute (printed beneath the table).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bryant, J. L., & Paulson, A. S. (1976). An extension of Tukey's method of multiple comparisons to experimental designs with random concomitant variables. *Biometrika, 63*, 631–638.

Bryant, J. L., & Bruvold, N. T. (1980). Multiple comparison procedures in the analysis of covariance. *Journal of the American Statistical Association, 75*(372), 874–880. doi:10.2307/2287175

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

See Also

`bryant_paulson` for the underlying distribution; `ci_c_ancova` for the per-comparison interval; `ancova` for an ANCOVA fit.

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
# Bryant and Bruvold (1980) worked example: 6 panels, 1 covariate, nu = 14,
# ANCOVA error MS = 0.01326. Here the design is a randomized block with
# s = 4 blocks, so the per-group size for the adjusted-mean SE is 4 and the
# error df (14) must be supplied directly. The multiplier for these
# intervals is a Bryant-Paulson quantile, which has no closed form; it is
# obtained by inverting the distribution function with a root search.
adj <- c(3.595, 3.619, 4.102, 4.515, 4.618, 4.876)
bp <- ci_c_ancova_bp(adj_means = adj, s_ancova = sqrt(0.01326),
                     n_per_group = 4, n_covariates = 1, df = 14)
bp
# The multiplier is 4.83 and every pairwise critical difference is 0.278,
```

```
# matching the paper; the 15 intervals hold jointly at the 95 percent level.
# The multiplier is kept on the result, so the critical difference can be
# rebuilt by hand as q * s_ancova * sqrt(1/n):
attr(bp, "critical_value")
attr(bp, "critical_value") * sqrt(0.01326) * sqrt(1 / 4)

# A complex contrast (say panels 1 and 2 against panels 3 through 6) is
# requested by passing its weights to c_weights, together with
# contrast_type = "allowance", the all-contrasts form of Eq. (2.4) of
# Bryant and Bruvold. That contrast of adjusted means is -0.921, with
# simultaneous limits of -1.199 and -0.643.
ci_c_ancova_bp(adj_means = adj, s_ancova = sqrt(0.01326),
               c_weights = c(0.5, 0.5, -0.25, -0.25, -0.25, -0.25),
               n_per_group = 4, n_covariates = 1, df = 14,
               contrast_type = "allowance")
```

`ci_dunnett` *Dunnett's Simultaneous Confidence Intervals Against a Control*

Description

Computes Dunnett's (1955, 1964) simultaneous confidence intervals for the $a - 1$ comparisons of $a - 1$ treatment means against a single control mean, with family-wise coverage at the specified `conf_level`. Returns the result in tidy long form.

Usage

```
ci_dunnett(
  x,
  group = NULL,
  control = NULL,
  alternative = c("two_sided", "less", "greater"),
  conf_level = 0.95
)
```

Arguments

`x`	Either (a) a fitted `lm` or `aov` object with a one-way factor predictor, or (b) a numeric vector of observations, in which case `group` must also be supplied.
`group`	When `x` is a vector, a factor of group labels of the same length.
`control`	Character name of the control level (must be one of the factor levels). If `NULL` (default), the first level is used.
`alternative`	One of `"two_sided"` (default; the base-R spelling `"two.sided"` is accepted as an alias), `"less"`, or `"greater"`.
`conf_level`	Family-wise confidence level. Default `0.95`.

Details

Critical value. The two-sided Dunnett critical value $d^{(2)}_{\alpha,a-1,\nu}$ is obtained from the multivariate t distribution with $a-1$ dimensions, common correlation 0.5 (the Dunnett correlation under balanced n; the function does not adjust for unequal n), and ν error degrees of freedom. The function uses the existing `cv_dunnett()` critical value.

Adjusted *p*-values. Computed exactly from the same equicorrelated multivariate t distribution. The one common correlation $1/2$ admits a one-factor representation, so the probability that all comparisons fall inside (or below) the observed statistic collapses to two nested one-dimensional integrals, evaluated by quadrature. The adjusted p-value is one minus that probability. The computation is deterministic (no Monte Carlo) and needs no additional package.

Value

A `data.frame` with one row per non-control level. Columns: `contrast`, `mean_difference`, `se`, `t_value`, `ci_lower`, `ci_upper`, `p_adjusted`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Dunnett, C. W. (1955). A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association, 50*(272), 1096–1121.

Dunnett, C. W. (1964). New tables for multiple comparisons with a control. *Biometrics, 20*(3), 482–491.

Hsu, J. C. (1996). *Multiple comparisons: Theory and methods*. Chapman & Hall.

See Also

`cv_dunnett`, `ci_tukey_kramer`, `ci_scheffe`

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# 1. Compare the SSRI and placebo arms of the depression_bdi study
#    against the wait list control:
fit <- lm(bdi_post ~ condition, data = depression_bdi)
ci_dunnett(fit, control = "wait_list")

# 2. One-sided: a treatment that works pulls the posttest BDI down,
#    so the directional alternative is "less":
ci_dunnett(fit, control = "wait_list", alternative = "less")
```

`ci_dunnett_t3` *Provides Dunnett's T3 Simultaneous Confidence Intervals for All Pairwise Comparisons Without Assuming Homogeneity of Variance*

Description

Forms simultaneous confidence intervals and multiplicity-adjusted *p*-values for all $a(a-1)/2$ pairwise differences among a group means without assuming that the groups share a variance. Each pair has its own standard error and Welch–Satterthwaite degrees of freedom, and its critical value is that of the studentized maximum modulus for the $a(a-1)/2$ comparisons at those degrees of freedom (Dunnett, 1980). This is the procedure SPSS reports as Dunnett's T3; `cv_smm` gives its critical value.

Usage

```
ci_dunnett_t3(x, group = NULL, conf_level = 0.95)
```

Arguments

`x`	Either (a) a fitted `lm` or `aov` object with a single factor predictor, or (b) a numeric vector of the outcome, in which case `group` must also be supplied. An infinite outcome value is an error, since it leaves its group's mean and variance undefined.
`group`	When `x` is a vector, a factor (or coercible to factor) giving the group membership of each observation. Observations with a missing outcome or a missing group are dropped.
`conf_level`	Family-wise confidence level. Default `0.95`.

Details

The procedure. For groups g and h, with means $\bar{Y}_g$ and $\bar{Y}_h$, variances s_g^2 and s_h^2, and sizes n_g and n_h, the standard error of the difference uses only those two groups' variances,

$$SE_{gh} = \sqrt{s_g^2/n_g + s_h^2/n_h},$$

and its degrees of freedom are the Welch–Satterthwaite value

$$df_{gh} = \frac{(s_g^2/n_g + s_h^2/n_h)^2}{s_g^4/[n_g^2(n_g-1)] + s_h^4/[n_h^2(n_h-1)]},$$

as in Maxwell, Delaney, and Kelley (2027, Equations 5.13 and 5.14) and in base R's `t.test` for two groups. With $C = a(a-1)/2$ comparisons in the family and α equal to one minus `conf_level`, the interval for $\mu_h - \mu_g$ is

$$(\bar{Y}_h - \bar{Y}_g) \pm m_{\alpha;\, C,\, df_{gh}}\, SE_{gh},$$

where $m_{\alpha;C,df}$ is the upper α critical value of the studentized maximum modulus for C comparisons on df degrees of freedom (Stoline & Ury, 1979), the value `cv_smm(alpha_level = 1 -`

`conf_level`, `df`, `n_comparisons = C`) returns. The adjusted *p*-value is the probability that a studentized maximum modulus variate for C comparisons on the pair's degrees of freedom exceeds the observed $|t| = |\bar{Y}_h - \bar{Y}_g|/SE_{gh}$, so a pair's `p_adjusted` falls below α when its interval excludes zero and not otherwise, to numerical precision: the critical value's tail area matches α to a relative 10^{-6} or better. Both are computed by numerical integration rather than simulation, and the degrees of freedom are not rounded. As with Games–Howell, the family-wise level holds approximately rather than exactly, since each pair is referred to a distribution indexed by its own estimated degrees of freedom.

Dunnett's T3 and Tamhane's T2. Tamhane (1979) compared procedures for pairwise comparisons of means with unequal variances. The one known as Tamhane's T2 uses the same standard errors and degrees of freedom but takes its critical value from the inequality of Šidák (1967), treating the C *t* statistics as if they were independent: each pair is tested at the level $1-(1-\alpha)^{1/C}$, so its critical value is the $(1+(1-\alpha)^{1/C})/2$ quantile of *t* on df_{gh} and its adjusted *p*-value is $1-(1-p_{gh})^C$, with p_{gh} the pair's two-sided Welch *p*-value. Dunnett (1980) replaced that bound with the studentized maximum modulus at the same degrees of freedom. The two distribution functions differ only in where the power is taken. With $S = \sqrt{\chi^2_{df}/df}$ and Φ the standard normal distribution function, the probability that all C statistics fall within $\pm c$ is $\{\mathrm{E}[2\Phi(cS) - 1]\}^C$ as T2 computes it and $\mathrm{E}\{[2\Phi(cS) - 1]^C\}$ under the studentized maximum modulus, which by Jensen's inequality is never the smaller of the two. T3's critical value therefore never exceeds T2's at the same degrees of freedom: its intervals are never wider, and its adjusted *p*-values never larger. On the reactivated-story scores of Maxwell, Delaney, and Kelley (2027, Chapter 12, Exercise 25), three groups of 13, the T2 *p*-values are .045, .001, and .027 and the T3 values .044, .001, and .026, which are the values SPSS reports for Dunnett's T3. SPSS offers the two as separate procedures, Tamhane's T2 and Dunnett's T3; the T in T3 comes from Tamhane, who developed its predecessor (Maxwell, Delaney, & Kelley, 2027, Chapter 5). A formula that raises the single *t* probability $2F_t(|t|) - 1$, with F_t the distribution function of *t* on df_{gh}, to the power C computes T2, whatever it is called.

Dunnett's T3 or Games–Howell. `ci_games_howell` uses the same standard errors and degrees of freedom but refers $|t|\sqrt{2}$ to the studentized range for a means. Dunnett (1980) found that Games–Howell becomes slightly liberal when the samples are small, and Maxwell, Delaney, and Kelley (2027, Chapter 5) therefore recommend Dunnett's T3 when the groups have fewer than roughly 50 observations each and Games–Howell for larger samples. When the variances are in fact homogeneous, `ci_tukey_kramer` pools them and so has more error degrees of freedom and more power. With two groups the T3 interval and *p*-value are exactly those of Welch's *t* test, `t.test(..., var.equal = FALSE)`, because the studentized maximum modulus for one comparison is the two-sided *t*.

Value

A `data.frame` with one row per pairwise comparison, in the order `ci_games_howell` uses: with the factor levels in their order, each later level is compared with each earlier one, and the `contrast` label (`"B - A"`) names the difference taken. The columns are `contrast`, `mean_difference`, `se` (the separate-variance standard error of the difference), `df` (the pair's Welch–Satterthwaite degrees of freedom, which is why it varies from row to row), `t_value` (`mean_difference` divided by `se`), `ci_lower`, `ci_upper`, and `p_adjusted`. The `conf_level` attribute records the family-wise level, and the `ci_adjustment` and `p_adjustment` attributes are `"dunnett_t3"`. The table prints through the `dmar_tbl` display layer and works with `tidy` and `glance` (see `dmar_tidiers`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Dunnett, C. W. (1980). Pairwise multiple comparisons in the unequal variance case. *Journal of the American Statistical Association, 75*(372), 796–800. doi:10.2307/2287161

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 5 on the multiple-comparisons problem, where the modifications of Tukey's HSD for unequal variances, Dunnett's T3 and Games–Howell, are developed.)

Šidák, Z. (1967). Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association, 62*(318), 626–633. doi:10.1080/01621459.1967.10482935

Stoline, M. R., & Ury, H. K. (1979). Tables of the Studentized maximum modulus distribution and an application to multiple comparisons among means. *Technometrics, 21*(1), 87–93. doi:10.1080/00401706.1979.10489726

Tamhane, A. C. (1979). A comparison of procedures for multiple comparisons of means with unequal variances. *Journal of the American Statistical Association, 74*(366), 471–480. doi:10.2307/2286358

See Also

`cv_smm` for the critical value it uses, `ci_games_howell` for the studentized range counterpart, `ci_tukey_kramer` for the homogeneity-assuming counterpart, `ci_dunnett` for many-to-one comparisons, and `dmar_tidiers` for the tidy methods.

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# The raw drinks_per_week outcome of the drinks_trial data: the arms hold
# 37, 32, and 19 patients, fewer than the roughly 50 per group below
# which Games-Howell runs slightly liberal, and the Standard arm's
# variance is more than three times that of either CRA arm.
ci_dunnett_t3(drinks_trial$drinks_per_week, drinks_trial$treatment)

# A fitted one-way model may be passed instead of the two vectors.
ci_dunnett_t3(aov(drinks_per_week ~ treatment, data = drinks_trial))

# Each half-width is the studentized maximum modulus critical value for
# the three comparisons, at that pair's degrees of freedom, times the
# pair's standard error.
t3 <- ci_dunnett_t3(drinks_trial$drinks_per_week, drinks_trial$treatment)
t3$ci_upper[1] - t3$mean_difference[1]
```

```
cv_smm(alpha_level = .05, df = t3$df[1], n_comparisons = 3)$value * t3$se[1]
```

`ci_eigenvalue` *Confidence Interval on the Largest Eigenvalue of a Sample Covariance Matrix*

Description

Computes an asymptotic confidence interval on the largest population eigenvalue λ_1 of a population covariance matrix, given a sample covariance matrix from N observations on p variables under multivariate normality. Useful in principal-components analysis and dimension-reduction settings to gauge whether the largest eigenvalue is well-separated from the second.

Usage

```
ci_eigenvalue(cov_matrix, N = NULL, conf_level = 0.95, which_eigenvalue = 1)
```

Arguments

`cov_matrix`	Sample covariance matrix (a symmetric, positive- semidefinite numeric matrix), or a data frame whose columns are the variables (in which case `cov_matrix` is computed internally).
`N`	Total sample size (number of rows of the original data). Required when `cov_matrix` is supplied as a matrix; ignored when `cov_matrix` is a data frame (`nrow(cov_matrix)` is used instead).
`conf_level`	Confidence level. Default `0.95`.
`which_eigenvalue`	Which eigenvalue (1 = largest, 2 = next, ...) to bracket. Default 1.

Details

Asymptotic distribution. Under multivariate normality with eigenvalues $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_p$, the sample eigenvalues $\hat{\lambda}_j$ are asymptotically independent and approximately normal with mean λ_j and variance $2\lambda_j^2/(N-1)$ when the eigenvalues are simple (well-separated) (Anderson, 2003, Theorem 13.3.1; Muirhead, 1982, Section 9.7). The asymptotic CI is therefore

$$\hat{\lambda}_j \cdot \exp\left(\pm z_{1-\alpha/2} \sqrt{\frac{2}{N-1}} \right),$$

on the multiplicative scale (equivalently, a Wald CI on $\log \lambda_j$ with variance $2/(N-1)$). The log scale is the natural variance-stabilizing transformation for an eigenvalue.

Caveats. The asymptotic CI assumes well-separated population eigenvalues. When the largest two eigenvalues are close, the sample eigenvalue exhibits a "repulsion" phenomenon and the CI is biased (typically too narrow). Diagnostic: if $\hat{\lambda}_1/\hat{\lambda}_2$ is close to 1, the asymptotic CI should not be relied upon; a bootstrap is preferable.

Value

A 3-row `data.frame` with rows ordered `"ci_lower"`, `"eigenvalue"` (the sample eigenvalue point estimate), and `"ci_upper"`, so the point estimate sits between its confidence limits.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Anderson, T. W. (2003). *An introduction to multivariate statistical analysis* (3rd ed.). Wiley. (See Chapter 13.)

Muirhead, R. J. (1982). *Aspects of multivariate statistical theory*. Wiley. (See Section 9.7.)

See Also

`prcomp`, `eigen`

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
# 1. From a data frame:
set.seed(113)
X <- data.frame(matrix(rnorm(200), nrow = 50))
ci_eigenvalue(X, which_eigenvalue = 1)

# 2. From an explicit covariance matrix:
S <- cov(X)
ci_eigenvalue(S, N = nrow(X), which_eigenvalue = 1)
```

`ci_eta_squared` *Confidence Interval for Eta Squared (Effect Size for ANOVA)*

Description

Computes the point estimate of eta squared (η^2) of each effect of an analysis of variance and an exact, noncentrality-based confidence interval for its population value, for the *partial* and the *total* estimand (see `eta_squared`). With `estimand = "all"`, the default, the function reports both; the interval from an effect's F statistic is an interval for the partial value, and the total has one only where it equals the partial value, in a one-way design (see Details). `"total"` or `"partial"` reports one estimand.

Usage

```
ci_eta_squared(
  object = NULL,
  estimand = c("all", "total", "partial"),
  F_value = NULL,
  df_effect = NULL,
  df_error = NULL,
  N = NULL,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL
)
```

Arguments

`object`	Optional. A fitted model object of class `aov`, `lm`, or `aovlist` (multi-stratum aov fit, e.g.\ `aov(y ~ A + Error(subject/A), data = d)`). For multi-stratum fits the function walks every error stratum and returns one row per non-`Residuals` effect, identifying the stratum used. Give either `object` or the F interface, not both.
`estimand`	Which estimand to report: `"all"` (the default, both, with `NA` where the inputs determine no estimate or no interval), `"total"`, or `"partial"`. An estimand asked for by name for which the inputs determine no interval is refused (see Details).
`F_value`	Observed F-value, or a vector of the F statistics of several effects tested against the same error term.
`df_effect`	Numerator degrees of freedom for the effect (one value for each effect, or one for all).
`df_error`	Error (residual) degrees of freedom.
`N`	Total sample size, the total number of observations (derived automatically from a fitted model, via `nobs(object)` for single-stratum fits, or, for `aovlist` fits, recovered as one more than the sum of the effect and residual degrees of freedom across every stratum, which is the total number of observations). The confidence limits for the noncentrality parameter are converted into limits for η^2 with `N`. For an F from a design with an `Error()` stratum, give the size of the effect's stratum instead (see Details): the number of subjects for an effect tested between subjects, and for an effect tested within subjects the stratum's total degrees of freedom, which is `df_effect + df_error` when the effect is the only one tested in its stratum, not the number of rows of the data. `N` must be at least `df_effect + df_error`.
`conf_level`	Desired confidence coverage; default `0.95`. Used only when `alpha_lower` and `alpha_upper` are both `NULL`.
`alpha_lower, alpha_upper`	Optional Type I error on the lower and upper side. If both are `NULL`, a symmetric interval at `conf_level` is used. If both are supplied, `conf_level` is recomputed as `1 - alpha_lower - alpha_upper`. The table records the coverage, `1 -`

alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound.

Details

The function takes a fitted model (aov, lm, or an aovlist from aov() with an Error() term) or an F statistic with its degrees of freedom and N. For factorial and within-subjects designs it returns one row per effect, the partial value computed against that effect's own error stratum.

Point estimates. The partial value is $\hat{\eta}^2_{\text{partial}} = df_{\text{effect}}F/(df_{\text{effect}}F + df_{\text{error}})$, which equals $SS_{\text{effect}}/(SS_{\text{effect}} + SS_{\text{error}})$; the total value of a fitted model is $SS_{\text{effect}}/SS_{\text{total}}$, with the corrected total sum of squares. They are the estimates of eta_squared, whose Details describe the two estimands and the sums of squares of a fitted model. An F statistic does not determine the total value, which the F interface reports as NA.

Confidence interval. The interval rests on the two principles Steiger (2004, pp. 167–168) sets out. A confidence interval for the noncentrality parameter λ of the effect's F test is found by inverting the noncentral F distribution (via ci_nc_F), and because η^2 is an increasing function of λ, the limits for λ map into limits for η^2:

$$\eta^2_{\text{bound}} = \frac{\lambda_{\text{bound}}}{\lambda_{\text{bound}} + N},$$

where N is the number of observations for an effect tested between subjects and the size of the effect's error stratum for an effect of an aovlist fit (see below). When the observed F is below the critical value that sets the lower limit, the lower limit for λ, and so for η^2, is 0. In a one-way design the interval is the one ci_pvaf and ci_omega_squared give, the population η^2 and ω^2 being the same quantity; the functions differ there only in their point estimates.

What the interval estimates: the partial value. The population value of an effect's partial η^2 is the value its sample counterpart converges to as the sample grows, which is how Steiger (2004, Eq. 35, p. 171) defines the population ρ^2 of a one-way design. For an effect tested between subjects, in a one-way or a factorial design, $\lambda = N\sigma^2_{\text{effect}}/\sigma^2_e$, with σ^2_{effect} the variance of the effect over all N observations and σ^2_e the within-cell variance, so the population partial η^2, $\sigma^2_{\text{effect}}/(\sigma^2_{\text{effect}} + \sigma^2_e)$, equals $\lambda/(\lambda + N)$ at every sample size (Steiger, 2004, Eqs. 13, 15, 16, and 19, pp. 169–170, where it is written ω^2 and described as the proportion of the variance that remains after the other effects have been partialed out; Kelley, 2007, Eqs. 44, 45, and 50, p. 12). The interval converted with N is therefore exact for it, and each effect of a fitted aov or lm model is converted with the number of observations.

The total value. The population total η^2 of an effect, σ^2_{effect} as a proportion of the variance due to every effect of the design and the error, depends on the noncentrality parameters of the other effects as well as on the effect's own, and the distribution of one F statistic carries no information about them. The intervals of Steiger (2004, pp. 169–171), Smithson (2003, pp. 43–44, where the correlation ratio of a factorial effect is "a squared partial correlation"), and Fidler and Thompson (2001, Table 8) for the effects of a fixed-effects factorial design are intervals for the partial value. In a one-way design the effect and the error are the whole design, the total and the partial value are the same quantity, and the interval of a fitted one-way model is reported for both. For an effect of a design with other effects, and for every effect of an aovlist fit (whose total includes the differences among subjects), no interval is formed for the total: with estimand = "all" its estimate is reported with NA limits, and estimand = "total" is refused with an error of class dmar_estimand_undetermined

(see dmar_metadata). The *F* interface determines neither the total value nor its interval, since *F* does not say whether the design has other effects; for a one-way design give the fitted model, or use `estimand = "partial"`, whose interval is then the total's too.

A conversion with the degrees of freedom. Smithson (2001, Eqs. 8 and 9, p. 618; 2003, Formulas 5.3 and 5.4, p. 43) defines the population η^2 of a factorial effect in the same way, through the effect's variance over all *N* cases, but converts the limits with $\lambda/(\lambda + u + v + 1)$, *u* and *v* being the effect and error degrees of freedom (2001, Eq. 11; 2003, Formula 5.6). That form comes from regression, where $u+v+1$ is the sample size (2001, Eq. 4, p. 612), and it agrees with his definition only when $u + v + 1 = N$, as in a one-way design. For a factorial effect it is a convention for the conversion rather than a definition of the parameter: its limits are exact for $\lambda/(\lambda + u + v + 1)$, a larger quantity that depends on the degrees of freedom of the design's other effects. For the 2 by 7 design with four observations per cell that both Steiger (2004, p. 171) and Smithson (2001, p. 619) analyze, the 90% limits for the interaction are [.008, .316] with *N* = 56 and [.009, .346] with 6 + 42 + 1 = 49, and those for the row effect are [.009, .234] with *N* = 56 and [.012, .280] with 1 + 42 + 1 = 44. Converted with 44, a 95% interval for that row effect covers the population value 94.5% of the time when the value is .02 and 91.9% of the time when it is .40. Fidler and Thompson (2001, Table 8, p. 596) computed their fixed-effects factorial intervals with Smithson's syntax, from the *F* values and their degrees of freedom alone. Supplying $u + v + 1$ as `N` with `estimand = "partial"` reproduces such published limits.

Within-subjects designs. An effect of an `aovlist` fit is tested in its own error stratum, and the number of rows of the data is not a sample size for that test. Under compound symmetry, Steiger (2004, Eq. 58, p. 177) gives the noncentrality parameter of a one-way within-subjects design with *n* subjects and *p* occasions as $\lambda = n\sum_j(\mu_j - \mu)^2/[\sigma^2(1 - \rho)]$. The sums of squares for the occasions and for their error grow as $n\sum_j(\mu_j - \mu)^2$ and $n(p - 1)\sigma^2(1 - \rho)$, so the value the sample partial η^2 converges to is $\lambda/[\lambda + n(p - 1)]$, where $n(p - 1)$ is the stratum's total degrees of freedom, $df_{\text{effect}} + df_{\text{error}}$. Under sphericity the same holds for every effect of a within-subjects stratum, with the stratum's total degrees of freedom (those of its effects and its error) in place of $n(p-1)$, and each such effect is converted with that total. An effect in the between-subjects stratum of a mixed design is tested among the subjects' means, as an effect of a between-subjects design is tested among its observations, and is converted with the number of subjects, one more than that stratum's total degrees of freedom. Converted with the number of rows, np, the interval of a within-subjects effect is for $\lambda/(\lambda + np)$, a smaller quantity, which the sample partial η^2 does not estimate. In the three-occasion analysis of pygmalion in `vignette("DMAR", package = "DMAR")`, $F(2, 618)$ converted with the 930 rows gives the 95% interval [.111, .194], which lies below the estimate of .212; converted with 2 + 618 = 620, the interval is [.157, .265].

The literature cited here does not settle the within-subjects case. Steiger (2004, p. 176) restricts his procedures to completely between-subjects designs, noting that the noncentrality parameter of a repeated measures design confounds the effects of the treatments with the correlation among the observations (pp. 176–177), and the factorial examples of Smithson (2001, 2003) and Fidler and Thompson (2001) are between-subjects designs. The conversion of a within-subjects stratum given here follows from Steiger's Eq. 58 and rests, as the stratum's *F* test does, on sphericity. A within-subjects partial η^2 also leaves the differences among subjects out of its denominator, so it is not comparable with a partial η^2 from a between-subjects design (Bakeman, 2005, p. 380).

The *F* interface. The limits are converted with `N` as supplied. The function cannot tell a between-subjects factorial *F*, for which the total sample size properly exceeds $df_{\text{effect}} + df_{\text{error}} + 1$, from a within-subjects *F* given with the number of rows, so it does not warn; for an *F* tested within subjects, supply the size of the effect's stratum as `N`.

Designs supported.

- *Between-subjects ANOVA*: fitted `aov`/`lm`, or *F*, df, and *N*. Each effect's limits are converted with the number of observations.
- *Within-subjects or mixed ANOVA*: fitted `aovlist`. Each effect's CI is built from its own stratum's *F* and residual *df* and converted with the size of that stratum: the number of subjects for the between-subjects stratum (the stratum of the outermost term of `Error()`, such as `subject` in `Error(subject/time)`) and the total degrees of freedom of a within-subjects stratum. The `N` column reports the total number of observations across all strata, and the `stratum` column identifies which error term each row used.

Sums of squares in factorial designs. When a fitted model is supplied, *F*-values are read from `anova()` (single-stratum) or `summary()` (multi-stratum), both of which use Type I (sequential) sums of squares in base R. For balanced designs Types I, II, and III agree; for unbalanced designs they differ. If Type II or III *F*-values are required, compute them with e.g.\ `car::Anova(object, type = 3)` and pass the relevant *F*, degrees of freedom, and *N* into the *F* interface. See the Details of `eta_squared` on the coding of factors that Type III tests depend on. `factorial_anova` applies sum-to-zero contrasts itself and reports partial η^2 and partial ω^2 with their confidence intervals for between-subjects factorial designs.

Value

A `data.frame` (class `dmar_ci_anova`, printed through `dmar_tbl`) with one row per effect and the columns `effect`; for each estimand reported, its estimate and limits, `eta_squared_total`, `eta_squared_total_ci_lower`, `eta_squared_total_ci_upper`, `eta_squared_partial`, `eta_squared_partial_ci_lo` and `eta_squared_partial_ci_upper`; `stratum` (for an `aovlist` fit); and `F_value`, `df_effect`, `df_error`, and `N`. With the *F* interface `effect` is `"overall"` (the names of `F_value`, when it has them, or `effect_1`, `effect_2`, and so on, for several). `N` is the number of observations (for the *F* interface, `N` as supplied). The limits of the *F* interface and of a single-stratum fit are converted with that `N`; the limits of an effect of an `aovlist` fit are converted with the size of its own stratum (see Details), so that its row's *F*, degrees of freedom, and that size, passed to the *F* interface, reproduce them. The size is not a column of the table: it is the sum of the `Df` column of the stratum's table in `summary(object)`, plus one for the stratum that lies between subjects (`id` in `Error(id / occasion)`). The table records the confidence level of its intervals in its `conf_level` attribute (`conf_level`, or `1 - alpha_lower - alpha_upper` when the tail areas are supplied), which the console prints beneath it and `results_sentence` reports; the `estimand` attribute records the estimand asked for, and the `estimand_notes` attribute names each estimate or interval that is `NA` with the reason, which the console also prints. `tidy()` gives one row per effect and estimand, with the limits in `ci_lower` and `ci_upper` (see `dmar_tidiers`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bakeman, R. (2005). Recommended effect size statistics for repeated measures designs. *Behavior Research Methods, 37*(3), 379–384. doi:10.3758/BF03192707

Bargmann, R. E. (1970). Interpretation and use of a generalized discriminant function. In R. C. Bose, I. M. Chakravarti, P. C. Mahalanobis, C. R. Rao, & K. J. C. Smith (Eds.), *Essays in probability and statistics* (pp. 35–60). University of North Carolina Press.

Fidler, F., & Thompson, B. (2001). Computing correct confidence intervals for ANOVA fixed- and random-effects effect sizes. *Educational and Psychological Measurement, 61*(4), 575–604. doi:10.1177/0013164401614003

Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40*(3), 659–670. doi:10.1177/001316448004000309

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*, 137–152. doi:10.1037/a0028086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on η^2, Chapter 7 on factorial designs, and Chapter 11 on generalized η^2 for within-subjects designs.)

Smithson, M. (2001). Correct confidence intervals for various regression effect sizes and parameters: The importance of noncentral distributions in computing intervals. *Educational and Psychological Measurement, 61*, 605–632. doi:10.1177/00131640121971392

Smithson, M. (2003). *Confidence intervals*. Thousand Oaks, CA: Sage Publications. doi:10.4135/9781412983761

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Venables, W. (1975). Calculation of confidence intervals for noncentrality parameters. *Journal of the Royal Statistical Society, Series B, 37*(3), 406–412. doi:10.1111/j.25176161.1975.tb01554.x

See Also

`eta_squared`, `ci_omega_squared`, `ci_pvaf`, `ci_nc_F`, `ci_eta_squared_generalized`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
# 1. The F interface. Bargmann's (1970) example, used in turn by
#    Venables (1975), Fleishman (1980), and Steiger (2004): a one-way
#    ANOVA with five groups of 11 and an observed F of 11.221 on 4 and
#    50 df, at 90%. F determines the partial value, which in a one-way
#    design is the total.
ci_eta_squared(F_value = 11.221, df_effect = 4, df_error = 50, N = 55,
               estimand = "partial", conf_level = 0.90)

# 2. One-way ANOVA from a fitted model: mean IQ gain differs across
#    the six grades of the pygmalion data (N = 310). The total and the
```

```
#    partial value coincide, and so do their intervals.
fit_one <- aov(iq_gain ~ factor(grade), data = pygmalion)
ci_eta_squared(fit_one)

# 3. Two-factor ANOVA for the manipulated expectancy treatment and the
#    measured grade classification (pygmalion data, N = 310). The
#    treatment by grade interaction is weak here (F = 1.19), so the
#    additive model is used. The totals are reported without limits.
fit_additive <- aov(iq_8 ~ treatment + factor(grade), data = pygmalion)
ci_eta_squared(fit_additive)

# 4. Within-subjects ANOVA. The partial value's interval is computed
#    against the within-subjects error stratum, named in 'stratum'.
set.seed(113)
n <- 20
rm_data <- data.frame(
  subject = factor(rep(seq_len(n), each = 3)),
  time    = factor(rep(c("Pre", "Mid", "Post"), n),
                   levels = c("Pre", "Mid", "Post")),
  y       = rnorm(n, sd = 1.5)[rep(seq_len(n), each = 3)] +
            0.7 * rep(1:3, n) + rnorm(n * 3, sd = 1.2)
)
fit_rm <- aov(y ~ time + Error(subject/time), data = rm_data)
ci_eta_squared(fit_rm, estimand = "partial")

# 5. The size with which the limits are converted. The interaction of
#    the 2 by 7 design with four observations per cell analyzed by
#    Steiger (2004, p. 171), F = 2.50 on 6 and 42 df, at 90%: the
#    total sample size of 56 gives the interval for the population
#    partial eta squared, and 6 + 42 + 1 = 49 gives the limits that
#    Smithson (2001) reports for the same F.
ci_eta_squared(F_value = 2.50, df_effect = 6, df_error = 42, N = 56,
               estimand = "partial", conf_level = 0.90)
ci_eta_squared(F_value = 2.50, df_effect = 6, df_error = 42, N = 49,
               estimand = "partial", conf_level = 0.90)

# 6. A within-subjects F from a report: the three IQ occasions of the
#    pygmalion data, F(2, 618) = 83.11 for 310 children. N is the
#    620 degrees of freedom of the occasion stratum, not 930 rows.
ci_eta_squared(F_value = 83.11, df_effect = 2, df_error = 618, N = 620,
               estimand = "partial")
```

`ci_eta_squared_generalized` *Confidence Interval for Generalized Eta Squared (Approximate)*

Description

Returns the point estimate of generalized eta squared (η^2_G; Olejnik & Algina, 2003) along with an *optional* confidence interval computed by one of two approximate methods. The default is to

return only the point estimate (`method = "none"`), because both available CI methods are approximations whose coverage properties have not been broadly validated for this estimand and warrant independent evaluation before being used in substantive inference.

Usage

```
ci_eta_squared_generalized(
  object = NULL,
  observed = NULL,
  SS_effect = NULL,
  SS_observed = NULL,
  SS_error = NULL,
  F_effect = NULL,
  df_effect = NULL,
  F_observed = NULL,
  df_observed = NULL,
  df_error = NULL,
  N = NULL,
  method = c("none", "parametric", "bootstrap"),
  B = 10000L,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  seed = NULL
)
```

Arguments

`object`	Optional. A fitted model object of class `aov`, `lm`, or `aovlist` (multi-stratum aov fit for within-subjects / mixed designs).
`observed`	Character vector of factor names treated as measured.
`SS_effect`, `SS_observed`, `SS_error`	Sums of squares (option 2 in `eta_squared_generalized`): `SS_effect` for the focal effect, `SS_error` for its error term, and `SS_observed` (a scalar or a vector) for every other source in the denominator of η^2_G. The estimate depends on `SS_observed` and `SS_error` only through their sum. The parametric interval is built on the F they imply, $(SS_{\text{effect}}/df_{\text{effect}})/(SS_{\text{error}}/df_{\text{error}})$, which is the F of the effect's test only when `SS_error` is the sum of squares that test divides by: the residual, adjusted for any covariate, in a design without an `Error()` term, and in a design with one the residual of the stratum in which the effect is tested (subjects within groups for an effect tested between subjects, and in a one-way within-subjects design the interaction of subjects and the within-subjects factor). `SS_observed` then holds the other sources that Olejnik and Algina (2003, Eq. 5, p. 437) place in the denominator: the measured factors, the interactions that contain one, the covariates, and the residuals of the other strata, such as that of the subjects stratum of a within-subjects design. The interval also needs `df_effect`, `df_error`, and `N`.

F_effect, df_effect, F_observed, df_observed, df_error
: F-values and degrees of freedom (option 3 in eta_squared_generalized). Each F divides a mean square by the focal effect's error mean square: F_effect is the effect's test, and F_observed covers the sources that SS_observed holds, so for a source in another stratum it is not the F that summary() of the fit reports. df_effect and df_error go with the sums of squares as well. Given with the sums of squares, F_effect, F_observed, and df_observed must describe the same test: the F the sums of squares imply for the effect, and the ratio of SS_observed to SS_error, must agree with the F interface's to a relative tolerance of 1e-6, the tolerance eta_squared_generalized applies to the two estimates, and the interval is then formed from the F-values. The two can agree on the estimate and still imply different F statistics for the effect, and so different intervals; such a call is refused.

N
: Total sample size, with which the confidence limits for the noncentrality parameter are converted (see Details). Required when method = "parametric" and no fitted model is supplied; ignored when method = "none" or when a fitted model is supplied, in which case a single-stratum fit is converted with nobs(object) and each effect of an aovlist fit with the size of its own stratum, as in ci_eta_squared. For an effect tested in an Error() stratum, give the size of that stratum instead: the number of subjects for an effect tested between subjects, and for an effect tested within subjects the stratum's total degrees of freedom, which is df_effect + df_error when the effect is the only one tested in its stratum. N must be finite and at least df_effect + df_error.

method
: One of "none" (default), "parametric", or "bootstrap". See Details.

B
: Number of bootstrap replications when method = "bootstrap", a single whole number. Default 10000. The fewest accepted is 1000 through the 98 percent level (this function's own floor) and 2000 at the 99 percent level, where 10 replications beyond each limit take that many; a smaller B is refused before anything is computed (class dmar_count_invalid). A reported interval deserves 10000 or more. See *Bootstrap replications* under *Details* and dmar_bootstrap.

conf_level
: Desired confidence coverage; default 0.95.

alpha_lower, alpha_upper
: Optional Type I error on the lower and upper side. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound.

seed
: Optional integer seed for the bootstrap, for reproducibility. Used locally: the caller's random number generator state is restored on exit. Default NULL draws the resamples from the current random number stream, which advances that stream.

Details

Why CI = "none" is the default. Confidence interval construction for η^2_G is not as settled as for partial η^2 or ω^2, because the denominator mixes sums of squares from heterogeneous sources (the focal effect, one or more measured factors, and the error term). No noncentral F transformation

maps the population noncentrality parameter directly to η^2_G. Both methods below are approximations and are exposed for exploration rather than as defaults.

`method = "parametric"`**.** The function first obtains a confidence interval for the population noncentrality parameter λ of the focal effect's F-test via `ci_nc_F` and maps its limits through the partial η^2 transformation $\eta^2_{p,\text{bound}} = \lambda_{\text{bound}}/(\lambda_{\text{bound}} + N)$, as `ci_eta_squared` converts them: N is the number of observations for a fit without an `Error()` term (the convention of `ci_pvaf` and `ci_omega_squared` as well), the size of the effect's stratum for an effect of an `aovlist` fit (see below), and `N` as supplied for the raw interfaces. The limits are then re-expressed as η^2_G bounds via

$$\eta^2_{G,\text{bound}} = \frac{r_{\text{bound}}}{r_{\text{bound}} + r_{\text{obs}} + 1},$$

where $r_{\text{bound}} = \eta^2_{p,\text{bound}}/(1 - \eta^2_{p,\text{bound}}) = \lambda_{\text{bound}}/N$ and r_{obs} is the sum of the other sums of squares in the denominator of η^2_G (those of the measured sources and, for an `aovlist` fit, the residuals of the other strata) divided by the focal effect's error sum of squares. The point estimate is the same map applied to $SS_{\text{effect}}/SS_{\text{error}} = F \cdot df_{\text{effect}}/df_{\text{error}}$, which for a fitted model converges to λ/N as the sample grows. When r_{obs} is 0, as in a fit without an `Error()` term in which nothing measured enters the denominator, η^2_G is the partial η^2, and the interval is the exact one `ci_eta_squared(estimand = "partial")` gives. Otherwise the interval is not exact, and no conversion of the limits for λ can make it so: η^2_G then depends on the value that r_{obs} converges to as well as on λ, while the distribution of the focal F depends on λ alone, so the transformation principle behind the exact intervals (Steiger, 2004, Proposition 1, p. 167), which maps the limits through a function of that one parameter, does not apply. The interval takes r_{obs} as known and equal to its sample value, which leaves the sampling variability of r_{obs} out of it. Its coverage has not been validated, and it should be treated as preliminary.

`method = "bootstrap"`**.** A residual bootstrap from the fitted model: the resampling unit is a residual, drawn nonparametrically from the model's own residuals rather than from a fitted distribution. For each of the `B` replications, the response is regenerated as $\hat{y}_i + \varepsilon^*_i$ where ε^* is sampled with replacement from the model's residuals; the model is refit; η^2_G is recomputed; and the percentile interval, the empirical quantiles of the `B` bootstrap values (Efron & Tibshirani, 1993), is reported. The percentile interval is the only bootstrap interval offered; there is no bias-corrected and accelerated (BCa) variant. Bootstrap results vary from run to run; supply `seed` for reproducibility. Requires a single-stratum `aov`/`lm` fit. **Not yet supported for** `aovlist` **(multi-stratum / within-subjects) fits**, since the bootstrap needs to respect the subject-level correlation structure, which naive residual resampling does not. Use `method = "parametric"` for `aovlist` fits. Coverage has not been broadly validated for this estimand.

Bootstrap replications. The limits are quantiles of the `B` replicated values of each effect, read at `alpha_lower` and `1 - alpha_upper`. The function requires at least 1000 replications, more than the 400 that 10 replications beyond each limit take at the 95 percent level; at the 99 percent level that count is 2000 and governs. A replication in which η^2_G for some effect is undefined (the refit fails, or the effect size cannot be computed from it) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls.

When more than half were dropped, the limits of every effect are NA, the estimate is returned, and the warning has class dmar_bootstrap_insufficient. With several effects, B_used is the fewest replications the limits of any effect rest on. options(dmar.bootstrap_exploratory = TRUE) lets a smaller B (2 or more) run for a first look or a demonstration; every such call gives a warning of class dmar_bootstrap_exploratory and the result is not one to report. See dmar_bootstrap for the rule and its reasons.

Within-subjects designs (aovlist). Each focal effect's parametric interval is built from its own stratum's *F* test, and its limits for λ are converted with the size of that stratum, as ci_eta_squared converts them: the stratum's total degrees of freedom for an effect tested within subjects, and the number of subjects for an effect in the between-subjects stratum of a mixed design. r_{obs} includes the residual sums of squares of the other strata, since every source of variance that involves subjects belongs to the denominator of η^2_G (Olejnik & Algina, 2003, Eq. 5, p. 437; Bakeman, 2005, p. 381). With r_{obs} at the value it converges to, the conversion maps each limit for λ to the corresponding value of the population η^2_G of Olejnik and Algina (2003, Eq. 2, p. 436), the value the estimate converges to as subjects are added. In a one-way within-subjects design with n subjects, p occasions, and compound symmetry, for which Steiger (2004, Eq. 58, p. 177) gives $\lambda = n\sum_j(\mu_j - \mu)^2/[\sigma^2(1-\rho)]$, that value is $\lambda(1-\rho)/[\lambda(1-\rho)+np]$: the variance of the occasion means, $\sum_j(\mu_j-\mu)^2/p$, over the sum of that variance and σ^2, which holds the subject, subject by occasion, and error variances that Olejnik and Algina (2003, p. 443, note 1) place in the denominator. Given λ, that value still depends on ρ, and the distribution of the occasions' F, which depends on λ alone, carries no further information about ρ. Steiger (2004, pp. 176–177) notes that in repeated measures designs λ confounds the effect with the correlation among the observations, and that the root-mean-square standardized effect of such a design cannot be estimated directly with the exact noncentral methods unless ρ is known. Converted with the number of rows, np, the limits would be for a smaller quantity than the one the estimate estimates: for the three IQ occasions of pygmalion analyzed in vignette("DMAR", package = "DMAR"), 930 rows against $F(2, 618)$ give the interval [.019, .037], below the estimate of .041; converted with 2 + 618 = 620, the interval is [.029, .054]. As eta_squared_generalized does, the function stops when an effect is tested in more than one stratum, as happens when observations are missing within subjects or a covariate varies within them, since no one stratum's sum of squares is then the effect's.

Value

A data.frame with one row per focal effect and the columns effect, eta_squared_generalized, ci_lower, ci_upper, and method. For aovlist fits a stratum column is also present. When method = "none", the limit columns contain NA. Otherwise the table records the confidence level of its intervals in its conf_level attribute: conf_level, or 1 - alpha_lower - alpha_upper when the tail areas are supplied. With method = "bootstrap" the attributes B, B_used, n_dropped, B_required, and bootstrap_status record the replications (see *Bootstrap replications* under Details).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J., Keselman, H. J., & Penfield, R. D. (2005). An alternative to Cohen's standardized mean difference effect size: A robust parameter and confidence interval in the two independent groups

case. *Psychological Methods, 10*(3), 317–328. doi:10.1037/1082989X.10.3.317

Bakeman, R. (2005). Recommended effect size statistics for repeated measures designs. *Behavior Research Methods, 37*(3), 379–384. doi:10.3758/BF03192707

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*, 137–152. doi:10.1037/a0028086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on η^2, Chapter 7 on factorial designs, and Chapter 11 on generalized η^2 for within-subjects designs.)

Olejnik, S., & Algina, J. (2003). Generalized eta and omega squared statistics: Measures of effect size for some common research designs. *Psychological Methods, 8*(4), 434–447. doi:10.1037/1082989X.8.4.434

Smithson, M. (2001). Correct confidence intervals for various regression effect sizes and parameters: The importance of noncentral distributions in computing intervals. *Educational and Psychological Measurement, 61*, 605–632. doi:10.1177/00131640121971392

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

See Also

`eta_squared_generalized`, `ci_eta_squared`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
# The pygmalion expectancy experiment: treatment is manipulated, while
# grade is a measured classification, so grade belongs in the denominator.
pyg <- pygmalion
pyg$grade <- factor(pyg$grade)
fit <- aov(iq_8 ~ treatment * grade, data = pyg)

# The default returns the point estimate and leaves the limits NA, because
# both interval methods are approximations.
ci_eta_squared_generalized(fit, observed = "grade")

# The parametric approximation maps a noncentral F interval for the focal
# effect through the observed sums of squares. Warnings mark the method as
# preliminary and report that the interaction's lower limit is clamped at
# 0; treat the limits accordingly.
```

```
ci_eta_squared_generalized(fit, observed = "grade", method = "parametric")

# The third option is a residual bootstrap, which refits the model once per
# replication. B = 100 keeps the example quick and is below the 1000 the
# function requires; the option lets it run and the warning says so. A
# reported interval deserves B = 10000 or more. The seed makes the
# interval reproducible and leaves the random number stream of the
# surrounding session as it was.
old <- options(dmar.bootstrap_exploratory = TRUE)
ci_eta_squared_generalized(fit, observed = "grade",
                           method = "bootstrap", B = 100, seed = 113)
options(old)

# The parametric treatment row from summary statistics, as an ANOVA table
# reports them: the sums of squares of treatment, of its error term (the
# residual), and of the measured sources in its denominator (grade and
# the treatment by grade interaction), with the degrees of freedom and
# the number of children for the interval.
tab <- anova(fit)
ci_eta_squared_generalized(
  SS_effect   = tab["treatment", "Sum Sq"],
  SS_observed = tab[c("grade", "treatment:grade"), "Sum Sq"],
  SS_error    = tab["Residuals", "Sum Sq"],
  df_effect   = tab["treatment", "Df"], df_error = tab["Residuals", "Df"],
  N = nobs(fit), method = "parametric")

# Within-subjects ANOVA. The parametric CI uses each effect's own stratum
# and converts its limits with that stratum's size.
set.seed(113)
n <- 20
rm_data <- data.frame(
  subject = factor(rep(seq_len(n), each = 3)),
  time    = factor(rep(c("Pre", "Mid", "Post"), n),
                   levels = c("Pre", "Mid", "Post")),
  y       = rnorm(n, sd = 1.5)[rep(seq_len(n), each = 3)] +
            0.7 * rep(1:3, n) + rnorm(n * 3, sd = 1.2)
)
fit_rm <- aov(y ~ time + Error(subject/time), data = rm_data)
ci_eta_squared_generalized(fit_rm, method = "parametric")
```

`ci_games_howell` *Provides Games–Howell Simultaneous Confidence Intervals for All Pairwise Comparisons Without Assuming Homogeneity of Variance*

Description

Provides Games–Howell Simultaneous Confidence Intervals for All Pairwise Comparisons Without Assuming Homogeneity of Variance

Usage

```
ci_games_howell(x, group = NULL, conf_level = 0.95)
```

Arguments

x	Either (a) a fitted lm or aov object with a single factor predictor, or (b) a numeric vector of the outcome, in which case group must also be supplied.
group	When x is a vector, a factor (or coercible to factor) giving the group membership of each observation.
conf_level	Family-wise confidence level. Default 0.95.

Details

Tukey's HSD and the Kramer modification for unequal *n* (ci_tukey_kramer) both pool the within-group variances into MS_W, so both assume homogeneity of variance. Neither is robust when that assumption fails. The Games–Howell procedure drops the assumption: it uses a separate error term for each pair and a Welch–Satterthwaite degrees of freedom for each pair, then takes its critical value from the studentized range.

For groups g and h, the standard error of the difference uses only those two groups' variances, and the degrees of freedom are

$$df = \frac{(s_g^2/n_g + s_h^2/n_h)^2}{s_g^4/[n_g^2(n_g-1)] + s_h^4/[n_h^2(n_h-1)]},$$

the same Welch–Satterthwaite expression that base R's t.test uses by default for two groups. A pair is declared different when the observed t exceeds $q/\sqrt{2}$, with q the studentized range critical value (cv_tukey_hsd) at the pair's degrees of freedom, so the interval for the difference of means is $(\bar{Y}_g - \bar{Y}_h) \pm q_{\alpha;a,df}\sqrt{(s_g^2/n_g + s_h^2/n_h)/2}$. Maxwell, Delaney, and Kelley (2027, Chapter 5) develop this as one of the two modifications of Tukey's HSD for heterogeneous variances (their Equations 5.13 and 5.14). The adjusted *p*-value is the probability that the studentized range with a means and the pair's degrees of freedom exceeds the observed $|q|$.

How the studentized range is computed. The per-pair degrees of freedom are often small and are not whole numbers, and base R's qtukey and ptukey are not accurate at small degrees of freedom (with 20 groups and 2 df, qtukey(.99, 20, 2) is 44.526 where the exact value is 37.943) and return NaN below 2 df, which a pair of small groups with unequal variances can produce. Where they are not accurate (fewer than 9 df for up to 6 groups, rising by steps to fewer than 111 with more than 3000 groups; from 10000 to 10^8 df; confidence levels above .999; with more than 50 groups, below the .75 point of the distribution (.80 with more than 1000 groups); and wherever qtukey stops short of its root), the quantile and the *p*-value come from the exact distribution, computed by integrating the probability integral of the range against the chi square distribution of the error variance; elsewhere they are qtukey and ptukey. cv_tukey_hsd describes the computation, the regions, and how they were measured.

When to use it. Reach for Games–Howell when the group variances are not interchangeable and the design is between subjects. It is the heterogeneity-robust counterpart of ci_tukey_kramer and, like it, controls the family-wise error rate across all $a(a-1)/2$ pairs. It handles unequal *n* as a matter of course, so it does not need a separate unequal-*n* variant.

When something else is better. Dunnett (1980) found that Games–Howell becomes slightly liberal (the family-wise error rate runs somewhat above the nominal level) when the samples are small. Maxwell, Delaney, and Kelley (2027, Chapter 5) therefore recommend Games–Howell for larger samples and Dunnett's T3 (`ci_dunnett_t3`), which takes its critical value from the studentized maximum modulus (`cv_smm`) rather than the studentized range, when the groups have fewer than roughly 50 observations each. When the variances are in fact homogeneous, use `ci_tukey_kramer` instead: it pools the variances, so it has more error degrees of freedom and more power. When only treatments are compared to a single control, use `ci_dunnett`.

With $a = 2$ groups the procedure is exactly Welch's t test: the interval and the p-value equal those from `t.test(..., var.equal = FALSE)`, because $q_{\alpha;2,df} = \sqrt{2}\, t_{1-\alpha/2,df}$.

Value

A `data.frame` with one row per pairwise comparison and columns `contrast`, `mean_difference`, `se`, `df`, `q_statistic`, `ci_lower`, `ci_upper`, and `p_adjusted`. The `df` column is the Welch–Satterthwaite degrees of freedom for that pair, which is why it varies from row to row. The table prints through the `dmar_tbl` display layer and works with `tidy` and `glance` (see `dmar_tidiers`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Games, P. A., & Howell, J. F. (1976). Pairwise multiple comparison procedures with unequal *n*'s and/or variances: A Monte Carlo study. *Journal of Educational Statistics, 1*(2), 113–125. doi:10.2307/1164979

Dunnett, C. W. (1980). Pairwise multiple comparisons in the unequal variance case. *Journal of the American Statistical Association, 75*(372), 796–800. doi:10.2307/2287161

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 5 on the multiple-comparisons problem, where the modifications of Tukey's HSD for unequal *n* and unequal variances are developed.)

See Also

`ci_tukey_kramer` for the homogeneity-assuming counterpart, `ci_dunnett_t3` for Dunnett's T3, the studentized maximum modulus counterpart for smaller samples, `ci_dunnett` for many-to-one comparisons, `ci_scheffe` for arbitrary contrasts, `cv_tukey_hsd` for the critical value it uses, and `dmar_tidiers` for the tidy methods.

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# The raw drinks_per_week outcome of the drinks_trial data: the
# Standard arm's variance is more than three times that of either CRA
# arm, so the pooled error term Tukey's HSD relies on is questionable.
ci_games_howell(drinks_trial$drinks_per_week, drinks_trial$treatment)

# A fitted one-way model may be passed instead of the two vectors.
ci_games_howell(aov(drinks_per_week ~ treatment, data = drinks_trial))

# With two groups the procedure is Welch's t test, so the limits agree;
# the trial's two enrollment cohorts give a two-group comparison.
ci_games_howell(drinks_trial$drinks_per_week, drinks_trial$cohort)$ci_lower
-t.test(drinks_per_week ~ cohort, data = drinks_trial)$conf.int[2]
```

`ci_mahalanobis` *Confidence Interval for the Squared Mahalanobis Distance*

Description

Computes the squared Mahalanobis distance D^2 together with an exact confidence interval for the population squared distance Δ^2, obtained by inverting Hotelling's T^2 statistic through its (noncentral) F-distribution as in Reiser (2001). Both the one-sample setting (mean vector against a hypothesized population mean) and the two-sample setting (between-groups distance from discriminant analysis or multivariate group comparison) are supported, and either raw data or a pre-computed D^2 with sample sizes can be supplied.

Usage

```
ci_mahalanobis(
  D2 = NULL,
  group_1 = NULL,
  group_2 = NULL,
  null_mean = NULL,
  n_1 = NULL,
  n_2 = NULL,
  p = NULL,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  ...
)
```

Arguments

`D2`	Optional pre-computed squared Mahalanobis distance. Ignored if `group_1` is supplied.

`group_1` Optional numeric matrix or data frame for the first sample ($n_1 \times p$). Rows are observations and columns are variables.

`group_2` Optional numeric matrix or data frame for the second sample ($n_2 \times p$). When `NULL`, the function operates in one-sample mode against `null_mean`.

`null_mean` Optional hypothesized population mean for the one-sample case (length-p numeric vector), the mean vector of the null hypothesis the sample centroid is compared with, $\boldsymbol{\mu}_0$ in the formulas of Details. Defaults to a vector of zeros.

`n_1` Sample size for group 1 (required when supplying `D2` directly).

`n_2` Sample size for group 2 (required for two-sample mode when supplying `D2` directly; leave `NULL` for one-sample mode).

`p` Dimensionality (number of variables) when supplying `D2` directly.

`conf_level` Confidence coverage for a symmetric interval (default `0.95`). Set it to `NULL` to specify the tails directly through `alpha_lower` and `alpha_upper`.

`alpha_lower, alpha_upper` Optional Type I error rates for the lower and upper tail. To use them, set `conf_level = NULL` and supply both (an asymmetric or one-sided interval with coverage `1 - alpha_lower - alpha_upper`); supplying either alongside a non-`NULL` `conf_level` is an error, as in `ci_nc_F`, to which they are passed. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound.

`...` Further control of the root search that `ci_nc_F` runs for the limits of the noncentrality parameter, given by name: `tol`, the convergence tolerance of `ci_nc_F`, and `maxiter`, `check.conv`, and `trace`, which it passes to `uniroot`. Any other name, and an argument without a name, is refused with a condition of class `dmar_argument_invalid` that names it, before anything is computed.

Details

Definition. For a p-vector $\mathbf{x}$ drawn from a multivariate normal with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, Mahalanobis's (1936) squared distance from a reference vector $\boldsymbol{\mu}_0$ (the argument `null_mean` in the one-sample case) is

$$\Delta^2 = (\boldsymbol{\mu} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{\mu}_0).$$

In the two-sample case the population distance between groups is $\Delta^2 = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$, assuming a common covariance. The corresponding sample estimates plug the sample means and the sample (or pooled) covariance into the same quadratic form.

Link to Hotelling's T^2. Hotelling's (1931) T^2 statistic is $T^2 = nD^2$ (one sample) or $T^2 = \{n_1 n_2/(n_1+n_2)\} D^2$ (two samples). Under multivariate normality

$$\frac{n_1 + n_2 - p - 1}{(n_1 + n_2 - 2)\, p}\, T^2 \sim F'\!\left(p,\, n_1 + n_2 - p - 1,\, \lambda = \frac{n_1 n_2}{n_1 + n_2}\, \Delta^2\right)$$

in the two-sample case, and analogously $\{(n-p)/[(n-1)p]\}\, T^2 \sim F'(p, n-p, n\Delta^2)$ in the one-sample case (see Anderson, 2003, Section 5.2).

Confidence interval. The CI on Δ^2 is obtained by inverting these distributional results (Reiser, 2001): a CI on the noncentrality parameter λ is constructed via `ci_nc_F` and then mapped back to Δ^2 by $\Delta^2 = \lambda\,(n_1 + n_2)/(n_1 n_2)$ (two sample) or $\Delta^2 = \lambda/n$ (one sample). When the observed F is below the lower-tail critical value of the central F-distribution at the requested confidence level, the lower CI on λ (and hence on Δ^2) is clamped to zero, in keeping with the `ci_nc_F` convention.

Bias. The plug-in estimator D^2 is upward biased for Δ^2; the CI from this function is exact for Δ^2 under multivariate normality and reflects the bias structure correctly, but the point estimate reported is the standard plug-in D^2.

Value

A one-row `data.frame` with columns `sample_type` (`"one-sample"` or `"two-sample"`), `D2` (point estimate of the squared distance), `ci_lower` and `ci_upper` (the confidence limits on the population squared distance Δ^2), `F_value`, `df_1`, `df_2`, `n_1`, `n_2` (`NA` in one-sample mode), and `p`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Anderson, T. W. (2003). *An introduction to multivariate statistical analysis* (3rd ed.). Wiley.

Hotelling, H. (1931). The generalization of Student's ratio. *The Annals of Mathematical Statistics, 2*(3), 360–378.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India, 2*(1), 49–55.

Reiser, B. (2001). Confidence intervals for the Mahalanobis distance. *Communications in Statistics–Simulation and Computation, 30*(1), 37–45. doi:10.1081/SAC100001856

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

See Also

`ci_nc_F`, `ci_smd`, `ci_R2`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
# Two-sample distance between the two schools of the holzinger_swineford
#     data on a four-test cognitive battery.
battery <- c("t1_visual_perception", "t2_cubes", "t4_lozenges",
```

```
            "t6_paragraph_comprehension")
g1 <- as.matrix(holzinger_swineford[
  holzinger_swineford$school == "Grant-White", battery])
g2 <- as.matrix(holzinger_swineford[
  holzinger_swineford$school == "Pasteur", battery])
ci_mahalanobis(group_1 = g1, group_2 = g2)

# One-sample distance: how far is the Grant-White centroid from a
#     reference vector of (29, 24, 18, 9)?
ci_mahalanobis(group_1 = g1, null_mean = c(29, 24, 18, 9))

# Pre-computed D^2 (no raw data needed): the two-school distance,
#     reproduced from reported summaries.
ci_mahalanobis(D2 = 0.608, n_1 = 145, n_2 = 156, p = 4)
```

`ci_nc_chisq` *Confidence Interval for the Noncentrality Parameter of a Noncentral Chi Square Distribution*

Description

Finds the noncentrality parameters of a noncentral chi square distribution that bracket an observed chi square value with the requested tail probabilities, giving a confidence interval on the population noncentrality parameter. Together with `ci_nc_t` and `ci_nc_F`, this is one of the low-level noncentral distribution workhorses on which the `ci_*` confidence interval functions are built; most analyses reach it through those functions rather than calling it directly. The function corresponds to `conf.limits.nc.chisq()` in MBESS; its name places it in the `ci_*` family because a confidence interval is what it computes.

Usage

```
ci_nc_chisq(
  chisq = NULL,
  conf_level = 0.95,
  df = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  tol = 1e-09,
  verbose = TRUE,
  ...
)
```

Arguments

`chisq`	The observed chi square value
`conf_level`	The desired degree of confidence for a symmetric interval
`df`	The degrees of freedom

alpha_lower	The proportion of values beyond the lower limit (cannot be used with conf_level)
alpha_upper	The proportion of values beyond the upper limit (cannot be used with conf_level)
tol	The convergence tolerance passed to uniroot
verbose	If TRUE (the default), the returned data frame additionally reports the achieved tail probabilities at each limit; if FALSE, only term and value are returned
...	Additional arguments forwarded to uniroot

Details

Each confidence limit is the noncentrality parameter $\lambda \geq 0$ of a noncentral chi square distribution with df degrees of freedom whose appropriate tail at the observed chisq contains the requested probability:

- the lower limit satisfies $P(X \geq \texttt{chisq}) = \texttt{alpha_lower}$;
- the upper limit satisfies $P(X \leq \texttt{chisq}) = \texttt{alpha_upper}$.

The two conditions run in opposite directions in λ: the lower-tail probability $P(X \leq \texttt{chisq})$ is continuous and strictly decreasing in the noncentrality parameter, so the upper-tail probability $P(X \geq \texttt{chisq})$ is continuous and strictly increasing in it. The lower limit is the λ at which the upper tail has grown to alpha_lower, and the upper limit is the λ at which the lower tail has shrunk to alpha_upper. Each is therefore the unique non-negative root of a one-dimensional equation, and both are located with uniroot on the decreasing lower-tail scale; extendInt is used to widen the search bracket if needed.

Because the noncentrality parameter is bounded below by zero, the lower limit is set to zero whenever the observed chisq is smaller than the alpha_lower critical value of the central chi square distribution (i.e., the data is consistent with $\lambda = 0$ at the requested confidence level). A warning is issued in that case, and the achieved probabilities reported in the output reflect the actual values at $\lambda = 0$.

Symmetrically, when the observed chisq is so small that even at $\lambda = 0$ the lower-tail probability is already at or below alpha_upper, no $\lambda \geq 0$ places as much as alpha_upper mass at or below chisq; the upper limit is undefined and is returned as NA, with a warning.

For a noncentrality above 10^5, the distribution function is summed directly as the Poisson mixture of central chi square distributions it is, $P(X \leq x) = \sum_j \pi_j(\lambda/2) P(\chi^2_{df+2j} \leq x)$, where $\pi_j(\mu)$ is the Poisson probability of j with mean μ, over the terms that carry all but 10^{-17} of the Poisson mass in each tail. R's pchisq() loses accuracy above about 10^5, as its help page warns, and above about 2×10^6 its series stops before it reaches the bulk of the Poisson weights, which once collapsed both limits to a single value below the statistic. Such a noncentrality arises only from a statistic in the millions (a large model fitted to a very large sample); at and below 10^5 the limits are those of pchisq().

Value

A data.frame with one row per confidence limit and the columns:

term	Either "ci_lower" or "ci_upper".

value	The noncentrality parameter at that limit. 0 when alpha_lower = 0 or the lower limit is unattainable; Inf when alpha_upper = 0; NA when the observed chisq is so small that even at $\lambda = 0$ the lower-tail probability is already at or below alpha_upper, so the upper limit is undefined (a warning is issued).
prob_less	(verbose = TRUE) The probability $P(X \leq \texttt{chisq})$ that an observation from the noncentral chi square distribution centered at the row's limit falls at or below the observed chisq.
prob_greater	(verbose = TRUE) The complementary probability $P(X \geq \texttt{chisq})$. By construction this equals alpha_lower on the ci_lower row and $1 - \texttt{alpha_upper}$ on the ci_upper row.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

See Also

ci_nc_t, ci_nc_F, stats::pchisq(), stats::qchisq(), uniroot

Other noncentral distribution confidence intervals: ci_nc_F(), ci_nc_t()

Examples

```
# A typical call to the function.
ci_nc_chisq(chisq = 30, conf_level = .95, df = 15)

# A one-sided (upper) confidence interval.
ci_nc_chisq(chisq = 30, alpha_lower = 0, alpha_upper = .05,
                    conf_level = NULL, df = 15)
```

---

ci_nc_F *Confidence Interval for the Noncentrality Parameter of a Noncentral* F *Distribution*

Description

Finds the noncentrality parameters of a noncentral F-distribution that bracket an observed F-value with the requested tail probabilities, giving a confidence interval on the population noncentrality parameter. Together with ci_nc_t and ci_nc_chisq, this is one of the low-level noncentral distribution workhorses on which the ci_* confidence interval functions (e.g., ci_pvaf, ci_snr, ci_R2) are built; most analyses reach it through those functions rather than calling it directly. The function corresponds to conf.limits.ncf() in MBESS; its name places it in the ci_* family because a confidence interval is what it computes.

Usage

```
ci_nc_F(
  F_value = NULL,
  conf_level = 0.95,
  df_1 = NULL,
  df_2 = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  tol = 1e-09,
  verbose = TRUE,
  ...
)
```

Arguments

F_value	The observed F-value
conf_level	The desired degree of confidence for a symmetric interval
df_1	The numerator degrees of freedom
df_2	The denominator degrees of freedom
alpha_lower	The proportion of values beyond the lower limit (cannot be used with conf_level)
alpha_upper	The proportion of values beyond the upper limit (cannot be used with conf_level)
tol	The convergence tolerance passed to uniroot
verbose	If TRUE (the default), the returned data frame additionally reports the achieved tail probabilities at each limit; if FALSE, only term and value are returned
...	Additional arguments forwarded to uniroot

Details

Each confidence limit is the noncentrality parameter $\lambda \geq 0$ of a noncentral F-distribution with df_1 and df_2 degrees of freedom whose appropriate tail at the observed F_value contains the requested probability:

- the lower limit satisfies $P(F \geq \texttt{F_value}) = \texttt{alpha_lower}$;
- the upper limit satisfies $P(F \leq \texttt{F_value}) = \texttt{alpha_upper}$.

The two conditions run in opposite directions in λ: the lower-tail probability $P(F \leq \texttt{F_value})$ is continuous and strictly decreasing in the noncentrality parameter, so the upper-tail probability $P(F \geq \texttt{F_value})$ is continuous and strictly increasing in it. The lower limit is the λ at which the upper tail has grown to `alpha_lower`, and the upper limit is the λ at which the lower tail has shrunk to `alpha_upper`. Each is therefore the unique non-negative root of a one-dimensional equation, and both are located with `uniroot` on the decreasing lower-tail scale; `extendInt` is used to widen the search bracket if needed.

Because the noncentrality parameter is bounded below by zero, the lower limit is set to zero whenever the observed `F_value` is smaller than the `alpha_lower` critical value of the central F-distribution (i.e., the data is consistent with $\lambda = 0$ at the requested confidence level). A warning is issued in that case, and the achieved probabilities reported in the output reflect the actual values at $\lambda = 0$ rather than the requested `alpha_lower`. The warning carries the condition class `dmar_nc_F_clamp`, so a caller that inverts the noncentral F repeatedly can muffle or deduplicate it by class.

Value

A `data.frame` with one row per confidence limit and the columns:

`term`	Either `"ci_lower"` or `"ci_upper"`.
`value`	The noncentrality parameter at that limit. `0` when `alpha_lower = 0` or the lower limit is unattainable; `Inf` when `alpha_upper = 0`; `NA` on the `ci_upper` row when the observed `F_value` is so small that even at $\lambda = 0$ the lower-tail probability is already at or below `alpha_upper`, leaving the upper noncentrality limit undefined (a warning is issued).
`prob_less`	(`verbose = TRUE`) The probability $P(F \leq \texttt{F_value})$ that an F-statistic from the noncentral F-distribution centered at the row's limit falls at or below the observed `F_value`.
`prob_greater`	(`verbose = TRUE`) The complementary probability $P(F \geq \texttt{F_value})$. By construction this equals `alpha_lower` on the `ci_lower` row and $1 -$ `alpha_upper` on the `ci_upper` row.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

See Also

`ss_aipe_R2`, `ci_R2`, `ci_nc_t`, `ci_nc_chisq`, `stats::pf()`, `stats::qf()`, `uniroot`

Other noncentral distribution confidence intervals: `ci_nc_chisq()`, `ci_nc_t()`

Examples

```
ci_nc_F(F_value = 5, conf_level = .95, df_1 = 5, df_2 = 100)

# A one-sided (upper) confidence interval.
ci_nc_F(F_value = 5, conf_level = NULL, df_1 = 5, df_2 = 100,
                alpha_lower = 0, alpha_upper = .05)
```

`ci_nc_t` *Confidence Interval for the Noncentrality Parameter of a Noncentral* t *Distribution*

Description

Finds the noncentrality parameters of a noncentral *t*-distribution that bracket an observed *t*-value with the requested tail probabilities, giving a confidence interval on the population noncentrality parameter. Together with `ci_nc_F` and `ci_nc_chisq`, this is one of the low-level noncentral distribution workhorses on which the `ci_*` confidence interval functions (e.g., `ci_smd`, `ci_smd_c`, `ci_cv`) are built; most analyses reach it through those functions rather than calling it directly. The function corresponds to `conf.limits.nct()` in MBESS; its name places it in the `ci_*` family because a confidence interval is what it computes.

Usage

```
ci_nc_t(
  ncp,
  df,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  tol = 1e-09,
  verbose = TRUE,
  ...
)
```

Arguments

`ncp`	The observed *t*-value, which is the estimate of the noncentrality parameter the interval is formed for
`df`	The degrees of freedom
`conf_level`	The level of confidence for a symmetric confidence interval. A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in Details)
`alpha_lower`	The proportion of values beyond the lower limit of the confidence interval (cannot be used with `conf_level`): 0, for an interval with no lower limit, or at least 1e-9 (see "Small tail areas" in Details)

alpha_upper	The proportion of values beyond the upper limit of the confidence interval (cannot be used with conf_level): 0, for an interval with no upper limit, or at least 1e-9
tol	The convergence tolerance passed to uniroot when locating each limit
verbose	If TRUE (the default), the returned data frame additionally reports the achieved tail probabilities at each limit; if FALSE, only term and value are returned
...	Additional arguments forwarded to uniroot

Details

Each confidence limit is the noncentrality parameter of a noncentral *t*-distribution with df degrees of freedom whose appropriate tail at the observed ncp contains the requested probability:

- the lower limit satisfies $P(T \geq \text{ncp}) =$ alpha_lower;
- the upper limit satisfies $P(T \leq \text{ncp}) =$ alpha_upper.

Each tail probability is continuous and strictly monotone in the noncentrality parameter, so each limit is the unique root of a one-dimensional equation. The roots are located with uniroot starting from a bracket centered on ncp with half-width scaled by the asymptotic standard error of the noncentrality estimator; extendInt is used to widen the bracket if needed.

Computation. R's pt() sums a series for the noncentral *t* distribution up to a noncentrality of 37.62 and with at most 400,000 degrees of freedom, and returns a normal approximation otherwise. Compared with the exact probability over a grid of degrees of freedom, noncentralities, and quantiles (from 10^{-6} to $1 - 10^{-6}$), it was within 10^{-9} up to a noncentrality of 37.62 with at most 2,000 degrees of freedom and up to 33 with more (the series stops short in the upper tail near the top of its range: the error was 10^{-6} at 34 and .001 at 36 with 100,000 degrees of freedom), while the normal approximation was off by as much as .14 with one degree of freedom, .008 with 38, and .001 with 1,000; below one degree of freedom it is not accurate at any noncentrality other than 0. Each limit is therefore found with pt() where the limit lies in that range (with at least one degree of freedom), and otherwise with the exact probability

$$P(T \leq t) = \int_0^\infty \Phi(ts - \lambda)\, f(s)\, ds,$$

where Φ is the standard normal distribution function, f the density of $S = \sqrt{V/\nu}$ for V chi square on ν degrees of freedom, and $T = (Z + \lambda)/S$ with Z standard normal, independent of V; the integral is computed numerically. The limits agreed with bisection on a second exact form, which conditions on the normal variate instead, to within 3.5×10^{-9} of the width of the interval in 400 random cases with 1 to 1,000,000 degrees of freedom. The tail probabilities in prob_less and prob_greater are computed the same way as the limit in their row. A limit outside the range takes about a millisecond; one inside it, a tenth of that.

Small tail areas. A nonzero tail area below 1e-9, or a conf_level above .999999998, is refused with an error of class dmar_argument_invalid whose reason is "tail_below_floor". A limit is the root of a tail probability, and pt() sums its series to an absolute tolerance of 10^{-12} and forms one tail as 1 minus the other, so the error of the probability becomes a larger part of the tail as the tail shrinks. Against the exact probability, the limits with equal tails at t = 5 on 47 degrees of freedom, .5 on 10, 2 on 200, 10 on 20, -3 on 30, and 1 on 1,000, at 66 tail areas from 10^{-6} down to 10^{-9}, were off by at most 1.7×10^{-5} at tail areas of 10^{-8} and above, 7.3×10^{-5} at 2×10^{-9} and

above, and 1.4×10^{-4} at 10^{-9} (the upper limit at t = 10 on 20 degrees of freedom, 6.0×10^{-6} of the width of the interval); at 10^{-10} the largest error was 1.1×10^{-3}, and at 10^{-20} the upper limit at t = 5 on 47 degrees of freedom was 7.6 percent too small. Every interval and plan whose limits come from this function keeps the floor, among them `ci_smd`, `ci_smd_c`, `ci_sm`, `ci_cv`, `ci_sc`, `ci_sc_ancova`, `ci_src`, and `ci_reg_coef` with `noncentral = TRUE`, and the planners of those intervals. The central intervals of a contrast, a correlation, and a regression coefficient (`ci_c`, `ci_r`, `ci_rc`, and `ci_reg_coef` with `noncentral = FALSE`) and their planners (`ss_aipe_c`, `ss_aipe_r`, and `ss_aipe_rc`) keep it too, so that an interval and the plan for it accept the same tail areas.

A limit past 37.62. Each limit is found by its own rule, so an interval can have its lower limit from `pt()` and its upper limit from the exact probability. The interval for the standardized mean difference of two groups of 20 at $d = 10$ with 99 percent confidence (`ci_smd`) is such an interval: it is [7.003, 13.099], 6.096 wide. In 100,000 samples of two groups of 20 with $\delta = 13$, this 99 percent interval fell below δ in .52 percent and above it in .48 percent (.50 percent each by design).

This function is especially useful for forming confidence intervals around standardized mean differences (Cohen's *d*, Glass's *g*, Hedges' *g*), standardized regression coefficients, and coefficients of variation.

Value

A `data.frame` with one row per confidence limit and the columns:

`term`	Either `"ci_lower"` or `"ci_upper"`.
`value`	The noncentrality parameter at that limit. `-Inf` when `alpha_lower = 0`; `Inf` when `alpha_upper = 0`.
`prob_less`	(`verbose = TRUE`) The probability $P(T \leq \text{ncp})$ that a *t*-statistic from the noncentral *t*-distribution centered at the row's limit falls at or below the observed `ncp`. By construction this equals `alpha_upper` on the `ci_upper` row and $1 -$ `alpha_lower` on the `ci_lower` row.
`prob_greater`	(`verbose = TRUE`) The complementary probability $P(T \geq \text{ncp})$. By construction this equals `alpha_lower` on the `ci_lower` row and $1 -$ `alpha_upper` on the `ci_upper` row.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Kelley, K. (2005). The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals, *Educational and Psychological Measurement, 65*, 51–69. doi:10.1177/0013164404264850

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`stats::pt()`, `stats::qt()`, `uniroot`, `ci_smd`, `ci_smd_c`, `ci_nc_F`, `ci_nc_chisq`

Other noncentral distribution confidence intervals: `ci_nc_F()`, `ci_nc_chisq()`

Examples

```
# Suppose observed t-value based on 'df'=126 is 2.83. Finding the lower
# and upper critical values for the population noncentrality parameter
# with a symmetric confidence interval with 95\% confidence is given as:
ci_nc_t(ncp = 2.83, df = 126, conf_level = .95)

# Modifying the above example so that a nonsymmetric 95% confidence interval
# can be formed:
ci_nc_t(ncp = 2.83, df = 126, alpha_lower = .01, alpha_upper = .04, conf_level = NULL)

# Modifying the above example so that a single-sided 95% confidence interval
# can be formed:
ci_nc_t(ncp = 2.83, df = 126, alpha_lower = 0, alpha_upper = .05, conf_level = NULL)
```

---

ci_omega_squared *Confidence Interval for Omega Squared (Effect Size for ANOVA)*

Description

Computes the point estimate of omega squared (ω^2) of each effect of an analysis of variance and an exact, noncentrality-based confidence interval for its population value, for the *partial* and the *total* estimand (see `omega_squared`). With `estimand = "all"`, the default, the function reports both; the interval from an effect's F statistic is an interval for the partial value, and the total has one only where it equals the partial value, in a one-way design (see Details). `"total"` or `"partial"` reports one estimand. The function takes a fitted model (`aov`, `lm`, or an `aovlist` from `aov()` with an `Error()` term) or an F statistic with its degrees of freedom and N.

Usage

```
ci_omega_squared(
  object = NULL,
  estimand = c("all", "total", "partial"),
  F_value = NULL,
  df_effect = NULL,
  df_error = NULL,
  N = NULL,
  conf_level = 0.95,
```

```
  alpha_lower = NULL,
  alpha_upper = NULL
)
```

Arguments

`object`	Optional. A fitted `aov`, `lm`, or `aovlist` object. When supplied, the function returns one row per effect, with the stratum named for an `aovlist`. Give either `object` or the *F* interface, not both.
`estimand`	Which estimand to report: `"all"` (the default, both, with `NA` where the inputs determine no estimate or no interval), `"total"`, or `"partial"`. An estimand asked for by name for which the inputs determine no interval is refused (see Details).
`F_value`	Observed *F*-value from the fixed-effects ANOVA, or a vector of the *F* statistics of several effects tested against the same error term.
`df_effect`	Numerator degrees of freedom for the effect (one value for each effect, or one for all).
`df_error`	Error (residual) degrees of freedom.
`N`	Total sample size, the number of observations (`nobs(object)` is used for a fitted model). For an *F* from a design with an `Error()` stratum, give the size of the effect's stratum as `N`, as `omega_squared` and `ci_eta_squared` use it for such a fit: the number of subjects for an effect tested between subjects, and for an effect tested within subjects the stratum's total degrees of freedom, which is `df_effect + df_error` when the effect is the only one tested in its stratum. `N` must be at least `df_effect + df_error`.
`conf_level`	Desired confidence coverage; default `0.95`. Used only when `alpha_lower` and `alpha_upper` are both `NULL`.
`alpha_lower, alpha_upper`	Optional Type I error on the lower and upper side. If both are `NULL`, a symmetric interval at `conf_level` is used. If both are supplied, `conf_level` is recomputed as `1 - alpha_lower - alpha_upper`. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound.

Details

Point estimates. The estimates are those of `omega_squared`: the partial value, which for a fixed effect with F on df_{effect} degrees of freedom and N observations is

$$\hat{\omega}^2_{\text{partial}} = \frac{df_{\text{effect}}(F-1)}{df_{\text{effect}}(F-1)+N},$$

and the total value of a fitted model with one error stratum,

$$\hat{\omega}^2_{\text{total}} = \frac{SS_{\text{effect}} - df_{\text{effect}} MS_{\text{error}}}{SS_{\text{total}} + MS_{\text{error}}}$$

(Hays, 1994; Keppel, 1991; Olejnik & Algina, 2000, Table 10). In a one-way design the two are the same number. Values below zero are reported as zero. An F statistic does not determine the total value, which the F interface reports as `NA`, and a fit with an `Error()` term has no total ω^2 (see `omega_squared`).

Confidence interval. The CI is constructed by Steiger's (2004, Proposition 1) confidence interval transformation principle: a CI for the noncentrality parameter λ of the F distribution is obtained (via `ci_nc_F`) and then mapped through

$$\omega^2_{\text{bound}} = \frac{\lambda_{\text{bound}}}{\lambda_{\text{bound}} + N},$$

with the size of its stratum in place of N for an effect of an `aovlist` fit. When the lower CI on λ is not identified (i.e., the observed F is below the one-sided critical value), the lower limit on ω^2 is set to 0, matching the convention used in `ci_pvaf`. The population ω^2 and η^2 are the same quantity, so the interval is the one `ci_eta_squared` gives (and, in a one-way design, `ci_pvaf`); the functions differ only in their *sample* estimators (an implication of the confidence interval transformation principle of Steiger, 2004).

Which estimand the interval is for. For an effect tested between subjects, $\lambda/(\lambda + N)$ is the population *partial* value, the effect's share of the variance due to the effect and the error (Steiger, 2004, Eqs. 13 and 16), so the interval is for the partial ω^2. In a one-way design that is also the total, and the interval of a fitted one-way model is reported for both estimands. For an effect of a design with other effects, the total depends on their noncentrality parameters too, which one F statistic does not inform, and no interval is formed for it: with `estimand = "all"` its estimate is reported with `NA` limits, and `estimand = "total"` is refused with an error of class `dmar_estimand_undetermined`. The Details of `ci_eta_squared` give the sources and the within-subjects case.

Sums of squares in factorial designs. When a fitted model is supplied, the function reads the F-values from `anova()`, which in base R uses Type I (sequential) sums of squares. For balanced designs, Types I, II, and III give identical F-values; for unbalanced designs they differ. If Type II or III F-values are required, compute them with e.g.\ `car::Anova(object, type = 3)` and pass the relevant F / df into the F interface. See the Details of `eta_squared` on the coding of factors that Type III tests depend on. `factorial_anova` applies sum-to-zero contrasts itself and reports partial η^2 and partial ω^2 with their confidence intervals for between-subjects factorial designs.

Value

A `data.frame` (class `dmar_ci_anova`, printed through `dmar_tbl`) with one row per effect and the columns `effect`; for each estimand reported, its estimate and limits, `omega_squared_total`, `omega_squared_total_ci_lower`, `omega_squared_total_ci_upper`, `omega_squared_partial`, `omega_squared_partial_ci_lower`, and `omega_squared_partial_ci_upper`; `stratum` (for an `aovlist` fit); and `F_value`, `df_effect`, `df_error`, and `N`. With the F interface `effect` is `"overall"` (the names of `F_value`, when it has them, or `effect_1`, `effect_2`, and so on, for several). `N` is the number of observations; the estimate and limits of an effect of an `aovlist` fit are computed with the size of its stratum, as in `ci_eta_squared`. The table records the confidence level of its intervals in its `conf_level` attribute (`conf_level`, or `1 - alpha_lower - alpha_upper` when the tail areas are supplied), which the console prints beneath it and `results_sentence` reports; the `estimand` attribute records the estimand asked for, and the `estimand_notes` attribute names each estimate or interval that is `NA` with the reason, which the console also prints. `tidy()` gives one row per effect and estimand, with the limits in `ci_lower` and `ci_upper` (see `dmar_tidiers`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bargmann, R. E. (1970). Interpretation and use of a generalized discriminant function. In R. C. Bose, I. M. Chakravarti, P. C. Mahalanobis, C. R. Rao, & K. J. C. Smith (Eds.), *Essays in probability and statistics* (pp. 35–60). University of North Carolina Press.

Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40*(3), 659–670. doi:10.1177/001316448004000309

Hays, W. L. (1994). *Statistics* (5th ed.). Fort Worth, TX: Harcourt Brace College Publishers.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*, 137–152. doi:10.1037/a0028086

Keppel, G. (1991). *Design and analysis: A researcher's handbook* (3rd ed.). Prentice Hall.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on ω^2 and its interval, and Chapter 7 on factorial designs.)

Olejnik, S., & Algina, J. (2000). Measures of effect size for comparative studies: Applications, interpretations, and limitations. *Contemporary Educational Psychology, 25*(3), 241–286. doi:10.1006/ceps.2000.1040

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Venables, W. (1975). Calculation of confidence intervals for noncentrality parameters. *Journal of the Royal Statistical Society, Series B, 37*(3), 406–412. doi:10.1111/j.25176161.1975.tb01554.x

See Also

omega_squared, ci_eta_squared, ci_pvaf, ci_nc_F

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# 1. The F interface. Bargmann's (1970) example, used in turn by
#    Venables (1975), Fleishman (1980), and Steiger (2004): a one-way
#    ANOVA with five groups of 11 and an observed F of 11.221 on 4 and
#    50 df, at 95% and at 90%. F determines the partial value, which in
#    a one-way design is the total.
ci_omega_squared(F_value = 11.221, df_effect = 4, df_error = 50, N = 55,
                 estimand = "partial")
```

```
ci_omega_squared(F_value = 11.221, df_effect = 4, df_error = 50, N = 55,
                 estimand = "partial", conf_level = 0.90)

# 2. One-way ANOVA from a fitted model: mean IQ gain differs across
#    the six grades of the pygmalion data (N = 310).
fit_one <- aov(iq_gain ~ factor(grade), data = pygmalion)
ci_omega_squared(fit_one)

# 3. Two-factor ANOVA for the manipulated expectancy treatment and the
#    measured grade classification (pygmalion data, N = 310). The
#    treatment by grade interaction is weak here (F = 1.19), so the
#    additive model is used.
fit_additive <- aov(iq_8 ~ treatment + factor(grade), data = pygmalion)
ci_omega_squared(fit_additive)
```

`ci_proportion` *Confidence Interval for a Single Proportion*

Description

The Wilson (1927) score interval for a binomial proportion, the package's default for proportion inference. Unlike the textbook Wald interval, it stays inside [0, 1] and has positive width when the number of successes is 0 or N. For a 95% interval, its coverage averaged over a uniformly distributed true proportion is .954 at $N = 10$ and .953 at $N = 20$, against .769 and .846 for the Wald interval; for some true proportions near 0 or 1 the Wilson coverage falls to about .84. For other intervals for a binomial proportion, see Brown, Cai, and DasGupta (2001). The Wald interval is available for instruction and comparison.

Usage

```
ci_proportion(successes, N, conf_level = 0.95, method = c("wilson", "wald"))
```

Arguments

`successes`	Number of successes, a single non-negative integer.
`N`	Number of trials, a single positive integer at least `successes`.
`conf_level`	Confidence level. Defaults to 0.95.
`method`	`"wilson"` (default) or `"wald"`.

Value

A `data.frame` (class `dmar_tbl`) with rows `ci_lower`, `proportion`, `ci_upper`, `successes`, and `N`, so the point estimate sits between its confidence limits.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Brown, L. D., Cai, T. T., & DasGupta, A. (2001). Interval estimation for a binomial proportion. *Statistical Science, 16*(2), 101–133. doi:10.1214/ss/1009213286

Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association, 22*(158), 209–212.

See Also

`responder_analysis`, which uses this interval for each group's responder proportion.

Other confidence intervals: `ci_crossed()`, `ci_proportion_paired()`, `ci_two_phase()`

Examples

```
ci_proportion(successes = 17, N = 50)

# The Wilson interval stays inside [0, 1] even at the boundary.
ci_proportion(successes = 0, N = 20)
```

---

ci_proportion_paired *Confidence Interval for the Difference Between Two Paired Proportions*

Description

Computes a confidence interval for the difference between two proportions measured on the same units, $\psi = p_x - p_y$: two diagnostic procedures applied to the same patients, the proportion of students answering an item correctly before and after instruction, or two raters' rates of assigning a category to the same cases. The two outcomes of a unit are correlated, so the interval for two independent proportions does not apply. The default is the score interval of Tango (1998), which inverts the score test of Nam (1997). It stays inside $[-1, 1]$ and is defined when no pair is discordant. When discordant pairs are rare, as in the case worked through under Details, it covers the difference more often than its nominal level, erring on the conservative side, where the familiar Wald interval falls well short of that level.

Usage

```
ci_proportion_paired(
  x = NULL,
  y = NULL,
  n_10 = NULL,
  n_01 = NULL,
  N = NULL,
  conf_level = 0.95,
  method = c("score", "wald")
)
```

Arguments

x, y	Numeric or logical vectors of the same length, one element per unit, coded 1 (success) and 0 (failure): the outcome of each unit under the first condition (x) and the second (y). A pair with a missing value in either vector is dropped. Supply either x and y or the three counts, not both.
n_10	Number of units that succeed under x and fail under y.
n_01	Number of units that fail under x and succeed under y.
N	Number of units (pairs), at least n_10 + n_01.
conf_level	Confidence level. Defaults to 0.95. A conf_level of 0.90 gives the interval whose limits are the one-sided 95% bounds read in noninferiority and equivalence decisions.
method	"score" (default; Tango, 1998) or "wald".

Details

Only the discordant pairs carry the difference. Classify the N units by their pair of outcomes: n_{11} succeed under both conditions, n_{10} under x only, n_{01} under y only, and n_{00} under neither. The two proportions are $(n_{11} + n_{10})/N$ and $(n_{11} + n_{01})/N$, so n_{11} cancels from their difference, $\hat{\psi} = (n_{10} - n_{01})/N$, and the concordant units enter only through N. That is why the counts interface asks for n_10, n_01, and N and nothing more. The variance of the per-unit difference, $p_{10} + p_{01} - \psi^2$, depends on a discordant probability as well as on ψ, so for inference on ψ the probability p_{01} is a nuisance parameter.

The score interval. For a hypothesized difference ψ_0 the score statistic is

$$Z(\psi_0) = \frac{n_{10} - n_{01} - N\psi_0}{\sqrt{N\{2\tilde{p}_{01} + \psi_0(1 - \psi_0)\}}},$$

where $\tilde{p}_{01}$ is the maximum likelihood estimate of p_{01} under the restriction $p_{10} - p_{01} = \psi_0$, the larger root of a quadratic (Nam, 1997, equations 1 and 3). The denominator is N times the variance of the per-unit difference, evaluated at the restricted estimates rather than at the observed proportions. With α = 1 - conf_level, the lower limit is the value of ψ_0 at which $Z(\psi_0) = z_{1-\alpha/2}$ and the upper limit the value at which $Z(\psi_0) = -z_{1-\alpha/2}$, found by root finding. At $\psi_0 = 0$ the statistic is McNemar's, $(n_{10} - n_{01})/\sqrt{n_{10} + n_{01}}$, so the score interval excludes 0 exactly when McNemar's test without a continuity correction rejects at level α: the interval extends that test to every other value of the difference. With no discordant pair the limits are $\mp z^2/(N + z^2)$, with $z = z_{1-\alpha/2}$, an interval whose width reflects the N units observed rather than a width of zero.

The Wald interval. The interval $\hat{\psi} \pm z_{1-\alpha/2}\{(n_{10} + n_{01})/N - \hat{\psi}^2\}^{1/2}/\sqrt{N}$ estimates the variance at the observed proportions instead, and is available for instruction and comparison. With few discordant pairs it is too short: it has zero width when no pair is discordant, and it can extend beyond $[-1, 1]$. Nam (1997) found by simulation that the test it inverts rejects more often than its nominal level, while the score test holds close to the nominal level and tends to the conservative side. Summing over every possible table with $p_{10} = .03$, $p_{01} = .01$, and $N = 50$, the 95% score interval covers ψ with probability .986 and the 95% Wald interval with probability .779.

Noninferiority and equivalence. The limits of a 100(1 - 2α)% interval are one-sided 100(1 - α)% confidence bounds, so conf_level = 0.90 gives the interval that equivalence_proportion_paired reads against its bounds at alpha_level = 0.05.

Value

A `data.frame` (class `dmar_tbl`) with rows `ci_lower`, `difference` ($\hat{\psi}$), `ci_upper`, `n_10`, `n_01`, and `N`, so the point estimate sits between its confidence limits. The confidence level is the `conf_level` attribute and the interval's method the `method` attribute.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

McNeil, B. J., Keeler, E., & Adelstein, S. J. (1975). Primer on certain elements of medical decision making. *New England Journal of Medicine, 293*(5), 211–215. doi:10.1056/NEJM197507312930501

Nam, J.-m. (1997). Establishing equivalence of two treatments and sample size requirements in matched-pairs design. *Biometrics, 53*(4), 1422–1430. doi:10.2307/2533508

Tango, T. (1998). Equivalence test and confidence interval for the difference in proportions for the paired-sample design. *Statistics in Medicine, 17*(8), 891–908. doi:10.1002/(SICI)10970258(19980430)17:8<891::AID-SIM780>3.0.CO;2B

See Also

`equivalence_proportion_paired` for noninferiority and equivalence tests on the same difference; `ci_proportion` for a single proportion; `responder_analysis` for two independent proportions; `mcnemar.test` for the test of no difference.

Other confidence intervals: `ci_crossed()`, `ci_proportion()`, `ci_two_phase()`

Examples

```
# Liver scan data from McNeil, Keeler, and Adelstein (1975), the
# example of Nam (1997, p. 1427): 344 patients classified as positive
# or negative by scan (x) and by pathology (y). Only the 32 and 27
# patients on whom the two disagree inform the difference in the
# positive rates.
liver <- matrix(c(231, 27, 32, 54), nrow = 2,
                dimnames = list(scan = c("positive", "negative"),
                                pathology = c("positive", "negative")))
liver
ci_proportion_paired(n_10 = liver["positive", "negative"],
                     n_01 = liver["negative", "positive"],
                     N = sum(liver))

# From raw outcomes: two judges assign the same 200 cases to one of
# three diagnoses. diagnosis_agreement holds one row per cell of their
# agreement table, so repeating each row by its frequency gives one row
# per case. How much more often does Judge A (x) than Judge B (y)
# diagnose a personality disorder?
cases <- diagnosis_agreement[rep(seq_len(nrow(diagnosis_agreement)),
                                 diagnosis_agreement$frequency), ]
ci_proportion_paired(x = cases$judge_a == "Personality disorder",
                     y = cases$judge_b == "Personality disorder")
```

```
# When the two conditions agree on all 40 units, the Wald interval has
# zero width, as though the difference were known exactly; the score
# interval still reflects how few units were observed.
ci_proportion_paired(n_10 = 0, n_01 = 0, N = 40)
ci_proportion_paired(n_10 = 0, n_01 = 0, N = 40, method = "wald")
```

`ci_pvaf` *Confidence Interval for the Proportion of Variance Accounted for (in the Dependent Variable by Knowing the Levels of the Factor)*

Description

Computes the exact confidence limits for the proportion of variance in the dependent variable accounted for by knowing the levels of the factor (group status in a single factor design) in a fixed effects analysis of variance, so an omnibus *F*-test is accompanied by an effect size with a statement of its precision.

Usage

```
ci_pvaf(
  F_value = NULL,
  df_1 = NULL,
  df_2 = NULL,
  N = NULL,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  ...
)
```

Arguments

`F_value`	Observed *F*-value from fixed effects analysis of variance
`df_1`	Numerator degrees of freedom
`df_2`	Denominator degrees of freedom
`N`	Sample size
`conf_level`	Confidence interval coverage (i.e., 1-Type I error rate); default is .95
`alpha_lower`	Type I error for the lower confidence limit
`alpha_upper`	Type I error for the upper confidence limit. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound.
`...`	Allows one to potentially include parameter values for inner functions

Details

The confidence level must be specified in one of following two ways: using confidence interval coverage (`conf_level`), or lower and upper confidence limits (`alpha_lower` and `alpha_upper`).

This function uses the confidence interval transformation principle (Steiger, 2004) to transform the confidence limits for the noncentrality parameter to the confidence limits for the population proportion of variance accounted for by knowing the group status. The confidence interval for the noncentral *F* parameter can be obtained from the function `ci_nc_F`, which is used within this function.

A lower limit of 0. When the observed *F* is below the `alpha_lower` critical value of the central *F* distribution (an *F*-test that is not significant at `alpha_lower`), no positive noncentrality parameter leaves as little as `alpha_lower` above the observed *F*, and the lower limit is 0, with a warning. The interval from 0 to the upper limit has the stated coverage, `1 - alpha_lower - alpha_upper`. A lower limit misses only when it lies above the population value, which happens with probability `alpha_lower` whatever that value is, and a lower limit of 0 never lies above it, so a limit of 0 gives up no coverage. In 20,000 simulated analyses of variance of 5 groups of 11 at each of five population values from 0 to .20, the 95% interval contained the population value in 95.0% to 95.1% of them (Monte Carlo standard error 0.15 percentage points), although its lower limit was 0 in 27% to 98% of them. An analysis whose *F* was so small that the central *F* distribution leaves `alpha_upper` or less below it has no upper limit (the `"ci_upper"` and `"actual_coverage"` rows are `NA`) and was counted as not containing the population value: 2.5% of the analyses at a population value of 0, and fewer at each larger value.

Value

A 4-row `data.frame` with columns `term`, `value`, `prob_less`, and `prob_greater`. The `term` values are `"ci_lower"` (the lower confidence limit on the proportion of variance accounted for, on the [0, 1] scale), `"pvaf"` (the sample proportion of variance accounted for, `df_1 * F_value / (df_1 * F_value + df_2)`, the same value that eta squared reports, so the point estimate sits between its confidence limits), `"ci_upper"` (the upper confidence limit), and `"actual_coverage"` (the coverage probability of the interval, `1 - alpha_lower - alpha_upper`; `NA` when the upper limit is undefined). The `prob_less` and `prob_greater` columns hold, on the row of each limit, the two tail areas at the observed `F_value` of the noncentral *F* distribution whose noncentrality parameter corresponds to that limit: `prob_less` is the area at or above `F_value` and `prob_greater` the area at or below it (the reverse of the columns of the same names in `ci_nc_F`, where `prob_less` is the area at or below `F_value`), so `prob_less` is `alpha_lower` on the `"ci_lower"` row and `prob_greater` is `alpha_upper` on the `"ci_upper"` row. With a tail area of 0 no limit is computed on that side (the lower limit is 0, or the upper limit is 1), and the row of that limit holds the requested area, 0, and its complement, 1. When `alpha_lower` is larger than 0 and the observed *F* is below the `alpha_lower` critical value, so that the lower limit is 0, the `"ci_lower"` row holds the tail areas of the central *F* distribution, whose area at or above `F_value` is the *p*-value of the *F*-test and is larger than `alpha_lower`; those areas describe the central distribution and not the coverage of the interval. Both columns are `NA` on the `"pvaf"` and `"actual_coverage"` rows, and on the `"ci_upper"` row when the upper limit is undefined.

Note

This function can be used for single or factorial ANOVA designs.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bargmann, R. E. (1970). Interpretation and use of a generalized discriminant function. In R. C. Bose, I. M. Chakravarti, P. C. Mahalanobis, C. R. Rao, & K. J. C. Smith (Eds.), *Essays in probability and statistics* (pp. 35–60). University of North Carolina Press.

Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40*(3), 659–670. doi:10.1177/001316448004000309

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*, 524–555. doi:10.1080/00273170802490632

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Steiger, J. H. (2004). Beyond the *F* Test: Effect size confidence intervals and tests of close fit in the Analysis of Variance and Contrast Analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Venables, W. (1975). Calculation of confidence intervals for noncentrality parameters. *Journal of the Royal Statistical Society, Series B, 37*(3), 406–412. doi:10.1111/j.25176161.1975.tb01554.x

See Also

`ci_nc_F`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
## Bargmann (1970) gave an example of a one-way ANOVA with an observed
## F value of 11.2213 on 4 and 50 degrees of freedom, which Venables (1975),
## Fleishman (1980), and Steiger (2004) used in turn. Steiger describes it
## as five groups of 11 and reports F as 11.221, the value used here.
## The exact confidence interval for the proportion of variance accounted for of that example:
ci_pvaf(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55)

ci_pvaf(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55, conf_level = .90)

ci_pvaf(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55, alpha_lower = 0, alpha_upper = .05)
```

ci_R2 *Confidence Interval for the Population Squared Multiple Correlation Coefficient*

Description

Constructs a confidence interval for the population squared multiple correlation coefficient ρ^2 by inverting the sampling distribution of the sample R^2. The confidence interval is for the population value ρ^2; the required input is the corresponding sample value, the observed sample squared multiple correlation coefficient R^2 (or, equivalently, the observed F-statistic and degrees of freedom). The right choice of sampling distribution, and so the right interval, depends on whether the predictors are treated as random draws from a joint multivariate normal distribution (the default and the typical case in the behavioral, educational, and social sciences) or as fixed by design (planned dosing levels, factorial covariates, etc.). The function selects the sampling distribution via `random_predictors` and inverts the corresponding noncentral distribution; the construction is the regression analog of a noncentral distribution based CI on a standardized effect size (Steiger & Fouladi, 1992; Kelley, 2007).

Usage

```
ci_R2(
  R2 = NULL,
  df_1 = NULL,
  df_2 = NULL,
  conf_level = 0.95,
  random_predictors = TRUE,
  F_value = NULL,
  N = NULL,
  p = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  tol = 1e-09
)
```

Arguments

`R2`	Observed value of the sample squared multiple correlation coefficient
`df_1`	Numerator degrees of freedom
`df_2`	Denominator degrees of freedom
`conf_level`	Confidence interval coverage; 1-Type I error rate
`random_predictors`	Whether or not the predictor variables are random or fixed (random is default)
`F_value`	Obtained F-value
`N`	Sample size
`p`	Number of predictors

alpha_lower	Type I error for the lower confidence limit
alpha_upper	Type I error for the upper confidence limit. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound.
tol	The convergence tolerance passed to uniroot when random_predictors = FALSE and the confidence limits are found by inverting the noncentral F distribution (see ci_nc_F); ignored when random_predictors = TRUE, where the Lee (1971) bisection uses its own fixed tolerance

Details

Fixed vs.\ random predictors. The two regression models give *different* sampling distributions for the sample R^2, and so different confidence intervals. Under fixed predictors the design matrix is treated as constant in hypothetical replications of the study, and the omnibus F-statistic $F = (R^2/p)/((1-R^2)/(N-p-1))$ follows a noncentral F with p and $N-p-1$ degrees of freedom and noncentrality $\lambda = N\rho^2/(1-\rho^2)$ (Cohen, 1988); the CI is obtained by inverting that distribution at the supplied confidence level (see ci_nc_F). Under random predictors the design matrix is itself a draw from a joint multivariate normal distribution and the unconditional sampling distribution of the sample R^2 is due to Fisher (1928); ci_R2 inverts the three-moment noncentral F approximation of Lee (1971, Section 5.1) to that distribution by bisection (the same construction Algina and Olejnik 2000 implemented in SAS). Gatsonis and Sampson (1989) document the comparison and show that treating random predictors as fixed tends to over-state precision (and so under-state the CI width); the discrepancy is modest at moderate to large N but non-trivial at small N with moderate-to-large effects. In the behavioral, educational, and social sciences predictor variables are almost always random, so the default is random_predictors = TRUE; pass random_predictors = FALSE for designs in which the predictor variables are fixed by design.

Value

A 3-row data.frame with columns term, value, prob_less, and prob_greater. The rows are ordered "ci_lower" (lower confidence limit on the population ρ^2), "R2" (the sample squared multiple correlation coefficient supplied by the user, the point estimate), and "ci_upper" (upper confidence limit on the population ρ^2), so the point estimate sits between its confidence limits. The prob_less and prob_greater columns are NA on the "R2" row. With fixed predictors (random_predictors = FALSE) they hold, on the row of each limit, the two tail areas at the observed F of the noncentral F distribution whose noncentrality parameter corresponds to that limit: prob_less is the area at or above the observed F and prob_greater the area at or below it (the reverse of the columns of the same names in ci_nc_F), so prob_less is alpha_lower on the "ci_lower" row and prob_greater is alpha_upper on the "ci_upper" row. With a tail area of 0 no limit is computed on that side (the lower limit is 0, or the upper limit is 1), and the row of that limit holds the requested area, 0, and its complement, 1. When alpha_lower is larger than 0 and the observed F is below the alpha_lower critical value, so that the lower limit is 0, the "ci_lower" row holds the tail areas of the central F distribution, whose area at or above the observed F is the p-value of the F-test and is larger than alpha_lower; those areas describe the central distribution and not the coverage of the interval. Both columns are NA on the "ci_upper" row when the upper limit is undefined. With random predictors (random_predictors = TRUE) the columns hold the requested

tail areas in the same arrangement, `alpha_lower` and its complement on the `"ci_lower"` row and the complement of `alpha_upper` and `alpha_upper` on the `"ci_upper"` row, whatever the limits are, except when the upper limit is undefined.

For random-predictor mode (`random_predictors = TRUE`) the limits are computed via the Lee (1971) bisection over the multiple-correlation sampling distribution; for fixed-predictor mode they are computed by inversion of the noncentral *F* distribution.

An undefined upper limit. The upper limit is the population value at which the probability of a sample R^2 at or below the observed one falls to `alpha_upper`. That probability is largest at $\rho^2 = 0$, so when the observed R^2 is at or below the `alpha_upper` quantile of its distribution at $\rho^2 = 0$, no population value is consistent with the sample at that level, and no upper limit exists. The `"ci_upper"` row is then `NA` in its value and in both tail columns, the lower limit is 0, and a warning of class `dmar_result_qualified` says why; this holds for random and for fixed predictors alike. With a 95 percent interval it happens in 2.5 percent of samples when $\rho^2 = 0$ and less often for a larger ρ^2; for example, $R^2 = .003$ with $N = 50$ and $p = 3$ is below the .025 quantile, .0046.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J. & Olejnik, S. (2000). Determining sample size for accurate estimation of the squared multiple correlation coefficient. *Multivariate Behavioral Research, 35*, 119–137. doi:10.1207/s15327906mbr3501_5

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Fisher, R. A. (1928). The general sampling distribution of the multiple correlation coefficient. *Proceedings of the Royal Society of London, Series A, 121*(788), 654–673. doi:10.1098/rspa.1928.0224

Gatsonis, C., & Sampson, A. R. (1989). Multiple correlation: Exact power and sample size calculations. *Psychological Bulletin, 106*(3), 516–524.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*, 524–555. doi:10.1080/00273170802490632

Lee, Y. S. (1971). Some results on the sampling distribution of the multiple correlation coefficient. *Journal of the Royal Statistical Society, Series B, 33*(1), 117–130.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Smithson, M. (2003). *Confidence intervals*. Thousand Oaks, CA: Sage Publications.

Steiger, J. H., & Fouladi, R. T. (1992). R2: A computer program for interval estimation, power calculations, sample size estimation, and hypothesis testing in multiple regression. *Behavior Research Methods, Instruments, & Computers, 24*(4), 581–582. doi:10.3758/BF03203611

See Also

ss_aipe_R2, ci_nc_F

Other confidence intervals for effect sizes: ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# For random predictor variables.
ci_R2(R2 = .25, N = 100, p = 5, conf_level = .95, random_predictors = TRUE)

ci_R2(F_value = 6.266667, N = 100, p = 5, conf_level = .95, random_predictors = TRUE)

# For fixed predictor variables.
ci_R2(R2 = .25, N = 100, p = 5, conf_level = .95, random_predictors = FALSE)

ci_R2(F_value = 6.266667, N = 100, p = 5, conf_level = .95, random_predictors = FALSE)

# One sided confidence intervals when predictors are random.
ci_R2(R2 = .25, N = 100, p = 5, alpha_lower = .05, alpha_upper = 0,
      conf_level = NULL, random_predictors = TRUE)

ci_R2(R2 = .25, N = 100, p = 5, alpha_lower = 0, alpha_upper = .05,
      conf_level = NULL, random_predictors = TRUE)

# One sided confidence intervals when predictors are fixed.
ci_R2(R2 = .25, N = 100, p = 5, alpha_lower = .05, alpha_upper = 0,
      conf_level = NULL, random_predictors = FALSE)

ci_R2(R2 = .25, N = 100, p = 5, alpha_lower = 0, alpha_upper = .05,
      conf_level = NULL, random_predictors = FALSE)
```

---

ci_ratio_of_means *Confidence Interval for the Ratio of Two Independent Means*

Description

Computes a confidence set for the ratio of two population means estimated from independent groups, $\rho = \mu_1/\mu_2$, or, with a third, baseline group, for the ratio of the two groups' advantages over the baseline, $\rho = (\mu_1 - \mu_b)/(\mu_2 - \mu_b)$. A ratio states an effect in relative terms (a new procedure's mean is 90 percent of the standard procedure's), which is the natural scale when the outcome has a meaningful zero. The second form is the fraction of a reference treatment's advantage over a placebo or a trivial baseline that a new treatment retains, the retention of effect of the three-arm noninferiority trial (Pigeot et al., 2003). The default interval is Fieller's, which inverts the t test of $\mu_1 - \rho\mu_2 = 0$ and is reported as what it is: a bounded interval, the complement of an interval, or the

whole line. The log-scale interval of the response ratio and percentile and BCa bootstrap intervals are offered beside it. For two means measured on the same units, see `ci_ratio_of_means_paired`.

Usage

```
ci_ratio_of_means(
  x = NULL,
  y = NULL,
  baseline = NULL,
  mean_1 = NULL,
  sd_1 = NULL,
  n_1 = NULL,
  mean_2 = NULL,
  sd_2 = NULL,
  n_2 = NULL,
  mean_baseline = NULL,
  sd_baseline = NULL,
  n_baseline = NULL,
  var_equal = FALSE,
  ci_method = c("fieller", "log", "percentile", "bca"),
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  B = 10000L,
  seed = NULL
)
```

Arguments

`x`, `y`	Numeric vectors of observations from the numerator group and the denominator group. The groups are independent and need not be the same size; `NA`s are removed from each vector separately.
`baseline`	An optional numeric vector of observations from a third, independent baseline group (a placebo arm, a trivial comparison condition). When it is supplied the ratio is that of the advantages, $(\mu_1 - \mu_b)/(\mu_2 - \mu_b)$.
`mean_1`, `mean_2`	Group sample means, for analysis from summary statistics in place of `x` and `y`.
`sd_1`, `sd_2`	Group sample standard deviations (with $n - 1$ in the denominator of the variance).
`n_1`, `n_2`	Group sample sizes.
`mean_baseline`, `sd_baseline`, `n_baseline`	The baseline group's sample mean, standard deviation, and size, for the ratio of the two groups' advantages over it from summary statistics (the three arms a noninferiority trial reports). Supply all three or none.
`var_equal`	Logical. `FALSE` (the default) lets the group variances differ and inverts the Welch-type test, whose degrees of freedom change with the hypothesized ratio; `TRUE` pools the variances, which gives Fieller's exact interval under normality and equal variances. Used by `"fieller"` and `"log"`; the bootstrap resamples each group from itself and needs neither.

`ci_method` The interval: `"fieller"` (the default), `"log"` (the response ratio interval, formed on the log scale and defined only when both means, or both advantages, are positive), `"percentile"`, or `"bca"` (bootstrap intervals, which need the raw data).

`conf_level` Confidence level (default `0.95`).

`alpha_lower` The tail area at the lower limit. An omitted confidence level yields to supplied tail areas; an explicit level must agree with them. Each tail area is at most .5.

`alpha_upper` The tail area at the upper limit, supplied together with `alpha_lower`. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound, unbounded on that side.

`B` Number of bootstrap replications for `"percentile"` and `"bca"`, a single whole number (ignored otherwise). Default `10000`: a replication of a ratio of means costs little, so a reported interval deserves the default, and the BCa adjustment moves the working quantiles into the tails, where a small `B` is least stable. The fewest accepted with 0.025 in each tail is 400 for the percentile interval and 2000 for BCa (200 and 1000 at a 0.90 level, 2000 and 10000 at 0.99); with unequal tail areas the smaller one decides. A smaller `B` is refused before anything is computed (class `dmar_count_invalid`). See *Bootstrap replications* under *Details* and `dmar_bootstrap`.

`seed` Optional integer seed for the bootstrap. The default `NULL` uses the current state of the random number generator; a supplied seed is set for the call and the prior state restored on exit.

Details

Fieller's confidence set. For a candidate ratio ρ the contrast $\bar{x}_1 - \rho\bar{x}_2$ (with a baseline, $(\bar{x}_1 - \bar{x}_b) - \rho(\bar{x}_2 - \bar{x}_b)$) estimates zero exactly when ρ is the population ratio, and its *t* statistic,

$$T(\rho) = \frac{\hat{\theta}_1 - \rho\hat{\theta}_2}{\sqrt{v_{11} - 2\rho v_{12} + \rho^2 v_{22}}},$$

where $\hat{\theta}_1$ and $\hat{\theta}_2$ are the numerator and denominator and the v terms their estimated variances and covariance, is a pivot: it involves the data and ρ alone (Fieller, 1941, 1954). The 1941 reference is Fieller's study of the biological standardization of insulin, which Fieller (1954) cites with the year 1940. The confidence set is every ρ that the test does not reject, the solution of a quadratic inequality in ρ. Its form is decided by the denominator's *t*, $\hat{\theta}_2/\sqrt{v_{22}}$, reported in the `t_denominator` row (Fieller, 1954, p. 177). The row keeps the sign of the denominator, and only its absolute value matters: when the absolute value of the denominator's *t* exceeds the critical value (with unequal tail areas, each side of the estimate is bounded when that absolute value exceeds the side's own critical value) the set is a bounded interval. When it does not, the set contains arbitrarily large values of the ratio, as it should when the data cannot distinguish the denominator from zero: it is then either the complement of an interval, every value outside a gap, or, when the data are consistent with every ratio at the chosen level, the whole line. The table reports all three as they are. The `ci_lower` and `ci_upper` rows are the outer ends of the set (`-Inf` and `Inf` when it is unbounded), and the `excluded_lower` and `excluded_upper` rows are the ends of the gap of a complement

(NA otherwise): the set is every value from ci_lower to ci_upper except those strictly between excluded_lower and excluded_upper. The confidence_set attribute names the form, and so does the line printed under the table; results_sentence writes the set in that form, for example as every value except those between the ends of the gap. With unequal tail areas one side of the set can be bounded while the other is not, which can also give an interval unbounded on one side or the union of two intervals. von Luxburg and Franz (2009) derive the same sets geometrically, from confidence intervals for the means of linear combinations of the two variables.

Equal or unequal variances. With var_equal = TRUE the variances are pooled over the groups, on $n_1 + n_2 - 2$ degrees of freedom ($n_1 + n_2 + n_b - 3$ with a baseline), and the set is Fieller's exact one under normality and equal variances. With the default var_equal = FALSE each group keeps its own variance and the contrast is tested as Welch's *t* tests a difference, with Satterthwaite degrees of freedom; those degrees of freedom depend on ρ, so the set has no closed form and is found by solving for the values of ρ at which the test's *p*-value equals its tail area. The interval therefore agrees with the test: without a baseline and with equal tail areas, each finite end L of the set (a limit or an end of the gap) is a value at which t.test(x, L * y) has *p*-value 1 - conf_level; with a baseline the test is that of the contrast $\bar{x}_1 - L\bar{x}_2 - (1 - L)\bar{x}_b$, with the Satterthwaite degrees of freedom of its three variances. The degrees of freedom of the test at each finite end of the set are in the df_limits attribute, and the df row holds those of the denominator's *t*. This inversion is the one the **mratios** function ttestratio() performs with iterativeCI = TRUE, which its documentation describes as searching iteratively for the limits by inverting the test; its default instead puts degrees of freedom computed at the estimate into Fieller's formula, and its documentation notes that those limits need not agree with the test. Hasler, Vonk, and Hothorn (2008) treat the three-arm design when the group variances differ.

The log ratio. ci_method = "log" forms the interval for $\log \rho$ from the delta method standard error, $\sqrt{v_{11}/\hat{\theta}_1^2 + v_{22}/\hat{\theta}_2^2 - 2v_{12}/(\hat{\theta}_1\hat{\theta}_2)}$, and maps it back by the exponential. This is the response ratio of meta-analysis (Hedges, Gurevitch, & Curtis, 1999), offered so a ratio reported here can be compared with one entered in a meta-analysis. It is defined only when both terms of the ratio are positive, and is refused otherwise with a condition of class dmar_log_ratio_undefined (which inherits dmar_refusal). It is always bounded; when the denominator is well determined it is close to Fieller's interval, and the two part company as the denominator's *t* falls. With unequal variances its degrees of freedom are the Satterthwaite degrees of freedom of the linearized log ratio.

The bootstrap. "percentile" and "bca" resample each group from itself (the baseline group too), as **boot**, the software of Davison and Hinkley (1997), does for several samples given as strata, and compute the ratio in every replication. The BCa interval reads its limits at quantile positions moved from those of the percentile interval by a bias correction, the median bias of the replications in normal units, and by an acceleration, the rate at which the standard error of the estimate changes with the parameter on a normalized scale (Efron & Tibshirani, 1993); its acceleration is estimated by the jackknife, deleting one observation at a time within its group, with each group's influence values scaled by its size. A bootstrap interval for a ratio is always bounded, whatever the denominator's *t*, so it can undercover badly when the denominator is near zero, exactly where Fieller's set becomes unbounded. When the Fieller set at the same level is unbounded, the line under the table says so. Pigeot et al. (2003) compare the percentile interval with Fieller's in the three-arm design. Summary statistics cannot be resampled, so a bootstrap from them is refused with a condition of class dmar_bootstrap_needs_data.

Bootstrap replications. The limits are quantiles of the B replicated ratios. A limit that cuts off a tail area α is located by the replications beyond it, so the percentile interval is run only when B puts at least 10 replications beyond each limit asked for (400 with 0.025 in each tail;

with unequal tail areas the smaller one decides, and a one-sided bound needs 10 beyond its one limit), and the BCa interval only with 50 beyond each nominal limit (2000 with 0.025 in each tail). A replication in which the ratio is undefined (a denominator of exactly zero, which discrete data can produce) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, and the line under the table states them, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the limits asked for are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. For the BCa interval the count of 10 is applied again where the bias correction and the acceleration place the limits, and when fewer than 10 replications lie beyond one of them no BCa interval is reported (limits `NA`, the same class, with the two positions as `positions` and a suggested `B` as `B_suggest`). A BCa interval whose bias correction is infinite (every replicate on one side of the estimate) or whose jackknife is undefined (deleting one observation leaves a denominator of zero) stops with an error that names `ci_method = "percentile"`. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the result is not one to report. See `dmar_bootstrap` for the rule and its reasons.

Reading the interval for noninferiority. With a baseline, a new treatment is shown to retain more than a fraction f of the reference treatment's advantage, at one-sided level `alpha_lower`, exactly when f is below the lower limit of a set that is a bounded interval (or, for a one-sided bound, an interval unbounded only above) and whose denominator is positive. The one-sided bound is `alpha_lower = 0.05, alpha_upper = 0`, and a two-sided 95 percent interval gives the bound at one-sided level .025. Reporting the limit lets each reader apply his or her own fraction; the function itself tests no fraction and states no verdict. When the denominator is negative, or the set is unbounded below, the lower limit does not carry that reading.

Value

A long `data.frame` (classes `dmar_ci_ratio_of_means`, `dmar_ci_long`, `dmar_tbl`) with columns `term` and `value`, whose rows are `ci_lower`, `ratio`, and `ci_upper` (the set's outer ends, `-Inf` or `Inf` when it is unbounded on that side); `excluded_lower` and `excluded_upper` (the ends of the excluded gap of a complement or a union of two intervals, `NA` otherwise); `mean_1`, `mean_2`, and `mean_baseline` (the group means; `mean_baseline` is `NA` without a baseline); `se_1`, `se_2`, and `se_baseline` (the standard errors of the means, from the pooled variance when `var_equal = TRUE`); `t_denominator` (the denominator, $\bar{x}_2$ or $\bar{x}_2 - \bar{x}_b$, over its standard error); `df` (the degrees of freedom of that t: pooled, or Satterthwaite's for the denominator when `var_equal = FALSE`); and `n_1`, `n_2`, and `n_baseline`. The rows are the same for every call. Attributes: `confidence_set` (`"interval"`, `"complement of an interval"`, `"whole line"`, or, with unequal tail areas only, `"union of two intervals"`; `NA` when a bootstrap interval was withheld and both limits are `NA`), `ci_method`, `var_equal`, `conf_level`, and the tail areas when they differ; `df_limits` when `var_equal = FALSE` and the interval is Fieller's or the log ratio's, the degrees of freedom of the test at each end of the set, a vector with elements `lower`, `upper`, `excluded_lower`, and `excluded_upper` that is `NA` at an end no test decides (an infinite limit, the open end of a one-sided bound, or a gap the set does

not have); B, B_used, n_dropped, B_required, and bootstrap_status for a bootstrap interval (see *Bootstrap replications* under Details); and fieller_set (the form of the Fieller set) when the interval is a log or bootstrap interval and the Fieller set at the same level is unbounded. tidy() returns the ratio, its limits, the ends of any excluded gap, and the form of the set in one row (see dmar_tidiers).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap methods and their application*. Cambridge University Press. doi:10.1017/CBO9780511802843

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Fieller, E. C. (1941). The biological standardization of insulin. *Supplement to the Journal of the Royal Statistical Society, 7*(1), 1–54. doi:10.2307/2983630

Fieller, E. C. (1954). Some problems in interval estimation. *Journal of the Royal Statistical Society, Series B, 16*(2), 175–185. doi:10.1111/j.25176161.1954.tb00159.x

Hasler, M., Vonk, R., & Hothorn, L. A. (2008). Assessing non-inferiority of a new treatment in a three-arm trial in the presence of heteroscedasticity. *Statistics in Medicine, 27*(4), 490–503. doi:10.1002/sim.3052

Hedges, L. V., Gurevitch, J., & Curtis, P. S. (1999). The meta-analysis of response ratios in experimental ecology. *Ecology, 80*(4), 1150–1156. doi:10.1890/00129658(1999)080[1150:TMAORR]2.0.CO;2

Pigeot, I., Schäfer, J., Röhmel, J., & Hauschke, D. (2003). Assessing non-inferiority of a new treatment in a three-arm clinical trial including a placebo. *Statistics in Medicine, 22*(6), 883–899. doi:10.1002/sim.1450

von Luxburg, U., & Franz, V. H. (2009). A geometric approach to confidence sets for ratios: Fieller's theorem, generalizations and bootstrap. *Statistica Sinica, 19*(3), 1095–1117.

See Also

ci_ratio_of_means_paired for two means measured on the same units; summary_t_test and welch_t for the difference between two means.

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# Drinks per week among the homeless alcohol-dependent adults of
# drinks_trial: the mean under the community reinforcement approach (CRA)
# relative to the mean under standard treatment. The group variances are
# not assumed equal.
```

```
drinks <- drinks_trial$drinks_per_week
cra <- drinks[drinks_trial$treatment == "CRA"]
standard <- drinks[drinks_trial$treatment == "Standard"]
ci_ratio_of_means(cra, standard)

# The drinking counts are strongly skewed, so the response ratio interval
# and a BCa bootstrap interval are worth setting beside Fieller's;
# B = 2000 keeps the example quick, and a reported interval deserves the
# default B = 10000.
ci_ratio_of_means(cra, standard, ci_method = "log")
ci_ratio_of_means(cra, standard, ci_method = "bca", B = 2000, seed = 113)

# From the summary statistics a paper reports, with pooled variances.
ci_ratio_of_means(mean_1 = 104, sd_1 = 14, n_1 = 40,
                  mean_2 = 96, sd_2 = 15, n_2 = 42, var_equal = TRUE)

# A three-arm trial reported as summary statistics: the fraction of the
# reference arm's advantage over placebo that the new arm retains.
ci_ratio_of_means(mean_1 = 60.1, sd_1 = 12.2, n_1 = 60,
                  mean_2 = 62.4, sd_2 = 10.1, n_2 = 60,
                  mean_baseline = 50.3, sd_baseline = 9.8,
                  n_baseline = 60)

# A three-arm trial: the fraction of the reference treatment's advantage
# over placebo that a new treatment retains, with a one-sided 95 percent
# lower bound. Noninferiority at a retained fraction f holds for every f
# below the lower limit.
set.seed(113)
placebo <- rnorm(60, mean = 50, sd = 10)
reference <- rnorm(60, mean = 62, sd = 10)
new_drug <- rnorm(60, mean = 60, sd = 12)
ci_ratio_of_means(new_drug, reference, baseline = placebo,
                  alpha_lower = 0.05, alpha_upper = 0)
```

`ci_ratio_of_means_paired` *Confidence Interval for the Ratio of Two Means From Paired Observations*

Description

Computes a confidence set for the ratio of two population means, $\rho = \mu_x/\mu_y$, when both variables are measured on the same units (the same participants before and after treatment, two instruments applied to the same cases, two procedures scored on the same items), or, with a third aligned variable, for the ratio of the two variables' advantages over it, $\rho = (\mu_x - \mu_z)/(\mu_y - \mu_z)$. The second form is the fraction of a reference procedure's advantage over a trivial baseline that a candidate procedure retains, the retention of effect of three-arm noninferiority designs (Pigeot et al., 2003) when all three are evaluated on the same units. The default interval is Fieller's, which inverts the one-sample t test of the per-unit contrast $x - \rho y$ and is reported as what it is: a bounded interval,

the complement of an interval, or the whole line. When the units are grouped (several items from each of a set of topics, several patients from each clinic), `cluster` gives the cluster-robust version. The log-scale interval of the response ratio and percentile and BCa bootstrap intervals are offered beside it. For independent groups, see `ci_ratio_of_means`.

Usage

```
ci_ratio_of_means_paired(
  x = NULL,
  y = NULL,
  baseline = NULL,
  cluster = NULL,
  mean_1 = NULL,
  sd_1 = NULL,
  mean_2 = NULL,
  sd_2 = NULL,
  r = NULL,
  N = NULL,
  ci_method = c("fieller", "log", "percentile", "bca"),
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  B = 10000L,
  seed = NULL
)
```

Arguments

`x`, `y`	Numeric vectors of the numerator and denominator variables, aligned by unit. A unit with a missing value on any of `x`, `y`, `baseline`, or `cluster` is dropped.
`baseline`	An optional numeric vector of a third variable aligned with `x` and `y` (a trivial or placebo condition measured on the same units). When it is supplied the ratio is that of the per-unit advantages, $x - z$ over $y - z$.
`cluster`	An optional vector (numeric, character, or factor) naming the cluster of each unit. When it is supplied the covariance matrix of the two means is the cluster-robust one, formed from the cluster sums, on $Q - 1$ degrees of freedom for Q clusters, and the bootstrap resamples whole clusters.
`mean_1`, `mean_2`	Means of the numerator and denominator variables, for analysis from summary statistics in place of `x` and `y`. With a baseline, supply the means, standard deviations, and correlation of the two advantages ($x - z$ and $y - z$).
`sd_1`, `sd_2`	Standard deviations of the two variables (with $N - 1$ in the denominator of the variance).
`r`	The correlation between the two variables over the units.
`N`	The number of units, the total sample size of the paired design.
`ci_method`	The interval: `"fieller"` (the default), `"log"` (the response ratio interval, formed on the log scale and defined only when both means, or both advantages, are positive), `"percentile"`, or `"bca"` (bootstrap intervals, which need the raw data).

`conf_level` Confidence level (default `0.95`).

`alpha_lower` The tail area at the lower limit. An omitted confidence level yields to supplied tail areas; an explicit level must agree with them. Each tail area is at most .5.

`alpha_upper` The tail area at the upper limit, supplied together with `alpha_lower`. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound, unbounded on that side.

`B` Number of bootstrap replications for `"percentile"` and `"bca"`, a single whole number (ignored otherwise). Default `10000`: a replication of a ratio of means costs little, so a reported interval deserves the default, and the BCa adjustment moves the working quantiles into the tails, where a small `B` is least stable. The fewest accepted with 0.025 in each tail is 400 for the percentile interval and 2000 for BCa (200 and 1000 at a 0.90 level, 2000 and 10000 at 0.99); with unequal tail areas the smaller one decides. A smaller `B` is refused before anything is computed (class `dmar_count_invalid`). See *Bootstrap replications* under *Details* and `dmar_bootstrap`.

`seed` Optional integer seed for the bootstrap. The default `NULL` uses the current state of the random number generator; a supplied seed is set for the call and the prior state restored on exit.

Details

Fieller's confidence set. For a candidate ratio ρ the per-unit contrast $d_i = x_i - \rho y_i$ (with a baseline, $(x_i - z_i) - \rho(y_i - z_i)$) has mean zero exactly when ρ is the population ratio, so the one-sample *t* test of d on $N - 1$ degrees of freedom tests each candidate, and the confidence set is every ρ it does not reject (Fieller, 1941, 1954). The 1941 reference is Fieller's study of the biological standardization of insulin, which Fieller (1954) cites with the year 1940. Written through the means and their covariance matrix V, the test statistic is

$$T(\rho) = \frac{\hat{\theta}_1 - \rho\hat{\theta}_2}{\sqrt{v_{11} - 2\rho v_{12} + \rho^2 v_{22}}},$$

with V the sample covariance matrix of the two variables divided by N, and the set is the solution of a quadratic inequality in ρ. With equal tail areas and no clusters, each finite end L of the set (a limit or an end of the gap) is a value at which `t.test(x - L * y)` has *p*-value `1 - conf_level` (with a baseline, `t.test((x - baseline) - L * (y - baseline))`), and under bivariate normality the coverage is exact. The form of the set is decided by the denominator's *t*, $\hat{\theta}_2/\sqrt{v_{22}}$, in the `t_denominator` row (Fieller, 1954, p. 177). The row keeps the sign of the denominator, and only its absolute value matters: when the absolute value of the denominator's *t* exceeds the critical value (with unequal tail areas, each side of the estimate is bounded when that absolute value exceeds the side's own critical value) the set is a bounded interval. When it does not, the set contains arbitrarily large values of the ratio, as it should when the data cannot distinguish the denominator from zero: it is then either the complement of an interval, every value outside a gap, or, when the data are consistent with every ratio at the chosen level, the whole line. The `ci_lower` and `ci_upper` rows are the outer ends of the set (`-Inf` and `Inf` when it is unbounded), and the `excluded_lower` and `excluded_upper` rows are the ends of the gap of a complement

(NA otherwise): the set is every value from ci_lower to ci_upper except those strictly between excluded_lower and excluded_upper. The confidence_set attribute names the form, and so does the line printed under the table; results_sentence writes the set in that form, for example as every value except those between the ends of the gap. With unequal tail areas one side of the set can be bounded while the other is not, which can also give an interval unbounded on one side or the union of two intervals. Fieller's own example in the examples below shows the three forms from one data set, at three confidence levels. von Luxburg and Franz (2009) derive the same sets geometrically, from confidence intervals for the means of linear combinations of the two variables.

Clustered units. When units come in clusters whose members are correlated, the per-unit standard error understates the uncertainty. With cluster, V is the cluster-robust covariance matrix formed from the cluster sums of the centered variables,

$$V = \frac{Q}{Q-1}\frac{1}{N^2}\sum_{q=1}^{Q} u_q u_q^\top,$$

where u_q holds the sums over cluster q of the two variables' deviations from their means; with $Q/(Q-1)$ as its small-sample factor it is the variance of an intercept that **sandwich**'s vcovCL(type = "HC1") gives. The reference distribution is t on $Q-1$ degrees of freedom, and the number of clusters is the n_clusters row.

The log ratio. ci_method = "log" forms the interval for $\log\rho$ from the delta method standard error, $\sqrt{v_{11}/\hat\theta_1^2 + v_{22}/\hat\theta_2^2 - 2v_{12}/(\hat\theta_1\hat\theta_2)}$, on the same degrees of freedom, and maps it back by the exponential. This is the response ratio of meta-analysis (Hedges, Gurevitch, & Curtis, 1999) with the covariance term that correlated means need (Lajeunesse, 2011). It is defined only when both terms of the ratio are positive, and is refused otherwise with a condition of class dmar_log_ratio_undefined (which inherits dmar_refusal). It is always bounded; when the denominator is well determined it is close to Fieller's interval, and the two part company as the denominator's t falls.

The bootstrap. "percentile" and "bca" resample units (pairs, or triples with a baseline), or whole clusters when cluster is supplied, and compute the ratio in every replication (Davison & Hinkley, 1997). The BCa interval reads its limits at quantile positions moved from those of the percentile interval by a bias correction, the median bias of the replications in normal units, and by an acceleration, the rate at which the standard error of the estimate changes with the parameter on a normalized scale (Efron & Tibshirani, 1993); its acceleration is estimated by the jackknife, deleting one unit (or one cluster) at a time. A bootstrap interval for a ratio is always bounded, whatever the denominator's t, so it can undercover badly when the denominator is near zero, exactly where Fieller's set becomes unbounded. When the Fieller set at the same level is unbounded, the line under the table says so. Summary statistics cannot be resampled, so a bootstrap from them is refused with a condition of class dmar_bootstrap_needs_data.

Bootstrap replications. The limits are quantiles of the B replicated ratios. A limit that cuts off a tail area α is located by the replications beyond it, so the percentile interval is run only when B puts at least 10 replications beyond each limit asked for (400 with 0.025 in each tail; with unequal tail areas the smaller one decides, and a one-sided bound needs 10 beyond its one limit), and the BCa interval only with 50 beyond each nominal limit (2000 with 0.025 in each tail). A replication in which the ratio is undefined (a denominator of exactly zero, which discrete data can produce) is dropped and counted, never replaced, and the interval is computed from the rest. The result records B (requested), B_used (the number the interval rests on), n_dropped, and B_required (the fewest replications a request may ask for; that count is held to B, not to B_used) as attributes, and the line under the table states them, with bootstrap_status ("complete",

"dropped", "exploratory", "insufficient_count", or "insufficient_share"). One warning of class dmar_bootstrap_replications_dropped states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of B that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the limits asked for are NA, the estimate is returned, and the warning has class dmar_bootstrap_insufficient. For the BCa interval the count of 10 is applied again where the bias correction and the acceleration place the limits, and when fewer than 10 replications lie beyond one of them no BCa interval is reported (limits NA, the same class, with the two positions as positions and a suggested B as B_suggest). A BCa interval whose bias correction is infinite (every replicate on one side of the estimate) or whose jackknife is undefined (deleting one observation leaves a denominator of zero) stops with an error that names ci_method = "percentile". options(dmar.bootstrap_exploratory = TRUE) lets a smaller B (2 or more) run for a first look or a demonstration; every such call gives a warning of class dmar_bootstrap_exploratory and the result is not one to report. See dmar_bootstrap for the rule and its reasons.

Reading the interval for noninferiority. With a baseline, the candidate is shown to retain more than a fraction f of the reference's advantage, at one-sided level alpha_lower, exactly when f is below the lower limit of a set that is a bounded interval (or, for a one-sided bound, an interval unbounded only above) and whose denominator is positive. The one-sided bound is alpha_lower = 0.05, alpha_upper = 0, and a two-sided 95 percent interval gives the bound at one-sided level .025. Reporting the limit lets each reader apply his or her own fraction; the function itself tests no fraction and states no verdict. When the denominator is negative, or the set is unbounded below, the lower limit does not carry that reading.

Value

A long data.frame (classes dmar_ci_ratio_of_means, dmar_ci_long, dmar_tbl) with columns term and value, whose rows are ci_lower, ratio, and ci_upper (the set's outer ends, -Inf or Inf when it is unbounded on that side); excluded_lower and excluded_upper (the ends of the excluded gap of a complement or a union of two intervals, NA otherwise); mean_1, mean_2, and mean_baseline (the means of x, y, and baseline; mean_baseline is NA without a baseline); se_numerator and se_denominator (the standard errors of the numerator and denominator, the means of x and y or of the two advantages); r (the correlation between the numerator and denominator estimates, which without clusters is the correlation of the two variables over the units); t_denominator (the denominator over its standard error); df; N (the number of units); and n_clusters (NA without cluster). The rows are the same for every call. Attributes: confidence_set ("interval", "complement of an interval", "whole line", or, with unequal tail areas only, "union of two intervals"; NA when a bootstrap interval was withheld and both limits are NA), ci_method, conf_level, and the tail areas when they differ; B, B_used, n_dropped, B_required, and bootstrap_status for a bootstrap interval (see *Bootstrap replications* under Details); and fieller_set (the form of the Fieller set) when the interval is a log or bootstrap interval and the Fieller set at the same level is unbounded. tidy() returns the ratio, its limits, the ends of any excluded gap, and the form of the set in one row (see dmar_tidiers).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap methods and their application*. Cambridge University Press. doi:10.1017/CBO9780511802843

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Fieller, E. C. (1941). The biological standardization of insulin. *Supplement to the Journal of the Royal Statistical Society, 7*(1), 1–54. doi:10.2307/2983630

Fieller, E. C. (1954). Some problems in interval estimation. *Journal of the Royal Statistical Society, Series B, 16*(2), 175–185. doi:10.1111/j.25176161.1954.tb00159.x

Hedges, L. V., Gurevitch, J., & Curtis, P. S. (1999). The meta-analysis of response ratios in experimental ecology. *Ecology, 80*(4), 1150–1156. doi:10.1890/00129658(1999)080[1150:TMAORR]2.0.CO;2

Lajeunesse, M. J. (2011). On the meta-analysis of response ratios for studies with correlated and multi-group designs. *Ecology, 92*(11), 2049–2055. doi:10.1890/110423.1

Pigeot, I., Schäfer, J., Röhmel, J., & Hauschke, D. (2003). Assessing non-inferiority of a new treatment in a three-arm clinical trial including a placebo. *Statistics in Medicine, 22*(6), 883–899. doi:10.1002/sim.1450

von Luxburg, U., & Franz, V. H. (2009). A geometric approach to confidence sets for ratios: Fieller's theorem, generalizations and bootstrap. *Statistica Sinica, 19*(3), 1095–1117.

See Also

ci_ratio_of_means for independent groups; probability_of_superiority_paired for another effect size of paired observations.

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# Depressive symptoms at the end of treatment relative to the start,
# for the ten participants given the SSRI: the ratio of the posttest
# mean to the pretest mean of the Beck Depression Inventory.
ssri <- depression_bdi[depression_bdi$condition == "ssri", ]
ci_ratio_of_means_paired(ssri$bdi_post, ssri$bdi_pre)

# Fieller's (1954, p. 180) example, the ratio of the mean effects of two
# hypnotic drugs given to the same ten patients, from his summary
# statistics (variances of the means 0.3201 and 0.4009, covariance
# 0.2848). Its denominator's t is 3.68, so the set is a bounded interval
# at 95 percent, the complement of an interval at 99.5 percent, and the
# whole line at 99.9 percent, as Fieller reported.
drug_a_sd <- sqrt(10 * 0.3201)
drug_b_sd <- sqrt(10 * 0.4009)
r_ab <- 0.2848 / sqrt(0.3201 * 0.4009)
ci_ratio_of_means_paired(mean_1 = 0.75, sd_1 = drug_a_sd, mean_2 = 2.33,
```

```
                          sd_2 = drug_b_sd, r = r_ab, N = 10)
ci_ratio_of_means_paired(mean_1 = 0.75, sd_1 = drug_a_sd, mean_2 = 2.33,
                          sd_2 = drug_b_sd, r = r_ab, N = 10,
                          conf_level = 0.995)
ci_ratio_of_means_paired(mean_1 = 0.75, sd_1 = drug_a_sd, mean_2 = 2.33,
                          sd_2 = drug_b_sd, r = r_ab, N = 10,
                          conf_level = 0.999)

# A retained fraction: a candidate procedure, a reference procedure, and
# a trivial baseline scored on the same 300 items, which come from 60
# topics of five items each. The ratio is the share of the reference's
# gain over the baseline that the candidate keeps, with a one-sided
# 95 percent lower bound and topics as clusters.
set.seed(113)
topic <- rep(1:60, each = 5)
topic_effect <- rnorm(60, sd = 0.08)[topic]
trivial <- 0.20 + topic_effect + rnorm(300, sd = 0.10)
reference <- trivial + 0.15 + rnorm(300, sd = 0.10)
candidate <- trivial + 0.13 + rnorm(300, sd = 0.10)
ci_ratio_of_means_paired(candidate, reference, baseline = trivial,
                          cluster = topic, alpha_lower = 0.05,
                          alpha_upper = 0)

# A percentile bootstrap interval for the SSRI ratio; B = 2000 keeps the
# example quick, and a reported interval deserves the default B = 10000.
ci_ratio_of_means_paired(ssri$bdi_post, ssri$bdi_pre,
                          ci_method = "percentile", B = 2000, seed = 113)
```

`ci_rc` *Confidence Interval for an Unstandardized Regression Coefficient*

Description

Computes a confidence interval for a population regression coefficient in its raw (unstandardized) metric, by the standard *t*-based approach or the noncentral *t* approach. A thin convenience wrapper around `ci_reg_coef`, which is the general engine; for the standardized coefficient use `ci_src`.

Usage

```
ci_rc(
  reg_coef_j = NULL,
  se_reg_coef_j = NULL,
  sd_Y = NULL,
  sd_X = NULL,
  N,
  p,
  R2_Y_X = NULL,
  R2_j_X_without_j = NULL,
```

```
  conf_level = 0.95,
  R2_Y_X_without_j = NULL,
  t_value = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  noncentral = FALSE
)
```

Arguments

reg_coef_j	Value of the regression coefficient for the *j*th predictor variable, or NULL (the default) when the other inputs determine it: t_value with R2_Y_X and R2_j_X_without_j, or the three squared multiple correlations, which fix its magnitude but not its sign (see ci_reg_coef)
se_reg_coef_j	Standard error for the *j*th predictor variable
sd_Y	Standard deviation of *Y*, the dependent variable
sd_X	Standard deviation of *X*, the predictor variable of interest
N	Sample size
p	The number of predictors
R2_Y_X	The squared multiple correlation coefficient predicting *Y* from the *p* predictor variables
R2_j_X_without_j	The squared multiple correlation coefficient predicting the *j*th predictor variable (i.e., the predictor of interest) from the remaining *p*-1 predictor variables
conf_level	Desired level of confidence for the computed interval (i.e., 1 - the Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused
R2_Y_X_without_j	The squared multiple correlation coefficient predicting *Y* from the *p*-1 predictor variable with the *j*th predictor of interest excluded
t_value	The *t*-value evaluating the null hypothesis that the population regression coefficient for the *j*th predictor equals zero
alpha_lower	The Type I error rate for the lower confidence interval limit
alpha_upper	The Type I error rate for the upper confidence interval limit. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in ci_nc_t).
noncentral	TRUE or FALSE statement specifying whether or not the noncentral approach to confidence intervals should be used

Details

Returns the confidence limits for the regression coefficient of interest from the standard approach to confidence interval formation or from the noncentral approach to confidence interval formation using the noncentral *t*-distribution.

Value

A 3-row `data.frame` with columns `term`, `value`, `prob_less`, and `prob_greater`. The `term` values are `"ci_lower"`, `"reg_coef"` (the coefficient as given, or as the other inputs determine it), and `"ci_upper"`, and `value` holds the coefficient and its confidence limits in the raw metric. The `prob_less` and `prob_greater` columns report the tail probabilities below and above each limit; when the noncentral *t* approach is used they are the achieved tail probabilities; they are `NA` on the row of the estimate. The estimate sits between its limits, as in `ci_src` and `ci_reg_coef`.

Note

Not all of the values need to be specified, only those that contain all of the necessary information in order to compute the confidence interval (options are thus given for the values that need to be specified).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

Smithson, M. (2003). *Confidence intervals*. Thousand Oaks, CA: Sage Publications.

Steiger, J. H. (2004). Beyond the *F* Test: Effect size confidence intervals and tests of close fit in the Analysis of Variance and Contrast Analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

See Also

`ss_aipe_reg_coef`, `ci_nc_t`, `ci_reg_coef`, `ci_src`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
ci_rc(reg_coef_j = 0.61319, se_reg_coef_j = 0.16098, N = 30, p = 6, conf_level = 0.95)
```

`ci_reg_coef` *Confidence Interval for a Regression Coefficient, Raw or Standardized*

Description

The general engine behind `ci_rc` (unstandardized) and `ci_src` (standardized): computes a confidence interval for a population regression coefficient by the standard *t*-based approach or the noncentral *t* approach, in whichever metric the inputs are supplied.

Usage

```
ci_reg_coef(
  reg_coef_j = NULL,
  se_reg_coef_j = NULL,
  sd_Y = NULL,
  sd_X = NULL,
  N,
  p,
  R2_Y_X = NULL,
  R2_j_X_without_j = NULL,
  conf_level = 0.95,
  R2_Y_X_without_j = NULL,
  t_value = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  noncentral = FALSE
)
```

Arguments

`reg_coef_j`	Value of the regression coefficient for the *j*th predictor variable, or `NULL` (the default) when the other inputs determine it: `t_value` with `R2_Y_X` and `R2_j_X_without_j`, or the three squared multiple correlations, which fix its magnitude but not its sign (see Details)
`se_reg_coef_j`	Standard error for the *j*th predictor variable
`sd_Y`	Standard deviation of *Y*, the dependent variable
`sd_X`	Standard deviation of X_j, the predictor variable of interest
`N`	Sample size
`p`	The number of predictors
`R2_Y_X`	The squared multiple correlation coefficient predicting `Y` from the `p` predictor variables

R2_j_X_without_j
: The squared multiple correlation coefficient predicting the j th predictor variable (i.e., the predictor of interest) from the remaining p-1 predictor variables

conf_level
: Desired level of confidence for the computed interval (i.e., 1 - the Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused

R2_Y_X_without_j
: The squared multiple correlation coefficient predicting Y from the p-1 predictor variable with the j th predictor of interest excluded

t_value
: The *t*-value evaluating the null hypothesis that the population regression coefficient for the j th predictor equals zero

alpha_lower
: The Type I error rate for the lower confidence interval limit

alpha_upper
: The Type I error rate for the upper confidence interval limit. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in ci_nc_t).

noncentral
: TRUE or FALSE, specifying whether or not the noncentral approach to confidence intervals should be used

Details

For standardized variables, do not specify the standard deviation of the variables and input the standardized regression coefficient for reg_coef_j.

Exactness. For a coefficient in the units of the data, the central interval (noncentral = FALSE, the default) is exact under normal errors given the values of the predictors, and so for random predictors as well. The noncentral interval multiplies the confidence limits for the noncentrality parameter by the estimated standard error, while the noncentrality parameter is the coefficient divided by its population standard error, so that interval is approximate on either scale. In a simulation with one predictor held fixed and normal errors (4,000 samples per condition, N of 15, 40, and 200, Monte Carlo standard errors of .001 to .004), the 95 percent noncentral interval for a coefficient in the units of the data covered .944 to .955 when the noncentrality parameter was 1, and from .961 (N = 200) to .993 (N = 15) when it was 7; the central interval covered .943 to .953 in every condition.

Standardized data. For a coefficient of standardized data, or one rescaled by the sample standard deviations, the standard error treats those standard deviations as fixed constants (Kelley & Maxwell, 2003, pp. 315–316). The standard deviation of Y varies from sample to sample in every design, and those of the predictors vary as well when the predictors are random, so this standard error is approximate either way and the interval does not have exactly its nominal coverage. In the simulation of multivariate normal data described in ci_src, the central 95 percent interval (the default here) covered the population standardized coefficient .957 to .993 of the time with a single random predictor correlated .3 to .7 with Y, .959 to .967 for coefficients of .34 to .52 among two to four random predictors correlated positively with Y and with one another and explaining less than 60 percent of its variance, and .952 to .956 in the conditions whose coefficient was 0 or near it, but only .882 to .952 for suppressors, and .938 and .881 with two uncorrelated predictors that explained 72 and 90 percent of the variance of Y. With the values of the predictors held fixed it covered .955 to .989 whenever the coefficient was not 0. The noncentral interval, the one ci_src

reports, covered more in every condition except those whose coefficient was 0 or near it, where the two agreed within Monte Carlo error.

When `reg_coef_j` is reconstructed from squared multiple correlations (that is, from `R2_Y_X`, `R2_Y_X_without_j`, and `R2_j_X_without_j` rather than a supplied `reg_coef_j`, `se_reg_coef_j`, or `t_value`), only the magnitude of the coefficient is identifiable; its sign is not. The positive root is returned and a warning is issued. If the coefficient is negative, negate the point estimate and swap and negate the confidence limits, or supply `reg_coef_j` directly.

Value

A 3-row `data.frame` with columns `term`, `value`, `prob_less`, and `prob_greater`. The rows are ordered `"ci_lower"`, `"reg_coef"` (the regression coefficient point estimate), and `"ci_upper"`, so the point estimate sits between its confidence limits. The lower and upper rows give the confidence limits on the regression coefficient. The `prob_less` and `prob_greater` columns report the achieved tail probabilities at each limit when the noncentral t method is used (they are `NA` for the `"reg_coef"` estimate row).

Note

Not all of the values need to be specified, only those that contain all of the necessary information in order to compute the confidence interval (options are thus given for the values that need to be specified).

The function `ci_rc` in DMAR also calculates the confidence interval for the population (unstandardized) regression coefficient. The function `ci_src` also calculates the confidence interval for the population (standardized) regression coefficient. These two functions perform the same tasks as `ci_reg_coef` does and are preferred to it because of simpler arguments.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

Smithson, M. (2003). *Confidence intervals*. Thousand Oaks, CA: Sage Publications.

See Also

ss_aipe_reg_coef, ci_nc_t, ci_rc, ci_src

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
ci_reg_coef(reg_coef_j = 0.61319, se_reg_coef_j = 0.16098, N = 30, p = 6)
```

---

ci_rmsea *Confidence Interval for the Population Root Mean Square Error of Approximation*

Description

Constructs a confidence interval for the population root mean square error of approximation (RMSEA), a population badness-of-fit index for structural equation models, from a reported RMSEA, its degrees of freedom, and the sample size. The interval is obtained by inverting the noncentral chi square distribution of the test statistic $T = a\hat{F}$, where $\hat{F}$ is the minimized fit function and a the sample size multiplier, and mapping the resulting noncentrality limits to the RMSEA metric (Steiger & Lind, 1980; Browne & Cudeck, 1993).

Usage

```
ci_rmsea(
  rmsea,
  df,
  N,
  conf_level = 0.9,
  alpha_lower = NULL,
  alpha_upper = NULL,
  sample_size_convention = c("N-1", "N")
)
```

Arguments

rmsea	Observed root mean square error of approximation
df	Degrees of freedom of the model, greater than 0 (a model with zero degrees of freedom is saturated, and its RMSEA is undefined)
N	Sample size: greater than 1 under the default sample_size_convention, and greater than 0 under "N"

`conf_level` Desired confidence level, between 0 and 1. The default, 0.90, gives the conventional interval for the RMSEA, whose two limits decide the tests of close and not close fit at .05 (see Details).

`alpha_lower` The Type I error rate for the lower tail

`alpha_upper` The Type I error rate for the upper tail. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound.

`sample_size_convention`
The multiplier a with which the reported RMSEA was computed: `"N-1"` (the default), so $a = N - 1$, the convention of Browne and Cudeck (1993), of Kelley and Lai (2011), and of `ss_aipe_rmsea`; or `"N"`, so $a = N$, the convention of lavaan's default normal likelihood and of Mplus.

Details

The RMSEA expresses the badness of model fit per degree of freedom on the noncentrality scale. Under the noncentral chi square model for the sample fit statistic, $T = a\hat{F}$ has approximate noncentral chi square distribution with df degrees of freedom and noncentrality parameter $\lambda = a \cdot df \cdot \mathrm{RMSEA}^2$. The function rebuilds $T = a \cdot df \cdot \mathrm{RMSEA}^2 + df$ from the reported RMSEA, obtains the interval for λ by inverting the noncentral chi square distribution at the requested confidence level (`ci_nc_chisq` does the inversion), and maps its limits back to the RMSEA scale as $\sqrt{\lambda/(a \cdot df)}$. When the lower noncentrality limit hits zero (*i.e.*, the data are compatible with a well-fitting model), the lower RMSEA limit is truncated at zero because RMSEA is nonnegative by construction.

The multiplier must be the one with which the RMSEA was computed. An RMSEA from lavaan under its default likelihood was computed with N; given to the default `"N-1"`, it gives an interval that matches neither convention, off by an amount of order $1/N$. Two further cases need the model's chi square statistic rather than its RMSEA, and `sem_fit_measures` takes it: an RMSEA truncated at zero, which cannot distinguish the statistics below df that produce it, and the RMSEA of a multiple group model, which carries the factor $\sqrt{G}$ for G groups (Steiger, 1998); this function assumes a single group.

The 90 percent CI (rather than the usual 95 percent) is the conventional reporting choice for the RMSEA, from the .05 and .95 percentile points of Steiger and Lind (1980) through Browne and Cudeck (1993), because a $100(1 - 2\alpha)$ percent interval decides the two one-sided tests at level α, so that the limits of the 90 percent interval settle them at level .05 (MacCallum, Browne, & Sugawara, 1996, p. 136 and Table 1): the hypothesis of close fit, $H_0 : \mathrm{RMSEA} \leq \epsilon_0$, is rejected when the lower limit exceeds ϵ_0, and the hypothesis of not close fit, $H_0 : \mathrm{RMSEA} \geq \epsilon_0$, is rejected, so that fit within ϵ_0 is supported, when the upper limit falls below ϵ_0. A lower limit of zero likewise says that the test of exact fit does not reject at .05. `ci_rmsea` therefore defaults to `conf_level = 0.90`, as every RMSEA interval in DMAR does. `conf_level = 0.95`, the level of the package's other intervals, gives the 95 percent interval, whose limits decide the one-sided tests at .025. The printed table states the level beneath it. See `equivalence_rmsea` for the test of not close fit with its p-value and verdict.

Value

A 3-row `data.frame` with columns `term` and `value`. The `term` values are `"ci_lower"` (the lower bound of the confidence interval on the population RMSEA, truncated at zero by definition), `"rmsea"` (the observed point estimate), and `"ci_upper"` (the upper bound).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Browne, M. W., & Cudeck, R. (1993). Alternative ways of assessing model fit. In K. A. Bollen & J. S. Long (Eds.), *Testing structural equation models* (pp. 136–162). Sage.

Kelley, K., & Lai, K. (2011). Accuracy in parameter estimation for the root mean square error of approximation: Sample size planning for narrow confidence intervals. *Multivariate Behavioral Research, 46*, 1–32. doi:10.1080/00273171.2011.543027

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130–149. doi:10.1037/1082989X.1.2.130

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Steiger, J. H. (1998). A note on multiple sample extensions of the RMSEA fit index. *Structural Equation Modeling, 5*(4), 411–419. doi:10.1080/10705519809540115

Steiger, J. H., & Lind, J. C. (1980). *Statistically-based tests for the number of common factors*. Paper presented at the annual Spring meeting of the Psychometric Society, Iowa City, IA.

See Also

`sem_fit_measures`, `equivalence_rmsea`, `ss_aipe_rmsea`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
# 1. The conventional 90 percent CI on RMSEA, the default. Its upper
#    limit, 0.052, lies just above 0.05, so the test of not close fit at
#    .05 does not reject an RMSEA of 0.05 or more, and fit within 0.05 is
#    not established even though the point estimate sits well below it.
ci_rmsea(rmsea = .035, df = 40, N = 425)

# 2. The 95 percent CI is wider.
ci_rmsea(rmsea = .035, df = 40, N = 425, conf_level = .95)

# 3. An RMSEA reported by lavaan under its default likelihood, which
#    forms the statistic with N.
```

```
ci_rmsea(rmsea = .0921, df = 24, N = 301, sample_size_convention = "N")

# 4. Fewer degrees of freedom and a smaller N: a wider CI.
ci_rmsea(rmsea = .055, df = 10, N = 100)
```

`ci_sc` *Confidence Interval for a Standardized Contrast in a Fixed Effects ANOVA*

Description

Computes the exact noncentral *t*-based confidence interval for a standardized contrast of means in a fixed effects analysis of variance: the contrast of interest divided by the error standard deviation, so groups measured in raw units are compared on a common standardized scale.

Usage

```
ci_sc(
  means = NULL,
  s_anova = NULL,
  c_weights = NULL,
  n_per_group = NULL,
  N = NULL,
  contrast = NULL,
  ncp = NULL,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  df_error = NULL,
  ...
)
```

Arguments

`means`	A vector of the group means or the means of the particular level of the effect (for fixed effect designs)
`s_anova`	The standard deviation of the errors from the ANOVA model (i.e., the square root of the mean square error); required, as a single positive number, on every path
`c_weights`	The contrast weights (choose weights so that the positive *c*-weights sum to 1 and the negative *c*-weights sum to -1; i.e., use fractional values such as `c(.5, .5, -1)`, not `c(1, 1, -2)`). The rule is applied to the values, so weights stored as integers, such as `-1:1`, are accepted.
`n_per_group`	Sample sizes *per group* or sample sizes for the level of the particular factor (if length 1 it is assumed that the sample size *per group* or for the level of the particular factor are are equal)

N	Total sample size
contrast	The (unstandardized) contrast effect, obtained by multiplying the *j*th mean by the *j*th contrast weight (this is the unstandardized effect)
ncp	The noncentrality parameter from the *t*-distribution
conf_level	Desired level of confidence for the computed interval (i.e., 1 - the Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused
alpha_lower	The Type I error rate for the lower confidence interval limit
alpha_upper	The Type I error rate for the upper confidence interval limit. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in `ci_nc_t`).
df_error	The degrees of freedom for the error. In one-way designs it is *N* minus the number of groups, `N - length(means)`, and need not be specified; it must be specified if the design has multiple factors.
...	Optional additional specifications for nested functions

Value

A 3-row `data.frame` with columns `term` and `value`. The `term` values are `"ci_lower"` (the lower confidence limit on the standardized contrast), `"std_contrast"` (the standardized contrast), and `"ci_upper"` (the upper limit).

Note

Be sure to use the standard deviation and not the error variance for `s_anova`, not the square of this value (the error variance) which would come from the source table (i.e., do not use the variance of the error but rather use its square root, the standard deviation).

Be sure to use fractional *c*-weights when doing complex contrasts (not integer values such as `c(1, 1, -2)`) to specify `c_weights`. For example, in an ANOVA of four groups, if the user wants to compare the mean of group 1 and 2 with the mean of group 3 and 4, `c_weights` should be specified as c(0.5, 0.5, -0.5, -0.5) rather than c(1, 1, -1, -1). Make sure the sum of the contrast weights are zero.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Steiger, J. H. (2004). Beyond the *F* Test: Effect size confidence intervals and tests of close fit in the Analysis of Variance and Contrast Analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

See Also

ci_nc_t, ci_src, ci_smd, ci_smd_c, ci_sm, ci_c ci_c_ancova

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# Here is a four group example. Suppose that the means of groups 1--4 are 2, 4, 9,
# and 13, respectively. Further, let the error variance be .64 and thus the standard
# deviation would be .80 (note we use the standard deviation in the function, not the
# variance). The standardized contrast of interest here is the average of groups 1 and 4
# versus the average of groups 2 and 3.

ci_sc(means = c(2, 4, 9, 13), s_anova = .80, c_weights = c(.5, -.5, -.5, .5),
      n_per_group = c(3, 3, 3, 3), N = 12, conf_level = .95)

# Here is an example with two groups.
ci_sc(means = c(1.6, 0), s_anova = .80, c_weights = c(1, -1),
      n_per_group = c(10, 10), N = 20, conf_level = .95)
```

---

ci_scheffe *Scheffe-Adjusted Simultaneous Confidence Intervals for Contrasts*

Description

Computes the Scheffe (1953, 1959) simultaneous confidence intervals on user-specified contrasts among the means of a one-way design. The Scheffe procedure controls the family-wise error rate for *any* set of contrasts, however many and however post-hoc, which makes it more conservative than Tukey-Kramer or Bonferroni for the specific case of all-pairwise comparisons but optimal for arbitrary post-hoc contrasts.

Usage

```
ci_scheffe(x, group = NULL, contrasts = NULL, conf_level = 0.95)
```

Arguments

x	A fitted lm or aov object with a single one-way factor predictor, or a numeric vector of observations with group supplied.
group	Optional factor of group labels when x is a numeric vector.
contrasts	An $a \times m$ matrix or vector of contrast coefficients (rows = levels, columns = contrasts). Each column must sum to zero. If NULL (default), the function returns intervals for all $a(a-1)/2$ pairwise contrasts.
conf_level	Family-wise confidence level. Default 0.95.

Details

Critical value. For a groups with ν error degrees of freedom, the Scheffe critical value is

$$S \;=\; \sqrt{(a-1)F_{1-\alpha,a-1,\nu}},$$

where $F_{1-\alpha,a-1,\nu}$ is the upper α quantile of the central F distribution. The Scheffe simultaneous CI on a contrast $\psi = \sum_i c_i \mu_i$ is

$$\hat{\psi} \;\pm\; S \cdot SE(\hat{\psi}),$$

where $SE(\hat{\psi}) = \sqrt{MS_E \sum_i c_i^2 / n_i}$.

Scope. The Scheffe family-wise coverage holds for *any* number of contrasts, pairwise, complex, or chosen after looking at the data. The trade-off is conservativeness: for all-pairwise comparisons, Tukey-Kramer is uniformly more powerful.

Value

A data.frame with one row per contrast. Columns: contrast (a printed label), estimate, se, F_value, ci_lower, ci_upper, p_adjusted.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Scheffe, H. (1953). A method for judging all contrasts in the analysis of variance. *Biometrika, 40*(1/2), 87–104.

Scheffe, H. (1959). *The analysis of variance*. Wiley.

See Also

cv_scheffe, ci_tukey_kramer, ci_dunnett

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# 1. All pairwise contrasts among the six marketing panels of the
#    test_market data via the default:
fit <- lm(brand_movement ~ panel, data = test_market)
ci_scheffe(fit)

# 2. A contrast chosen after inspecting the means: the two panels with
#    the highest brand movement (5 and 6) against the two with the
#    lowest (1 and 2). The Scheffe coverage holds for a contrast picked
#    this way, and the interval still excludes zero even though none of
#    the pairwise intervals above does.
cmat <- matrix(c(-0.5, -0.5, 0, 0, 0.5, 0.5), nrow = 6,
               dimnames = list(levels(test_market$panel),
                               "panels 5,6 - panels 1,2"))
ci_scheffe(fit, contrasts = cmat)
```

`ci_sc_ancova` *Confidence Interval for a Standardized Contrast in ANCOVA With One Covariate*

Description

Calculate the confidence interval for a standardized contrast in ANCOVA with one covariate. The standardizer (i.e., the divisor) can be either the error standard deviation of the ANOVA model (i.e., the model excluding the covariate) or of the ANCOVA model.

Usage

```
ci_sc_ancova(
  contrast = NULL,
  adj_means = NULL,
  s_anova = NULL,
  s_ancova = NULL,
  standardizer = "s_ancova",
  c_weights,
  n_per_group,
  cov_means,
  SSwithin_x,
  conf_level = 0.95
)
```

Arguments

`contrast`	Unstandardized contrast of adjusted means
`adj_means`	The vector that contains the adjusted mean of each group on the dependent variable

s_anova	The standard deviation of the errors from the ANOVA model (i.e., the square root of the mean square error from ANOVA); required, as a single positive number, when standardizer = "s_anova"
s_ancova	The standard deviation of the errors from the ANCOVA model (i.e., the square root of the mean square error from ANCOVA); required, as a single positive number, with either standardizer
standardizer	Which error standard deviation the user wants to use, the value of which can be either "s_ancova" or "s_anova"
c_weights	The contrast weights (chose weights so that the positive *c*-weights sum to 1 and the negative *c*-weights sum to -1; i.e., use fractional values not integers).
n_per_group	Either a single number that indicates the sample size per group, or a vector that contains the sample size of each group
cov_means	A vector that contains the group means of the covariate
SSwithin_x	The sum of squares within groups obtained from the summary table for ANOVA on the covariate
conf_level	The desired confidence interval coverage, (i.e., 1 - Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in ci_nc_t)

Details

The argument SSwithin_x is the sum of squares within groups for the covariate, taken from the ANOVA source table in which the covariate (not the outcome) is the dependent variable. Published reports do not always print this quantity directly. When a report gives the covariate group means, the group sample sizes, and the *F* statistic from the one-way ANOVA on the covariate, SSwithin_x can be recovered algebraically. The worked example below follows Lai and Kelley (2012): three groups of sizes 19, 18, and 19 (so $N = 56$) have covariate means 60.08, 57.08, and 57.97, and the covariate ANOVA reports $F = 0.756$ with 2 and 53 degrees of freedom. The sum of squares between groups for the covariate, computed from the group means and sample sizes, is approximately 88.5, so the mean square between groups is approximately $88.5/2 = 44.3$. Because *F* is the ratio of the mean square between groups to the mean square within groups, the mean square within groups is approximately $44.3/0.756 = 58.6$, and the sum of squares within groups is that mean square times its degrees of freedom, approximately $58.6 \times 53 = 3103$. That recovered value is what you would pass to SSwithin_x. The "Examples" section reproduces this computation in code.

Value

A 3-row data.frame with columns term and value (numeric). The term values are "ci_lower" (the lower confidence limit on the standardized ANCOVA contrast), "std_contrast" (the standardized contrast), and "ci_upper" (the upper limit). The divisor used in standardization (either "s_anova" or "s_ancova") is attached as the "standardizer" attribute of the returned data.frame.

Note

Be sure to use the standard deviations and not the error variances for s_anova and s_ancova, not the squares of these values which would come from the source tables (i.e., do not use the variance of the errors but rather use its square root, the standard deviation).

If n_per_group receives a single number, that number is considered as the sample size per group. If n_per_group is assigned to a vector, the vector is considered as the sample size of each group.

Be sure to use fractional *c*-weights when doing complex contrasts (not integers) to specify c_weights. For example, in an ANCOVA of four groups, if the user wants to compare the mean of group 1 and 2 with the mean of group 3 and 4, c_weights should be specified as c(0.5, 0.5, -0.5, -0.5) rather than c(1, 1, -1, -1). Make sure the sum of the contrast weights are zero.

The argument to be assigned to standardizer must be either "s_ancova" or "s_anova".

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*, 363–385. doi:10.1037/1082989X.11.4.363

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

ci_c_ancova, ci_sc

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# Maxwell, Delaney, & Kelley (2027) offer an example that 30 depressive
# individuals are randomly assigned to three groups, 10 in each, and ANCOVA
# is performed on the posttest scores using the participants' pretest
# scores as the covariate. The means of pretest scores of group 1, 2, and 3 are
# 17, 17.7, and 17.4, respectively, whereas the adjusted means of groups 1, 2, and 3
# are 7.5, 12, and 14, respectively. The error variance in ANCOVA is 29 and thus
# 5.385165 is the error standard deviation, with the sum of squares within groups
# from an ANOVA on the covariate is 752.5.
```

```
# To obtained the confidence interval for the standardized adjusted mean difference
# between group 1 and 2, using the ANCOVA error standard deviation:
ci_sc_ancova(adj_means = c(7.5, 12, 14), s_ancova = 5.385165, c_weights = c(1, -1, 0),
             n_per_group = 10, cov_means = c(17, 17.7, 17.4), SSwithin_x = 752.5)

# Or, with less error in rounding:
ci_sc_ancova(adj_means = c(7.54, 11.98, 13.98), s_ancova = 5.393, c_weights = c(-1, 0, 1),
             n_per_group = 10, cov_means = c(17, 17.7, 17.4), SSwithin_x = 752.5)

# Now, using the standard deviation from ANOVA (and not ANCOVA as above), we have:
ci_sc_ancova(adj_means = c(7.54, 11.98, 13.98), s_anova = 6.294, s_ancova = 5.393,
             c_weights = c(-1, 0, 1),n_per_group = 10, cov_means = c(17, 17.7, 17.4),
             SSwithin_x = 752.5, standardizer = "s_anova", conf_level = .95)

# Recovering SSwithin_x from a covariate ANOVA F when a report does not print
# it directly (see the Details section). This example follows Lai and Kelley
# (2012): three groups of sizes 19, 18, and 19 have covariate means 60.08,
# 57.08, and 57.97, and the one-way ANOVA on the covariate reports F = 0.756.
cov_means_ex  <- c(60.08, 57.08, 57.97)
n_ex          <- c(19, 18, 19)
grand_x       <- sum(n_ex * cov_means_ex) / sum(n_ex)
ss_between_x  <- sum(n_ex * (cov_means_ex - grand_x)^2)
ms_between_x  <- ss_between_x / (length(n_ex) - 1)
ms_within_x   <- ms_between_x / 0.756
SSwithin_x_ex <- ms_within_x * (sum(n_ex) - length(n_ex))

ci_sc_ancova(adj_means = c(63.88, 62.39, 56.48), s_ancova = 20.267,
             c_weights = c(0.5, 0.5, -1), n_per_group = n_ex, cov_means = cov_means_ex,
             SSwithin_x = SSwithin_x_ex)
```

---

ci_sm *Confidence Interval for the Standardized Mean*

Description

Computes the exact confidence interval for the standardized mean, the mean divided by the standard deviation, by inverting the noncentral t distribution. The standardized mean is the one-sample analog of the standardized mean difference and shares its noncentral interval theory.

Usage

```
ci_sm(
  sm = NULL,
  mean = NULL,
  sd = NULL,
  ncp = NULL,
  N = NULL,
  conf_level = 0.95,
```

```
  alpha_lower = NULL,
  alpha_upper = NULL,
  ...
)
```

Arguments

sm	Standardized mean
mean	Mean
sd	Standard deviation
ncp	Noncentral parameter
N	Sample size
conf_level	Confidence interval coverage (i.e., 1 - Type I error rate); default is .95. A level above .999999998, whose tail areas are below 1e-9, is refused
alpha_lower	Type I error for the lower confidence limit
alpha_upper	Type I error for the upper confidence limit. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in `ci_nc_t`).
...	Allows one to potentially include parameter values for inner functions

Details

The user must specify the standardized mean in one and only one of the three ways: a) mean and standard deviation (`mean` and `sd`), b) standardized mean (`sm`), and c) noncentral parameter (`ncp`). The confidence level must be specified in one of following two ways: using confidence interval coverage (`conf_level`), or lower and upper confidence limits (`alpha_lower` and `alpha_upper`). This function uses the exact confidence interval method based on noncentral *t*-distributions. The confidence interval for noncentral *t*-parameter can be obtained from the `ci_nc_t` function in DMAR.

Value

A 3-row `data.frame` with columns `term` and `value`. The `term` values are `"ci_lower"` (the lower confidence limit on the standardized mean), `"std_mean"` (the standardized mean), and `"ci_upper"` (the upper confidence limit).

Note

The standardized mean is the mean divided by the standard deviation.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`ci_nc_t`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
ci_sm(sm = 2.037905, N = 13, conf_level = .95)
ci_sm(mean = 30, sd = 14.721, N = 13, conf_level = .95)
ci_sm(ncp = 7.347771, N = 13, conf_level = .95)
ci_sm(sm = 2.037905, N = 13, alpha_lower = .05, alpha_upper = 0)
ci_sm(mean = 50, sd = 10, N = 25, conf_level = .95)
```

`ci_smd` *Confidence Interval for the Standardized Mean Difference (Two Independent Groups)*

Description

Constructs an exact-coverage confidence interval for the population standardized mean difference $\delta = (\mu_1 - \mu_2)/\sigma$ (Cohen's d when expressed as a sample quantity) for two independent groups whose scores are normal within each group with a common variance. The interval is obtained by inverting the noncentral t sampling distribution of the rescaled statistic $t = \hat{d}\sqrt{n_1 n_2/(n_1+n_2)}$, which under the independent groups equal variances model is exactly noncentral t with $n_1 + n_2 - 2$ degrees of freedom and noncentrality parameter $\lambda = \delta\sqrt{n_1 n_2/(n_1+n_2)}$ (Hedges, 1981). The confidence limits are then rescaled back to the δ metric. This is the same construction Steiger and Fouladi (1997) and Kelley (2007) describe for noncentral effect size CIs.

Usage

```
ci_smd(
  ncp = NULL,
  smd = NULL,
  n_1 = NULL,
  n_2 = NULL,
```

```
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  tol = 1e-09,
  ...
)
```

Arguments

ncp	The estimated noncentrality parameter, this is generally the observed *t*-statistic from comparing the two groups and assumes homogeneity of variance
smd	The standardized mean difference (using the pooled standard deviation in the denominator)
n_1	The sample size for Group 1
n_2	The sample size for Group 2
conf_level	The confidence level (1-Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused
alpha_lower	The Type I error rate for the lower tail
alpha_upper	The Type I error rate for the upper tail. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in `ci_nc_t`).
tol	The tolerance of the iterative method for determining the critical values
...	Allows one to potentially include parameter values for inner functions

Details

ncp-input vs. smd-input paths. The function accepts the effect size in either of two equivalent metrics: the observed *t*-statistic (via `ncp`) or the sample standardized mean difference (via `smd`). The two paths are mathematically equivalent under the equal variances assumption (since $t = \hat{d}\sqrt{n_1 n_2/(n_1+n_2)}$); pick whichever is easier to obtain. Supply exactly one. Both paths internally call `ci_nc_t` to invert the noncentral *t* distribution at the specified two-tailed (or asymmetric, via `alpha_lower` / `alpha_upper`) confidence level.

Independent vs.\ paired comparison. `ci_smd` assumes two *independent* groups with a common variance. DMAR does not currently provide a confidence interval for the standardized mean difference in a paired or within-subject design, whose sampling distribution depends on the correlation between the paired measurements; applying the independent groups interval to paired data gives the wrong coverage. (`ci_smd_c` is not a paired interval either; it is the interval for Glass's estimator, which standardizes the difference between two independent groups by the control group standard deviation.)

Bias correction (Hedges' g). `ci_smd` reports the CI on *d*; if the bias-corrected *g* is desired, multiply the bounds by the Hedges and Olkin (1985) correction factor $J(\nu) = 1 - 3/(4\nu - 1)$ (with $\nu = n_1 + n_2 - 2$). Because $J(\nu)$ is a constant, the rescaling preserves coverage.

Value

A 3-row `data.frame` with columns `term` and `value`. The `term` values are `"ci_lower"` (the lower bound of the confidence interval on the standardized mean difference), `"smd"` (the point estimate), and `"ci_upper"` (the upper bound).

Warning

This function uses `ci_nc_t`, which has as one of its arguments `tol` (and can be modified with `tol` of the present function). If the present function fails to converge (i.e., if it runs but does not report a solution), it is likely that the `tol` value is too restrictive and should be increased by a factor of 10, but probably by no more than 100. Running the function `ci_nc_t` directly will report the actual probability values of the limits found. This should be done if any modification to `tol` is necessary in order to ensure acceptable confidence limits for the noncentral *t* parameter have been achieved.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. Academic Press.

Kelley, K. (2005). The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals. *Educational and Psychological Measurement, 65*(1), 51–69. doi:10.1177/0013164404264850

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

smd, smd_c, ci_smd_c, ss_aipe_smd, ss_power_smd, plot_smd, ci_nc_t

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
# Steiger and Fouladi (1997) example values.
ci_smd(ncp = 2.6, n_1 = 10, n_2 = 10, conf_level = 1 - .05)
ci_smd(ncp = 2.4, n_1 = 300, n_2 = 300, conf_level = 1 - .05)
```

---

ci_smd_c *Confidence Limits for the Standardized Mean Difference Using the Control Group Standard Deviation as the Divisor*

Description

Computes the exact noncentral *t*-based confidence limits for the standardized mean difference that uses the control group standard deviation as the divisor (Glass's *g*). Standardizing by the control group alone keeps the scale of the effect anchored in the untreated population, which matters when the treatment may alter variability as well as the mean.

Usage

```
ci_smd_c(
  ncp = NULL,
  smd_c = NULL,
  n_C = NULL,
  n_E = NULL,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  tol = 1e-09,
  ...
)
```

Arguments

ncp	The estimated noncentrality parameter, this is generally the observed *t*-statistic from comparing the control and experimental group (assuming homogeneity of variance)
smd_c	The standardized mean difference (using the control group standard deviation in the denominator)
n_C	The sample size for the control group
n_E	The sample size for experimental group
conf_level	The confidence level (1-Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused
alpha_lower	The Type I error rate for the lower tail
alpha_upper	The Type I error rate for the upper tail. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in `ci_nc_t`).
tol	The tolerance of the iterative method for determining the critical values
...	Potentially include parameter for inner functions

Value

A 3-row `data.frame` with columns `term` and `value`. The `term` values are `"ci_lower"` (the lower bound of the confidence interval), `"smd_c"` (the standardized mean difference standardized by the control group standard deviation), and `"ci_upper"` (the upper bound).

Warning

This function uses `ci_nc_t`, which has as one of its arguments `tol` (and can be modified with `tol` of the present function). If the present function fails to converge (i.e., if it runs but does not report a solution), it is likely that the `tol` value is too restrictive and should be increased by a factor of 10, but probably by no more than 100. Running the function `ci_nc_t` directly will report the actual probability values of the limits found. This should be done if any modification to `tol` is necessary in order to ensure acceptable confidence limits for the noncentral *t* parameter have been achieved.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Glass, G. V. (1976). Primary, secondary, and meta-analysis of research. *Educational Researcher, 5*, 3–8.

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`smd_c`, `smd`, `ci_smd`, `ci_nc_t`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_snr()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
ci_smd_c(smd_c = .5, n_C = 100, n_E = 100, conf_level = .95)
```

---

ci_snr *Confidence Interval for the Signal-to-Noise Ratio*

Description

Computes the exact confidence interval for the signal-to-noise ratio in a fixed effects analysis of variance, the variance due to the factor of interest divided by the error variance, expressing the magnitude of an effect relative to the unexplained variability.

Usage

```
ci_snr(
  F_value = NULL,
  df_1 = NULL,
  df_2 = NULL,
  N = NULL,
```

```
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  ...
)
```

Arguments

F_value	Observed *F*-value from the analysis of variance
df_1	Numerator degrees of freedom
df_2	Denominator degrees of freedom
N	Sample size
conf_level	Confidence interval coverage (i.e., 1 - Type I error rate), default is .95
alpha_lower	Type I error for the lower confidence limit
alpha_upper	Type I error for the upper confidence limit. The table records the coverage, `1 - alpha_lower - alpha_upper`, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, `results_sentence`, and the `alpha_lower` and `alpha_upper` columns of `tidy()` state; with a tail area of 0 the interval is a one-sided confidence bound.
...	Allows one to potentially include parameter values for inner functions

Details

The confidence level must be specified in one of following two ways: using confidence interval coverage (`conf_level`), or lower and upper confidence limits (`alpha_lower` and `alpha_upper`).

This function uses the confidence interval transformation principle (Steiger, 2004) to transform the confidence limits for the noncentrality parameter to the confidence limits for the population's signal-to-noise ratio. The confidence interval for noncentral *F* parameter can be obtained from the `ci_nc_F` function in DMAR, which is used internally within this function.

Value

A 2-row `data.frame` with columns `term` and `value`. The `term` values are `"ci_lower"` and `"ci_upper"`, giving the lower and upper confidence limits on the signal-to-noise ratio.

Note

The signal to noise ratio is defined as the variance due to the particular factor over the error variance (i.e., the mean square error).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bargmann, R. E. (1970). Interpretation and use of a generalized discriminant function. In R. C. Bose, I. M. Chakravarti, P. C. Mahalanobis, C. R. Rao, & K. J. C. Smith (Eds.), *Essays in probability and statistics* (pp. 35–60). University of North Carolina Press.

Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40*(3), 659–670. doi:10.1177/001316448004000309

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Steiger, J. H. (2004). Beyond the *F* Test: Effect size confidence intervals and tests of close fit in the Analysis of Variance and Contrast Analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Venables, W. (1975). Calculation of confidence intervals for noncentrality parameters. *Journal of the Royal Statistical Society, Series B, 37*(3), 406–412. doi:10.1111/j.25176161.1975.tb01554.x

See Also

`ci_srsnr`, `ci_nc_F`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_src()`, `ci_srsnr()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
## Bargmann (1970) gave an example of a one-way ANOVA with an observed
## F value of 11.2213 on 4 and 50 degrees of freedom, which Venables (1975),
## Fleishman (1980), and Steiger (2004) used in turn. Steiger describes it
## as five groups of 11 and reports F as 11.221, the value used here.
## The exact confidence interval for the signal-to-noise ratio of that example:

ci_snr(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55)

ci_snr(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55, conf_level = .90)

ci_snr(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55, alpha_lower = .02, alpha_upper = .03)
```

---

ci_src *Confidence Interval for a Standardized Regression Coefficient*

Description

Computes a confidence interval for a population standardized regression coefficient, by the standard *t*-based approach or the noncentral *t* approach. A thin convenience wrapper around `ci_reg_coef`, which is the general engine; for the coefficient in its raw metric use `ci_rc`.

Usage

```
ci_src(
  std_reg_coef_j = NULL,
  se_std_reg_coef_j = NULL,
  N = NULL,
  p = NULL,
  R2_Y_X = NULL,
  R2_j_X_without_j = NULL,
  conf_level = 0.95,
  R2_Y_X_without_j = NULL,
  t_value = NULL,
  reg_coef_j = NULL,
  se_reg_coef_j = NULL,
  sd_Y = NULL,
  sd_X = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL
)
```

Arguments

std_reg_coef_j	The standardized regression coefficient, or NULL (the default) when it is given in the units of the data as reg_coef_j or when the other inputs determine it (see Details)
se_std_reg_coef_j	The standard error of the standardized regression coefficient
N	Sample size
p	The number of predictors
R2_Y_X	The squared multiple correlation coefficient predicting *Y* from the *p* predictor variables
R2_j_X_without_j	The squared multiple correlation coefficient predicting the *j*th predictor variable (i.e., the predictor of interest) from the remaining *p*-1 predictor variables
conf_level	Desired level of confidence for the computed interval (i.e., 1 - the Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused
R2_Y_X_without_j	The squared multiple correlation coefficient predicting *Y* from the *p*-1 predictor variable with the *j*th predictor of interest excluded
t_value	The *t*-value evaluating the null hypothesis that the population regression coefficient for the *j*th predictor equals zero
reg_coef_j	The unstandardized regression coefficient
se_reg_coef_j	The standard error of the unstandardized regression coefficient
sd_Y	Standard deviation of *Y*, the dependent variable
sd_X	Standard deviation of *X*, the predictor variable of interest
alpha_lower	The Type I error rate for the lower confidence interval limit

alpha_upper — The Type I error rate for the upper confidence interval limit. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound. A nonzero tail area below 1e-9 is refused (see "Small tail areas" in ci_nc_t).

Details

For standardized variables, do not specify the standard deviation of the variables and input the standardized regression coefficient for std_reg_coef_j.

The interval needs N, p, and one of these: the coefficient (std_reg_coef_j, or reg_coef_j with sd_Y and sd_X) with its standard error (se_std_reg_coef_j or se_reg_coef_j), with t_value, or with R2_Y_X and R2_j_X_without_j; t_value with se_std_reg_coef_j, or with R2_Y_X and R2_j_X_without_j; or the three squared multiple correlations R2_Y_X, R2_Y_X_without_j, and R2_j_X_without_j. The squared multiple correlations fix the magnitude of the coefficient but not its sign: the positive root is returned, with a warning, as in ci_reg_coef. A call that gives none of these is refused with a condition of class dmar_argument_invalid. The interval is computed by ci_reg_coef, and its conditions name the arguments of this function (std_reg_coef_j, not reg_coef_j). A standardized coefficient larger than 1.1 in absolute value is reported with a warning of class dmar_result_qualified, since a coefficient that large is more often one in the units of the data.

Fixed and random predictors. The standard error of the standardized coefficient, whether given as se_std_reg_coef_j, rescaled from se_reg_coef_j by sd_X / sd_Y, or computed from the squared multiple correlations, treats the sample standard deviations of *Y* and of X_j as fixed constants (Kelley & Maxwell, 2003, pp. 315–316). The standard deviation of *Y* varies from sample to sample in every design, and those of the predictors vary as well when the predictors are random, as they are in most observational studies, so this standard error is approximate either way and the interval does not have exactly its nominal coverage. When the values of the predictors are set by the design, the standard error is, in large samples, too large whenever the population coefficient is not 0, so the interval covers above its nominal level. When the predictors are random, the standard error can be too large or too small, and which it is depends on the correlations among *Y* and the predictors, not on R^2 alone. In 2,193 two-predictor structures drawn with each of the three correlations uniform between -.9 and .9 (structures whose correlation matrix had an eigenvalue below .02 were set aside), the large-sample standard error of the first coefficient was too small in a quarter of them: in none of the 743 in which the predictors explained at most 30 percent of the variance of *Y*, in 8 percent of those explaining 30 to 60 percent, 56 percent of those explaining 60 to 80, and 79 percent of those explaining 80 to 90, and in all 70 that explained more than 90 percent. A structure outside that range can reverse the last: with one predictor correlated .96 with *Y*, a second correlated .1 with *Y*, and the two uncorrelated ($R^2 = .93$), the large-sample standard error of the first coefficient is 2.2 times its sampling standard deviation, and at *N* = 200 both intervals covered it in every one of 4,000 samples.

In a simulation of multivariate normal data with random predictors (20 conditions with 1 to 4 predictors and *N* from 50 to 1,000; 4,000 or 20,000 samples per condition, with Monte Carlo standard errors of .001 to .005), the 95 percent interval of this function covered the population standardized coefficient .961 to .998 of the time with a single predictor correlated .3 to .7 with *Y*; .964 to .979 for coefficients of .34 to .52 among two to four predictors correlated positively with *Y* and with one another and explaining less than 60 percent of its variance; .952 to .956 in the five conditions

whose coefficient was 0 or near it (at most .17 in absolute value); and .955 to .965 for suppressors (a predictor correlated 0 to .1 with *Y* and .6 to .8 with another). With two uncorrelated predictors that explained 72 and 90 percent of the variance of *Y*, it covered .982 and .937. The central *t* interval with the same standard error (that of `ci_reg_coef` with `noncentral = FALSE`) covered less: .882 to .952 for the suppressors, and .938 and .881 in the last two conditions. With the values of the predictors held fixed from sample to sample instead (six conditions), this function's interval covered .971 to .998 whenever the coefficient was not 0.

Value

A 3-row `data.frame` with columns `term`, `value`, `prob_less`, and `prob_greater`. The `term` values are `"ci_lower"`, `"src"` (the standardized regression coefficient point estimate), and `"ci_upper"`, so the estimate sits between its confidence limits. The `prob_less` and `prob_greater` columns report the achieved tail probabilities at each limit when the noncentral t method is used (`NA` for the estimate row).

Note

This function calls upon `ci_reg_coef` in DMAR, but has a different naming scheme. See `ci_reg_coef` for more details.

To form a confidence interval for the unstandardized regression coefficient, use `ci_rc`. This function is used to form a confidence interval for the standardized regression coefficient.

Not all of the values need to be specified, only those that contain all of the necessary information in order to compute the confidence interval (options are thus given for the values that need to be specified).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

Smithson, M. (2003). *Confidence intervals*. Thousand Oaks, CA: Sage Publications.

Steiger, J. H. (2004). Beyond the *F* Test: Effect size confidence intervals and tests of close fit in the Analysis of Variance and Contrast Analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

See Also

ss_aipe_reg_coef, ci_nc_t, ci_reg_coef, ci_rc

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_srsnr(), contrast_adjusted(), plot_smd()

Examples

```
ci_src(std_reg_coef_j = .6707, .1761, N = 30, p = 6, conf_level = .95)
```

---

ci_srsnr *Confidence Interval for the Square Root of the Signal-to-Noise Ratio*

Description

Computes the exact confidence interval for the square root of the signal-to-noise ratio, the standard deviation of the group means relative to the error standard deviation. On this root scale the quantity is an effect size in standard deviation units, the multi-group analog of the standardized mean difference.

Usage

```
ci_srsnr(
  F_value = NULL,
  df_1 = NULL,
  df_2 = NULL,
  N = NULL,
  means = NULL,
  var_within = NULL,
  n_per_group = NULL,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  ...
)
```

Arguments

F_value	Observed *F*-value from the analysis of variance. Use this argument when re-analyzing existing data.
df_1	Numerator degrees of freedom
df_2	Denominator degrees of freedom
N	Sample size

means	Numeric vector of group means, one per group and at least two, given with var_within and n_per_group in place of F_value: the observed sample means of a one-way design, from which the function computes the observed *F*, or population means for a hypothetical interval (see Details).
var_within	The within-group variance that goes with means, a single positive number: the pooled within-group mean square (the mean square error of the one-way analysis of variance), or a population within-group variance for a hypothetical interval.
n_per_group	The group sizes that go with means: a single size for equal groups, or one size per group, each a whole number of at least 2.
conf_level	Confidence interval coverage (i.e., 1 - Type I error rate); default is .95
alpha_lower	Type I error for the lower confidence limit
alpha_upper	Type I error for the upper confidence limit. The table records the coverage, 1 - alpha_lower - alpha_upper, as its level and, when the two tail areas differ, the areas themselves, which the line beneath the printed table, results_sentence, and the alpha_lower and alpha_upper columns of tidy() state; with a tail area of 0 the interval is a one-sided confidence bound.
...	Allows one to potentially include parameter values for inner functions

Details

The confidence level is given in one of two ways: as the coverage, conf_level, or as the two tail areas, alpha_lower and alpha_upper.

The square root of the signal-to-noise ratio is the standard deviation of the group means, each weighted by its share of the total sample size n_j/N, divided by the within-group standard deviation; it is $\sqrt{\lambda/N}$ for the noncentrality parameter λ of the *F* test, and it is Cohen's f with those weights (see cohen_f). This function uses the confidence interval transformation principle (Steiger, 2004) to transform the confidence limits for the noncentrality parameter to the confidence limits for the square root of the signal-to-noise ratio. The confidence interval for the noncentrality parameter itself is given by ci_nc_F.

The interval has two routes. The first takes the observed *F*, its degrees of freedom, and *N*. The second takes the group means $\bar{y}_j$, the pooled within-group mean square MS_{within}, and the group sizes n_j of a one-way design with *k* groups, and computes

$$F = \frac{\sum_j n_j(\bar{y}_j - \bar{y})^2/(k-1)}{MS_{within}}, \qquad \bar{y} = \frac{1}{N}\sum_j n_j\bar{y}_j,$$

with $df_1 = k - 1$, $df_2 = N - k$, and $N = \sum_j n_j$. With the observed sample means and mean square, this is the observed *F*, and the interval is the one the first route gives for the same data. Values of df_1, df_2, or N given with means must agree with these. Group means that are all equal give $F = 0$, which is refused, as it is on the first route: at $F = 0$ the upper confidence limit is undefined.

Population means and a population within-group variance can be given in the same places. The result is then a hypothetical interval: the interval a study with these group sizes would report if its sample means and its pooled within-group mean square equaled the values given. It is not an interval from observed data, and it is not the expected interval of such a study: the limits of the interval vary from sample to sample, and the interval computed at the population values need not equal their average.

Value

A 2-row `data.frame` with columns `term` and `value`. The `term` values are `"ci_lower"` and `"ci_upper"`, giving the square roots of the corresponding signal-to-noise-ratio confidence limits.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bargmann, R. E. (1970). Interpretation and use of a generalized discriminant function. In R. C. Bose, I. M. Chakravarti, P. C. Mahalanobis, C. R. Rao, & K. J. C. Smith (Eds.), *Essays in probability and statistics* (pp. 35–60). University of North Carolina Press.

Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40*(3), 659–670. doi:10.1177/001316448004000309

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Steiger, J. H. (2004). Beyond the *F* Test: Effect size confidence intervals and tests of close fit in the Analysis of Variance and Contrast Analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Venables, W. (1975). Calculation of confidence intervals for noncentrality parameters. *Journal of the Royal Statistical Society, Series B, 37*(3), 406–412. doi:10.1111/j.25176161.1975.tb01554.x

See Also

`ci_snr`, `ci_nc_F`

Other confidence intervals for effect sizes: `ci_R2()`, `ci_c()`, `ci_c_ancova()`, `ci_c_ancova_bp()`, `ci_correlation`, `ci_cv()`, `ci_eta_squared()`, `ci_eta_squared_generalized()`, `ci_mahalanobis()`, `ci_omega_squared()`, `ci_pvaf()`, `ci_ratio_of_means()`, `ci_ratio_of_means_paired()`, `ci_rc()`, `ci_reg_coef()`, `ci_rmsea()`, `ci_sc()`, `ci_sc_ancova()`, `ci_sm()`, `ci_smd()`, `ci_smd_c()`, `ci_snr()`, `ci_src()`, `contrast_adjusted()`, `plot_smd()`

Examples

```
## Bargmann (1970) gave an example of a one-way ANOVA with an observed
## F value of 11.2213 on 4 and 50 degrees of freedom, which Venables (1975),
## Fleishman (1980), and Steiger (2004) used in turn. Steiger describes it
## as five groups of 11 and reports F as 11.221, the value used here.
## The exact confidence interval for the square root of the signal-to-noise ratio of that example:

ci_srsnr(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55)

ci_srsnr(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55, conf_level = .90)

ci_srsnr(F_value = 11.221, df_1 = 4, df_2 = 50, N = 55, alpha_lower = .02, alpha_upper = .03)

# The interval from the summaries of an observed one-way design: the
# group means, the pooled within-group mean square, and the group sizes
```

```
# of the three arms of drinks_trial. The F route gives the same interval
# for the same data.
fit <- stats::lm(log_drinks ~ treatment, data = drinks_trial)
ci_srsnr(means = tapply(drinks_trial$log_drinks, drinks_trial$treatment, mean),
         var_within = sum(stats::residuals(fit)^2) / stats::df.residual(fit),
         n_per_group = as.vector(table(drinks_trial$treatment)))
ci_srsnr(F_value = stats::anova(fit)[1, "F value"], df_1 = 2, df_2 = 85, N = 88)

# A hypothetical interval: with population means of 94, 91, 92, and 83, a
# within-group variance of 67.375, and 20 per group, the interval a study
# would report if its summaries equaled these values. It is not the
# expected interval of that study.
ci_srsnr(means = c(94, 91, 92, 83), var_within = 67.375, n_per_group = 20)
```

`ci_tukey_kramer` *Tukey-Kramer Simultaneous Confidence Intervals for Pairwise Contrasts*

Description

Computes the Tukey-Kramer simultaneous confidence intervals for all $a(a-1)/2$ pairwise contrasts among a group means in a one-way design with possibly unequal group sample sizes (Tukey, 1953; Kramer, 1956; Hayter, 1984), and returns the result in tidy long form. Each interval has individual coverage at least the specified `conf_level` and the family-wise coverage is at least `conf_level`.

Usage

```
ci_tukey_kramer(x, group = NULL, conf_level = 0.95)
```

Arguments

`x`	Either (a) a fitted `lm` or `aov` object with a single one-way factor predictor, or (b) a numeric vector of observations, in which case `group` must also be supplied.
`group`	When `x` is a vector, a factor (or coercible to factor) of group labels, same length as `x`.
`conf_level`	Family-wise confidence level. Default `0.95`.

Details

Formula. For groups i, j with means $\bar{y}_i, \bar{y}_j$ and sample sizes n_i, n_j, the Tukey-Kramer simultaneous CI is

$$\bar{y}_i - \bar{y}_j \;\pm\; q_{\alpha,a,\nu}\sqrt{\frac{MS_E}{2}\left(\frac{1}{n_i}+\frac{1}{n_j}\right)},$$

where $q_{\alpha,a,\nu}$ is the upper α quantile of the studentized-range distribution with a groups and ν error degrees of freedom. The quantile is base R's `qtukey` where that is accurate and the exact quantile where it is not: `qtukey` and `ptukey` lose accuracy at small error df (fewer than 9 for up to

6 groups, rising by steps to fewer than 111 with more than 3000 groups), above 10000 and up to 10^8 df, at confidence levels above .999, and, with more than 50 groups, below the .75 point of the distribution (.80 with more than 1000 groups; with 20 groups and 2 error df, `qtukey(.99, 20, 2)` is 44.526 where the exact value is 37.943), and `qtukey` can stop short of its root anywhere. There the distribution function is computed by integrating the probability integral of the range against the chi square distribution of the error variance, and the quantile is its root. `cv_tukey_hsd` describes the computation, the regions, and how they were measured.

Why Tukey-Kramer. Hayter (1984) proved that the Tukey-Kramer procedure is conservative for unbalanced designs (the coverage probability is at least `conf_level`). For balanced designs it reduces to Tukey's HSD and the coverage is exactly `conf_level`.

Adjusted *p*-values. Each pairwise p-value is computed from the studentized-range distribution, $p = P(Q_{a,\nu} > |q|)$, by the same computation as the quantile (`ptukey` where it is accurate, the exact distribution function where it is not).

Value

A `data.frame` with one row per pairwise contrast. Columns: `contrast` (e.g., `"B - A"`), `mean_difference`, `se`, `q_statistic` (the studentized-range q), `ci_lower`, `ci_upper`, `p_adjusted`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Hayter, A. J. (1984). A proof of the conjecture that the Tukey-Kramer multiple comparisons procedure is conservative. *Annals of Statistics, 12*(1), 61–75.

Kramer, C. Y. (1956). Extension of multiple range tests to group means with unequal numbers of replications. *Biometrics, 12*(3), 307–310.

Tukey, J. W. (1953). *The problem of multiple comparisons*. Unpublished manuscript, Princeton University.

See Also

`cv_tukey_hsd`, `ci_dunnett`, `ci_scheffe`, `TukeyHSD`

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# 1. Balanced one-way: the six marketing panels of the test_market
#    data, four outlets per panel, so the procedure is exactly Tukey's
#    HSD. Panels 5 and 6 separate from panel 1.
fit <- lm(brand_movement ~ panel, data = test_market)
```

```
ci_tukey_kramer(fit)

# 2. Same data via vector / group interface:
ci_tukey_kramer(test_market$brand_movement, group = test_market$panel)
```

`ci_two_phase` *Confidence Interval for a Mean Under Two-Phase Sampling With a Proxy Measurement*

Description

The confidence interval for the population mean of a target measurement that is costly to obtain and so is observed on only a random subsample of units, when an inexpensive proxy measurement is observed on every unit of a larger pool. The design is two-phase, or double, sampling (Neyman, 1938): the first phase draws the pool and measures the proxy on all of it, and the second draws a simple random subsample of the pool, the labeled units, and measures the target on them. The regression estimator of the target mean, the row `mean_Y` of the table,

$$\hat{\mu}_Y = \bar{y}_n + \hat{b}\,(\bar{x}_M - \bar{x}_n),$$

adjusts the labeled units' target mean $\bar{y}_n$ by how far their proxy mean $\bar{x}_n$ sits from the proxy mean of the whole pool $\bar{x}_M$, through the least squares slope $\hat{b}$ of the target on the proxy in the labeled units. When the two measurements are strongly correlated, the interval is much narrower than the one the labeled units give alone, which is the point of the design.

Usage

```
ci_two_phase(
  y,
  x,
  x_unlabeled,
  conf_level = 0.95,
  method = c("regression", "prediction_powered"),
  cluster = NULL,
  cluster_unlabeled = NULL
)
```

Arguments

`y`	The target measurement on the n labeled units (numeric, or logical for a binary measurement, taken as 0 and 1).
`x`	The proxy measurement on the same n units, aligned with `y`.
`x_unlabeled`	The proxy measurement on the units of the pool that have no target value, $N = M - n$ of them, where M is the size of the pool.
`conf_level`	Confidence level (default `0.95`).

`method` `"regression"` (default), the regression estimator above; or `"prediction_powered"`, the prediction-powered estimate of Angelopoulos et al. (2023), which gives the proxy the fixed weight $\lambda = 1$, the row `proxy_weight` (see Details).

`cluster`, `cluster_unlabeled`
Optional cluster identifiers for the labeled and the unlabeled units (for example the respondent who contributed several units, or the prompt that generated several items), supplied together or not at all. When given, the standard error is computed from the cluster totals of each unit's contribution to the estimate, and the limits use the *t* distribution on $Q_n - 1$ degrees of freedom, where Q_n is the number of clusters among the labeled units (see Details).

Details

The estimand. The interval is for μ_Y, the mean of the target measurement in the population from which the pool was drawn at random, the labeled units being a simple random subsample of the pool. The proxy enters only as a covariate. It need not be on the target's scale, unbiased for it, or even measured in its units, and whatever its bias, that bias does not enter the estimate; what the proxy contributes is its correlation ρ with the target. In the evaluation of a system whose outputs an automated judge labels, for example, the target is a human assessor's label of an output, the proxy is the judge's label of the same output, and the pool is the set of outputs the judge labels.

Standard error. To first order,

$$\mathrm{Var}(\hat{\mu}_Y) = \frac{\sigma_Y^2(1-\rho^2)}{n} + \frac{\rho^2\sigma_Y^2}{M},$$

the first term the residual variation the regression leaves among the n labeled units and the second the sampling variation of the pool itself. The reported standard error estimates $\sigma_e^2/n + \beta^2\sigma_X^2/M$, with the residual mean square of the target on the proxy in the labeled units (on $n-2$ degrees of freedom) for σ_e^2 and $\hat{b}^2$ times the proxy's variance over the whole pool for $\beta^2\sigma_X^2$, and the limits use the *t* distribution on the $n-2$ degrees of freedom of the first term, which dominates when the pool is large. If the pool is itself the whole population of interest, a fixed collection of units every one of which carries the proxy, the estimand is the pool's own target mean and the reported standard error is conservative for it: the second term is then not needed, and the first can be multiplied by $1 - n/M$.

Relation to prediction-powered inference. Writing $\bar{x}_N$ for the proxy mean of the unlabeled units, $\hat{\mu}_Y = \lambda\bar{x}_N + (\bar{y}_n - \lambda\bar{x}_n)$ with $\lambda = \hat{b}N/M$. With $\lambda = 1$ in its place this is the prediction-powered estimate of Angelopoulos et al. (2023): the proxy mean of the unlabeled units, corrected by the mean target-minus-proxy difference of the labeled units. That estimate is unbiased for any proxy, but it uses the proxy efficiently only when the proxy is on the target's scale with a slope near one, while the regression estimator learns the weight from the data; the `proxy_weight` row reports λ, the weight each method applied. `method = "prediction_powered"` reports the interval with standard error $\{s_{Y-X}^2/n + s_{X,N}^2/N\}^{1/2}$ and the *t* distribution on $n-1$ degrees of freedom; Angelopoulos et al. state it with the normal quantile, which the *t* interval approaches as n grows.

Clusters. With `cluster` and `cluster_unlabeled`, each unit's contribution to the linearized estimate is summed within its cluster, and the standard error is $\{Q_n/(Q_n-1)\sum_q T_q^2\}^{1/2}$, the sum running over every cluster of the pool, T_q the total of cluster q, and Q_n the number of clusters among the labeled units. For the regression estimate a labeled unit contributes $e_i/n + \hat{b}(x_i - \bar{x}_M)/M$ and an unlabeled unit $\hat{b}(x_j - \bar{x}_M)/M$, where e_i is its residual; for the prediction-powered estimate a

labeled unit contributes its target-minus-proxy difference less the mean difference, over n, and an unlabeled unit its proxy less $\bar{x}_N$, over N. The term that dominates the variance, the residual term, comes from the labeled clusters alone, so they, and not all the clusters of the pool, set both the degrees of freedom, $Q_n - 1$, and the small-sample factor $Q_n/(Q_n - 1)$. When the labeled units come from a few clusters this matters: the interval then rests on those few, however many clusters the pool has. With one unit per cluster the labeled part of the prediction-powered standard error reduces to s^2_{Y-X}/n, its form without clusters. At least two labeled clusters are required.

A constant proxy or target. The regression estimate needs both measurements to vary among the labeled units. A proxy that takes one value there leaves the slope undefined, and a target that takes one value (a binary target that is 1 on every labeled unit, for example) makes the residual variation and the slope both zero, so the interval would have zero width. Each is refused with an error rather than reported as an interval; more labeled units are the remedy. The prediction-powered estimate does not rest on the regression and is defined in both cases.

Missing values. A labeled unit enters only when its target, its proxy, and (if given) its cluster are all present, and an unlabeled unit only when its proxy (and cluster) is present; the others are set aside, which leaves the design intact when values are missing completely at random. The `n_labeled` and `n_pool` rows count the units used.

Value

A `data.frame` (class `dmar_tbl`), its `method` attribute naming the method, with rows

`mean_Y` the estimate of μ_Y, the population mean of the target.

`se`, `df` its standard error and the degrees of freedom of the *t* reference distribution.

`ci_lower`, `ci_upper` the confidence limits.

`slope` the least squares slope $\hat{b}$ of the target on the proxy in the labeled units. The regression estimate applies it; the prediction-powered estimate does not, and the row is reported for comparison (`NA` when the proxy is constant on the labeled units).

`proxy_weight` the weight λ the estimate gives the proxy mean of the unlabeled units: $\hat{b}N/M$ for the regression estimate and 1 for the prediction-powered estimate.

`correlation` the correlation of target and proxy in the labeled units.

`proxy_bias` the proxy mean minus the target mean in the labeled units, the proxy's bias when the two share a scale. It does not enter the regression estimate.

`se_ratio_target_only` the standard error divided by that of the labeled units' target mean alone ($s_Y/\sqrt{n}$, or its cluster counterpart), so values below 1 are the precision the proxy buys.

`n_labeled`, `n_pool` the numbers of labeled units and of units in the pool, n and M.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Angelopoulos, A. N., Bates, S., Fannjiang, C., Jordan, M. I., & Zrnic, T. (2023). Prediction-powered inference. *Science, 382*(6671), 669–674. doi:10.1126/science.adi6000

Neyman, J. (1938). Contribution to the theory of sampling human populations. *Journal of the American Statistical Association, 33*(201), 101–116. doi:10.1080/01621459.1938.10503378

See Also

`ss_aipe_two_phase` and `ss_aipe_two_phase_cost` to plan the numbers of labeled and pool units, `ss_aipe_two_phase_sensitivity` to evaluate a plan, and `ss_seq_two_phase` for the sequential rule that needs no planning values.

Other confidence intervals: `ci_crossed()`, `ci_proportion()`, `ci_proportion_paired()`

Examples

```
# A brief screening scale (the proxy) is scored for all 2,000
# respondents of a survey; a structured interview (the target) is given
# to a random 150 of them.
set.seed(113)
screen    <- rnorm(2000, mean = 50, sd = 10)
interview <- 20 + 0.6 * screen + rnorm(2000, sd = 5)
labeled   <- sample(2000, 150)
ci_two_phase(y = interview[labeled], x = screen[labeled],
             x_unlabeled = screen[-labeled])

# The screener is not on the interview's scale: the interview rises
# about 0.6 points per screener point. The prediction-powered estimate
# gives the screener a weight of 1 regardless, so it stays unbiased but
# its interval is wider than the regression estimator's.
ci_two_phase(y = interview[labeled], x = screen[labeled],
             x_unlabeled = screen[-labeled],
             method = "prediction_powered")
```

---

cles *Common-Language Effect Size (McGraw & Wong, 1992)*

Description

Computes the common-language (CL) effect size for two independent groups, defined as the probability that a randomly drawn observation from group 1 exceeds a randomly drawn observation from group 2 under bivariate normality with equal variances:

$$\mathrm{CL} \;=\; \Pr(Y_1 > Y_2) \;=\; \Phi\big(\delta/\sqrt{2}\big),$$

where δ is the population standardized mean difference and Φ is the standard normal cumulative distribution function. When sample sizes are supplied, the confidence interval on CL is constructed by transforming the noncentral t-based CI on Cohen's d (Steiger & Fouladi, 1997; Kelley, 2007)

through $\Phi(\cdot/\sqrt{2})$, which is monotone-increasing so the coverage probability is preserved exactly. This is preferred over the normal-approximation CI on CL commonly seen in applied work (Brooks, Dalal, & Nolan, 2014).

Usage

```
cles(
  smd,
  n_1 = NULL,
  n_2 = NULL,
  conf_level = 0.95,
  smd_ci_lower = NULL,
  smd_ci_upper = NULL
)
```

Arguments

`smd`	Sample standardized mean difference (Cohen's d); a numeric scalar. Positive means group 1 exceeds group 2.
`n_1`, `n_2`	Sample sizes in the two groups; both required when a confidence interval on CL is desired.
`conf_level`	Confidence level for the CI. Default `0.95`.
`smd_ci_lower`, `smd_ci_upper`	Optional pre-computed confidence limits on d. If supplied, these are used directly and the noncentral computation is skipped.

Details

The common-language idea extends to other effect sizes; the common language effect size for correlations is developed by Liu, Carlson, and Kelley (2019).

Background. McGraw & Wong (1992) introduced the CL effect size to make Cohen's d more interpretable: instead of "the means differ by 0.5 SD," one can say "in 64 treated person scores higher than the control person." Under bivariate normality with equal variances, the population probability $\Pr(Y_1 > Y_2)$ equals $\Phi(\delta/\sqrt{2})$, where $\delta = (\mu_1 - \mu_2)/\sigma$ (McGraw & Wong, 1992).

Connection to other measures. CL is identical to the AUC (Area Under the Curve) interpretation of d in receiver-operating analysis. Vargha & Delaney (2000) generalized CL to the nonparametric setting (their A measure) by replacing the population p with its empirical Mann-Whitney estimate; under bivariate normality the two coincide. The success-rate-difference and number-needed-to-treat scales (Kraemer & Kupfer, 2006; see `nnt_from_smd`) are linear transformations of CL: $\mathrm{SRD} = 2\mathrm{CL} - 1$.

Confidence interval construction. Because $\Phi(\cdot/\sqrt{2})$ is monotone-increasing, the CI on CL is obtained by transforming the CI on d: $[\Phi(d_L/\sqrt{2}),\ \Phi(d_U/\sqrt{2})]$. This is an exact-coverage interval (under the noncentral t sampling model) and is more accurate than the normal-approximation CI on CL that uses a Wald-style variance for $\hat{p}$ (Brooks, Dalal, & Nolan, 2014).

Value

A `data.frame` with rows for the point estimate (`cles`) and, when sample sizes or limits on *d* are supplied, the lower and upper CI limits (`cles_ci_lower`, `cles_ci_upper`), the names `randomization_test` gives the same quantity. The *d*-equivalent of each row is also reported for transparency (`smd`, `smd_ci_lower`, `smd_ci_upper`). With an interval the table carries the attributes `conf_level` and `ci_source`, which names `smd` as the estimate whose interval the interval for CL is computed from, so that `results_sentence` reports CL on the table and on its `tidy()` and `glance()` results, and `tidy()` gives the confidence level on the rows of the interval for CL alone.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Brooks, M. E., Dalal, D. K., & Nolan, K. P. (2014). Are common language effect sizes easier to understand than traditional effect sizes? *Journal of Applied Psychology, 99*(2), 332–340. doi:10.1037/a0034745

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kraemer, H. C., & Kupfer, D. J. (2006). Size of treatment effects and their importance to clinical research and practice. *Biological Psychiatry, 59*(11), 990–996. doi:10.1016/j.biopsych.2005.09.014

Liu, X. S., Carlson, R., & Kelley, K. (2019). Common language effect size for correlations. *The Journal of General Psychology, 146*(3), 325–338. doi:10.1080/00221309.2019.1585321

McGraw, K. O., & Wong, S. P. (1992). A common language effect size statistic. *Psychological Bulletin, 111*(2), 361–365. doi:10.1037/00332909.111.2.361

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

Vargha, A., & Delaney, H. D. (2000). A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics, 25*(2), 101–132. doi:10.3102/10769986025002101

See Also

`smd`, `ci_smd`, `nnt_from_smd`

Other effect size estimates: `cliff_delta()`, `correction_for_attenuation()`, `eta_squared()`, `eta_squared_generalized()`, `expected_partial_r()`, `expected_r()`, `expected_smd()`, `nnt_from_smd()`, `omega_squared()`, `probability_of_superiority_paired()`, `proportion_of_superiority()`, `responder_analysis()`, `smd_trimmed()`

Examples

```
# 1. Point estimate only.
cles(smd = 0.5)

# 2. With a noncentral t CI from sample sizes (preferred):
```

```
cles(smd = 0.5, n_1 = 50, n_2 = 50, conf_level = 0.95)

# 3. With a pre-computed CI on d:
cles(smd = 0.5, smd_ci_lower = 0.20, smd_ci_upper = 0.80)

# 4. CL at three reference d values:
cles(smd = 0.2)
cles(smd = 0.5)
cles(smd = 0.8)
```

`cliff_delta` *Cliff's δ Ordinal Effect Size*

Description

Takes two independent samples of scores that can be ordered and returns Cliff's δ, the probability that a randomly drawn observation from the first population exceeds one from the second minus the probability of the reverse, with a confidence interval and the proportions of pairs won, lost, and tied. The statistic uses only the order of the observations, so it needs no assumption of normality or of equal variances and no transformation of a skewed outcome, and its interval comes from a variance formula, not from resampling.

Usage

```
cliff_delta(group_1, group_2, conf_level = 0.95)
```

Arguments

`group_1`, `group_2`	Numeric or logical vectors, or ordered factors with the same levels in the same order, holding the observations in the two groups. The statistic uses only their order. Missing values are removed group by group (see the `n_missing` attribute). An unordered factor and a character vector are refused; see *What is refused, and why*.
`conf_level`	Confidence level for the CI. Default `0.95`.

Details

Estimand and interval. For independent observations Y_1 from the first population and Y_2 from the second,

$$\delta \;=\; \Pr(Y_1 > Y_2) - \Pr(Y_1 < Y_2) \;=\; 2 \cdot A - 1,$$

where A is the Vargha-Delaney (2000) statistic. The sampling units are the $n_1 + n_2$ individuals, each observed once and each in one group. The interval is a large-sample confidence interval for the population δ, formed on the inverse hyperbolic tangent scale. δ ranges over $[-1, 1]$; 0 says that neither group tends to score higher, and 1 or -1 that the two distributions do not overlap. Feng and Cliff (2004, p. 322) credit the statistic to Cliff (1993).

Sample estimate.

$$\hat{\delta} = \frac{\#\{(i,j) : y_{1i} > y_{2j}\} - \#\{(i,j) : y_{1i} < y_{2j}\}}{n_1 n_2},$$

the mean of the $n_1 n_2$ signs $d_{ij} = \operatorname{sign}(y_{1i} - y_{2j})$. A tied pair has sign 0 and counts for neither side. The rows `p_y1_greater`, `p_y1_less`, and `p_tied` give the three proportions of pairs.

Assumptions. The two samples are independent of each other and the observations within each are independent and identically distributed; for paired observations use `probability_of_superiority_paired`. The response is at least ordinal. The estimate needs nothing more: no distributional shape, no equality of variances, and ties are allowed. The interval is a large-sample result; *When the estimated variance is zero* says how it behaves in small samples with a large effect.

Variance. The variance of $\hat{\delta}$ is

$$\operatorname{Var}(\hat{\delta}) \;=\; \frac{(n_2 - 1)\sigma_{d_1}^2 + (n_1 - 1)\sigma_{d_2}^2 + \sigma_d^2}{n_1 n_2}$$

(Feng & Cliff, 2004, Equation 2, p. 323, who credit Cliff, 1993), where $\sigma_{d_1}^2$ is the variance of the row means $d_{i\cdot}$, the mean sign of one observation of the first group against the whole second population, $\sigma_{d_2}^2$ is the variance of the column means $d_{\cdot j}$, and σ_d^2 is the variance of the signs d_{ij}. (The equation as printed there attaches $(n_1 - 1)$ to the variance of the first group's row means and $(n_2 - 1)$ to the second's. The coefficients are the other way around, as the variance of a two-sample U statistic shows: as n_2 grows, $\hat{\delta}$ becomes the mean of n_1 independent row means, whose variance is $\sigma_{d_1}^2/n_1$. The difference matters only for unequal group sizes.) The function estimates the three components by their sample values: the sample variances (divisors $n_1 - 1$ and $n_2 - 1$) of the observed row means and column means, and the sample variance (divisor $n_1 n_2 - 1$) of the $n_1 n_2$ signs. Feng and Cliff's Equation 4, an unbiased estimate with a minimum allowed value, is not used. The result is the row `var_cliff_delta`.

Confidence interval. The interval is formed on the scale of $\operatorname{arctanh}(\delta)$, as Fisher's *Z* interval for a correlation is, and transformed back, so that the limits stay inside $[-1, 1]$:

$$\tanh\left\{ \operatorname{arctanh}(\hat{\delta}) \pm z_{1-\alpha/2} \frac{\sqrt{\widehat{\operatorname{Var}}(\hat{\delta})}}{1 - \hat{\delta}^2} \right\},$$

the standard error on the transformed scale coming from the delta method. Feng and Cliff (2004, Equation 5, p. 324) give an asymmetric interval of another form, which this function does not compute.

Relation to Vargha and Delaney's *A*. $A = (\delta + 1)/2$, and because $\operatorname{arctanh}(2A - 1) = \operatorname{logit}(A)/2$, the interval here and the logit interval of `vargha_delaney_A` are the same construction on the two scales. The two functions estimate the variance differently, however (the three-term estimate above here; the two placement variances of DeLong, DeLong, and Clarke-Pearson, 1988, there), so their limits are not the same after the change of scale. For `group_1 = 1:5` and `group_2 = 3:7`, $\hat{\delta} = -0.64$ with limits -0.935 and 0.177; `vargha_delaney_A()` gives $\hat{A} = 0.18$ with limits 0.034 and 0.576, which are -0.931 and 0.151 on the scale of δ.

Scale and missing values. The observations enter only through their order, so any increasing rescoring of the response gives the same result, and an ordinal response may be supplied as an ordered factor. Missing values (`NA` and `NaN`) are removed group by group; the numbers removed are in the `n_missing` attribute. Infinite values are ordered as usual. Dates and date-times (classes `Date` and `POSIXct`) are ordered by their numeric values, as are two time differences (class `difftime`) in

the same units. A one-column matrix and a one-dimensional array are read as the vectors they are. Each group needs at least two non-missing observations; with fewer the function stops, since no variance can be estimated.

What is refused, and why. The statistic uses the order of the observations and nothing else, so the function must be given objects whose order is the order meant. It refuses, with a condition of class `dmar_scores_not_ordered` (see `dmar_metadata`), an unordered factor (its codes follow the order of `levels()`, alphabetical unless set, so the levels low, mid, high are stored as 2, 3, 1, and $\hat{\delta}$ computed from the codes of the example in *Examples* is -0.19 where the ordered levels give 0.31); a character vector (strings sort alphabetically, `"10"` before `"9"`); an ordered factor beside anything that is not an ordered factor with the same levels in the same order (each factor numbers its own levels from 1); time differences in different units, or one beside plain numbers (each such object is stored in its own units); a `Date` beside a `POSIXct`; and a list, data frame, or matrix of several columns. Nothing is computed in any of these cases. The message says which case applies and what to supply: an ordered factor with its levels from lowest to highest, `factor(x, levels = c("low", "mid", "high"), ordered = TRUE)`, or numeric scores.

When the estimated variance is zero. Two kinds of data give a variance estimate of exactly 0, and they are the only two. *Complete separation*: every observation in one group exceeds every observation in the other, so the estimate is at the end of its range ($\hat{\delta}$ -1 or 1). *No variation*: every observation has the same value, so the estimate is 0. In both, the two limits are reported equal to the estimate, `var_cliff_delta` is 0, the table's `interval_degenerate` attribute names the case, and a warning of class `dmar_interval_degenerate` is given whose `reason` element is `"complete_separation"` or `"no_variation"`; the condition also carries `estimate`, `n` (the two group sizes), and `limit_feng_cliff`. The reported limits are not a confidence interval with the stated coverage. The zero describes the sample: the variance estimate measures how much the comparisons differ from one observation to the next, and without overlap they do not differ. Complete separation is common when the effect is large and the samples are small. With five observations per group from normal populations, it occurred in 19 percent of samples when the population value of δ was 0.71 and in 59 percent when it was 0.92, and across all samples the reported 95 percent limits then covered the population value 79 and about 40 percent of the time, against 98 and 96 percent among the samples that were not separated (20,000 simulated samples each). Feng and Cliff (2004, p. 324, Equations 6 to 8) treat this case by assuming the largest variance $\hat{\delta}$ can have, which gives the limit $(n_b - z^2)/(n_b + z^2)$ when $\hat{\delta} = 1$ (and its negative when $\hat{\delta} = -1$), with n_b the smaller group size; they describe the result as conservative. At the 95 percent level it is -0.123, 0.020, 0.131, 0.445, and 0.678 for n_b of 3, 4, 5, 10, and 20. The warning states that limit. It is not the limit this function reports.

Particular cases. Infinite values are ordered like any others, so a group made up of `Inf` beside finite scores is completely separated, and two groups made up of the same infinite value have no variation. Two constant groups at different values are completely separated. Groups that only touch, the largest score of one equal to the smallest of the other, are not separated: the tied pair gives a positive variance and an ordinary interval, which is wide (`c(3, 4, 5)` against `c(1, 2, 3)` gives $\hat{\delta} = 0.889$ with limits -0.167 and 0.995). The mark travels with the result: `results_sentence` writes the estimate and says that there is no confidence interval, the printed table says so beneath itself, `tidy()` and `glance()` carry it in the column `interval_degenerate`, `randomization_test` passes this warning on, and `ss_aipe_cliff_delta_sensitivity` counts a separated replication as one without an interval. The randomization test remains valid in both cases.

Relation to other measures. Cliff's δ is a linear function of the Vargha-Delaney (2000) A statistic ($\delta = 2A - 1$) and of the Mann-Whitney U statistic ($U/(n_1 n_2) = A$). The common language effect

size computed by cles is $\Pr(Y_1 > Y_2)$ for normal distributions with a common variance; $(\delta + 1)/2$ estimates that probability (with ties split) without those assumptions, which suits skewed outcomes and ordinal scales.

Reading the output. In the third example below the 32 participants given the Community Reinforcement Approach are group 1 and the 37 given standard care are group 2, so there are 1,184 pairs. The CRA participant reports more drinks per week in 29.3 percent of the pairs (p_y1_greater), fewer in 59.3 percent (p_y1_less), and the same number in 11.4 percent (p_tied), which gives $\hat{\delta} = 0.293 - 0.593 = -0.300$, with 95 percent limits -0.535 and -0.021.

Value

A data.frame (classes dmar_cliff_delta and dmar_tbl) with columns term and value and these rows:

cliff_delta The estimate $\hat{\delta}$, in $[-1, 1]$.

ci_lower, ci_upper The confidence limits at conf_level. Both equal the estimate when the estimated variance is 0, and are then not a confidence interval.

var_cliff_delta The estimated variance of $\hat{\delta}$ (see *Variance*).

p_y1_greater The proportion of the $n_1 n_2$ pairs, one observation from each group, in which the observation from group 1 is the larger.

p_y1_less The proportion in which it is the smaller.

p_tied The proportion of tied pairs. The three proportions sum to 1, and cliff_delta is the first minus the second.

The table carries the attributes conf_level; n_missing, the numbers of missing values removed from the two groups, as c(group_1 = , group_2 =); and interval_degenerate, present only when the estimated variance is 0, with the value "complete_separation" or "no_variation". The tidy() and glance() verbs (see dmar_tidiers) add a logical column interval_degenerate.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin, 114*(3), 494–509. doi:10.1037/00332909.114.3.494

DeLong, E. R., DeLong, D. M., & Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics, 44*(3), 837–845. doi:10.2307/2531595

Feng, D., & Cliff, N. (2004). Monte Carlo evaluation of ordinal d with improved confidence interval. *Journal of Modern Applied Statistical Methods, 3*(2), 322–332. doi:10.22237/jmasm/1099267560

Smith, J. E., Meyers, R. J., & Delaney, H. D. (1998). The community reinforcement approach with homeless alcohol-dependent individuals. *Journal of Consulting and Clinical Psychology, 66*(3), 541–548.

Vargha, A., & Delaney, H. D. (2000). A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics, 25*(2), 101–132. doi:10.3102/10769986025002101

See Also

vargha_delaney_A, probability_of_superiority_paired, cles, randomization_test, nnt_from_smd, ss_aipe_cliff_delta

Other effect size estimates: cles(), correction_for_attenuation(), eta_squared(), eta_squared_generalized(), expected_partial_r(), expected_r(), expected_smd(), nnt_from_smd(), omega_squared(), probability_of_superiority_paired(), proportion_of_superiority(), responder_analysis(), smd_trimmed()

Examples

```
# 1. Two groups of different sizes, no ties:
set.seed(113)
a <- rnorm(30, mean = 0, sd = 1)
b <- rnorm(40, mean = 0.5, sd = 1)
cliff_delta(a, b)

# 2. With ties (ordinal data):
o1 <- c(1, 2, 2, 3, 3, 3, 4, 4, 5)
o2 <- c(2, 3, 3, 4, 4, 5, 5, 5)
cliff_delta(o1, o2)

# 3. Robust to right skew. Cliff's delta on the raw, untransformed
#    drinking outcome from the Smith, Meyers, and Delaney (1998)
#    trial, comparing the Community Reinforcement Approach (CRA)
#    against standard care. Because the statistic uses only ranks it
#    needs no normalizing transformation of the heavily skewed
#    outcome, unlike the standardized mean difference.
data(drinks_trial)
cra <- drinks_trial$drinks_per_week[drinks_trial$treatment == "CRA"]
std <- drinks_trial$drinks_per_week[drinks_trial$treatment == "Standard"]
cliff_delta(cra, std)

# 4. An ordinal response is supplied as an ordered factor with its
#    levels stated from lowest to highest. The unordered factors
#    rating_1 and rating_2 themselves are refused, with a message that
#    says why: their codes follow the alphabetical order of the labels
#    (high, low, mid).
rating_1 <- factor(c("low", "mid", "high", "high"))
rating_2 <- factor(c("low", "low", "mid", "high"))
lev <- c("low", "mid", "high")
cliff_delta(factor(rating_1, levels = lev, ordered = TRUE),
            factor(rating_2, levels = lev, ordered = TRUE))

# 5. Completely separated groups: the estimated variance is 0, the two
#    limits equal the estimate, and a warning says that they are not a
#    confidence interval and gives the limit of Feng and Cliff (2004).
separated <- cliff_delta(6:10, 1:5)
```

```
separated
results_sentence(separated)
```

---

cohen_f *Cohen's* f *Effect Size*

Description

Computes Cohen's *f*, the standard deviation of the group means divided by the standard deviation within the groups, from one of three complete specifications: the group means with the within-group variance (and the group sizes, when the groups differ in size), eta squared, or the two standard deviations.

Usage

```
cohen_f(
  means = NULL,
  var_within = NULL,
  n_per_group = NULL,
  eta_squared = NULL,
  sd_means = NULL,
  sd_within = NULL
)
```

Arguments

means	Numeric vector of group means, one per group and at least two: population means for the population *f*, or observed sample means for the plug-in value. Give with var_within.
var_within	The within-group variance, a single positive number: the population variance common to the groups, or, for the plug-in value, the pooled within-group mean square (the mean square error of the one-way analysis of variance). Give with means.
n_per_group	Optional, and used only with means: the group sizes, one per group, which weight each group mean by its share of the total, n_j/N. A single number states equal groups and gives the same result as NULL, the default, which weights the means equally ($1/k$ each, for *k* groups).
eta_squared	Eta squared, the proportion of the variance that the groups account for, a single number in [0, 1). Give it alone. A population value gives the population *f*. A sample eta squared gives the plug-in value $\sqrt{df_1 F/df_2}$, which is larger than the value from the observed means and the pooled mean square (see Details).
sd_means	The standard deviation of the group means, σ_m, a single number of at least 0, computed with divisor *k* (or with the weights n_j/N), not with the divisor *k* - 1 that stats::sd() uses. Give with sd_within.
sd_within	The within-group standard deviation, σ, a single positive number, or, for the plug-in value, the square root of the pooled within-group mean square. Give with sd_means.

Details

Cohen's f is defined with population quantities (Cohen, 1988, p. 276): $f = \sigma_m/\sigma$, where σ_m is the standard deviation of the population means and σ is the standard deviation within the populations. Given population values, the function returns the population *f*. Given sample summaries, it returns the plug-in value from the same calculation. The plug-in value depends on which summaries are given: from the same data, a sample eta squared gives a larger value than the observed group means with the pooled within-group mean square (see below). Sample inputs do not make the result an unbiased estimate of the population f: the spread of sample means includes their sampling error as well as the spread of the population means. Unequal group sizes determine how the group means are weighted; without `n_per_group` the means are weighted equally.

A call gives exactly one complete specification: `means` with `var_within` (and `n_per_group` if wanted), `eta_squared` alone, or `sd_means` with `sd_within`. Each determines f on its own, so an argument of one specification given with an argument of another is refused, as is one half of a pair given alone, or `n_per_group` without `means`; the refusal has class `dmar_argument_invalid`.

From the means, with weights $w_j = 1/k$ by default or $w_j = n_j/N$ when `n_per_group` is given,

$$\bar{\mu} = \sum_j w_j \mu_j, \qquad \sigma_m^2 = \sum_j w_j (\mu_j - \bar{\mu})^2, \qquad f = \sqrt{\sigma_m^2/\sigma^2}.$$

The divisor of σ_m^2 is *k* (or *N*), because the variance is that of a fixed set of means. From eta squared, $f = \sqrt{\eta^2/(1-\eta^2)}$ (Kim, 2016, p. 233), and from the two standard deviations, $f = \sigma_m/\sigma$. Given population values for the same populations, with the same weighting of the means, the three give the same *f*.

Given sample summaries, the three agree only when the within-group variance is defined the same way in each. With the observed means, the group sizes, and the pooled within-group mean square, $SS_{within}/(N-k)$, of a one-way design, $N\sigma_m^2$ is the between-groups sum of squares, so the plug-in value equals $\sqrt{df_1 F/N}$, with *F*, $df_1 = k - 1$, and $df_2 = N - k$ from the analysis of variance. A sample eta squared, $SS_{between}/SS_{total}$, pairs with the within-group variance SS_{within}/N rather than with the mean square, so the value from it, $\sqrt{SS_{between}/SS_{within}} = \sqrt{df_1 F/df_2}$, is larger than the value from the means and the pooled mean square by the factor $\sqrt{N/(N-k)}$. The two are equal when `var_within` is SS_{within}/N (or `sd_within` is its square root). Without `n_per_group`, the means of unequal groups are weighted equally and the value differs from both. The square root of the signal-to-noise ratio of `ci_srsnr` is this f with the weights n_j/N and the pooled mean square; that function gives its confidence interval from *F* or from the same summaries.

Value

A one-row `data.frame` (printed through `dmar_tbl`) with columns `term` and `value`; `term` is `"cohen_f"` and `value` is f at full precision.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Kim, H.-Y. (2016). Statistical notes for clinical researchers: Sample size calculation 3. Comparison of several means using one-way ANOVA. *Restorative Dentistry & Endodontics, 41*(3), 231–234. doi:10.5395/rde.2016.41.3.231

See Also

ci_srsnr, ci_snr, ss_power_one_way_anova, ss_power_R2

Examples

```
# Kim (2016, Table 1) plans a four-group study around means of 18.4,
# 22.2, 25.1, and 32.1 with a common standard deviation of 8.6, and
# reports f as .58. At full precision, this f plans a balanced study of
# 40, ten per group, for power .80 at the .05 level, as the article does.
f_kim <- cohen_f(means = c(18.4, 22.2, 25.1, 32.1), var_within = 8.6^2)
f_kim
ss_power_one_way_anova(a = 4, true_f = f_kim$value, desired_power = .80)

# Cohen (1988, p. 277) places seven population means over half a
# within-population standard deviation, one at each end and five at the
# midpoint, and gives f as .13 (.267 times .5, his Equation 8.2.8).
cohen_f(means = c(-.25, 0, 0, 0, 0, 0, .25), var_within = 1)

# The plug-in value from the summaries of an observed one-way design
# with unequal groups: the group means, the pooled within-group mean
# square, and the group sizes of the three arms of drinks_trial.
fit <- stats::lm(log_drinks ~ treatment, data = drinks_trial)
cohen_f(means = tapply(drinks_trial$log_drinks, drinks_trial$treatment, mean),
        var_within = sum(stats::residuals(fit)^2) / stats::df.residual(fit),
        n_per_group = as.vector(table(drinks_trial$treatment)))

# The sample eta squared of the same fit gives a larger plug-in value,
# by the factor sqrt(N / (N - k)) = sqrt(88 / 85) (see Details).
ss <- stats::anova(fit)[["Sum Sq"]]
cohen_f(eta_squared = ss[1] / sum(ss))

# From eta squared, and from the two standard deviations:
cohen_f(eta_squared = .10)
cohen_f(sd_means = 4, sd_within = 8)
```

---

cohen_h *Cohen's* h *Effect Size for a Difference Between Two Proportions*

Description

Computes Cohen's h, the effect size for the difference between two proportions on the arcsine (variance-stabilizing) scale,

$$h = \varphi_1 - \varphi_2, \qquad \varphi_i = 2 \arcsin\sqrt{p_i}.$$

The arcsine transform spaces proportions so that a given *h* carries the same detectability wherever the proportions sit, which a raw difference $p_1 - p_2$ does not: a shift from .01 to .05 is easier to detect than one from .41 to .45, and *h* reflects that while the raw difference does not. Cohen's *h* is the proportion analog of the standardized mean difference (smd): the effect size on which power and sample size planning for a difference between two proportions is conventionally based.

Usage

```
cohen_h(p_1, p_2)
```

Arguments

p_1, p_2	The two proportions, each in $[0, 1]$. h is $\varphi(p_1) - \varphi(p_2)$, so it is positive when p_1 is the larger.

Details

Cohen's h is a population quantity: supplied with population proportions it returns the population value, supplied with sample proportions it returns the corresponding sample value. It is signed, positive when p_1 exceeds p_2; its magnitude abs() is the size of the effect irrespective of direction.

Value

A 1-row data.frame with columns term and value; term is "cohen_h" and value is the signed effect size.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum. (See Chapter 6.)

See Also

smd for the standardized mean difference, cohen_f, ci_proportion

Examples

```
# A shift from .40 to .55.
cohen_h(p_1 = 0.55, p_2 = 0.40)

# The same raw difference near the floor is a larger h, since a difference is
# easier to detect where the proportions are small.
cohen_h(p_1 = 0.20, p_2 = 0.05)

# Its magnitude is the size irrespective of direction.
abs(cohen_h(p_1 = 0.40, p_2 = 0.55)$value)
```

cohen_kappa *Cohen's Kappa Coefficient of Inter-Rater Agreement*

Description

Computes Cohen's (1960) kappa coefficient of agreement between two raters on a categorical variable, with optional weighting (linear, quadratic, or custom) for ordinal categories, following Cohen's (1968) weighted kappa. Input can be the two raters' classification vectors or a published $k \times k$ frequency table. A confidence interval and a Wald test of $H_0 : \kappa = 0$ are returned, using the asymptotic standard error of Fleiss, Cohen, and Everitt (1969).

Usage

```
cohen_kappa(
  rater_1 = NULL,
  rater_2 = NULL,
  table = NULL,
  weights = "unweighted",
  weight_scaling = c("agreement", "disagreement"),
  categories = NULL,
  conf_level = 0.95,
  ci_method = c("wald", "percentile", "bca"),
  B = 10000L,
  seed = NULL
)
```

Arguments

rater_1	First rater's classification vector (categorical, factor or coercible to factor).
rater_2	Second rater's classification vector (same length as rater_1; categorical, factor or coercible to factor). Together rater_1 and rater_2 give the two raters' classifications on the same set of subjects; pairs in which either is NA are dropped before computation. SPSS user-defined missing values, which haven::read_sav(user_na = TRUE) keeps as values of a haven_labelled_spss vector, are NA here (as SPSS and **haven**'s is.na() treat them), whether or not **haven** is loaded, so a code declared missing is never counted as a category.
table	A $k \times k$ frequency table of joint assignments (rows are rater 1's categories, columns rater 2's, in the same order), supplied instead of rater_1 and rater_2. This is the form in which published agreement studies usually report their data. Row and column dimnames, when present, must match and are used as the category labels.
weights	Either the string "unweighted" (the default; appropriate for nominal categories), "linear", or "quadratic", or a user-supplied square numeric weight matrix. A custom matrix is interpreted on the scale named by weight_scaling. Asymmetric matrices are allowed; see *Details*.

`weight_scaling` How a custom `weights` matrix is scaled: `"agreement"` (the default; diagonal 1, decreasing off-diagonal entries, typically in $[0, 1]$) or `"disagreement"` (Cohen's 1968 ratio-scaled disagreement weights: zero diagonal, larger entries for graver disagreements). The two scalings give identical κ_W; see *Details*.

`categories` Optional character vector listing the full category set in display order (used both to set the order of the confusion matrix's rows/columns and to map ordinal categories to integers $1, \ldots, k$ for linear and quadratic weighting). When `NULL` (the default) and both raters are factors with the same level set, the factor levels are used in their own order; otherwise the sorted union of values observed in `rater_1` and `rater_2` is used. When supplied, the set must be unique, non-missing, and contain every observed rating; an omitted category raises an error, because silently dropping the corresponding ratings while still dividing by the original sample size would corrupt kappa. With `table` input, supply `categories` only when the table has no `dimnames`.

`conf_level` Confidence level for the interval (default `0.95`).

`ci_method` Interval method: `"wald"` (the default, the asymptotic interval from the Fleiss, Cohen, and Everitt standard error), `"percentile"` (bootstrap percentile), or `"bca"` (bootstrap bias-corrected and accelerated).

`B` Number of bootstrap replications when `ci_method` is `"percentile"` or `"bca"`, a single whole number (ignored for `"wald"`). Default `10000`, because kappa is cheap to recompute on a resample. The fewest accepted at the default `conf_level` is 400 for the percentile interval (200 at 0.90, 2000 at 0.99) and 2000 for the BCa interval (1000 at 0.90, 10000 at 0.99), whose adjustment moves the working quantiles into the tails; a smaller or fractional `B` is refused before anything is computed (class `dmar_count_invalid`). See *Bootstrap replications* under *Details* and `dmar_bootstrap`.

`seed` Optional integer seed for the bootstrap. The default `NULL` uses the current state of the random number generator; a supplied seed is set internally and the prior state restored on exit.

Details

For two raters and a confusion matrix P of joint proportions (rater 1 $\times$ rater 2), the weighted kappa is

$$\kappa_W = \frac{p_o^{(W)} - p_e^{(W)}}{1 - p_e^{(W)}}, \quad p_o^{(W)} = \sum_{i,j} W_{ij}\, P_{ij}, \quad p_e^{(W)} = \sum_{i,j} W_{ij}\, p_{i\cdot}\, p_{\cdot j},$$

where $p_{i\cdot}$ and $p_{\cdot j}$ are the row and column marginals. For `weights = "unweighted"` (the diagonal of W is 1, off- diagonal 0) this collapses to Cohen's original formulation.

Disagreement scaling. Cohen (1968) develops weighted kappa by ratio scaling *disagreement*: each cell receives a weight $v_{ij} \geq 0$, zero on the agreement diagonal, with, for example, a weight of 6 representing twice as much disagreement as 3. The weights are part of the definition of agreement (and of any hypothesis tested about it), so they must be fixed before the data are collected. κ_W is invariant to multiplying the v_{ij} by any positive constant, and a disagreement matrix is related to an agreement matrix by $w_{ij} = 1 - v_{ij}/v_{\max}$ (Cohen, 1968, Footnote 3), which is the conversion applied internally when `weight_scaling = "disagreement"`. Either scaling therefore yields the same κ_W; supply whichever is more natural.

Asymmetric weights and validity. Nothing in κ_W requires $W_{ij} = W_{ji}$. Symmetric weights suit reliability, where the two sources have equal status; asymmetric weights suit validity, where one source is a criterion and the other a predictor and the two directions of a confusion can carry different costs (Cohen, 1968). The examples reproduce Cohen's computer-diagnosis illustration.

Standard error. The Fleiss-Cohen-Everitt (1969) asymptotic variance for weighted kappa is used:

$$\operatorname{Var}(\hat{\kappa}_W) = \frac{1}{N(1-p_e^{(W)})^2}\Big[\sum_{i,j} P_{ij}\big(W_{ij} - (\bar{W}_{i\cdot} + \bar{W}_{\cdot j})(1-\hat{\kappa}_W)\big)^2 - \big(\hat{\kappa}_W - p_e^{(W)}(1-\hat{\kappa}_W)\big)^2\Big],$$

with $\bar{W}_{i\cdot} = \sum_j W_{ij}\, p_{\cdot j}$ and $\bar{W}_{\cdot j} = \sum_i W_{ij}\, p_{i\cdot}$. The Wald confidence interval is $\hat{\kappa} \pm z_{1-\alpha/2}\,\widehat{\mathrm{SE}}$. Cohen's (1968) own Formulas 10 and 13 for the standard error of κ_W preceded this result and were superseded by it; the examples reproduce his Table 1 arithmetic for the historical record while the function reports the Fleiss-Cohen-Everitt interval.

Choice of weights. Use `"unweighted"` for nominal categories. For ordinal categories, `"quadratic"` is the most common choice (and mathematically equivalent to the intraclass correlation under certain conditions; Fleiss & Cohen, 1973); `"linear"` is also defensible. Cohen (1968) further shows that with equal marginals and quadratic-pattern disagreement weights, κ_W equals the product-moment correlation between the category scores.

Small samples and the bootstrap. The Wald interval can have poor coverage for small N or extreme values of $\hat{\kappa}$; a bootstrap interval is more dependable in those regimes (Blackman & Koval, 2000). With `ci_method = "percentile"` or `"bca"` the subjects (the rated pairs) are resampled with replacement `B` times, kappa is recomputed on each resample with the same categories and weights, and the interval is read off the bootstrap distribution: the percentile interval takes the empirical quantiles, and the BCa interval adjusts the quantile positions for median bias (estimated from the bootstrap distribution) and for acceleration (estimated from the jackknife), making it second-order accurate where the percentile interval is first-order accurate (Efron & Tibshirani, 1993). `table` input is expanded to the equivalent paired ratings and resampled the same way. The `se`, `z_value`, and `p_value` columns keep their asymptotic definitions under every `ci_method`; only the interval changes. Bootstrap results vary from run to run; supply `seed` for reproducibility (the RNG state is set locally and the caller's state restored on exit).

Bootstrap replications. The percentile interval reads its limits from the quantiles of the `B` replicated values of kappa and is run only when `B` puts at least 10 replications beyond each limit (400 at the 95 percent level); the BCa interval, whose limits move farther into the tails, only with 50 beyond each nominal limit (2000). A replication in which kappa is undefined (chance agreement of 1 in the resample, as when every resampled rating falls in one category) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, both limits are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. For the BCa interval the count of 10 is applied again where the bias correction and the acceleration place the limits, and when fewer than 10 replications lie beyond one of them no BCa interval is reported (limits `NA`, the same class, with the two positions as `positions` and a suggested `B` as `B_suggest`). When kappa is undefined with

some case left out (chance agreement 1), when every bootstrap estimate falls on one side of the sample estimate, or when the acceleration is not finite, the BCa interval is refused with an error of class `dmar_bca_undefined`. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the result is not one to report. See `dmar_bootstrap` for the rule and its reasons.

Value

A one-row `data.frame` (class `dmar_tbl`) with columns `weights` (the form used), `kappa`, `se` (asymptotic standard error), `ci_lower`, `ci_upper`, `z_value`, `p_value` (Wald test of $H_0 : \kappa = 0$), `N` (number of paired ratings), and `n_categories` (k). With a bootstrap interval the result carries the attributes `B`, `B_used`, `n_dropped`, `B_required`, and `bootstrap_status` (see *Bootstrap replications* under Details), which `dmar_metadata` reads.

The per-cell detail behind the coefficient travels with the result as the `cells` attribute, in the form of Cohen's (1968) Table 1: a `data.frame` with one row per cell of the confusion matrix giving `rater_1` and `rater_2` (the cell's categories), `observed_proportion`, `expected_proportion` (the product of the marginal proportions, the cell's chance expectation), `weight` (the agreement-scale weight used in the computation), and, when `weight_scaling = "disagreement"`, the supplied `disagreement_weight`. Retrieve it with `attr(result, "cells")`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Blackman, N. J.-M., & Koval, J. J. (2000). Interval estimation for Cohen's kappa as a measure of agreement. *Statistics in Medicine, 19*(5), 723–741.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37–46.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin, 70*(4), 213–220.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Fleiss, J. L., Cohen, J., & Everitt, B. S. (1969). Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin, 72*(5), 323–327.

Fleiss, J. L., & Cohen, J. (1973). The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and Psychological Measurement, 33*(3), 613–619.

See Also

`diagnosis_agreement` (Cohen's 1968 Table 1 as a data set), `fleiss_kappa`, `icc`

Other reliability: `diagnosis_agreement`, `fleiss_kappa()`, `icc()`, `reliability()`, `reliability_H()`, `reliability_alpha()`, `reliability_kr20()`, `reliability_omega()`, `reliability_omega_categorical()`

Examples

```
# ------------------------------------------------------------------------
# Cohen (1968, Table 1): two judges assign N = 200 cases to three
# diagnostic categories. The table ships as the diagnosis_agreement
# data set in the paper's own layout, Judge B in rows and Judge A in
# columns, carrying Cohen's per-cell disagreement weights, observed
# proportions, and chance-expected proportions.
data(diagnosis_agreement)
tab <- xtabs(frequency ~ judge_b + judge_a, data = diagnosis_agreement)
v   <- unclass(xtabs(disagreement_weight ~ judge_b + judge_a,
                     data = diagnosis_agreement))
tab

# Unweighted kappa, all disagreements equal (Cohen's Formula 4): .492.
cohen_kappa(table = tab)

# A percentile bootstrap interval for the same table, which expands
# it to its 200 paired ratings and resamples the subjects. The point
# estimate and the asymptotic standard error are unchanged; only the
# interval is read off the bootstrap distribution. B = 2000 keeps the
# example quick; a reported interval deserves the default B = 10000.
cohen_kappa(table = tab, ci_method = "percentile", B = 2000,
            seed = 113)

# Cohen's ratio-scaled disagreement weights: a neurosis-psychosis
# confusion (weight 6) is six times as grave as a personality
# disorder-neurosis confusion (weight 1). Weighted kappa = .348,
# smaller than the unweighted .492: these judges disagree less than
# chance expectation where it matters little and at about the chance
# level where it matters most.
res <- cohen_kappa(table = tab, weights = v,
                   weight_scaling = "disagreement")
res

# The per-cell quantities of Cohen's Table 1 travel with the result:
# observed and chance-expected proportions and both weight scalings.
attr(res, "cells")

# Interchanging the 6 and 1 weights reverses the story: kappa_w = .574.
v_swap <- v
v_swap[v == 6] <- 1
v_swap[v == 1] <- 6
cohen_kappa(table = tab, weights = v_swap,
            weight_scaling = "disagreement")

# Cohen's own Table 1 arithmetic (his Formulas 8, 10, and 13),
# computed straight from the data set's per-cell columns and
# reproduced for the historical record. The function reports the
# Fleiss-Cohen-Everitt (1969) standard error, which superseded
# Formulas 10 and 13.
q_o <- with(diagnosis_agreement,
            sum(disagreement_weight * observed_proportion))   # = .90
```

```
q_c <- with(diagnosis_agreement,
            sum(disagreement_weight * expected_proportion))   # = 1.38
1 - q_o / q_c                          # kappa_w = .348   (Formula 8)
v2_o <- with(diagnosis_agreement,
             sum(disagreement_weight^2 * observed_proportion))
v2_c <- with(diagnosis_agreement,
             sum(disagreement_weight^2 * expected_proportion))
sqrt((v2_o - q_o^2) / (200 * q_c^2))   # = .0901   (Formula 10)
sqrt((v2_c - q_c^2) / (200 * q_c^2))   # = .0916   (Formula 13)
(1 - q_o / q_c) + c(-1, 1) * qnorm(0.975) * 0.0901  # 95% CI [.171, .524]
(1 - q_o / q_c) / 0.0916               # z = 3.80, p < .001

# Cohen's validity reinterpretation: Judge A is a diagnostic panel
# (the criterion), Judge B a computer diagnosis (the predictor), and
# the weights are asymmetric because the two directions of a
# confusion carry different costs. Oriented to the table's layout
# (rows = Judge B = computer, columns = Judge A = panel), the weights
# below reproduce Cohen's published quantities: sum(v * p_o) = .86,
# sum(v * p_c) = 1.33, kappa_w = .353, with his Formulas 10 and 13
# giving .0887 and .0915. The weighted kappa vignette works this
# example in full, including the orientation of the paper's printed
# weight display relative to these values.
v_validity <- matrix(
  c(0, 1, 2,
    1, 0, 2,
    4, 6, 0), nrow = 3, byrow = TRUE)
cohen_kappa(table = tab, weights = v_validity,
            weight_scaling = "disagreement")

# ---------------------------------------------------------------------
# Raw rater vectors and ordinal categories (quadratic agreement
# weights, e.g., Likert-style severity).
set.seed(113)
x <- sample(1:5, 100, replace = TRUE)
y <- pmin(pmax(x + sample(-1:1, 100, replace = TRUE), 1), 5)
cohen_kappa(x, y, weights = "quadratic")
```

`combine_p` *Combine Independent P-Values Across Studies*

Description

Combines the one-tailed p-values from k independent tests of a common directional hypothesis into a single test, by any of the four classical methods that Raudenbush (1984) applied to the teacher expectancy experiments: Fisher's (1938) "adding logs" chi square, Edgington's (1972) "adding ps", the Mosteller and Bush (1954) "adding Zs" (Stouffer) method, and a weighted adding-Zs variant (weights are typically the studies' degrees of freedom). Combined significance tests answer the narrow question "is there an effect in at least some studies?"; they do not estimate its size. Pair them with `meta_smd` or `meta_es`, which estimate the size of the effect with a confidence interval and, under a random effects model, its heterogeneity across studies.

Usage

```
combine_p(
  p,
  method = c("fisher", "edgington", "stouffer", "stouffer_weighted"),
  weights = NULL
)
```

Arguments

p	Numeric vector of one-tailed p-values, each in (0, 1), oriented so that small values support the common directional hypothesis.
method	Character vector naming the methods to compute: any of "fisher", "edgington", "stouffer", "stouffer_weighted"; the default computes all four (the "stouffer_weighted" row appears only when weights is supplied).
weights	Optional non-negative weights for "stouffer_weighted", one per study; degrees of freedom are the conventional choice (Mosteller & Bush, 1954).

Details

Fisher's statistic is $-2\sum \log p_i$, distributed chi square with $2k$ degrees of freedom under the joint null. Edgington's statistic is the plain sum $\sum p_i$, referred to a normal approximation with mean $k/2$ and variance $k/12$ (accurate for $k \geq 10$; for smaller k it is conservative in the tails). The Stouffer statistic is $\sum z_i/\sqrt{k}$ with $z_i = \Phi^{-1}(1-p_i)$, and the weighted variant is $\sum w_i z_i/\sqrt{\sum w_i^2}$. All four are reported with one-tailed combined p-values, matching the directional inputs.

Methods can disagree, and the disagreement is informative: Rosenthal (1978) notes there is no uniformly best test. In the published analysis, Raudenbush (1984) found three of the four rejecting the null at the .05 level while the df-weighted variant did not, an early warning that large studies were finding smaller effects. Computed from the study-level p-values as tabled, the example below shows two of the four rejecting: Fisher's ($p = .004$) and Stouffer's ($p = .014$) tests reject, Edgington's sits just above the level ($p = .051$; the tabled values sum to 7.00 where the paper's Table 2, p. 90, prints a sum of 6.84 with $p = .04$), and the df-weighted variant is not close ($p = .192$).

Value

A data.frame (class dmar_tbl) with, per requested method, its statistic row(s) and a one-tailed <method>_p row, plus a final n_studies row. The p rows print to fixed decimals via the p_terms attribute.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Edgington, E. S. (1972). An additive method for combining probability values from independent experiments. *The Journal of Psychology, 80*(2), 351–363.

Fisher, R. A. (1938). *Statistical methods for research workers* (7th ed.). Oliver & Boyd.

Mosteller, F., & Bush, R. R. (1954). Selected quantitative techniques. In G. Lindzey (Ed.), *Handbook of social psychology* (Vol. 1). Addison-Wesley.

Raudenbush, S. W. (1984). Magnitude of teacher expectancy effects on pupil IQ as a function of the credibility of expectancy induction: A synthesis of findings from 18 experiments. *Journal of Educational Psychology, 76*(1), 85–97.

Rosenthal, R. (1978). Combining results of independent studies. *Psychological Bulletin, 85*(1), 185–193.

See Also

`meta_smd` and `meta_es` for estimating the pooled effect rather than only testing it; `meta_contrast` for differences among study effects; `teacher_expectancy` for the data behind the examples.

Other meta-analysis: `meta_contrast()`, `meta_es()`, `meta_r()`, `meta_smd()`, `plot_forest()`

Examples

```
# Raudenbush (1984), Table 2: the four combined tests over the 18
# teacher expectancy studies (Pellegrini & Hicks at its study-level
# values), weighting the Z method by degrees of freedom.
data(teacher_expectancy)
study <- teacher_expectancy[-c(4, 5), ]
p18  <- append(study$p_one_tailed, .010, after = 3)
df18 <- append(study$n_experimental + study$n_control - 2, 42, after = 3)
combine_p(p18, weights = df18)
# Fisher chi square 62.17 on 36 df; Edgington sum near 7; Stouffer
# z near 2.2; and the df-weighted z under 1: the large studies disagree.
```

`common_method_adjusted_r` *Correlations Adjusted for Common Method Variance, With Intervals and Tests*

Description

When every variable in a study is measured the same way, often by one respondent on one questionnaire, part of each correlation can come from the shared method rather than from the constructs. The marker variable technique of Lindell and Whitney (2001) takes a correlation that should be zero on substantive grounds, the correlation r_M of a marker variable with the substantive variables, as an estimate of the method effect common to every correlation, and removes it from each substantive correlation. `common_method_adjusted_r()` returns, for one or more values of r_M, every adjusted correlation with its confidence interval and Lindell and Whitney's t test, so that a reader can see whether a relationship survives the adjustment and how that conclusion depends on the marker correlation assumed.

The marker correlation is itself an estimate, and the proxy used when no marker was measured (the smallest positive correlation among the substantive variables) is a choice. Reading the adjusted correlations over a grid of values of r_M, from no adjustment to a generous one, is the treatment of

that uncertainty: a correlation whose interval excludes zero across the plausible range does not depend on the value chosen. common_method_marker reports the adjusted matrix for a single marker correlation without intervals or tests, since it does not take N; this function takes N and gives both.

Usage

```
common_method_adjusted_r(R, N, marker_r = NULL, conf_level = 0.95)

## S3 method for class 'dmar_common_method_adjusted_r'
tidy(x, ...)

## S3 method for class 'dmar_common_method_adjusted_r'
glance(x, ...)
```

Arguments

R	A correlation matrix among the substantive variables (not the marker): symmetric, with a unit diagonal. Its column (or row) names name the variables; an unnamed matrix gets V1, V2, and so on.
N	The sample size on which R rests, at least 4.
marker_r	One or more marker correlations r_M, each in $(-1, 1)$, in the order the rows should follow. When NULL (the default) the smallest positive off-diagonal correlation in R is used as the proxy marker, as in common_method_marker.
conf_level	Confidence level of the intervals (default 0.95).
x	A dmar_common_method_adjusted_r table returned by common_method_adjusted_r().
...	Unused.

Details

The adjustment. Each substantive correlation r_{ij} is adjusted as

$$r_{ij}^{A} = \frac{r_{ij} - r_M}{1 - r_M}$$

(Lindell & Whitney, 2001, Equation 4), the partial correlation of i and j with the method effect removed when that effect contributes the same r_M to every correlation. The adjusted matrix for each value of marker_r is the "adjusted" attribute; for a single value it equals the "adjusted" attribute of common_method_marker.

The test. Lindell and Whitney (2001, Equation 5) test each adjusted correlation as a partial correlation with one variable partialed out,

$$t = \frac{r^{A}}{\sqrt{(1 - (r^{A})^2)/(N - 3)}},$$

referred to the *t* distribution with $N - 3$ degrees of freedom, with a two-sided *p*-value.

The interval. With r_M treated as known, the adjustment is an increasing linear function of the population correlation ρ_{ij}, so a confidence interval for ρ_{ij} maps into one for the adjusted correlation with the same coverage: each limit of the Fisher's Z interval for ρ_{ij} (the interval of ci_r, based on

the variance $1/(N-3)$) is adjusted by the same equation as the correlation. A lower limit that the map carries below -1 is set to -1, the bound of a correlation. Because the map is increasing, that happens exactly when the lower limit for ρ_{ij} falls below $2r_M - 1$, whatever the sign of r_{ij} or of the adjusted correlation: never when $r_M \leq 0$; for $0 < r_M < .5$ only when the lower limit for ρ_{ij} is negative, as it is for a negative correlation, one near zero, or one from a small sample (with $r_{ij} = .35$, $N = 5$, and $r_M = .3$ the adjusted correlation is .071 and its lower limit is set to -1); and for $r_M \geq .5$ even when that limit is positive. The interval is conditional on r_M: it carries the sampling variability of r_{ij} and none of that of the marker correlation, which is what the grid over `marker_r` is for.

An adjusted correlation at or beyond ± 1 lies outside the range of a correlation. It arises from a correlation below $2r_M - 1$, which a method effect of r_M common to every correlation cannot produce (or from a correlation of exactly 1), and it gets no interval and no test: its `ci_lower`, `ci_upper`, `t_value`, and `p_value` are `NA`.

The interval and the test are different approximations. The interval excludes zero exactly when the Fisher's Z interval for ρ_{ij} excludes r_M, while the test treats the adjusted correlation as a sample partial correlation, so near the boundary of significance the two can disagree. The test is kept for parity with the paper; the interval is what to report beside the estimate. Neither is adjusted for the number of pairs or of marker values.

Reading the results. The proxy marker is one of the substantive correlations, so the pair that supplied it is adjusted to exactly zero and its test says nothing about that pair. The technique presumes the variables are reflected so that their intercorrelations are positive; a negative correlation is pushed further from zero rather than toward it, so reverse code as needed before computing `R`. A correlation that remains different from zero after the adjustment is consistent with a relationship that is not only a method artifact, under the assumption that the method effect is the same for every pair; it is not evidence that no method variance is present.

Value

A `data.frame` (class `dmar_tbl`) with one row per value of `marker_r` and pair of variables, the pairs in the order (1, 2), (1, 3), ..., (2, 3), ... within each marker value, and the columns `marker_r`, `variable_1`, `variable_2`, `r` (the unadjusted correlation), `estimate` (the adjusted correlation), `ci_lower`, `ci_upper`, `t_value`, `df` ($N - 3$), and `p_value`. The attributes are `adjusted` (a list of the adjusted correlation matrices, one per value of `marker_r`, named by it), `N`, `conf_level`, and `marker_rule` (`"smallest_positive"` when the proxy was used, `"supplied"` otherwise). `tidy()` returns one row per pair and marker value, with `term` naming the pair (`"variable_1:variable_2"`) and the columns `marker_r`, `r`, `estimate`, `ci_lower`, `ci_upper`, `statistic` (the *t*), `df`, `p_value`, and `conf_level`; `glance()` returns `n_pairs`, `n_markers`, `N`, and `conf_level`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lindell, M. K., & Whitney, D. J. (2001). Accounting for common method variance in cross-sectional research designs. *Journal of Applied Psychology, 86*(1), 114–121. doi:10.1037/0021-9010.86.1.114

See Also

common_method_marker for the adjusted matrix and its summary at one marker correlation; common_method_single_factor for the single-factor screen; ci_r for the interval of an unadjusted correlation.

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_marker(), common_method_single_fa compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
# Visual, textual, and speed composites of the nine Holzinger and
# Swineford tests in the primary set, each the mean of three tests.
hs <- holzinger_swineford
scores <- data.frame(
  visual  = rowMeans(hs[c("t1_visual_perception", "t2_cubes",
                          "t4_lozenges")]),
  textual = rowMeans(hs[c("t6_paragraph_comprehension", "t7_sentence",
                          "t9_word_meaning")]),
  speed   = rowMeans(hs[c("t10_addition", "t12_counting_groups_of_dots",
                          "t13_straight_and_curved_capitals")]))
R <- stats::cor(scores)

# The smallest positive correlation as the proxy marker; the pair that
# supplied it (textual with speed) is adjusted to zero. Visual with
# speed shows the interval and the test parting near the boundary: the
# interval includes zero while the p-value is below .05.
common_method_adjusted_r(R, N = nrow(scores))

# A grid of marker correlations, from no adjustment to a large one.
grid <- common_method_adjusted_r(R, N = nrow(scores),
                                 marker_r = c(0, 0.1, 0.2, 0.3))
grid[grid$variable_1 == "visual" & grid$variable_2 == "textual", ]
attr(grid, "adjusted")[["0.2"]]
```

---

common_method_marker *Marker-Variable Adjustment for Common Method Variance*

Description

The marker-variable technique of Lindell and Whitney (2001) estimates common method variance from the correlation of a *marker* variable that is theoretically unrelated to at least one of the substantive variables: any non-zero correlation it shows with that variable is attributed to shared method, and that amount is partialed out of the substantive correlations. When no a priori marker is available, the smallest positive correlation among the substantive items is used as a proxy, the common

marker-free variant of the method. A correlation that remains statistically significant after the adjustment, by the paper's t test of the adjusted correlation with $N - 3$ degrees of freedom (their Equation 5), is evidence that the relationship is not an artifact of method variance. This function works from the correlation matrix alone and does not take N; `common_method_adjusted_r` takes N and gives that test and a confidence interval for every adjusted correlation, over one or more marker correlations.

Usage

```
common_method_marker(R, marker_r = NULL)
```

Arguments

`R`	A correlation matrix among the substantive items.
`marker_r`	The marker variable's (CMV) correlation. When `NULL` (default) the smallest positive off-diagonal correlation in `R` is used as the proxy marker.

Details

Writing r_M for the marker (or proxy) correlation, each substantive correlation is adjusted as $r_{ij}^A = (r_{ij} - r_M)/(1 - r_M)$ (Lindell & Whitney, 2001, Equation 4). The CMV-adjusted correlation matrix is returned as the `"adjusted"` attribute; the reported table summarizes the marker correlation and the average absolute correlation before and after adjustment.

The method presumes the variables are reflected so that their intercorrelations are positive; a negative substantive correlation is pushed further from zero by the adjustment rather than attenuated, so reverse-code as needed before adjusting.

Value

A `data.frame` (class `dmar_tbl`) with rows `marker_correlation`, `mean_abs_r_unadjusted`, and `mean_abs_r_adjusted` in the `value` column. The full adjusted correlation matrix is the `"adjusted"` attribute.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lindell, M. K., & Whitney, D. J. (2001). Accounting for common method variance in cross-sectional research designs. *Journal of Applied Psychology, 86*(1), 114–121. doi:10.1037/0021-9010.86.1.114

See Also

`common_method_adjusted_r` for the test and interval of each adjusted correlation over a grid of marker correlations; `common_method_single_factor` for the single-factor screen.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`,

common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
R <- matrix(c(1, .5, .4, .5, 1, .45, .4, .45, 1), 3, 3,
            dimnames = list(c("a", "b", "c"), c("a", "b", "c")))
res <- common_method_marker(R, marker_r = 0.10)
res
attr(res, "adjusted")
```

---

common_method_single_factor

Single-Common-Factor Screen for Common Method Variance

Description

This function implements Harman's single-factor test, a screen for common method variance: fit a one-factor model to all of the items by maximum likelihood and inspect how much of their variance the common factor accounts for. The rationale is that if a single method factor dominated the responses, one common factor would capture a large share of the variance. A factor accounting for more than half of the variance is the customary red flag (Podsakoff, MacKenzie, Lee, & Podsakoff, 2003). The screen is coarse and cannot by itself rule method variance in or out, since a dominant factor can reflect the constructs as well as the method; the marker-variable and latent method factor approaches estimate the method variance itself (see common_method_marker).

Usage

```
common_method_single_factor(data = NULL, S = NULL, R = NULL)
```

Arguments

data	A data.frame or numeric matrix of item responses. Supply this, a covariance matrix S, or a correlation matrix R (exactly one).
S	A symmetric covariance matrix among the items, when raw data are not available but the summary statistics a paper reports are. It is converted to a correlation matrix internally, so the test acts on the same scale-free quantity regardless of which input is supplied.
R	A correlation matrix among the items, when raw data are not available.

Details

Harman's single-factor test (the proportion of variance explained by one common factor) is related to but distinct from the marker-variable technique. A *marker variable* (or common-method marker) is a variable chosen to be theoretically unrelated to the substantive constructs under study, so that any observed correlation between it and the substantive items is attributable to shared method rather than to a true relationship; it is used to estimate or partial out common method variance (Lindell & Whitney, 2001). The single-factor test uses no such marker, it asks only whether a single dimension dominates the item set, so it can flag a strong common factor but cannot identify whether that factor is method or substance.

The one-factor model is fit to the item correlation matrix by maximum likelihood with `factanal`, and the statistic is the proportion of total variance the common factor accounts for: the sum of the squared standardized loadings divided by the number of items (equivalently, the mean communality). Much of the applied literature computes the screen from the largest eigenvalue of the correlation matrix, which describes the first principal component, not a factor; the test is implemented factor analytically here, in the psychometric tradition, because a principal component absorbs unique as well as common variance and so overstates the share a common factor accounts for. Correlations from raw data use pairwise-complete observations. A supplied covariance matrix is first standardized to a correlation matrix with `cov2cor`. The one-factor model requires at least three items.

Value

A `data.frame` (class `dmar_tbl`) with rows `variance_explained` (the proportion of total variance the single common factor accounts for) and `n_items` in the `value` column.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Harman, H. H. (1976). *Modern factor analysis* (3rd ed.). University of Chicago Press.

Lindell, M. K., & Whitney, D. J. (2001). Accounting for common method variance in cross-sectional research designs. *Journal of Applied Psychology, 86*(1), 114–121. doi:10.1037/0021-9010.86.1.114

Podsakoff, P. M., MacKenzie, S. B., Lee, J.-Y., & Podsakoff, N. P. (2003). Common method biases in behavioral research: A critical review of the literature and recommended remedies. *Journal of Applied Psychology, 88*(5), 879–903. doi:10.1037/00219010.88.5.879

See Also

`common_method_marker` for the marker-variable adjustment.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`,

sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
set.seed(113)
f <- rnorm(200)
d <- data.frame(
  x1 = f + rnorm(200), x2 = f + rnorm(200), x3 = f + rnorm(200),
  x4 = rnorm(200),     x5 = rnorm(200),     x6 = rnorm(200))
common_method_single_factor(d)

# The same screen from the summary statistics a paper reports.
common_method_single_factor(S = cov(d))
```

---

compare_cov_structures

Likelihood-Ratio Comparison of Covariance Structures

Description

Fits a long-format within-subjects regression under a menu of variance-covariance structures, from independence through the unstructured form, and returns a comparison table of log-likelihood, AIC, BIC, and pairwise likelihood-ratio tests against the most general structure (UN). Wraps gls.

Usage

```
compare_cov_structures(
  data,
  outcome,
  subject,
  time,
  fixed_effects = NULL,
  structures = c("IND", "CS", "CSH", "AR1", "ARH1", "TOEP", "TOEPH", "UN")
)
```

Arguments

data	Long-format data.frame with one row per subject-by-condition observation.
outcome	Character name of the response column.
subject	Character name of the subject-id column.
time	Character name of the time / within-subjects factor column.
fixed_effects	Right-hand-side formula for the fixed effects (default: ~ time).
structures	Character vector of structures to fit. Any subset of c("IND", "CS", "CSH", "AR1", "ARH1", "TOEP", "TOEPH","UN") (default: all eight). Matching is case insensitive, so lowercase aliases such as "cs", "ar1", "csh", "arh1", "toep", and "un" are accepted and normalized to their canonical uppercase labels.

Details

Structures. Every structure below is nested in UN, so the likelihood-ratio test against UN is well defined for each.

- `IND`: independent observations within subject (`correlation = NULL` in `gls`). Provided as a baseline.
- `CS`: compound symmetry, a constant correlation and a single variance across time points: `nlme::corCompSymm()`.
- `CSH`: heterogeneous compound symmetry, a constant correlation with a separate variance at each time point: `nlme::corCompSymm()` with `nlme::varIdent()`.
- `AR1`: first-order autoregressive correlation with a single variance: `nlme::corAR1()`.
- `ARH1`: heterogeneous first-order autoregressive correlation with a separate variance at each time point: `nlme::corAR1()` with `nlme::varIdent()`.
- `TOEP`: Toeplitz (banded), a separate correlation at each lag with a single variance: `nlme::corARMA()` with autoregressive order one less than the number of time points and no moving-average term.
- `TOEPH`: heterogeneous Toeplitz, the Toeplitz correlation with a separate variance at each time point: `nlme::corARMA()` with `nlme::varIdent()`.
- `UN`: unstructured, every variance and covariance free: `nlme::corSymm()` with `nlme::varIdent()`.

LRT. Each restricted structure is compared against UN by the likelihood-ratio test. Both fits are re-estimated under ML (not REML) for the LRT, following `nlme` convention. The chi square statistic is $-2(\log L_{\text{restricted}} - \log L_{\text{UN}})$ on degrees of freedom equal to the difference in parameter count.

Caveats. The likelihood-ratio test against UN is valid because each listed structure is a restriction of UN. Two structures that are not nested in each other (for example CS and AR(1)) should be compared by AIC or BIC rather than by an LRT.

Value

A `data.frame` with one row per structure. Columns: `structure`, `log_lik`, `AIC`, `BIC`, `n_par`, `LRT_vs_UN_chisq`, `LRT_vs_UN_df`, `LRT_vs_UN_p`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., & Schabenberger, O. (2006). *SAS for mixed models* (2nd ed.). SAS Institute.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 15.)

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-effects models in S and S-PLUS*. Springer.

See Also

gls, logLik, anova

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# Four repeated measures on each of 30 subjects.
set.seed(113)
n <- 30; k <- 4
subj <- factor(rep(1:n, each = k))
tm   <- factor(rep(1:k, times = n))
y    <- as.vector(t(matrix(rnorm(n * k), n, k) +
                    rep(rnorm(n, 0, 1), each = k)))
d <- data.frame(y, subj, tm)

# All eight structures at once. Read the table by comparing AIC and BIC
# across rows, and use the likelihood-ratio test only for the nested
# comparison it reports, each structure against UN.
compare_cov_structures(d, outcome = "y", subject = "subj",
                       time = "tm")

# A subset, requested with lowercase aliases (matching is case
# insensitive).
compare_cov_structures(d, outcome = "y", subject = "subj",
                       time = "tm",
                       structures = c("cs", "csh", "ar1", "arh1"))
```

---

compare_sem *Compare Structural Equation Models by an Information Criterion*

Description

Ranks a set of structural equation models fitted to the same data by one information criterion of sem_information_criteria (BIC by default) and reports, for each model, its difference from the best, its relative weight within the set, and, for the criteria that approximate a log marginal likelihood, the approximate log Bayes factor of the best model over it. The models may be nested or not. A criterion compares models only through their log likelihoods, so before anything is computed the function checks that the log likelihoods are comparable: the same cases and data values, the same observed variables and groups, the same missing data method and mean structure, and the same exogenous observed variables conditioned on. Requires **lavaan** (in Suggests).

Usage

```
compare_sem(
  fits,
  criterion = c("BIC", "SABIC", "HBIC", "IBIC", "SPBIC", "AIC"),
  prior_means = NULL
)

## S3 method for class 'dmar_compare_sem'
tidy(x, ...)

## S3 method for class 'dmar_compare_sem'
glance(x, ...)
```

Arguments

`fits`	A named list of two or more fitted **lavaan** models: maximum likelihood fits of continuous variables (estimator ML or its robust variants MLM, MLMV, and MLR, which maximize the same likelihood) fitted to the same data. That is, the same cases, data values, observed variables (in any order), and groups; the same missing data method (`missing`) and mean structure (`meanstructure`); and, when `fixed.x` or `conditional.x` is `TRUE`, the same exogenous observed variables. Models fitted to summary statistics (`sample.cov`) are compared when they share the sample moments and *N*. Two-level models and models fitted with sampling weights are refused, and the criterion can impose further conditions (see `sem_information_criteria`).
`criterion`	The criterion: `"BIC"` (the default), `"SABIC"`, `"HBIC"`, `"IBIC"`, `"SPBIC"`, or `"AIC"`, in either case.
`prior_means`	For SPBIC, an optional list of prior mean vectors, one per model in the order of `fits` (and with its names, if named), each in the order of `lavaan::coef()` of its model. `NULL` applies the default rule of `sem_information_criteria`: free loadings and regression coefficients centered at zero and the other free parameters at their estimates. A saturated model needs an explicit prior mean.
`x`	A `dmar_compare_sem` table returned by `compare_sem()`.
`...`	Unused.

Details

Differences, Weights, and Evidence: For the criterion values C_k of the models, $\Delta_k = C_k - \min_j C_j$ (column `delta`), and the relative weight of model *k* is $\exp(-\Delta_k/2)/\sum_j \exp(-\Delta_j/2)$. BIC, HBIC, IBIC, and SPBIC each approximate -2 times the log marginal likelihood of a model (Bollen, Ray, Zavisca, & Harden, 2012), so $\exp(-\Delta_k/2)$ approximates the Bayes factor of model *k* against the best model, the weights approximate the posterior probabilities of the models when every model in the set has the same prior probability (Burnham & Anderson, 2004), and $\Delta_k/2$ (column `log_evidence_best_over_model`) approximates the log Bayes factor of the best model over model *k*. For AIC the weights are Akaike weights, the weight of evidence that each model is the one of the set closest to the data generating process in Kullback-Leibler discrepancy (Burnham & Anderson, 2004); they are not posterior probabilities under equal prior odds, and the log

evidence column is NA. SABIC rests on the minimum description length principle rather than on a marginal likelihood, so its weights carry no further interpretation and its log evidence column is NA too. So it is for every criterion of models fitted by two-stage maximum likelihood (`missing = "two.stage"` or `"robust.two.stage"`), whose criteria are those of Lai (2021): built from the complete data normal log likelihood at the EM estimates of the means and covariances, they select in large samples the model complete data would select, but they are not a likelihood of the observed data and approximate no marginal likelihood (see `sem_information_criteria`).

Every weight is conditional on the set of models compared: adding or removing a model changes all of them, and none of them says how well a model fits in absolute terms (`sem_fit_measures` speaks to that). Nor do the weights carry the sampling variability of the criteria; the uncertainty of the selection itself is not estimated.

What Is Checked: Each model must have converged to an admissible solution and have a log likelihood the criteria are defined on (see `sem_information_criteria` for what is refused and why). Then every model is compared with the first, and the first difference found is reported with the two models it concerns and what to do.

Data The same groups, the same observed variables, the same number of cases, the same cases, and the same data values (or, for models fitted to summary statistics, the same sample moments). The order of the variables does not matter; a set of different variables does, since a log likelihood is a density of the variables it models.

Missing data The same missing data method. Listwise deletion, FIML, and the two-stage method give different likelihoods, and two listwise fits of different variables keep different cases. The two two-stage settings, `missing = "two.stage"` and `"robust.two.stage"`, count as one method: they fit the same EM moments by the same likelihood and differ only in their standard errors and test statistic, which no criterion uses. Two-stage fits must also share the auxiliary variables of their first stage (lavaan's `aux`), since those change the EM estimates their log likelihoods are evaluated at. A two-stage fit whose first stage read cases observed on an auxiliary variable but on none of the model's variables, which lavaan leaves out of its N, is compared by the criterion `sem_information_criteria` computes for it with every case the first stage read, its log likelihood and the N of the penalty included (in several groups held together by a constraint, at the estimates with every case read, which move with the groups' counts), and a line under the table says so; such a fit and the fit `fa_confirmatory(missing = "two_stage")` or `measurement_invariance` returns for the same data count the same cases.

Mean structure The same `meanstructure` setting: without a mean structure lavaan leaves the saturated means out of the parameter count, which moves every criterion by an amount that depends on the setting.

Exogenous variables With `fixed.x = TRUE` (the default of `lavaan::sem()`) or `conditional.x = TRUE`, lavaan's log likelihood is the density conditional on the exogenous observed variables, the joint density less the marginal density of those variables, and its parameter count leaves out their moments. Two models that treat different variables as exogenous therefore have log likelihoods of different conditional distributions, whose difference can be hundreds of points for two saturated models of the same three variables. Models with the same exogenous variables are compared, and the others are refused with the advice to refit every model with `fixed.x = FALSE`.

Identification For IBIC, and for SPBIC under its default prior, every latent variable of every model must be identified the same way (each by a marker indicator, or each by a fixed variance), since both criteria change with the identification even though the likelihood does not (see `sem_information_criteria`). A set that uses both is refused, also when each model

mixes them in the same way: one identification throughout is a convention that can be stated and reported, and it catches the models that differ, which may share no latent variable to compare. Within one data set the units of the observed variables are shared, but an IBIC ranking still depends on them.

Value

A `dmar_tbl` (class `dmar_compare_sem`) with one row per model, sorted by the criterion (ties keep the order of `fits`), and the columns `term` (the model's name), `value` (its criterion), `delta`, `relative_weight`, `log_evidence_best_over_model`, and `npar` (its number of free parameters). The `criterion` attribute names the criterion, the `comparison` attribute records what was checked (*N*, the number of groups, whether the models were fitted to raw data, the `missing` settings used, the mean structure, the exogenous observed variables conditioned on, and for IBIC and SPBIC the identification), and the `sem_information` attribute holds the details of each model's `sem_information_criteria` result, named by model. `tidy()` returns every column with the criterion beside it, and `glance()` one row with the criterion, the number of models, *N*, the best model, its weight, and the difference of the second best.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bollen, K. A., Harden, J. J., Ray, S., & Zavisca, J. (2014). BIC and alternative Bayesian information criteria in the selection of structural equation models. *Structural Equation Modeling, 21*(1), 1–19. doi:10.1080/10705511.2014.856691

Bollen, K. A., Ray, S., Zavisca, J., & Harden, J. J. (2012). A comparison of Bayes factor approximation methods including two new methods. *Sociological Methods & Research, 41*(2), 294–324. doi:10.1177/0049124112452393

Burnham, K. P., & Anderson, D. R. (2004). Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research, 33*(2), 261–304. doi:10.1177/0049124104268644

Lai, K. (2021). Using information criteria under missing data: Full information maximum likelihood versus two-stage estimation. *Structural Equation Modeling, 28*(2), 278–291. doi:10.1080/10705511.2020.1780925

See Also

`sem_information_criteria` for the criteria of one model and what each assumes, `sem_fit_measures` for the fit of each model in absolute terms, and the vignette `vignette("sem_fit_measures", package = "DMAR")`.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
# One factor against two correlated factors for six of the
# holzinger_swineford tests.
one <- lavaan::cfa(
  "g =~ t1_visual_perception + t2_cubes + t4_lozenges +
        t6_paragraph_comprehension + t7_sentence + t9_word_meaning",
  data = holzinger_swineford)
two <- lavaan::cfa(
  "visual  =~ t1_visual_perception + t2_cubes + t4_lozenges
   textual =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning",
  data = holzinger_swineford)
x <- compare_sem(list(one_factor = one, two_factors = two))
x
generics::glance(x)

# The same comparison by SPBIC, under its default prior.
compare_sem(list(one_factor = one, two_factors = two), criterion = "SPBIC")
```

`content_validity_index`

Content Validity Index From Expert Ratings

Description

Quantifies how well a pool of candidate items covers the construct it is meant to measure, using the relevance ratings of a panel of subject matter experts. Validation begins before any respondent data are collected: experts rate each item for relevance, and the content validity index summarizes their agreement (Lynn, 1986; Polit & Beck, 2006; Bandalos, 2018). The function returns the item level index (I-CVI) with an exact binomial confidence interval, the chance corrected modified kappa of Polit, Beck, and Owen (2007), Lawshe's (1975) content validity ratio, and the two scale level summaries S-CVI/Ave and S-CVI/UA. The confidence interval is what keeps a small panel from being read as more informative than it is: with five experts an I-CVI of 0.80 carries an interval roughly seven tenths of the width of the scale.

Usage

```
content_validity_index(
  ratings,
  relevant = c(3, 4),
  essential = NULL,
  conf_level = 0.95
)
```

Arguments

`ratings`	A matrix or `data.frame` of expert ratings with items in rows and experts in columns, on the conventional 4 point relevance scale (1 = not relevant, 2 = somewhat relevant, 3 = quite relevant, 4 = highly relevant). Every non-missing entry

must be one of 1, 2, 3, or 4. Row names, when present, name the items; otherwise items are labeled `item_1`, `item_2`, and so on. At least 1 item and at least 2 experts are required. `NA` is allowed and means that expert did not rate that item; a missing entry lowers that item's expert count and thereby the denominator of its I-CVI, kappa, and CVR, leaving other items untouched. A row with no ratings at all is an error.

`relevant` The rating values counted as relevant. Default `c(3, 4)`, the standard dichotomization of the 4 point scale into relevant (3 or 4) versus not relevant (1 or 2). Must be a subset of `1:4`.

`essential` Optional. The rating values counted as "essential" for Lawshe's content validity ratio, when the panel answered the essential-versus-not question on a separate part of the scale. Must be a subset of `1:4`. When `NULL` (default), the content validity ratio is computed from the same dichotomization as relevance, that is from `relevant`.

`conf_level` Confidence level for the exact binomial interval on each I-CVI. Default `0.95`. Must be in (0, 1).

Details

Let N be the number of experts who rated an item and A the number of those who rated it relevant.

Item level index. The item level content validity index is the proportion of rating experts who called the item relevant,

$$\text{I-CVI} = A/N.$$

Lynn (1986) gives the conventional criteria: with five or fewer experts an item is expected to reach 1.00, and with six to ten experts at least 0.78.

Confidence interval. The I-CVI is a binomial proportion, so the interval reported here is the exact (Clopper-Pearson) interval from `binom.test` at `conf_level`. Expert panels are small by design and the interval states plainly how little a handful of ratings pins down the population proportion. This is the accuracy in parameter estimation view of content validity: if the interval is too wide to act on, the remedy is more experts.

Chance corrected agreement. Some of the observed agreement on relevance would occur if experts responded at random with probability 0.5, so Polit, Beck, and Owen (2007) correct the index in the manner of a kappa. The probability of chance agreement is the binomial point probability

$$p_c = \frac{N!}{A!\,(N-A)!}\,0.5^N,$$

and the modified kappa is

$$\kappa = \frac{\text{I-CVI} - p_c}{1 - p_c}.$$

The binomial coefficient is evaluated on the log scale with `lchoose` and then exponentiated, so a large panel does not overflow the way `factorial()` would.

Content validity ratio. Lawshe (1975) asked a panel whether each item measures behavior that is essential to the performance domain. With n_e experts calling the item essential,

$$\text{CVR} = \frac{n_e - N/2}{N/2},$$

which equals 1 when every expert says essential, 0 when exactly half do, and -1 when none do.

Scale level summaries. S-CVI/Ave is the mean of the I-CVIs over items, the averaging approach Polit and Beck (2006) recommend reporting. S-CVI/UA is the universal agreement proportion, the fraction of items whose I-CVI equals 1, a stricter and considerably more conservative summary.

Value

A `data.frame` (class `dmar_tbl`) with one row per item, in the order the items appear in `ratings`, and columns:

`item` The item name, from the row names of `ratings` when present.

`n_experts` Number of experts who rated the item (missing ratings excluded).

`n_relevant` Number of those experts whose rating was in `relevant`.

`i_cvi` The item level content validity index, `n_relevant / n_experts`.

`ci_lower`, `ci_upper` Limits of the exact binomial confidence interval for `i_cvi` at `conf_level`.

`kappa` The modified kappa of Polit, Beck, and Owen (2007), the I-CVI adjusted for chance agreement.

`cvr` Lawshe's content validity ratio, computed from `essential` when supplied and from `relevant` otherwise.

Attributes: `"s_cvi_ave"`, the mean of the `i_cvi` column; `"s_cvi_ua"`, the proportion of items with `i_cvi` equal to 1; `"relevant"`, the rating values counted as relevant; `"essential"`, the rating values counted as essential for the content validity ratio (equal to `"relevant"` when `essential` was `NULL`); and `"conf_level"`, the confidence level used.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bandalos, D. L. (2018). *Measurement theory and applications for the social sciences*. Guilford Press.

Lawshe, C. H. (1975). A quantitative approach to content validity. *Personnel Psychology, 28*(4), 563–575.

Lynn, M. R. (1986). Determination and quantification of content validity. *Nursing Research, 35*(6), 382–385.

Polit, D. F., & Beck, C. T. (2006). The content validity index: Are you sure you know what's being reported? Critique and recommendations. *Research in Nursing and Health, 29*(5), 489–497. doi:10.1002/nur.20147

Polit, D. F., Beck, C. T., & Owen, S. V. (2007). Is the CVI an acceptable indicator of content validity? Appraisal and recommendations. *Research in Nursing and Health, 30*(4), 459–467. doi:10.1002/nur.20199

See Also

gwet_ac, fleiss_kappa for agreement among raters on a common set of units.

Other agreement and measurement: R2_mixed_effects(), gwet_ac(), icc_lmer(), krippendorff_alpha(), limits_of_agreement(), lin_ccc(), variance_components_mls()

Examples

```
# Six experts rate five candidate items on the 4 point relevance scale.
ratings <- rbind(
  item_1 = c(4, 4, 3, 4, 4, 3),
  item_2 = c(4, 3, 4, 4, 3, 2),
  item_3 = c(2, 3, 1, 2, 3, 2),
  item_4 = c(4, 4, 4, 4, 4, 4),
  item_5 = c(3, 4, 4, 3, NA, 4))
colnames(ratings) <- paste0("expert_", 1:6)
cvi <- content_validity_index(ratings)
cvi

# The scale level summaries travel with the table as attributes.
attr(cvi, "s_cvi_ave")
attr(cvi, "s_cvi_ua")

# The worked example of Polit, Beck, and Owen (2007): 6 experts, 5 of whom
# rate the item relevant, gives I-CVI = 0.83, p_c = 0.094, and a modified
# kappa of 0.816 (the paper reports 0.81, carrying its rounded I-CVI).
content_validity_index(matrix(c(4, 4, 3, 4, 3, 1), nrow = 1))

# A wide interval is the point: with 5 experts an I-CVI of 0.80 is
# compatible with a population proportion anywhere from about 0.28 to 0.99.
content_validity_index(matrix(c(4, 4, 3, 4, 1), nrow = 1))

# The broom verbs on the earlier result: one row per item, and the
# scale-level summary.
generics::tidy(cvi)
generics::glance(cvi)
```

---

contrast_adjusted *Confidence Interval for a Contrast of Covariate-Adjusted Cell Means in a Factorial ANCOVA*

Description

Given a fitted lm or aov object for a factorial analysis of covariance (one or more crossed factors plus one or more covariates) and a numeric contrast vector over the cells of the factorial design, contrast_adjusted() forms the contrast of the covariate-adjusted cell means, $\hat{\psi} = \sum_j c_j \hat{\bar{Y}}_j$, where each $\hat{\bar{Y}}_j$ is the model's predicted mean for cell j evaluated at the mean of every covariate (the adjusted, or least-squares, cell mean). It returns the point estimate, a t confidence interval on

the model's residual degrees of freedom, and the accompanying t statistic and two-sided p-value for $H_0 : \psi = 0$.

Usage

```
contrast_adjusted(model, contrast, conf_level = 0.95)
```

Arguments

`model`	A fitted `lm` or `aov` object for a factorial ANCOVA: one or more crossed factors and one or more numeric covariates on the right-hand side of the formula.
`contrast`	A numeric vector of contrast weights, one weight per cell of the factorial design (the crossing of the model's factors). Its length must equal the number of cells. The weights typically sum to zero.
`conf_level`	The confidence level for the interval (default `0.95`).

Details

The adjusted cell means are the means the ANCOVA actually tests: the predicted outcome for each combination of factor levels, holding every covariate at its sample mean. Writing L for the linear map that sends the model coefficients to that contrast of adjusted means (built by evaluating the model's design matrix at each cell with the covariates set to their means and combining the rows with the contrast weights), the point estimate is $\hat{\psi} = L'\hat{\beta}$ and its standard error is the square root of the quadratic form $L' \operatorname{vcov}(\hat{\beta})\, L$. The interval is $\hat{\psi} \pm t_{1-\alpha/2,\,\nu}\,\mathrm{SE}$, with ν the residual degrees of freedom of the fitted model.

Because L is read off the fitted model's own design matrix, the function is agnostic to how the factors are parameterized: a cell-means parameterization (`y ~ 0 + cell + x`) and the crossed-factor parameterization (`y ~ A * B + x`) give the same contrast estimate and standard error, provided the contrast vector is ordered to match the cells of the reference grid (see `Note`).

The t interval returned here has exact per-comparison coverage for a single contrast chosen in advance. For a family of contrasts examined together, adjust the critical value for multiplicity (for example the Scheffe critical value for the full cell space, `cv_scheffe`, or a Bryant–Paulson simultaneous interval, `ci_c_ancova_bp`).

Value

A five-row `dmar_tbl` (a `data.frame` with columns `term` and `value`). The `term` values are `"contrast"` (the point estimate $\hat{\psi}$ of the contrast of adjusted cell means), `"ci_lower"` and `"ci_upper"` (the confidence limits), `"t_value"` (the t statistic), and `"p_value"` (the two-sided p-value). The stored `value` column is numeric at full precision.

Note

The contrast weights are matched to the cells of the reference grid, which is the crossing of the model's factors in the order the factors appear in the model formula, with the first factor varying fastest (the order `expand.grid` produces over the factor levels). For a single factor this is simply the order of its levels. When in doubt, fit the cell-means form `y ~ 0 + cell + covariates` with `cell = interaction(A, B, ...)` and order the weights to match `levels(cell)`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on designs with covariates.)

See Also

contrast_test for contrasts of unadjusted group means in a one-way design; ci_c_ancova for a single-covariate ANCOVA contrast from summary statistics.

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), plot_smd()

Examples

```
# A 2 x 2 factorial ANCOVA with one covariate.
set.seed(113)
d <- data.frame(
  A = factor(rep(c("a1", "a2"), each = 40)),
  B = factor(rep(rep(c("b1", "b2"), each = 20), 2)),
  x = rnorm(80)
)
d$y <- 5 + 2 * (d$A == "a2") + 1.5 * (d$B == "b2") +
  0.8 * d$x + rnorm(80)
fit <- lm(y ~ A * B + x, data = d)

# Cells in reference-grid order: (a1,b1), (a2,b1), (a1,b2), (a2,b2).
# Main effect of A, averaged over B: mean(a2 cells) - mean(a1 cells).
contrast_adjusted(fit, contrast = c(-0.5, 0.5, -0.5, 0.5))
```

---

contrast_test *Tests One or More Contrasts of Group Means in a One-Way Design*

Description

Given a fitted one-way aov or lm object and a set of contrast weights, computes for every contrast the estimate $\hat{\psi} = \sum_i c_i \bar{Y}_i$, its standard error, *t*-statistic, degrees of freedom, two-sided *p*-value, and confidence interval. Supports several common multiple-comparison adjustments and either equal-variance (pooled) or Welch-style unequal-variance inference.

Usage

```
contrast_test(
  object,
  contrasts = "pairwise",
  adjust = "none",
  conf_level = 0.95,
  var_equal = TRUE
)
```

Arguments

`object`	A fitted `aov` or `lm` object for a one-way design (a single grouping factor on the right-hand side of the formula).
`contrasts`	Specification of one or more contrasts. Any of: `"pairwise"` **(default)** All pairwise comparisons among the group means. **a named list of numeric vectors** Each vector is one contrast and its name is used as the row label. **a numeric matrix** Each row is one contrast; `rownames`, if present, are used as labels. **a numeric vector** Treated as a single contrast. Each contrast vector must have length equal to the number of groups, and the weights are typically chosen to sum to zero.
`adjust`	Multiple-comparison adjustment. One of `"none"` (default), `"bonferroni"`, `"scheffe"`, `"tukey"` (pairwise contrasts only), or any of the sequential methods supported by `p.adjust` (`"holm"`, `"hochberg"`, `"BH"`, `"BY"`).
`conf_level`	Confidence level for the interval (default `0.95`).
`var_equal`	Logical. If `TRUE` (default), uses the pooled error variance MS_{error} and the residual degrees of freedom from `object`. If `FALSE`, uses each group's own sample variance and a Welch-Satterthwaite approximate df per contrast.

Details

Test statistic. For a contrast with weights $c_1, \ldots, c_k$ (k = number of groups), the estimate is $\hat{\psi} = \sum_i c_i \bar{Y}_i$. Under equal variances, the standard error is $\sqrt{MS_{\text{error}} \sum_i c_i^2/n_i}$ with $df = N - k$; under unequal variances, the standard error is $\sqrt{\sum_i c_i^2 s_i^2/n_i}$ with the Welch-Satterthwaite df,

$$df_{\text{Welch}} = \frac{\left(\sum_i c_i^2 s_i^2/n_i\right)^2}{\sum_i (c_i^2 s_i^2/n_i)^2/(n_i - 1)}.$$

The unadjusted p-value is two-sided based on the t reference distribution.

Adjustments. The `p_adjusted` and confidence interval critical value are computed as follows.

- `"none"`: no adjustment; the CI uses $t_{1-\alpha/2, df}$.
- `"bonferroni"`: $p_{\text{adj}} = \min(1, m\,p)$ for m contrasts, with CI based on $t_{1-\alpha/(2m), df}$.
- `"scheffe"`: appropriate for any contrast (or family of contrasts). p_{adj} comes from the upper tail of an F reference distribution applied to $t^2/(k-1)$, and the CI uses $\sqrt{(k-1)\, F_{1-\alpha,\, k-1, df}}$.

- `"tukey"`: requires every contrast to be pairwise. Uses the studentized range distribution $Q_{k,df}$, so that $p_{\text{adj}} = P(Q_{k,df} > |t|\sqrt{2})$ and the CI uses $q_{1-\alpha,\,k,df}/\sqrt{2}$. Both come from base R's `ptukey` and `qtukey` where those are accurate, and from the exact distribution where they are not (at small df, for example fewer than 9 df with up to 6 groups; from 10000 to 10^8 df; at confidence levels above .999; with more than 50 groups, below the .75 point of the distribution (.80 with more than 1000 groups); and wherever `qtukey` stops short of its root; with 20 groups and 2 df `qtukey(.99, 20, 2)` is 44.526 where the exact value is 37.943); `cv_tukey_hsd` describes the computation and the regions.
- `"holm"`, `"hochberg"`, `"BH"`, `"BY"`: `p.adjust` is applied to the unadjusted p-values; the CI uses the unadjusted t-critical value because these methods do not give simultaneous CIs in closed form.

Variance assumption with adjustments. The Tukey and Scheffé procedures assume equal variances; combining them with `var_equal = FALSE` is at the user's risk (the resulting Type I error rate is no longer guaranteed). For unequal variances, common alternatives are Games-Howell (Tukey-style) and Brown-Forsythe (Scheffé-style); these are not currently supported here.

Scope. Only one-way designs are supported in v1 (one outcome, one grouping factor). Multi-way designs throw an informative error.

Value

A `data.frame` with one row per contrast and columns `contrast`, `estimate`, `se`, `t_value`, `df`, `p_value`, `p_adjusted`, `ci_lower`, and `ci_upper`. The adjustment, confidence level, and variance assumption are stored as `attr(*, "adjust")`, `attr(*, "conf_level")`, and `attr(*, "var_equal")`. With `adjust` `"bonferroni"`, `"scheffe"`, or `"tukey"`, the recorded level is the simultaneous level of the family of intervals, the convention of the post hoc intervals and of `pairwise_within` with `bonferroni_ci = TRUE`; under Bonferroni the line beneath the table also names the level of each interval, $1 - \alpha/m$. The table prints through the `dmar_tbl` display layer and works with `tidy` and `glance` (see `dmar_tidiers`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Hsu, J. C. (1996). *Multiple comparisons: Theory and methods*. Chapman & Hall.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Scheffe, H. (1953). A method for judging all contrasts in the analysis of variance. *Biometrika, 40*, 87–104.

Smith, J. E., Meyers, R. J., & Delaney, H. D. (1998). The community reinforcement approach with homeless alcohol-dependent individuals. *Journal of Consulting and Clinical Psychology, 66*(3), 541–548.

Tukey, J. W. (1953). The problem of multiple comparisons. Unpublished manuscript, Princeton University.

See Also

TukeyHSD, pairwise.t.test, p.adjust, cv_tukey_hsd, and dmar_tidiers for the tidy methods

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# All pairwise comparisons among the three arms of the depression_bdi
# treatment study.
fit <- aov(bdi_post ~ condition, data = depression_bdi)
contrast_test(fit, contrasts = "pairwise")

# Custom contrasts with a Tukey-protected family-wise error rate.
contrast_test(fit, contrasts = "pairwise", adjust = "tukey")

# A user-defined contrast: the SSRI arm vs. the average of the placebo
# and wait list arms. With levels ordered ssri, placebo, wait_list, the
# weights c(1, -0.5, -0.5) estimate that difference.
contrast_test(
  fit,
  contrasts = list("ssri vs non-drug arms" = c(1, -0.5, -0.5)),
  adjust = "scheffe"
)

# Welch-style inference: the wait list variance is about twice the
# SSRI variance, so the pooled error term is worth questioning.
contrast_test(fit, contrasts = "pairwise", var_equal = FALSE)

# Pairwise treatment comparisons in the Smith, Meyers, and Delaney
# (1998) drinking trial, on the normalizing log scale. Each row is
# one pairwise contrast of the three treatment means.
fit_drinks <- aov(log_drinks ~ treatment, data = drinks_trial)
contrast_test(fit_drinks, contrasts = "pairwise")

# An a priori contrast: the two active CRA arms (averaged) versus
# standard care. With levels ordered Standard, CRA, CRA + Disulfiram,
# the weights c(-1, 0.5, 0.5) compare the active arms against Standard.
contrast_test(
  fit_drinks,
  contrasts = list("CRA arms vs Standard" = c(-1, 0.5, 0.5))
)
```

convert_cor_cov *Correlation Matrix to Covariance Matrix Conversion*

Description

Rescales a correlation matrix into the covariance matrix implied by a set of standard deviations, the inverse of the standardization that produces a correlation matrix from a covariance matrix. Useful when a published article reports correlations and standard deviations but an analysis needs the covariances.

Usage

```
convert_cor_cov(cor_mat, sd, discrepancy = 1e-05)
```

Arguments

cor_mat	The correlation matrix to be converted
sd	A vector that contains the standard deviations of the variables in the correlation matrix
discrepancy	A small nonnegative tolerance (near 0; default 1e-5). A value on the main diagonal of the correlation matrix is treated as equal to 1 when it is within discrepancy of 1, that is, when $\|d - 1\| \leq$ discrepancy

Details

The correlation matrix to convert can be either symmetric or triangular. The covariance matrix returned is always a symmetric matrix.

Value

A square numeric matrix giving the covariance matrix implied by the supplied correlation matrix and standard deviations, with the same row / column names as cor_mat.

Note

The correlation matrix input should be a square matrix, and the length of sd should be equal to the number of variables in the correlation matrix (i.e., the number of rows/columns). Sometimes the correlation matrix input may not have exactly 1's on the main diagonal, due to, e.g., rounding; discrepancy specifies the allowable discrepancy so that the function still considers the input as a correlation matrix and can proceed (but the function does not change the numbers on the main diagonal).

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

Other parameterization conversions: convert_F_chisq(), convert_R2, convert_Z_r(), convert_d_or(), convert_d_r(), convert_r_Z(), convert_t_smd, convert_z_normal()

Examples

```
Cor.Mat <- rbind(c(1.0000, 0.8254, 0.4261, 0.6237, 0.5901, 0.1564, 0.1551),
              c(0.8254, 1.0000, 0.5583, 0.5967, 0.6692, 0.1877, 0.2246),
              c(0.4261, 0.5583, 1.0000, 0.4933, 0.4455, 0.1472, 0.3433),
              c(0.6237, 0.5967, 0.4933, 1.0000, 0.6403, 0.1160, 0.5316),
              c(0.5901, 0.6692, 0.4455, 0.6403, 1.0000, 0.3769, 0.5742),
              c(0.1564, 0.1877, 0.1472, 0.1160, 0.3769, 1.0000, 0.2833),
              c(0.1551, 0.2246, 0.3433, 0.5316, 0.5742, 0.2833, 1.0000))
colnames(Cor.Mat) <- rownames(Cor.Mat) <- c("rating", "complaints", "privileges",
"learning", "raises", "critical", "advance")

SDs <- c(12.172562, 13.314757, 12.235430, 11.737013, 10.397226, 9.894908, 10.288706)
convert_cor_cov(cor_mat=Cor.Mat, sd=SDs)
```

---

convert_d_or *Convert Between the Standardized Mean Difference and the Odds Ratio*

Description

Invertible conversions between a two-group standardized mean difference (Cohen's *d*) and an odds ratio, by the logistic-distribution method of Hasselblad and Hedges (1995): a continuous outcome split at a threshold under logistic errors implies

$$d = \log(\mathrm{OR}) \cdot \frac{\sqrt{3}}{\pi}, \qquad \mathrm{OR} = \exp\!\left(d \cdot \pi / \sqrt{3}\right).$$

These conversions let binary-outcome studies enter a synthesis on the standardized mean difference scale, or mean-difference studies enter one on the odds ratio scale (Borenstein, Hedges, Higgins, & Rothstein, 2009, Chapter 7).

Usage

```
convert_d_or(d)

convert_or_d(or)
```

Arguments

d	The standardized mean difference.
or	The odds ratio, a single positive number.

Value

A data.frame (class dmar_tbl) with a single row: term odds_ratio (for convert_d_or) or smd (for convert_or_d) and its value.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Borenstein, M., Hedges, L. V., Higgins, J. P. T., & Rothstein, H. R. (2009). *Introduction to meta-analysis*. Wiley.

Hasselblad, V., & Hedges, L. V. (1995). Meta-analysis of screening and diagnostic tests. *Psychological Bulletin, 117*(1), 167–178. doi:10.1037/00332909.117.1.167

See Also

convert_d_r / convert_r_d for the correlation leg of the same triangle.

Other parameterization conversions: convert_F_chisq(), convert_R2, convert_Z_r(), convert_cor_cov(), convert_d_r(), convert_r_Z(), convert_t_smd, convert_z_normal()

Examples

```
# d = 0.5 corresponds to an odds ratio of about 2.48.
convert_d_or(d = 0.5)

# And back, exactly.
convert_or_d(or = convert_d_or(d = 0.5)$value)

# The null maps to the null: d = 0 is an odds ratio of 1.
convert_d_or(d = 0)
```

---

convert_d_r *Convert Between the Standardized Mean Difference and the Correlation*

Description

Invertible conversions between a two-group standardized mean difference (Cohen's *d*) and the (point-biserial) correlation between the outcome and group membership. convert_d_r() maps *d* to *r*; convert_r_d() maps *r* back to *d*. These are the standard conversions used to bring effect sizes reported in different metrics onto a common scale, for example when synthesizing a literature in which some studies report mean differences and others report correlations (Borenstein, Hedges, Higgins, & Rothstein, 2009, Chapter 7).

Usage

```
convert_d_r(d, n_1 = NULL, n_2 = NULL)

convert_r_d(r, n_1 = NULL, n_2 = NULL)
```

Arguments

d	The standardized mean difference.
n_1, n_2	Optional per-group sample sizes. When supplied, the conversion uses the unequal-group factor $a = (n_1 + n_2)^2/(n_1 n_2)$; when omitted, equal group sizes are assumed, for which $a = 4$.
r	The point-biserial correlation, in $(-1, 1)$.

Details

With $a = (n_1 + n_2)^2/(n_1 n_2)$ (equal to 4 for equal groups), the two directions are

$$r = \frac{d}{\sqrt{d^2 + a}}, \qquad d = \frac{\sqrt{a}\, r}{\sqrt{1 - r^2}},$$

exact inverses of one another for a given a. The same n_1 and n_2 must be supplied to both directions for the round trip to be exact.

Value

A data.frame (class dmar_tbl) with a single row: term r (for convert_d_r) or smd (for convert_r_d) and its value.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Borenstein, M., Hedges, L. V., Higgins, J. P. T., & Rothstein, H. R. (2009). *Introduction to meta-analysis*. Wiley.

See Also

convert_d_or / convert_or_d for the odds ratio leg of the same triangle; smd and ci_r for estimating the quantities being converted.

Other parameterization conversions: convert_F_chisq(), convert_R2, convert_Z_r(), convert_cor_cov(), convert_d_or(), convert_r_Z(), convert_t_smd, convert_z_normal()

Examples

```
# Equal groups: d = 0.5 corresponds to r about .243.
convert_d_r(d = 0.5)

# And back, exactly.
convert_r_d(r = convert_d_r(d = 0.5)$value)

# Unequal groups change the conversion factor.
convert_d_r(d = 0.5, n_1 = 20, n_2 = 80)
```

---

convert_F_chisq *Convert Between an* F *Value and a Chi Square Value*

Description

Converts an observed *F* value into a chi square value, and a chi square value into an *F* value. Both functions return the converted statistic itself, a single number, not a *p*-value.

Two conversions are available, selected by `df_denominator`.

- **Scaling (the default).** With `df_denominator = Inf`, `convert_F_chisq()` returns `df_numerator * F_value` and `convert_chisq_F()` returns `chisq / df`. This is the standard conversion between the two test statistics and needs no denominator degrees of freedom.
- **Probability matching.** With a finite `df_denominator`, `convert_F_chisq()` returns the chi square value that has the same upper-tail probability (the same *p*-value) as the *F* value, and `convert_chisq_F()` returns the *F* value with the same upper-tail probability as the chi square value.

Usage

```
convert_F_chisq(F_value, df_numerator, df_denominator = Inf)

convert_chisq_F(chisq, df, df_denominator = Inf)
```

Arguments

`F_value`	Observed *F* value. Must be nonnegative.
`df_numerator`	Numerator degrees of freedom of the *F*, which is also the degrees of freedom of the chi square.
`df_denominator`	Denominator degrees of freedom of the *F*, the degrees of freedom on which the error variance is estimated. The default, `Inf`, treats the error variance as known and gives the scaling conversion ($\chi^2 = \nu_1 F$); a finite value gives the probability-matching conversion. See *Details*.
`chisq`	Observed chi square value. Must be nonnegative.
`df`	Degrees of freedom of the chi square, which becomes the numerator degrees of freedom of the *F*.

Details

Why there are two conversions. An F statistic is the ratio of two independent chi squares, each divided by its degrees of freedom,

$$F(\nu_1, \nu_2) = \frac{\chi^2_{\nu_1}/\nu_1}{\chi^2_{\nu_2}/\nu_2},$$

where the denominator is the estimated error variance scaled to have a mean of 1. A chi square is what the numerator becomes when that error variance is known rather than estimated. This is the entire difference between the two, and it is why the conversion depends on how the error variance is treated.

Scaling: `df_denominator = Inf`**.** As the denominator degrees of freedom grow, the estimated error variance converges to the true one and $\nu_1 F \to \chi^2(\nu_1)$. The default therefore treats the error variance as known and returns exactly

$$\chi^2 = \nu_1\, F, \qquad F = \chi^2/\nu_1,$$

with ν_1 the numerator degrees of freedom (`df_numerator`, which is also the degrees of freedom of the chi square). This is the value an *F* table prints in its infinite-denominator row, and it is the usual conversion between a Wald *F* and a Wald chi square. It involves no probabilities and needs no `df_denominator`.

Probability matching: finite `df_denominator`**.** When the error variance is estimated on ν_2 degrees of freedom, the scaling above runs high, because $\nu_1 F$ is more dispersed than $\chi^2(\nu_1)$. Supplying `df_denominator` returns instead the chi square value at the same upper-tail probability. Writing `pf` and `qchisq` for R's distribution and quantile functions, the computation is exactly

```
p <- pf(F_value, df_numerator, df_denominator, lower.tail = FALSE)
chi_square <- qchisq(p, df_numerator, lower.tail = FALSE)
```

and `convert_chisq_F()` composes the same two functions in the other order. The upper tail is used so the *p*-value is represented accurately for large statistics (the lower-tail probability rounds to 1 in double precision by about $F = 500$ at small ν_2, which would send `qchisq()` to infinity; the upper-tail value stays accurate past $F = 10^{20}$). No logarithms are involved and the returned value is the chi square itself, not the *p*-value used to find it. The map is strictly increasing and therefore one to one, and `convert_chisq_F()` is its exact inverse.

How much the two differ. At $\nu_1 = 3$ and $F = 2.75$, probability matching gives $\chi^2 = 6.290$ at $\nu_2 = 10$, 7.711 at $\nu_2 = 50$, and 8.220 at $\nu_2 = 1000$, approaching the scaling value $\nu_1 F = 8.25$ only as ν_2 grows. Scaling and probability matching agree in the limit and diverge as ν_2 shrinks; for a small ν_2, the scaled value understates the *p*-value (with $\nu_1 = 3, \nu_2 = 5$, a result whose true *p* is .05 reads as .001 if $\nu_1 F$ is referred to χ^2).

Noncentrality. Both conversions are defined by the central distributions and preserve the *p*-value; neither transports a noncentrality parameter (a noncentral *F* is not carried to a noncentral chi square with the same λ, except in the $\nu_2 \to \infty$ limit). For noncentral work use `ci_nc_F` and `ci_nc_chisq`.

Special case. With $\nu_1 = 1$ this is the squared form of the relation between *t* and *z*, since $F(1, \nu) = t(\nu)^2$ and $\chi^2(1) = z^2$.

Value

A 1-row `data.frame` with columns `term` and `value`. The `term` is `"chi_square_from_F"` for `convert_F_chisq` and `"F_from_chi_square"` for `convert_chisq_F`, and `value` is the converted statistic (a chi square value or an F value, respectively; never a p-value).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Johnson, N. L., Kotz, S., & Balakrishnan, N. (1995). *Continuous univariate distributions* (2nd ed., Vol. 2). Wiley.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

`cv_f`, `cv_chisq`, `ci_nc_F`, `ci_nc_chisq`

Other parameterization conversions: `convert_R2`, `convert_Z_r()`, `convert_cor_cov()`, `convert_d_or()`, `convert_d_r()`, `convert_r_Z()`, `convert_t_smd`, `convert_z_normal()`

Examples

```
# Scaling (the default): chi square = df_numerator * F.
convert_F_chisq(2.75, df_numerator = 3)

# The inverse: F = chi square / df.
convert_chisq_F(8.25, df = 3)

# Probability matching at 3 and 50 degrees of freedom: the chi square
# with the same p-value as the F.
convert_F_chisq(2.75, df_numerator = 3, df_denominator = 50)

# The p-value is preserved, which is what probability matching means.
f <- 2.75
x <- convert_F_chisq(f, df_numerator = 3, df_denominator = 50)$value
c(p_from_F = pf(f, 3, 50, lower.tail = FALSE),
  p_from_chi_square = pchisq(x, 3, lower.tail = FALSE))
```

---

convert_R2 *Convert Between* F, *R^2, and Their Noncentral Parameters*

Description

Given values of test statistics (and the appropriate additional information) the value of the noncentral values can be obtained. Likewise, given noncentral values (and the appropriate additional information) the value of the test statistic can be obtained.

Usage

```
convert_R2_f(R2 = NULL, df_1 = NULL, df_2 = NULL, p = NULL, N = NULL)

convert_f_R2(F_value = NULL, df_1 = NULL, df_2 = NULL)

convert_lambda_R2(ncp = NULL, N = NULL)

convert_R2_lambda(R2 = NULL, N = NULL)
```

Arguments

R2	Squared multiple correlation coefficient (population or observed)
df_1	Degrees of freedom for the numerator of the F-distribution
df_2	Degrees of freedom for the denominator of the F-distribution
p	Number of predictor variables for R2
N	Sample size
F_value	The obtained F value from a test of significance for the squared multiple correlation coefficient
ncp	The noncentrality parameter (λ) of an F-distribution, the input of convert_lambda_R2()

Details

These functions are especially helpful in the search for confidence intervals for noncentral parameters, as they convert to and from related quantities.

Value

Each of the four functions returns a 1-row data.frame with columns term and value. The term entry identifies the conversion performed ("r2_f", "f_r2", "lambda_r2", or "r2_lambda") and value is the converted scalar. The conversions are exact inverses of one another (with the appropriate degrees-of-freedom / sample size inputs supplied), which is what makes them useful inside the noncentrality-parameter confidence interval machinery of ci_R2 and ci_nc_F.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

See Also

ss_aipe_R2, ci_R2, ci_nc_t, ci_nc_F

Other parameterization conversions: convert_F_chisq(), convert_Z_r(), convert_cor_cov(), convert_d_or(), convert_d_r(), convert_r_Z(), convert_t_smd, convert_z_normal()

Examples

```
convert_R2_lambda(R2 = .5, N = 100)
```

`convert_r_Z` *Convert a Correlation Coefficient (*r*) Into the Scale of Fisher's* Z

Description

This function converts a correlation coefficient into the scale of Fisher's *Z*, the variance-stabilizing transformation of a correlation. Many authors call this map the z-prime transform and write the transformed value as z'. The capital *Z* is meaningful: Fisher's *Z* is not a z-score (it is not a standardized variate, that is, an observation centered and divided by a standard deviation). It is the transform $Z = \operatorname{atanh}(r)$ of a correlation coefficient, applied because the sampling distribution of *Z* is approximately normal with a variance that does not depend on the population correlation, which makes *Z* convenient for forming confidence intervals.

Usage

```
convert_r_Z(r)
```

Arguments

`r`	Correlation coefficient (between two variables)

Details

This function is typically used in the context of forming a confidence interval for a population correlation coefficient. Note that, in that situation, the two variables are assumed to follow a bivariate normal distribution (e.g., Hays, 1994).

Value

A 1-row `data.frame` with columns `term` and `value`. The `term` is `"Z_from_r"` and `value` is Fisher's *Z* corresponding to the supplied correlation coefficient. The inverse direction is `convert_Z_r`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Hays, W. L. (1994). *Statistics* (5th ed.). Fort Worth, TX: Harcourt Brace College Publishers.

See Also

convert_Z_r, ci_r

Other parameterization conversions: convert_F_chisq(), convert_R2, convert_Z_r(), convert_cor_cov(), convert_d_or(), convert_d_r(), convert_t_smd, convert_z_normal()

Examples

```
# From Hays (1994, pp. 649--650)
convert_r_Z(.35)
```

---

convert_t_smd *Conversion Functions for Noncentral* t*-distribution*

Description

Functions useful for converting a standardized mean difference to a noncentrality parameter, and vice versa.

Usage

```
convert_delta_lambda(smd, n_1, n_2)

convert_lambda_delta(ncp, n_1, n_2)
```

Arguments

smd	standardized mean difference (δ), a population value or the observed standardized mean difference as its estimate
n_1	Sample size in group 1
n_2	Sample size in group 2
ncp	noncentrality parameter of a t-distribution (λ), or the observed t-value as its estimate

Details

Although ncp is the population noncentral value, an estimate of it is the observed value of a t-statistic. Likewise, the population standardized mean difference can be estimated by the observed standardized mean difference, given as smd. Thus, the observed standardized mean difference can be converted to the observed t-value. These functions are especially helpful in the context of forming confidence intervals for the population standardized mean difference.

Value

Each function returns a 1-row data.frame with columns term and value. The term entry identifies the conversion ("delta_lambda" or "lambda_delta") and value is the converted scalar. The two functions are exact inverses given the *per-group* sample sizes n_1 and n_2.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

See Also

smd, ci_smd, ss_aipe_smd

Other parameterization conversions: convert_F_chisq(), convert_R2, convert_Z_r(), convert_cor_cov(), convert_d_or(), convert_d_r(), convert_r_Z(), convert_z_normal()

Examples

```
convert_lambda_delta(ncp = 2, n_1 = 113, n_2 = 113)
convert_delta_lambda(smd = .266076, n_1 = 113, n_2 = 113)
```

---

convert_z_normal *Convert a Standard Normal* z *Value to the Corresponding Value on a Normal Distribution*

Description

This function maps a value on the standard normal distribution (the z-distribution, with mean 0 and variance 1) to the equivalent point on a normal distribution with arbitrary mean and standard deviation, $N(mean, sd^2)$.

Usage

```
convert_z_normal(z, mean = 0, sd = 1)
```

Arguments

z	A value on the standard normal distribution (with mean 0 and variance 1).
mean	The mean of the target normal distribution.
sd	The standard deviation of the target normal distribution.

Details

The conversion is value = mean + z * sd, which places the returned value at the same percentile of $N(mean, sd^2)$ that z occupies on the standard normal distribution. Equivalently, value = qnorm(pnorm(z), mean, sd). With the defaults (mean = 0, sd = 1) the value is returned unchanged, since the target distribution is then the standard normal distribution itself.

Value

A 1-row `data.frame` with columns `term` and `value`. The `term` is `"value_from_z"` and `value` is the point on $N(mean, sd^2)$ that lies at the same percentile as `z` does on the standard normal distribution.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

`cv_z`

Other parameterization conversions: `convert_F_chisq()`, `convert_R2`, `convert_Z_r()`, `convert_cor_cov()`, `convert_d_or()`, `convert_d_r()`, `convert_r_Z()`, `convert_t_smd`

Examples

```
# A z value of 1.96 on the standard normal distribution maps to the
# corresponding point on a normal distribution with mean 100 and sd 15.
convert_z_normal(z = 1.96, mean = 100, sd = 15)

# With the default standard normal target, the value is returned unchanged.
convert_z_normal(z = 1.96)
```

---

convert_Z_r *Convert Fisher's* Z *Into the Scale of a Correlation Coefficient* (r)

Description

Converts Fisher's Z back into the scale of a correlation coefficient (r). Fisher's Z is the variance-stabilizing transformation of a correlation; many authors call it the z-prime transform and write the transformed value as z'. The capital Z is meaningful: Fisher's Z is not a z-score (it is not a standardized variate, that is, an observation centered and divided by a standard deviation). This function applies the inverse transform $r = \tanh(Z)$ to return to the scale of a correlation coefficient.

Usage

```
convert_Z_r(Z)
```

Arguments

`Z`	Fisher's Z (the variance-stabilizing transform of a correlation, which many authors call z')

Details

This function is typically used in the context of forming a confidence interval for a population correlation coefficient. Note that, in that situation, the two variables are assumed to follow a bivariate normal distribution (e.g., Hays, 1994).

Value

A 1-row `data.frame` with columns `term` and `value`. The `term` is `"r_from_Z"` and `value` is the correlation coefficient corresponding to the supplied Fisher's *Z*. The inverse direction is `convert_r_Z`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Hays, W. L. (1994). *Statistics* (5th ed.). Fort Worth, TX: Harcourt Brace College Publishers.

See Also

`convert_r_Z`, `ci_r`

Other parameterization conversions: `convert_F_chisq()`, `convert_R2`, `convert_cor_cov()`, `convert_d_or()`, `convert_d_r()`, `convert_r_Z()`, `convert_t_smd`, `convert_z_normal()`

Examples

```
# From Hays (1994, pp. 649--650)
convert_Z_r(0.3654438)
```

`correction_for_attenuation`

Correct a Correlation for Attenuation Due to Measurement Error

---

Description

Applies the Spearman (1904) correction for attenuation: the observed correlation between two fallible measures understates the correlation between the constructs they measure, and dividing by the square root of the product of the two reliabilities recovers it,

$$r_c = \frac{r_{XY}}{\sqrt{\rho_{XX'}\,\rho_{YY'}}}.$$

Within classical test theory (Lord & Novick, 1968), the disattenuated correlation estimates the correlation between the true scores, that is, how strongly the two constructs would correlate if each

were measured without error. When `N` is supplied, a confidence interval for the corrected correlation is formed by disattenuating the endpoints of the Fisher's Z interval for the observed correlation, the standard practice when the reliabilities are treated as known.

Usage

```
correction_for_attenuation(
  r,
  reliability_x,
  reliability_y,
  N = NULL,
  conf_level = 0.95
)
```

Arguments

`r`	The observed correlation between the two measures, in $[-1, 1]$.
`reliability_x`	Reliability of the first measure, in $(0, 1]$. Any reliability estimate appropriate to the use may be supplied (for example, coefficient alpha or omega from `reliability`).
`reliability_y`	Reliability of the second measure, in $(0, 1]$. For a correlation between a measure and an error-free criterion, supply 1 for that side (correcting for criterion unreliability only yields what the validity generalization literature calls the operational validity).
`N`	Optional sample size on which `r` is based. When supplied, a `conf_level` confidence interval for the corrected correlation is reported by correcting the endpoints of the Fisher's Z interval for `r`.
`conf_level`	Confidence level for the interval when `N` is supplied. Defaults to 0.95.

Details

The correction treats the two reliabilities as known constants, which is the conventional assumption; uncertainty in the reliabilities themselves would widen the interval further. Because the observed correlation can exceed what the supplied reliabilities allow (sampling error, or reliabilities that understate the truth), the corrected value can exceed 1 in magnitude; when that happens the value is reported as computed, with a warning, rather than silently truncated, since a corrected correlation beyond 1 is itself diagnostic information about the inputs.

Prefer the factor model when you have the items. The Spearman formula is the summary-statistics route: it is exactly right when all you have are the observed correlation and reliability estimates. When the item-level data are available, the better practice is to estimate the construct-level correlation directly as the factor correlation in a two-factor model (each scale loading on its own factor, factors free to correlate): the latent correlation is then estimated jointly with the measurement model rather than assembled from plug-in reliabilities, and it comes with a standard error that propagates the sampling variability of all the moving parts. The example below shows both routes on the same data, the formula route using `reliability_omega`, the model route using **lavaan**; with congeneric items the two agree closely, and when they disagree the factor model is the one to trust.

Value

A data.frame (class dmar_tbl) in term / value layout with the observed correlation (correlation_observed), the corrected correlation (correlation_corrected), the corrected interval (ci_lower, ci_upper; present only when N is supplied), and the reliability_x, reliability_y, and N inputs. With an interval the table carries the attributes conf_level and ci_estimate, which names correlation_corrected as the row the interval belongs to, so that results_sentence reports the corrected correlation with it.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lord, F. M., & Novick, M. R. (1968). *Statistical theories of mental test scores*. Addison-Wesley.

Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology, 15*(1), 72–101.

See Also

reliability and its family for estimating the reliabilities supplied here; cfa_1 and **lavaan** for the latent variable route the Details recommend when items are available; ci_r for inference on the observed correlation itself; convert_r_Z and convert_Z_r for the Fisher transformation the interval uses.

Other effect size estimates: cles(), cliff_delta(), eta_squared(), eta_squared_generalized(), expected_partial_r(), expected_r(), expected_smd(), nnt_from_smd(), omega_squared(), probability_of_superiority_paired(), proportion_of_superiority(), responder_analysis(), smd_trimmed()

Examples

```
# An observed correlation of .30 between measures with reliabilities .80
# and .70 corresponds to a construct-level correlation of about .40.
correction_for_attenuation(r = 0.30, reliability_x = 0.80, reliability_y = 0.70)

# With the sample size, the corrected interval comes along.
correction_for_attenuation(r = 0.30, reliability_x = 0.80, reliability_y = 0.70,
                   N = 120)

# Correct one side only (error-free criterion).
correction_for_attenuation(r = 0.30, reliability_x = 0.80, reliability_y = 1)

# The two routes to the construct-level correlation, on the same data
# (requires lavaan). Two congeneric scales of three items each whose
# latent variables correlate .50:
set.seed(113)
n <- 400
fx <- rnorm(n); fy <- 0.5 * fx + sqrt(1 - 0.25) * rnorm(n)
lam <- c(.8, .7, .6)
items <- data.frame(
```

```
  x1 = lam[1] * fx + rnorm(n, 0, sqrt(1 - lam[1]^2)),
  x2 = lam[2] * fx + rnorm(n, 0, sqrt(1 - lam[2]^2)),
  x3 = lam[3] * fx + rnorm(n, 0, sqrt(1 - lam[3]^2)),
  y1 = lam[1] * fy + rnorm(n, 0, sqrt(1 - lam[1]^2)),
  y2 = lam[2] * fy + rnorm(n, 0, sqrt(1 - lam[2]^2)),
  y3 = lam[3] * fy + rnorm(n, 0, sqrt(1 - lam[3]^2)))

# Route 1, summary statistics: omega reliabilities into the formula.
x_score <- rowMeans(items[, 1:3]); y_score <- rowMeans(items[, 4:6])
om_x <- reliability_omega(data = items[, 1:3])$value[1]
om_y <- reliability_omega(data = items[, 4:6])$value[1]
correction_for_attenuation(r = cor(x_score, y_score),
                           reliability_x = om_x, reliability_y = om_y,
                           N = n)

# Route 2, the factor model: the latent correlation estimated directly.
fit <- lavaan::cfa("X =~ x1 + x2 + x3\nY =~ y1 + y2 + y3",
                   data = items, std.lv = TRUE)
lavaan::parameterEstimates(fit)[
  lavaan::parameterEstimates(fit)$op == "~~" &
  lavaan::parameterEstimates(fit)$lhs == "X" &
  lavaan::parameterEstimates(fit)$rhs == "Y", ]
```

`correlations_test` *Formatted Correlation Matrix With* p*-values and Confidence Intervals*

Description

Computes a correlation matrix along with, for every pair of variables, the two-sided *p*-value, a confidence interval, and the pairwise sample size, and arranges them into a single annotated table. The table is a convenience for inspecting the correlations, their significance, and their intervals at a glance; it is not intended as a finished, publication-ready exhibit. Output formats include plain text (for the console or to paste into a Word document), HTML (best for Word via browser copy-paste), and LaTeX.

Usage

```
correlations_test(
  x,
  method = "pearson",
  conf_level = 0.95,
  listwise = FALSE,
  stars = FALSE,
  decimals_r = 2,
  decimals_p = 4,
  format = "text",
  file = NULL
)
```

Arguments

x	A data.frame, tibble, or matrix. Non-numeric columns are dropped with no warning; the remaining numeric, integer, and logical columns are used.
method	The correlation method: "pearson" (default), "spearman", or "kendall".
conf_level	Confidence level for the interval (default 0.95).
listwise	Logical. If TRUE, apply listwise deletion before any correlation is computed so that every pair uses the same sample. If FALSE (the default), each pair uses all of its available complete observations ("pairwise" deletion).
stars	Logical. If TRUE, append conventional significance stars (* $p < .05$, ** $p < .01$, *** $p < .001$) to each correlation and add an explanatory footnote to the table. The exact p-value is always shown as well.
decimals_r	Number of decimals for correlations and confidence interval limits (default 2).
decimals_p	Number of decimals for p-values (default 4, matching the package-wide digits_p convention used by dmar_tbl); values below $10^{-\text{decimals_p}}$ are printed as "< .0001" (with the threshold tracking decimals_p).
format	One of "text" (default), "html", or "latex". Controls the returned/printed table. See Details.
file	Optional file path. If supplied, the formatted table is written to this file. The HTML path writes a self-contained HTML document (the kable wrapped in a minimal document head that pulls in Bootstrap CSS from a CDN) so the file opens directly in a browser without any pandoc / webshot machinery. The LaTeX path writes the raw tabular fragment; embed it in a document that loads \usepackage{makecell} and \usepackage{booktabs}. The text path uses writeLines.

Details

Layout. Each lower-triangle cell stacks four values: the correlation (with optional significance stars), the two-sided p-value, the confidence interval, and the pairwise sample size. The upper triangle is left blank so the same information is not repeated.

***p*-values.** Computed with cor.test using the requested method. For Spearman and Kendall with ties, cor.test cannot compute an exact p-value and falls back to a normal-approximation p-value; the associated warnings are suppressed for a cleaner table.

Confidence intervals. All three methods use Fisher's variance- stabilizing transformation, $Z(r) = \operatorname{atanh}(r)$, and back-transform through $\tanh(\cdot)$, but the standard error on the Fisher's Z scale is selected to match the sampling distribution of the chosen correlation coefficient:

- Pearson (method = "pearson"): $\mathrm{SE}(Z) = 1/\sqrt{n-3}$. This is the classical Fisher (1921) interval. Under bivariate normality the coverage of this interval matches cor.test's conf.int exactly; for departures from bivariate normality both intervals lose coverage in the same way. See Kelley (2007) and Maxwell, Delaney, & Kelley (2027, Chapter 9) for discussion and worked examples.
- Spearman (method = "spearman"): $\mathrm{SE}(Z) = \sqrt{(1 + r^2/2)/(n-3)}$. This is the Bonett and Wright (2000) adjustment, which uses Fisher's transformation but inflates the standard error to account for the heavier-than-Pearson tails of the Spearman sampling distribution. This is

the form Bonett and Wright recommend for practical use; a plain Fisher $1/\sqrt{n-3}$ standard error tends to produce intervals that are too narrow for Spearman correlations.

- Kendall (`method = "kendall"`): $\mathrm{SE}(Z) = \sqrt{0.437/(n-4)}$. This is Bonett and Wright's (2000, equation 2) Fisher's Z interval for Kendall's τ. The constant 0.437 is the asymptotic variance factor of Fieller, Hartley, and Pearson (1957), derived under bivariate normality and stated by Bonett and Wright as accurate for $|\tau| < .8$ (the Spearman variance above is likewise stated as accurate for $|\rho_s| < .95$). Requires $n \geq 5$; for smaller pairwise samples the interval is returned as `NA`.

Fisher's Z transformation is finite only for $|r| < 1$. When $r = \pm 1$ (a perfect correlation in the sample), $Z(r)$ is infinite, and DMAR reports both limits of the interval as `NA`, for every `method`. That is DMAR's own choice, made because the interval is built on the Fisher's Z scale, where the estimate is then infinite. It is not the convention of `cor.test`, which for a perfect Pearson correlation reports the interval [1, 1] (or [-1, -1]) as its `conf.int`. The correlation and its p-value are reported as usual.

When to use each correlation. Pick `method = "pearson"` when both variables are continuous, approximately linearly related, and roughly bivariate normal (or at least without heavy tails and influential outliers). Pick `method = "spearman"` or `method = "kendall"` when the relationship is monotone but not necessarily linear, when one or both variables are ordinal, or when influential outliers would distort Pearson's *r*. Kendall's τ is often preferred over Spearman's ρ for small samples and for samples with many tied ranks because it has better small-sample properties and a more interpretable concordance-based meaning. See Maxwell, Delaney, & Kelley (2027, Chapter 9) for an extended discussion of effect size choice and interval estimation.

HTML/LaTeX output. Built with `knitr::kable` (a `Suggests` dependency). Cell content and variable names are escaped for the target format so that "$p < .001$" renders correctly and that variable names containing characters such as `_`, `&`, or `%` do not break LaTeX compilation. LaTeX output uses `\makecell`, which requires `\usepackage{makecell}` in the document preamble.

Pasting into Word. The cleanest path is `format = "html"` with a `file` argument; the function writes a small self-contained HTML document (no pandoc dependency). Open the result in a browser and copy/paste the table into Word. Formatting (including the stacked-cell layout) is preserved.

Value

An object of class `"correlations_test"` containing the matrices `r`, `p`, `ci_lower`, `ci_upper`, and `n` (all $p \times p$ with variable names as row/column names), plus the arguments used. When `format = "html"` or `format = "latex"` and `file` is `NULL`, a `kable` object is returned instead so that the table renders inside R Markdown / Quarto documents. When `format = "text"`, the table is printed to the console and the raw object is returned invisibly.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G., & Wright, T. A. (2000). Sample size requirements for estimating Pearson, Kendall and Spearman correlations. *Psychometrika, 65*(1), 23–28. doi:10.1007/BF02294183

Fieller, E. C., Hartley, H. O., & Pearson, E. S. (1957). Tests for rank correlation coefficients. I. *Biometrika, 44*(3/4), 470–481. doi:10.1093/biomet/44.34.470

Fisher, R. A. (1915). Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika, 10*(4), 507–521. doi:10.1093/biomet/10.4.507

Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron, 1*, 3–32.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

`descriptives`, `cor.test`, `cor`, `ci_r`

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# Worked example using four cognitive tests from the holzinger_swineford
# data (301 children in two schools). The goal of
# correlations_test() is to produce a formatted correlation matrix that
# reports, for every variable pair, the correlation, its two-sided
# p-value, a confidence interval on the population correlation, and the
# pairwise sample size. See Kelley (2007) and Maxwell, Delaney, & Kelley
# (2027, Chapter 9) for discussion of why effect sizes should be
# accompanied by confidence intervals.
hs_tests <- holzinger_swineford[, c("t1_visual_perception", "t2_cubes",
                                    "t4_lozenges",
                                    "t6_paragraph_comprehension")]

# Pearson correlations (the default). Each lower-triangle cell stacks r,
# the two-sided p-value, the 95\% confidence interval (Fisher's Z
# transformation; Fisher, 1915, 1921), and the pairwise N.
correlations_test(hs_tests)

# Add significance stars and an explanatory footnote.
correlations_test(hs_tests, stars = TRUE)

# Spearman correlations at a 99\% confidence level. The interval uses
# Bonett and Wright's (2000) Fisher's Z standard error
# sqrt((1 + r^2/2) / (n - 3)), which corrects the plain Fisher interval
# for the heavier tails of Spearman's sampling distribution.
correlations_test(hs_tests, method = "spearman", conf_level = 0.99)

# Kendall's tau, also using Bonett and Wright's (2000) Fisher's Z standard
```

```
# error sqrt(0.437 / (n - 4)). Kendall is often preferred over Spearman
# for small samples and for samples with many tied ranks, and these
# integer test scores carry many ties.
correlations_test(hs_tests, method = "kendall")

# Save a formatted HTML table that opens directly in a browser
# (then copy into Word). No pandoc required.
tmp_html <- tempfile(fileext = ".html")
correlations_test(hs_tests, stars = TRUE, format = "html", file = tmp_html)
```

`covmat_from_cfa` *Model Implied Covariance Matrix of a One-Factor Confirmatory Factor Model*

Description

Computes the covariance matrix of the indicators that a single-factor confirmatory factor model implies, from the factor loadings and the residual variances of the indicators:

$$\Sigma = \lambda\lambda^\top + \Theta, \qquad \Theta = \mathrm{diag}(\theta_1, \ldots, \theta_p).$$

The loadings and residual variances may be population values, for a covariance matrix to plan a study or to simulate from, or estimates from a fitted model, for the covariance matrix the fitted model implies. The calculation is the same in both cases, and the result is the model implied covariance matrix: the population covariance matrix of the model when the inputs are population values, and the fitted model's covariance matrix when they are estimates.

Usage

```
covmat_from_cfa(factor_loading, residual_var, tol_det = 1e-05)
```

Arguments

`factor_loading`	A numeric vector of factor loadings, one per indicator (at least two), on the metric of a factor with variance 1. A single-row or single-column matrix is accepted and read as a vector. Population loadings or estimates from a fitted model.
`residual_var`	A numeric vector of residual variances (unique or error variances), one per indicator, or a single value used for every indicator (the equal residual variance model). Population values or estimates from a fitted model.
`tol_det`	The value below which the determinant of the implied covariance matrix triggers a warning that the matrix may not be positive definite (default `1e-05`).

Details

Under the single-factor model, each indicator score is $x_i = \lambda_i \xi + \delta_i$, where ξ is the common factor and δ_i is the residual of indicator i, with variance θ_i, uncorrelated with the factor and with the other residuals. The general form of the model implied covariance matrix, $\Lambda\Phi\Lambda^\top + \Theta$, reduces to $\lambda\lambda^\top + \Theta$ here: Λ is the $p \times 1$ column of loadings $\lambda = (\lambda_1, \ldots, \lambda_p)^\top$ (`factor_loading`), the factor variance Φ is fixed at 1, and Θ is the diagonal matrix of the residual variances (`residual_var`). Because Θ is diagonal, the residuals are uncorrelated by construction; a model with correlated residuals needs a more general formulation than this function provides.

Estimates from a fit that fixed one loading at 1 and estimated the factor variance ϕ are put on this metric by multiplying each loading by $\sqrt{\phi}$; the residual variances are unchanged, and so is the implied matrix. `cfa_1` fixes the factor variance at 1, so its loadings and residual variances can be given as they are.

Estimated residual variances at or below 0 (a Heywood case) are used as given; the function does not check their sign. The determinant check warns only when the determinant of the implied matrix is below `tol_det`, which a Heywood case need not produce: with loadings .9, .8, and 1.1 and residual variances .2, .3, and -.01, the implied matrix is positive definite and no warning is given. A determinant above `tol_det` does not show that the matrix is positive definite either, because an even number of negative eigenvalues gives a positive determinant. Check the residual variances themselves for a Heywood case, and the eigenvalues of the result for positive definiteness.

The `cfa` spelling matches the `cfa_1` naming.

Value

A list with the single element `model_implied_cov`: the model implied covariance matrix of the indicators, a symmetric $p \times p$ matrix.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

`cfa_1`, `ss_aipe_reliability`

Examples

```
# Five indicators with equal loadings and equal residual variances
covmat_from_cfa(factor_loading = rep(0.7, 5), residual_var = rep(0.51, 5))

# Unequal loadings
covmat_from_cfa(factor_loading = c(0.5, 0.6, 0.7, 0.8),
                residual_var   = c(0.75, 0.64, 0.51, 0.36))

# From the estimates of a fitted one-factor model, whose factor variance
# cfa_1() fixes at 1: the covariance matrix the fitted model implies for
# four verbal tests of holzinger_swineford. The tests have no missing
# scores, so the model is fit to their covariance matrix, which gives
# the estimates the raw scores give.
items <- c("t5_general_information", "t6_paragraph_comprehension",
```

```
          "t7_sentence", "t9_word_meaning")
fit <- cfa_1(S = cov(holzinger_swineford[, items]), N = 301)
covmat_from_cfa(
  factor_loading = fit$estimate[grepl("=~", fit$syntax)],
  residual_var = fit$estimate[fit$syntax %in% paste(items, "~~", items)])
```

`cov_fiml` *Maximum Likelihood Covariance Matrix and Means of Incomplete Data*

Description

For continuous variables some of whose values are missing, `cov_fiml()` returns the maximum likelihood estimates of their covariance matrix and mean vector under the multivariate normal model, computed from every observed value rather than from the complete cases alone. These are the estimates of the saturated (unrestricted) model fitted by full information maximum likelihood (FIML), the model in which every variance, covariance, and mean is free. They are computed by the expectation maximization (EM) algorithm of Dempster, Laird, and Rubin (1977) and are often called the EM covariance matrix and the EM means (Enders & Peugh, 2004). The function requires **lavaan**, whose EM implementation does the computing.

The usual alternatives each give something up. Listwise deletion keeps only the complete cases, and its estimates are consistent in general only when values are missing completely at random (MCAR). Pairwise deletion estimates each covariance from a different subset of cases, so the matrix need not be positive definite and has no single sample size (Marsh, 1998). The EM estimates use every case and are consistent when values are missing at random (MAR) given the variables in the model (Schafer & Graham, 2002). They are therefore the matrix to hand to a method that starts from a covariance or correlation matrix, such as an exploratory factor analysis or a parallel analysis of incomplete items.

Usage

```
cov_fiml(data, aux = NULL)
```

Arguments

`data`	A `data.frame` or numeric matrix of continuous variables, with missing values coded `NA`. Every column that `aux` does not name is an analysis variable, a row and column of the returned `S`. Every column must be numeric; a factor, character, or logical column is refused, as is a column with fewer than two distinct observed values.
`aux`	Optional character vector naming auxiliary columns of `data`: variables that are not wanted in `S` but carry information about the missing values, because they predict which values are missing, the missing values themselves, or both. They enter the saturated model with the analysis variables, so their information reaches `S` and `mu`, and are then left out of what is returned (the saturated correlates treatment of Graham, 2003, the one the reliability family's `aux` argument uses).

Details

What is estimated. With complete data the estimates are the sample moments with the maximum likelihood divisor: `S` equals `cov(data) * (N - 1) / N` and `mu` equals `colMeans(data)`. With missing values, the EM algorithm alternates two steps until the estimates stop changing. The expectation step fills in the sums and cross products each case would contribute, replacing a missing value by its regression prediction from the observed values of its case under the current estimates and adding the residual covariance to the cross products; the maximization step recomputes the moments from those sums. No iteration lowers the likelihood of the observed data. Meeting the stopping criterion establishes numerical stability of the iterations, not a finite global maximum or uniqueness. The estimates are those of the unrestricted (h1) model of every FIML fit in **lavaan**: `lavaan::lavInspect(fit, "sampstat.h1")` returns them for a model fitted with `missing = "ml"` to the same variables. `cov_fiml()` computes them with **lavaan**'s EM (`lavaan::lavCor()` with `missing = "ml"`) and its default convergence criterion, which stops the iterations when no mean or covariance changes by more than 10^{-5}. That is a criterion on the absolute change, and in the units of variables whose standard deviations are near .001 it is met after an iteration or two, far from the maximum. `cov_fiml()` therefore runs the algorithm on the variables centered at their means and divided by their standard deviations, each computed from the variable's observed values, and transforms the estimates back, so that they do not depend on the units: multiplying a variable by a constant multiplies its mean, and its row and column of `S`, by that constant. They agree with **lavaan**'s own estimates of the same moments to the precision of the criterion, about seven significant digits. When the algorithm stops at its iteration limit instead, `converged` is `FALSE`, a warning says so, and `S` and `mu` are the last iterate rather than the maximum likelihood estimates.

When the estimates are consistent. The estimates are consistent when the values are MAR given the variables in the saturated model: the probability that a value is missing may depend on the observed values of the variables included, but not on the missing values themselves (Schafer & Graham, 2002). MAR is defined relative to the variables analyzed (Enders, 2001). If whether an item is missing depends on a variable that is left out, MAR need not hold for the items alone, and naming that variable in `aux` brings it back into the model without making it part of `S`. Including such variables (the inclusive strategy) costs little and can remove a bias that leaving them out would leave in place (Collins, Schafer, & Kam, 2001). Under missingness that depends on the missing values themselves (not at random), no choice of auxiliaries guarantees consistency. Normality enters as the working likelihood: Yuan, Yang-Wallentin, and Bentler (2012) summarize results showing that the normal theory estimates can remain consistent when the population distribution is unknown, and their simulations found the normal theory variance estimates biased at smaller sample sizes when the data were not normal, a bias that shrank as the sample grew.

Cases used. A case on which every variable, analysis and auxiliary, is missing carries no information and is set aside, as **lavaan** would set it aside. Every other case is used, including a case whose analysis variables are all missing but whose auxiliaries are observed. Under MAR given an auxiliary, such cases carry the auxiliary values of exactly the people whose analysis variables are missing, which is how the model learns how those people differ, and setting them aside would pull the estimates back toward those of the cases that responded. `N` counts the cases used and `N_complete` the cases complete on the analysis variables, all that listwise deletion would keep.

Refusals. Two variables that are never observed together have a covariance the data say nothing about: the likelihood is flat in that direction, and an EM estimate of it reflects the starting values and the other covariances rather than the data. The same holds, in substance, for two variables observed together only in rows where one of them takes a single value (or in one row only): those rows say nothing about how the two vary together, and the estimate of their covariance rests on the form of

the normal model, or is not determined at all when one of them is observed nowhere else. The function refuses either kind of pair and names it; missing_coverage reports the number of cases behind every pair.

Using the matrix downstream. N is the number of cases used, not an effective sample size, and the matrix carries less information than N complete cases would. A method that takes a covariance matrix and a sample size and treats them as complete data gets consistent point estimates from S, but its standard errors and test statistics are too optimistic, because they ignore the information the missing values took away (Savalei & Bentler, 2009). Enders and Peugh (2004) studied which sample size to give such a method when fitting a confirmatory factor model: the smallest number of cases behind any covariance gave Type I error rates near the nominal level when every indicator had missing values (the smallest number behind any variance did when only some indicators had them), and the harmonic mean of the numbers behind the variances gave confidence intervals with coverage near the nominal level. missing_coverage reports the number of cases behind every variance and covariance. The two-stage approach of Savalei and Bentler (2009) corrects the standard errors and the test statistic of a model fitted to the EM moments (**lavaan**'s missing = "two.stage"), and a model fitted to the raw data by FIML needs no correction. The fraction of missing information (fraction_missing_information) says what the missing values cost each parameter of such a fit. For point estimates, such as the eigenvalues of a factor analysis or of a parallel analysis, the EM matrix is the consistent basis.

Standard errors. The function returns point estimates only. The sampling variability that matters is that of whatever is computed from the matrix, and the analysis that computes it is where its interval belongs.

Value

A list with the same six elements whatever the data. A matrix is what every use of the result takes (stats::cov2cor(), stats::factanal(covmat =), a function's S argument), so the result holds the matrix itself, as cov_sem and covmat_from_cfa return theirs, rather than a table of its elements:

S The maximum likelihood covariance matrix of the analysis variables (divisor N), with their names as row and column names, in the order of the columns of data.

mu The maximum likelihood mean vector of the analysis variables, named, in the same order.

N The number of cases used: those with at least one observed value among the analysis variables and the auxiliaries.

N_complete The number of cases observed on every analysis variable.

patterns The number of distinct patterns of observed and missing values among the cases used, over every variable in the saturated model (the analysis variables and the auxiliaries); missing_patterns tabulates them.

converged TRUE when the EM algorithm met its convergence criterion.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Collins, L. M., Schafer, J. L., & Kam, C.-M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods, 6*(4), 330–351. doi:10.1037/1082989X.6.4.330

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological), 39*(1), 1–38.

Enders, C. K. (2001). A primer on maximum likelihood algorithms available for use with missing data. *Structural Equation Modeling, 8*(1), 128–141.

Enders, C. K., & Peugh, J. L. (2004). Using an EM covariance matrix to estimate structural equation models with missing data: Choosing an adjusted sample size to improve the accuracy of inferences. *Structural Equation Modeling, 11*(1), 1–19.

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

Marsh, H. W. (1998). Pairwise deletion for missing data in structural equation models: Nonpositive definite matrices, parameter estimates, goodness of fit, and adjusted sample sizes. *Structural Equation Modeling, 5*(1), 22–36. doi:10.1080/10705519809540087

Savalei, V., & Bentler, P. M. (2009). A two-stage approach to missing data: Theory and application to auxiliary variables. *Structural Equation Modeling, 16*(3), 477–497. doi:10.1080/10705510903008238

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods, 7*(2), 147–177. doi:10.1037/1082989X.7.2.147

Yuan, K.-H., Yang-Wallentin, F., & Bentler, P. M. (2012). ML versus MI for missing data with violation of distribution conditions. *Sociological Methods & Research, 41*(4), 598–629. doi:10.1177/0049124112460373

See Also

`missing_patterns` and `missing_coverage` to describe the missing values first; `fraction_missing_information` for what they cost the parameters of a FIML fit; `reliability_alpha` and `reliability_omega`, whose `missing = "fiml"` paths also bring auxiliary variables into a saturated model.

Other missing data: `df_barnard_rubin()`, `df_missing()`, `fraction_missing_information()`, `missing_coverage()`, `missing_patterns()`

Examples

```
# The six spatial tests of the Holzinger and Swineford data. The revised
# paper form board (t25) and flags (t26) tests were given only at
# Grant-White, so the 156 Pasteur pupils are missing both of them.
spatial <- c("t1_visual_perception", "t2_cubes", "t3_paper_form_board",
             "t4_lozenges", "t25_paper_form_board_r", "t26_flags")
em <- cov_fiml(holzinger_swineford[spatial])
em$N
em$N_complete
em$patterns
round(stats::cov2cor(em$S), 2)
```

```
# Listwise deletion would keep the 145 Grant-White pupils alone and
# estimate every correlation from them.
round(stats::cor(holzinger_swineford[spatial], use = "complete.obs"), 2)

# Values missing at random given a variable that is not analyzed: word
# meaning (t9) goes missing more often for pupils who scored high on
# general information (t5). The two visual tests say little about those
# pupils, so without t5 the mean of t9 stays near that of the pupils who
# took it; named as an auxiliary, t5 recovers most of the difference.
set.seed(113)
d <- holzinger_swineford[c("t1_visual_perception", "t4_lozenges",
                           "t9_word_meaning", "t5_general_information")]
z <- as.numeric(scale(d$t5_general_information))
p_missing <- stats::plogis(-0.5 + 2 * z)
d$t9_word_meaning[stats::runif(nrow(d)) < p_missing] <- NA
without_aux <- cov_fiml(d[1:3])
with_aux <- cov_fiml(d, aux = "t5_general_information")
rbind(without_aux = without_aux$mu, with_aux = with_aux$mu,
      full_data = colMeans(holzinger_swineford[names(d)[1:3]]))
```

`cov_sem` *Model Implied Covariance Matrix From a Lavaan-Specified SEM*

Description

Given a structural equation model written in lavaan model syntax with all of its parameters fixed to their population values, compute the model implied population covariance matrix $\Sigma(\theta)$ of the observed variables and, when the model has a mean structure, the model implied population mean vector $\mu(\theta)$. This function requires **lavaan** to be installed.

This is the helper that drives the *population* side of the sample size planning workflow for SEM: it lets the user state a population model, obtain the $\Sigma(\theta)$ (and $\mu(\theta)$) those fixed values imply, and then pass that population to `ss_aipe_sem_path`, `ss_aipe_sem_path_sensitivity`, `ss_aipe_rmsea_sensitivity`, `ss_power_composite_sem`, or `ss_aipe_composite_sem`.

Usage

```
cov_sem(model)
```

Arguments

`model`	A single character string giving a structural equation model in lavaan model syntax (see `model.syntax`), with every parameter fixed to its population value. A factor loading, a structural path, a variance, or a covariance is fixed by prefixing the numeric value to the variable with the `*` operator, for example `"f1 =~ 1*y1 + 0.8*y2 + 0.8*y3"` for the loadings, `"f2 ~ 0.5*f1"` for a structural path, and `"y1 ~~ 0.5*y1"` for a residual variance. A model with a mean structure (for example a latent growth curve model) also fixes every intercept and latent mean,

for example `"t1 ~ 0*1"` and `"s ~ 0.3*1"`. lavaan syntax embeds the values and knows which variables are latent, so no separate parameter vector or list of latent variables is needed.

Details

The function builds a non-fitted lavaan object from `model` with all parameters held at the population values written into the syntax, and reads back the model implied covariance matrix of the observed variables. Because the object is created with `do.fit = FALSE`, no estimation is performed and the placeholder sample covariance lavaan needs to construct the object is never used; the returned $\Sigma(\theta)$ comes entirely from the fixed parameter values. The observed-variable names are taken from the model syntax and fix the row and column order of the returned matrix.

Value

A list with components:

`sigma_theta` The model implied population covariance matrix of the observed variables, with rows and columns named.

`mu_theta` The model implied population mean vector of the observed variables, named, in the row order of `sigma_theta`. A vector of zeros when the model has no mean structure.

`observed_vars` Character vector of observed variable names in the row/column order of `sigma_theta`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2011). Accuracy in parameter estimation for targeted effects in structural equation modeling: Sample size planning for narrow confidence intervals. *Psychological Methods, 16*(2), 127–148. doi:10.1037/a0021764

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

See Also

`sem`, `model.syntax`, `ss_aipe_sem_path`, `ss_aipe_sem_path_sensitivity`, `ss_aipe_rmsea_sensitivity`, `covmat_from_cfa`.

Examples

```
# Population model with all parameters fixed to their values: two factors,
# three indicators each, and a structural path of 0.5 from the first
# factor to the second.
pop_model <- "
  f1 =~ 1*y1 + 0.8*y2 + 0.8*y3
  f2 =~ 1*y4 + 0.8*y5 + 0.8*y6
  f2 ~ 0.5*f1
  f1 ~~ 1*f1
```

```
  f2 ~~ 0.75*f2
  y1 ~~ 0.5*y1; y2 ~~ 0.5*y2; y3 ~~ 0.5*y3
  y4 ~~ 0.5*y4; y5 ~~ 0.5*y5; y6 ~~ 0.5*y6
"
cov_sem(pop_model)$sigma_theta

# A population model with a mean structure: a linear latent growth curve
# over four waves. The intercepts and latent means are fixed too, and
# mu_theta carries the model implied wave means, which start at 5.0 and
# rise by 0.3 per wave.
pop_lgm <- "
  i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
  s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4
  i ~~ 1*i
  s ~~ 0.2*s
  i ~~ -0.15*s
  t1 ~~ 0.5*t1; t2 ~~ 0.5*t2; t3 ~~ 0.5*t3; t4 ~~ 0.5*t4
  t1 ~ 0*1; t2 ~ 0*1; t3 ~ 0*1; t4 ~ 0*1
  i ~ 5*1
  s ~ 0.3*1
"
cov_sem(pop_lgm)$mu_theta
```

---

cv *Coefficient of Variation (Sample Value or Nearly Unbiased Estimate)*

Description

Computes the sample coefficient of variation $k = s/M$, the sample standard deviation divided by the sample mean, or, optionally, the nearly unbiased estimate of Kelley (2007, Equation 4) under normality. Either supply a precomputed `cv` or supply the raw `mean` and `sd`; with `unbiased = TRUE` the value is multiplied by $(1 + 1/(4N))$. The (biased) sample coefficient of variation (the default, `unbiased = FALSE`) is the form usually reported. To accompany it with a confidence interval, use `ci_cv`.

Usage

```
cv(cv = NULL, mean = NULL, sd = NULL, N = NULL, unbiased = FALSE)
```

Arguments

`cv`	The sample coefficient of variation, $k = s/M$, a single finite number. Optional; if supplied, `mean` and `sd` must not be.
`mean`	Sample mean. A single finite number other than 0. A negative mean gives a negative coefficient; `ci_cv` requires a positive one.
`sd`	Sample standard deviation, using $N - 1$ in the variance denominator. A single finite number that is not negative.

N	Sample size. Required when unbiased = TRUE, where it must exceed 1.
unbiased	Logical. If TRUE, multiplies the sample value by $(1 + 1/(4N))$. The result is nearly, not exactly, unbiased (see Details). Default FALSE.

Details

The sample coefficient of variation $k = s/M$ is the form usually reported in applied work. It is what this function returns by default (unbiased = FALSE). A point estimate is most informative when paired with a confidence interval; ci_cv computes one for the population value κ.

Under normality the sample coefficient is biased. Its expected value is approximately $\kappa[1-1/(4(N-1)) + \kappa^2/N]$, the first and second order terms of Expression 3 of Sokal and Braumann (1980, p. 51), which Kelley (2007) reproduces as his Equation 3. The term $-1/(4(N-1))$ comes from the sample standard deviation and the term κ^2/N from the sample mean in the denominator, so k underestimates κ on average when κ is below about one half and overestimates it when κ is larger. Multiplying k by $(1 + 1/(4N))$, a correction that Sokal and Braumann (1980) and Kelley (2007) credit to Haldane (1955), offsets the first term and not the second. The relative bias that remains is about $\kappa^2/N - 9/(32N^2)$; the second of these terms is derived from the expected value of s under normality and is not taken from those sources. Kelley (2007, Equation 4) calls the result a nearly unbiased estimate and recommends it when a point estimate is reported; the sample value k is the one used to form the confidence interval.

At $\kappa = 0.1$ and $N = 10$ the relative bias is about -2.6 percent for k and -0.2 percent for the corrected value; at $\kappa = 0.3$ and $N = 10$ it is about -1.8 percent and $+0.6$ percent. With 10 cases the two are about equally biased, in opposite directions, at a coefficient near 0.38 (near 0.35 in large samples, and above 0.4 with 5 cases or fewer), and for a larger κ the corrected value is the more biased of the two. These statements describe a mean that stays away from 0: under an unrestricted normal distribution the expectation of $1/M$ does not exist as an ordinary integral.

For confidence intervals on κ, ci_cv inverts the noncentral t distribution of $\sqrt{N}/k$ (Johnson & Welch, 1940; McKay, 1932; Kelley, 2007). McKay (1932) and Vangel (1996) also give approximations based on the chi square distribution, which ci_cv does not use. The first-order variance of k under normality, $\kappa^2(1/2 + \kappa^2)/(N-1)$, is in var_cv, whose page relates it to Expression 5 of Sokal and Braumann (1980, p. 52).

Value

A 1-row data.frame with columns term and value. The term value is "cv" and value is either the sample coefficient (default) or the nearly unbiased estimate (when unbiased = TRUE).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Haldane, J. B. S. (1955). The measurement of variation. *Evolution, 9*(4), 484. doi:10.1111/j.1558-5646.1955.tb01559.x

Johnson, N. L., & Welch, B. L. (1940). Applications of the non-central *t*-distribution. *Biometrika, 31*(3–4), 362–389. doi:10.1093/biomet/31.34.362

Kelley, K. (2007). Sample size planning for the coefficient of variation from the accuracy in parameter estimation approach. *Behavior Research Methods, 39*(4), 755–766. doi:10.3758/BF03192966

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

McKay, A. T. (1932). Distribution of the coefficient of variation and the extended *t* distribution. *Journal of the Royal Statistical Society, 95*(4), 695–698.

Sokal, R. R., & Braumann, C. A. (1980). Significance tests for coefficients of variation and variability profiles. *Systematic Zoology, 29*(1), 50–66. doi:10.1093/sysbio/29.1.50

Vangel, M. G. (1996). Confidence intervals for a normal coefficient of variation. *The American Statistician, 50*(1), 21–26. doi:10.1080/00031305.1996.10473537

See Also

ci_cv, var_cv, ss_aipe_cv

Examples

```
# 1. Point estimate from raw mean and SD.
cv(mean = 100, sd = 15)

# 2. Nearly unbiased estimate at N = 50; the multiplier is close to 1
#    at this sample size.
cv(mean = 100, sd = 15, N = 50, unbiased = TRUE)

# 3. The same at N = 10, where the multiplier is larger.
cv(cv = .15, N = 10, unbiased = TRUE)
```

---

cv_bonferroni_f *Provides the Bonferroni-Adjusted Critical Value for an* F *Test of One of Several Contrasts*

Description

Provides the Bonferroni-Adjusted Critical Value for an *F* Test of One of Several Contrasts

Usage

```
cv_bonferroni_f(
  alpha_level = 0.05,
  df_denominator,
  n_comparisons,
  df_numerator = 1,
  verbose = TRUE
)
```

Arguments

alpha_level	The family-wise Type I error rate (i.e., the rate for the set of n_comparisons tests taken together). Default 0.05, the level Maxwell, Delaney, and Kelley (2027) tabulate.
df_denominator	The denominator (error) degrees of freedom (a positive number). In a one-way design with N observations and a groups this is $N - a$.
n_comparisons	The number of comparisons in the family, C (a positive integer).
df_numerator	The numerator degrees of freedom. Default 1, because a contrast carries a single degree of freedom, which is the case the Appendix table covers.
verbose	Provides extra information about areas under the curve.

Details

The Bonferroni adjustment tests each of C contrasts at α/C rather than α, which holds the family-wise error rate at or below α whatever the contrasts are and however they are correlated. The critical value is therefore an ordinary upper-tail F quantile read at the smaller per-comparison rate,

$$F_{\alpha/C;\,\mathrm{df}_{\mathrm{num}},\,\mathrm{df}_{\mathrm{den}}},$$

which is what cv_f would return if handed alpha / C. This function exists because the adjustment is worth naming: the whole of it is the division, and seeing α/C reported back in area_greater is the point. Maxwell, Delaney, and Kelley (2027) tabulate these values for one numerator degree of freedom and a family-wise alpha of .05 in their Appendix Table A.3.

The default df_numerator = 1 covers the case the table addresses and the one that arises in practice, since a contrast among means is a single-degree-of-freedom question. Supply a larger value to Bonferroni adjust a family of multiple-degree-of-freedom model comparisons.

The procedure is often called Dunn's, after Dunn (1961), who first applied the Bonferroni inequality to multiple contrasts. It is not the rank-sum procedure of dunn_test, which the same author published three years later.

When something else is better. Bonferroni makes no use of the structure of the family, so a procedure built for a particular structure beats it there: cv_tukey_hsd is more powerful for all pairwise comparisons, and cv_dunnett is more powerful for comparing several treatments to one control. Bonferroni's advantage is generality; it applies to any set of contrasts chosen in advance, and it can beat Tukey's method when only a few of the pairwise comparisons were planned. Because it is conservative, a step-down variant such as Holm's is uniformly more powerful while controlling the same rate, and is available through the method argument of contrast_adjusted and p.adjust.

Value

Returns the critical value in a output style (a data.frame following the format used by cv_t). When verbose = TRUE the area_greater column reports the per-comparison error rate α/C that the adjustment spends on each test.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association, 56*(293), 52–64. doi:10.2307/2282330

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 5 on the multiple-comparisons problem; Appendix Table A.3 reports these critical values.)

See Also

cv_f, cv_tukey_hsd, cv_dunnett, cv_scheffe, contrast_adjusted

Other critical values: cv_bryant_paulson(), cv_chisq(), cv_dunnett(), cv_f(), cv_scheffe(), cv_smm(), cv_t(), cv_tukey_hsd(), cv_z()

Examples

```
# Five planned contrasts with 20 error degrees of freedom, holding the
# family-wise error rate at .05. The area_greater column shows the .01
# per-comparison rate the adjustment spends on each test.
cv_bonferroni_f(alpha_level = .05, df_denominator = 20, n_comparisons = 5)

# It is the F critical value read at alpha / C.
cv_bonferroni_f(alpha_level = .05, df_denominator = 20, n_comparisons = 5,
                verbose = FALSE)$value
cv_f(alpha_level = .05 / 5, df_numerator = 1, df_denominator = 20,
     verbose = FALSE)$value[2]

# With one comparison there is nothing to adjust.
cv_bonferroni_f(alpha_level = .05, df_denominator = 20, n_comparisons = 1,
                verbose = FALSE)$value
```

---

cv_bryant_paulson *Provides the Critical Value for the Bryant–Paulson ANCOVA Multiple-Comparison Procedure*

Description

Computes the critical value of the Bryant–Paulson generalized studentized range, the reference distribution for multiple comparisons of adjusted means in an analysis of covariance with random covariates. The single-step value is the multiplier for simultaneous confidence intervals on pairwise differences of adjusted means; the "duncan" option instead returns the significant range of the stepwise Duncan multiple-range procedure.

Usage

```
cv_bryant_paulson(
  alpha_level,
  df,
  n_groups,
  n_covariates = 1,
  procedure = c("tukey", "duncan"),
  verbose = TRUE
)
```

Arguments

alpha_level	Type I error rate (i.e., the false positive rate). As with cv_tukey_hsd, the full alpha_level applies to the upper tail of the (non-negative) generalized studentized range distribution; it is not split between two tails (see Details).
df	The ANCOVA error degrees of freedom (a positive number; in a one-way ANCOVA, df = N - n_groups - n_covariates).
n_groups	The number of groups whose adjusted means are being compared (an integer of at least 2).
n_covariates	The number of *random* covariates in the ANCOVA, the parameter p of the Bryant–Paulson distribution (a non-negative integer). With n_covariates = 0 the critical value reduces to the ordinary studentized range and the result equals $\sqrt{2}$ times cv_tukey_hsd (see Details).
procedure	One of "tukey" (default) or "duncan". "tukey" returns the single-step *simultaneous* critical value $q_{\alpha;p,k,\nu}$ used for familywise (Tukey–Kramer-type) confidence intervals; "duncan" returns the stepwise Duncan multiple-range significant range $r_{\alpha;p,k,\nu}$ tabled by Bryant and Bruvold (1980, Table 2) (see Details).
verbose	Provides extra information (the tail areas) about the critical value.

Details

The Bryant–Paulson procedure is the analysis-of-covariance generalization of Tukey's method (cv_tukey_hsd) for comparing adjusted means when the covariate is *random*. Because the covariate adjustment must be estimated, the studentized range of adjusted means is stochastically larger than the ordinary studentized range, so the Bryant–Paulson critical value exceeds Tukey's; using the latter would give intervals that are too narrow and a familywise error rate above alpha_level. The single-step (procedure = "tukey") value is the multiplier for a family of simultaneous confidence intervals on the pairwise differences of adjusted means that jointly hold at level $1 - \alpha$. Maxwell, Delaney, and Kelley (2027, Chapter 9) develop multiple comparisons of adjusted means in the analysis of covariance, the setting this critical value serves.

The reference distribution is the Bryant–Paulson generalized studentized range, implemented in qbryant_paulson. Its quantiles are not a standard base-R distribution and are not the multivariate *t* quantiles that cv_dunnett and cv_smm obtain from **mvtnorm**; they are computed directly by qbryant_paulson, so this function depends on neither base-R nor **mvtnorm** multiple-mean machinery.

Scale. The returned value is on the studentized-range scale, $q_{\alpha;p,k,\nu}$, the scale of Bryant and Paulson's (1976) and Bryant and Bruvold's (1980) tables and of Eq. (2.4) of the latter. A pair

of adjusted means is declared different when $|\hat{\theta}_i - \hat{\theta}_j| > q_{\alpha;p,k,\nu}\,\hat{\sigma}_{y|x}\sqrt{1/n}$. This differs from `cv_tukey_hsd`, which divides its value by $\sqrt{2}$ to report on the pairwise mean-difference scale; divide the value here by $\sqrt{2}$ to obtain that scale. With `n_covariates = 0`, `cv_bryant_paulson` returns exactly $\sqrt{2}\times$ `cv_tukey_hsd`.

Duncan multiple-range. For `procedure = "duncan"` the value is the "significant range" of Duncan's stepwise test as extended to ANCOVA by Bryant and Bruvold (1980, Section 4). With variable protection levels $\alpha_k = 1 - (1-\alpha)^{k-1}$,

$$r_{\alpha;p,2,\nu} = q_{\alpha;p,2,\nu}, \qquad r_{\alpha;p,k,\nu} = \max\{\, r_{\alpha;p,k-1,\nu},\; q_{\alpha_k;p,k,\nu} \,\}, \quad k > 2.$$

These are the values in Bryant and Bruvold's Table 2, reproduced by this function to the tabled two-decimal precision (see the package tests).

Value

Returns the critical value in a output style (a `data.frame` with class `dmar_tbl` and one row per critical value, following the format used by `cv_tukey_hsd` and `cv_t`). The `value` is on the *studentized-range scale* (the scale on which Bryant and Paulson tabulate their critical values and on which `ci_c_ancova_bp` uses them). When `verbose = TRUE` and `procedure = "tukey"`, the upper- and lower-tail areas of the Bryant–Paulson distribution at the critical value are also returned; for `procedure = "duncan"` the tail areas are `NA` because the significant range is a stepwise quantity rather than a single quantile.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bryant, J. L., & Paulson, A. S. (1976). An extension of Tukey's method of multiple comparisons to experimental designs with random concomitant variables. *Biometrika, 63*, 631–638.

Bryant, J. L., & Bruvold, N. T. (1980). Multiple comparison procedures in the analysis of covariance. *Journal of the American Statistical Association, 75*(372), 874–880. doi:10.2307/2287175

Duncan, D. B. (1955). Multiple range and multiple F tests. *Biometrics, 11*, 1–42.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9, where multiple comparisons of adjusted means in the analysis of covariance are developed; Appendix Table A.8 reports these critical values.)

See Also

`cv_tukey_hsd`, `cv_scheffe`, `qbryant_paulson`, `ci_c_ancova_bp`

Other critical values: `cv_bonferroni_f()`, `cv_chisq()`, `cv_dunnett()`, `cv_f()`, `cv_scheffe()`, `cv_smm()`, `cv_t()`, `cv_tukey_hsd()`, `cv_z()`

Examples

```
# Multiple comparisons of adjusted means in ANCOVA (the setting of Maxwell,
# Delaney, and Kelley, 2027, Chapter 9), using the worked example of Bryant
# and Bruvold (1980): 6 panels, 1 random covariate, 14 error df,
# alpha_level = .05. The single-step value is the multiplier for simultaneous
# confidence intervals on the pairwise differences of adjusted means. With a
# covariate present there is no closed form for it: the Bryant-Paulson
# distribution function is integrated numerically and then inverted. The
# value is 4.83, the entry in Table 1 of Bryant and Paulson (1976) and the
# multiplier behind the simultaneous intervals of the 1980 worked example.
cv_bryant_paulson(alpha_level = .05, df = 14, n_groups = 6, n_covariates = 1)

# With no covariates the reference distribution is the ordinary studentized
# range, which base R supplies directly. The critical value is then sqrt(2)
# times the Tukey HSD critical value.
cv_bryant_paulson(alpha_level = .05, df = 14, n_groups = 6, n_covariates = 0)$value
sqrt(2) * cv_tukey_hsd(alpha_level = .05, df = 14, n_groups = 6)$value

# The stepwise Duncan multiple-range significant range comes from
# procedure = "duncan". With n_covariates = 0 it reduces to Duncan's (1955)
# own significant studentized range, 3.37 for a stretch of 6 groups on 14
# error degrees of freedom.
cv_bryant_paulson(alpha_level = .05, df = 14, n_groups = 6, n_covariates = 0,
                  procedure = "duncan")
# One random covariate raises that range to 3.50, the entry in Table 2 of
# Bryant and Bruvold (1980). Being stepwise, the value is a running maximum
# of Bryant-Paulson quantiles, one inverted for every stretch from 2 to 6
# groups at that stretch's own protection level; the package tests check
# the full sequence against the paper's Section 4 example.
cv_bryant_paulson(alpha_level = .05, df = 14, n_groups = 6, n_covariates = 1,
                  procedure = "duncan")
```

---

cv_chisq *Provides the Critical Value(s) for a Chi Square Distribution*

Description

Provides the Critical Value(s) for a Chi Square Distribution

Usage

```
cv_chisq(
  alpha_level,
  df,
  alternative = "greater",
  alpha_lower,
  alpha_upper,
  ncp = 0,
```

```
  verbose = TRUE
)
```

Arguments

alpha_level	Type I error rate (i.e., the false positive rate).
df	The number of degrees of freedom (a positive number).
alternative	The type of alternative hypothesis of interest. The default, "greater", puts the whole of alpha_level in the upper tail, which is how the chi square distribution is used to test a model or an association (see Details).
alpha_lower	The error rate in the lower tail of the distribution.
alpha_upper	The error rate in the upper tail of the distribution.
ncp	The noncentral parameter (if zero, the default, it is the central chi square distribution).
verbose	Provides extra information about areas under the curve.

Details

Like the F distribution and unlike t and z, the chi square distribution is not symmetric and takes only non-negative values. Its common uses are one-sided in the upper tail: a test of association in a contingency table, a likelihood ratio test, and a test of model fit all reject for large values, because a poorly fitting model produces a large discrepancy, never a small one. That is why alternative defaults to "greater" here whereas it defaults to "not_equal" in cv_t. Maxwell, Delaney, and Kelley (2027) tabulate these upper-tail values in their Appendix Table A.9.

Both tails remain available for the situations that need them, most commonly an interval for a variance, which uses an upper and a lower chi square quantile. Set alternative = "not_equal", or give alpha_lower and alpha_upper directly. When a tail is given zero area its critical value is the boundary of the support, so lower_cv is 0 under the default.

A noncentral parameter can be supplied, which is what a power analysis for a test of model fit needs, though it would not be used for a standard null hypothesis significance test. See ci_nc_chisq for confidence limits on the noncentral parameter itself.

Value

Returns the critical value(s), based on the input specifications, in a output style (a data.frame with a row for the lower and the upper critical value, following the format used by cv_t).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (Appendix Table A.9 reports these critical values.)

See Also

cv_t, cv_f, ci_nc_chisq

Other critical values: cv_bonferroni_f(), cv_bryant_paulson(), cv_dunnett(), cv_f(), cv_scheffe(), cv_smm(), cv_t(), cv_tukey_hsd(), cv_z()

Examples

```
# The critical value for a test on 3 degrees of freedom at the .05 level.
cv_chisq(alpha_level = .05, df = 3)

# Simple output.
cv_chisq(alpha_level = .05, df = 3, verbose = FALSE)

# Both tails, as an interval for a variance would need.
cv_chisq(alpha_level = .05, df = 10, alternative = "not_equal")
```

---

cv_dunnett *Provides the Critical Value for Dunnett's Many-to-One Comparisons Procedure*

Description

Provides the Critical Value for Dunnett's Many-to-One Comparisons Procedure

Usage

```
cv_dunnett(
  alpha_level,
  df,
  n_comparisons,
  alternative = "not_equal",
  verbose = TRUE
)
```

Arguments

alpha_level	Type I error rate (i.e., the family-wise false-positive rate).
df	Error degrees of freedom (typically $N-k$, where k is the total number of groups including the control). May be Inf, in which case the known-variance (normal) limit is used.
n_comparisons	The number of treatment-versus-control comparisons (i.e., $k-1$ where k is the total number of groups).
alternative	The form of the alternative hypothesis: one of "not_equal" (two-sided; default), "greater" (treatments greater than control), or "less" (treatments less than control).
verbose	Provides extra information about areas under the curve.

Details

Dunnett's procedure controls the family-wise error rate for the special case of comparing each of $k-1$ treatments to a single control (the many-to-one comparisons setting), using a multivariate *t* reference with constant pairwise correlation $1/2$. That correlation is exact for a balanced design: each comparison is $(\bar{Y}_i - \bar{Y}_0)$, all sharing the one control mean $\bar{Y}_0$ of variance σ^2/n, while each comparison has variance $2\sigma^2/n$, so any two comparisons correlate $(\sigma^2/n)/(2\sigma^2/n) = 1/2$. Maxwell, Delaney, and Kelley (2027, Chapter 5) develop the many-to-one comparisons problem and tabulate these critical values.

How it is computed. The critical value is a quantile of a $(k-1)$-dimensional multivariate *t* distribution whose correlation matrix has a unit diagonal and off-diagonal entries of 1/2. Because that correlation is a single common value, the comparisons have the one-factor representation $Z_i = \sqrt{1/2}\,W + \sqrt{1/2}\,U_i$ with a shared factor W and independent U_i, all standard normal. Conditioning on W and on the common scale estimate $S = \sqrt{\chi^2_{df}/df}$ makes the comparisons independent, so the $(k-1)$-dimensional integral collapses to two nested one-dimensional integrals:

$$P\Big(\max_i T_i \le d\Big) = \int_0^\infty \int_{-\infty}^\infty \Big[\Phi\big((d\,s - \sqrt{1/2}\,w)/\sqrt{1/2}\big)\Big]^{k-1} \phi(w)\,dw\; f_S(s)\,ds$$

for the one-sided value (the two-sided value replaces the bracket with the probability that $|T_i| \le d$). This package evaluates those integrals with `integrate` and inverts with `uniroot`, so the returned value is deterministic and accurate to the solver tolerance, with no Monte Carlo simulation and no random seed. When $k - 1 = 1$ the value is the ordinary one- or two-sided *t* critical value.

Value

Returns the critical value in a output style (a `data.frame` following the format used by `cv_t`).

Note

The constant-correlation assumption holds for balanced designs (equal n per group); for severely unbalanced designs use `glht` instead.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Dunnett, C. W. (1955). A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association, 50*(272), 1096–1121.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 5 on the multiple-comparisons problem; Dunnett's many-to-one comparisons procedure is developed there, and Appendix Tables A.6 and A.7 report its critical values for two- and one-tailed tests.)

See Also

cv_t, cv_tukey_hsd, cv_smm, cv_scheffe, contrast_test

Other critical values: cv_bonferroni_f(), cv_bryant_paulson(), cv_chisq(), cv_f(), cv_scheffe(), cv_smm(), cv_t(), cv_tukey_hsd(), cv_z()

Examples

```
# Following the many-to-one comparisons setting of Maxwell, Delaney, and
# Kelley (2027, Chapter 5): three treatment groups each compared to a single
# control (k = 4 groups, so 3 comparisons) with 36 error degrees of freedom.
# The two-sided critical value controls the family-wise error rate at .05.
cv_dunnett(alpha_level = .05, df = 36, n_comparisons = 3)

# When the treatments are expected only to exceed the control, a one-sided
# ("treatments greater than control") critical value is smaller.
cv_dunnett(alpha_level = .05, df = 36, n_comparisons = 3, alternative = "greater")
```

---

cv_f *Provides the Critical Value(s) for an* F *Distribution*

Description

Provides the Critical Value(s) for an *F* Distribution

Usage

```
cv_f(
  alpha_level,
  df_numerator,
  df_denominator,
  alternative = "greater",
  alpha_lower,
  alpha_upper,
  ncp = 0,
  verbose = TRUE
)
```

Arguments

alpha_level	Type I error rate (i.e., the false positive rate).
df_numerator	The numerator degrees of freedom (a positive number). In a model comparison this is the difference in the number of parameters between the two models.
df_denominator	The denominator (error) degrees of freedom (a positive number).
alternative	The type of alternative hypothesis of interest. The default, "greater", puts the whole of alpha_level in the upper tail, which is how the *F* distribution is used to test a model comparison (see Details).

alpha_lower	The error rate in the lower tail of the distribution.
alpha_upper	The error rate in the upper tail of the distribution.
ncp	The noncentral parameter (if zero, the default, it is the central F distribution).
verbose	Provides extra information about areas under the curve.

Details

Unlike the t and z distributions, the F distribution is not symmetric and takes only non-negative values, and the usual test of a model comparison is one-sided: a restricted model fits worse than a full model, so evidence against the restriction shows up as a large F, never a small one. That is why `alternative` defaults to `"greater"` here whereas it defaults to `"not_equal"` in `cv_t`. Maxwell, Delaney, and Kelley (2027) tabulate these upper-tail values in their Appendix Table A.2.

Both tails remain available for the situations that need them, such as an interval for a ratio of variances, either by setting `alternative = "not_equal"` or by giving `alpha_lower` and `alpha_upper` directly. When a tail is given zero area its critical value is the boundary of the support, so `lower_cv` is 0 under the default.

A noncentral parameter can be supplied, which is what a power analysis needs, though it would not be used for a standard null hypothesis significance test. See `ci_nc_F` for confidence limits on the noncentral parameter itself.

Value

Returns the critical value(s), based on the input specifications, in a output style (a `data.frame` with a row for the lower and the upper critical value, following the format used by `cv_t`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3, where the F test of a model comparison is developed; Appendix Table A.2 reports these critical values.)

See Also

`cv_t`, `cv_chisq`, `cv_bonferroni_f`, `cv_scheffe`, `ci_nc_F`

Other critical values: `cv_bonferroni_f()`, `cv_bryant_paulson()`, `cv_chisq()`, `cv_dunnett()`, `cv_scheffe()`, `cv_smm()`, `cv_t()`, `cv_tukey_hsd()`, `cv_z()`

Examples

```
# The critical value for a model comparison with 3 numerator and 20
# denominator degrees of freedom, at the .05 level.
cv_f(alpha_level = .05, df_numerator = 3, df_denominator = 20)

# An omnibus test of four groups with 24 participants: a - 1 = 3 and
```

```
# N - a = 20 degrees of freedom; simple output.
cv_f(alpha_level = .05, df_numerator = 3, df_denominator = 20, verbose = FALSE)

# Both tails, as an interval for a ratio of variances would need.
cv_f(alpha_level = .05, df_numerator = 3, df_denominator = 20,
     alternative = "not_equal")
```

`cv_scheffe` *Provides the Critical Value for the Scheffé Procedure*

Description

Provides the Critical Value for the Scheffé Procedure

Usage

```
cv_scheffe(alpha_level, df_numerator, df_denominator, verbose = TRUE)
```

Arguments

`alpha_level`	Type I error rate (i.e., the family-wise false-positive rate).
`df_numerator`	The numerator degrees of freedom (typically the number of groups minus 1, $k - 1$, in a one-way ANOVA).
`df_denominator`	The denominator (error) degrees of freedom (typically $N - k$ in a one-way ANOVA).
`verbose`	Provides extra information about areas under the curve.

Details

The Scheffé critical value protects the family-wise error rate for the simultaneous test of *any* contrast (or family of contrasts) in a fixed-effects ANOVA, including data-driven contrasts selected after looking at the data. It is therefore the most conservative of the standard procedures.

The critical value, on the scale of a *t*-statistic, is

$$t_{\text{crit}}^{\text{Scheffe}} = \sqrt{(k-1)\, F_{1-\alpha,\, k-1,\, df_{\text{denominator}}}},$$

so that a contrast $\hat{\psi}$ with standard error $SE_{\hat{\psi}}$ is declared significant when $|\hat{\psi}/SE_{\hat{\psi}}| > t_{\text{crit}}^{\text{Scheffe}}$. The corresponding simultaneous confidence interval is $\hat{\psi} \pm t_{\text{crit}}^{\text{Scheffe}} \cdot SE_{\hat{\psi}}$.

Like the Studentized range distribution underlying Tukey HSD, the Scheffé reference is one-sided (the underlying *F* statistic is non-negative), so `alpha_level` is *not* split between two tails.

The Scheffé critical value is a function of a univariate *F* quantile, which base R supplies through `qf`, so unlike `cv_dunnett` and `cv_smm` this function needs no multivariate distribution machinery. Scheffé's procedure earns its simultaneous protection over the infinite family of all possible contrasts by projecting onto the overall *F* test rather than by integrating a multivariate *t* density; that is why a single univariate quantile suffices and the **mvtnorm** package is not required here. Maxwell, Delaney, and Kelley (2027, Chapter 5) develop the Scheffé method as the procedure for arbitrary contrasts within the multiple-comparisons problem.

Value

Returns the critical value in a output style (a `data.frame` with one row per critical value, following the format used by `cv_t` and `cv_tukey_hsd`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Scheffe, H. (1953). A method for judging all contrasts in the analysis of variance. *Biometrika, 40*, 87–104.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 5 on the multiple-comparisons problem, where the Scheffé method for arbitrary contrasts is developed.)

See Also

`cv_t`, `cv_tukey_hsd`, `contrast_test`

Other critical values: `cv_bonferroni_f()`, `cv_bryant_paulson()`, `cv_chisq()`, `cv_dunnett()`, `cv_f()`, `cv_smm()`, `cv_t()`, `cv_tukey_hsd()`, `cv_z()`

Examples

```
# Following the arbitrary-contrasts setting of Maxwell, Delaney, and Kelley
# (2027, Chapter 5): a one-way ANOVA with k = 4 groups and 36 error degrees
# of freedom (e.g., n = 10 per group). The Scheffé critical value protects
# the family-wise error rate over any contrast, including contrasts chosen
# after looking at the data.
cv_scheffe(alpha_level = .05, df_numerator = 3, df_denominator = 36)

# The price of that protection is a larger multiplier than the unadjusted
# t critical value at the same error degrees of freedom:
cv_t(alpha_level = .05, df = 36)
```

---

cv_smm — *Provides the Critical Value of the Studentized Maximum Modulus Distribution*

Description

Provides the Critical Value of the Studentized Maximum Modulus Distribution

Usage

```
cv_smm(alpha_level, df, n_comparisons, verbose = TRUE)
```

Arguments

`alpha_level`	Type I error rate (i.e., the family-wise false-positive rate).
`df`	Error degrees of freedom (typically $N - k$ in a one-way ANOVA). May be `Inf`, in which case the known-variance (normal) limit is used.
`n_comparisons`	The number of simultaneous comparisons (m) for which a family-wise critical value is desired.
`verbose`	Provides extra information about areas under the curve.

Details

The Studentized maximum modulus (SMM) distribution is the distribution of

$$\max_{i=1,\ldots,m} |Z_i| \,/\, S,$$

where $Z_1, \ldots, Z_m$ are independent standard normal variates and S is independent of the Z_i and equal to $\sqrt{\chi^2_{df}/df}$.

What the SMM is used for. The SMM critical value is the multiplier that turns a set of m individual estimates into a family of simultaneous confidence intervals, or equivalently a family of tests, that jointly control the family-wise error rate at level `alpha_level`. Because the modulus is the largest of m standardized statistics in absolute value, requiring that maximum to clear the critical value bounds the chance of any one of the m intervals failing to cover (or any one of the m tests producing a false positive). Maxwell, Delaney, and Kelley (2027, Chapter 5) describe this use in the context of the multiple-comparisons problem: when a researcher forms several means or contrasts and wants the stated coverage to hold across the whole set rather than one interval at a time, the SMM supplies the simultaneous critical value. A pair-by-pair construction, applied to m comparisons, is $\hat{\psi}_i \pm c_{\alpha;m,df}\, SE_{\hat{\psi}_i}$ with $c_{\alpha;m,df}$ the SMM critical value returned here. When $m = 1$ it reduces to the ordinary two-sided *t* critical value. `ci_dunnett_t3` uses it, at each pair's Welch–Satterthwaite degrees of freedom, for all pairwise comparisons of means without assuming homogeneity of variance (Dunnett's T3).

How it is computed. The m statistics are independent given the common scale estimate S, so the joint distribution factorizes after conditioning on S and the m-dimensional integral that defines the maximum modulus collapses to a single one-dimensional integral,

$$P\Big(\max_i |Z_i|/S \le c\Big) = \int_0^\infty \big[\, 2\,\Phi(c\,s) - 1 \,\big]^m \, f_S(s)\, ds,$$

where f_S is the density of $S = \sqrt{\chi^2_{df}/df}$. This package evaluates that integral with `integrate` and inverts it with `uniroot`, so the returned value is deterministic and accurate to the solver tolerance (there is no Monte Carlo simulation and no random seed). When `df = Inf` the scale is degenerate at 1 and the closed form $c = \Phi^{-1}\big((1 + (1-\alpha)^{1/m})/2\big)$ is returned; when $m = 1$ the value is exactly $t_{1-\alpha/2,\,df}$.

When `df` is small and the critical value large (`df` below 1 at conventional levels, for example below about .8 at an `alpha_level` of .05 and below about .6 at .10, or `df` of about 2 or less with an `alpha_level` of .02 or less), the integrand rises from 0 to 1 in a narrow band near $s = 0$ that an

adaptive rule can miss. Every root is therefore checked against the upper tail computed another way, by conditioning on $W = \max_i |Z_i|$ instead of on S,

$$P\Big(\max_i |Z_i|/S > c\Big) = \int_0^\infty F_S(w/c)\, g_W(w)\, dw,$$

where F_S is the distribution function of S and $g_W(w) = 2m\,[2\Phi(w) - 1]^{m-1}\phi(w)$ the density of W, and the root is found on that tail instead when the two disagree.

Value

Returns the critical value as a `data.frame`, following the format used by `cv_t`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Stoline, M. R., & Ury, H. K. (1979). Tables of the Studentized maximum modulus distribution and an application to multiple comparisons among means. *Technometrics, 21*(1), 87–93. doi:10.1080/00401706.1979.10489726

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 5 on the multiple-comparisons problem, where simultaneous confidence intervals for several means or contrasts are developed; Appendix Table A.5 reports SMM critical values.)

See Also

`ci_dunnett_t3`, which uses this critical value for all pairwise comparisons without assuming homogeneity of variance; `cv_t`, `cv_tukey_hsd`, `cv_dunnett`, `cv_scheffe`

Other critical values: `cv_bonferroni_f()`, `cv_bryant_paulson()`, `cv_chisq()`, `cv_dunnett()`, `cv_f()`, `cv_scheffe()`, `cv_t()`, `cv_tukey_hsd()`, `cv_z()`

Examples

```
# Following the simultaneous-intervals setting of Maxwell, Delaney, and
# Kelley (2027, Chapter 5): a researcher forms m = 5 comparisons and wants
# all five confidence intervals to hold simultaneously at the .05 level,
# with 36 error degrees of freedom.
cv_smm(alpha_level = .05, df = 36, n_comparisons = 5)

# When m = 1, the SMM critical value reduces to the two-sided
# t critical value:
cv_smm(alpha_level = .05, df = 36, n_comparisons = 1)$value
cv_t(alpha_level = .05, df = 36)$value[2]   # upper_cv from cv_t
```

cv_t *Provides the Critical Value(s) for a* t*-distribution*

Description

Provides the Critical Value(s) for a *t*-distribution

Usage

```
cv_t(
  alpha_level,
  df,
  alternative = "not_equal",
  alpha_lower,
  alpha_upper,
  ncp = 0,
  verbose = TRUE
)
```

Arguments

alpha_level	Type I error rate (i.e., the false positive rate).
df	The number of degrees of freedom (a positive number)
alternative	The type of alternative hypothesis of interest.
alpha_lower	The error rate on the lower (negative) side of the distribution.
alpha_upper	The error rate on the upper (positive) side of the distribution.
ncp	The noncentral parameter (if zero, the default, it is the central *t*-distribution).
verbose	Provides extra information about areas under the curve.

Details

Though a noncentral parameter can be included, that would not be done for a standard null hypothesis significance test.

With a nonzero ncp the critical values are quantiles of the noncentral *t* distribution, computed exactly. R's qt() and pt() with ncp switch to a normal approximation above a noncentrality of 37.62 (and above 33 with more than 2,000 degrees of freedom), where they can be far off: the .975 quantile with ncp = 50 and df = 3 is 186.49, and qt() gives 429.18. There the quantiles and the tail areas are computed by integrating over the sampling distribution of the standard deviation; inside that range they are R's. With fewer than one degree of freedom (df may be any positive number) pt() is not accurate inside that range either. At the exact quantiles of probability .0005 to .9995, with noncentralities up to 37.5 in absolute value, the tail probability it gave was off by as much as .14 with df = 0.05, .07 with 0.1, .01 with 0.2, .0013 with 0.3, 3.8×10^{-5} with 0.5, and 2×10^{-11} with 0.8, so with df < 1 every noncentral quantile and tail area is integrated. The critical values then differ from qt()'s by orders of magnitude with df of 0.3 or less (and by as much as 15 percent with 0.5): with df = 0.1, ncp = 0.5, and alpha_level = .05 they are -1.0428×10^{10} and

5.7095×10^{13}, where `qt()` gives -33554432 and 305284064. With so few degrees of freedom the tails are heavy enough that a critical value can lie beyond the largest number R represents (about 1.8×10^{308}); it is then returned as `-Inf` or `Inf`, as `qt()` returns it with `ncp = 0` (with `df = 0.001` one or both of the critical values are infinite at every `alpha_level` from .000001 to .4 and noncentrality from -100 to 100 tried).

Value

Returns the critical value(s), based on the input specifications, in a output style.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

Other critical values: `cv_bonferroni_f()`, `cv_bryant_paulson()`, `cv_chisq()`, `cv_dunnett()`, `cv_f()`, `cv_scheffe()`, `cv_smm()`, `cv_tukey_hsd()`, `cv_z()`

Examples

```
# A basic call for finding critical values with equal area in the two tails.
cv_t(alpha_level = .05, df = 13)

# A basic call for a single-sided confidence interval (for "a greater than" alternative hypothesis)
cv_t(alpha_level = .05, df = 13, alternative = "greater")

# A single-sided confidence interval (for "a greater than" alternative hypothesis); simple output.
cv_t(alpha_lower = 0, alpha_upper = .05, df = 13, verbose = FALSE)

# For a nonsymmetric 95% confidence interval.
cv_t(alpha_lower = .01, alpha_upper = .04, df = 13)
```

`cv_tukey_hsd` *Provides the Critical Value for the Tukey Honestly Significant Difference (HSD) Test*

Description

Provides the Critical Value for the Tukey Honestly Significant Difference (HSD) Test

Usage

```
cv_tukey_hsd(alpha_level, df, n_groups, verbose = TRUE)
```

Arguments

alpha_level	Type I error rate (i.e., the false positive rate). For the Tukey HSD test, the full alpha_level applies to the upper tail of the Studentized range distribution (see Details).
df	The error degrees of freedom from the ANOVA (a positive number; typically N - n_groups).
n_groups	The number of groups whose means are being compared (an integer of at least 2).
verbose	Provides extra information about areas under the curve.

Details

The Tukey HSD test compares all pairs of group means using the Studentized range distribution. Because the Studentized range is the absolute difference between the largest and smallest sample means (scaled by a standard error), it is non-negative and the associated distribution has support on $[0, \infty)$. As a consequence, the Type I error rate alpha_level is *not* split between two tails in the way it is for the (symmetric) t- and z-distributions in cv_t and cv_z; rather, the full alpha_level applies to the upper tail.

The reported critical value is on the scale used for pairwise comparisons of group means, i.e., $q_{1-\alpha,k,df}/\sqrt{2}$, where k is the number of groups. A pair of means is declared significantly different when the absolute standardized difference between them exceeds this critical value.

The Tukey HSD critical value is a quantile of the Studentized range distribution, so unlike cv_dunnett and cv_smm this function needs no multivariate distribution machinery and does not require the **mvtnorm** package. Maxwell, Delaney, and Kelley (2027, Chapter 5) develop the Tukey method as the procedure for all-pairwise comparisons within the multiple-comparisons problem.

How the quantile is computed. Over most of its range the critical value is base R's qtukey, which R documents as accurate to the fourth decimal place (it inverts ptukey by a secant search), and the areas are ptukey, which evaluates its integral with a 16-point Legendre formula and for which R states no accuracy. Compared with an independent computation, the two fall short in five places. (1) At small error df, increasingly so as the number of groups and the confidence level grow: with 20 groups and 2 error df, qtukey(.99, 20, 2) is 44.526 where the exact quantile is 37.943, so the critical value would be 31.485 rather than 26.830; below 2 df both functions return NaN. (2) At large error df: qtukey(.999, 3, 18000) is 1.2×10^{-5} too large, and beyond 25000 df both return the infinite-df distribution. (3) Far in either tail: qtukey(.9999, 10, 11) is 12.008462 where the exact quantile is 12.008580, so cv_tukey_hsd(1e-4, df = 11, n_groups = 10) would be 8.491265 rather than 8.491348; at .99999 qtukey can be off by several units (with 2000 groups and 1000 error df it is 12.930 where the exact quantile is 9.872); in the lower tail it was more than 10^{-5} off at .0001 (with 75 groups) and at .00001 can be far off (qtukey(.00001, 3, 150) is 0.000092 where the exact quantile is 0.006023) or NaN. (4) With more than 50 groups ptukey is off by more than 10^{-6} below its .75 point (by up to 8.3×10^{-6} with 2000 groups, and with 1500 or more groups it can return 0 where the probability is .001), and with thousands of groups at very large error df it is off just above that point as well (ptukey(7.6250574, 5000, Inf) is .7500010 where the exact probability is .7500000). (5) Anywhere, the secant search can stop short of the root of ptukey: qtukey(.975, 10, 12) is 6.003766 where the exact quantile is 6.003807, and qtukey(.9975, 5, 23) is 6.018665 where it is 6.018546.

In these places DMAR computes the distribution function exactly, as the probability integral of the range, $P(W \leq w) = k \int \phi(z)[\Phi(z) - \Phi(z - w)]^{k-1} dz$ for the range W of k standard normal variates (k the number of groups), integrated against the chi square distribution of the error variance by adaptive quadrature, and takes the critical value as the root of that function. It does so (a) below an error df that grows with the number of groups: fewer than 9 error df for up to 6 groups, fewer than 11 for up to 10, 15 for up to 20, 21 for up to 50, 31 for up to 100, 41 for up to 250, 47 for up to 400, 57 for up to 700, 71 for up to 1000, 81 for up to 1500, 91 for up to 2000, 101 for up to 3000, and 111 for more; (b) above 10000 and up to 10^8 error df; (c) for every quantile at a probability above .999 (an `alpha_level` below .001) or below .001, at any error df; (d) with more than 50 groups, for every probability below .75 and every quantile below the .75 point (below the .80 point with more than 1000 groups), at any error df; and (e) wherever `qtukey` fails, or its value is more than 2×10^{-6} from the root of `ptukey` (the residual of `ptukey` there divided by its slope). These rules were set by comparison with two independent computations of the studentized range, as the cells in which `qtukey` was more than 10^{-5} from the exact quantile or `ptukey` more than 10^{-6} from the exact probability, with a margin of at least one measured step. R's values were compared with 2 to 10000 groups at error df up to 20000, with 2 to 20000 groups at infinite df (which R uses for every df above 25000), and with 3 to 10000 groups at $10^8 + 1$, 10^9, and 10^{10} df, at probabilities from .00001 to .99999 (52,892 cells for the second computation alone). Elsewhere the values are those of `qtukey` and `ptukey`; at the measured cells there they were within 9.6×10^{-6} of the exact quantile and 9.1×10^{-7} of the exact probability. The largest quantile error (700 groups, 57 df, at .9965) came from a dense probe between the grid's cells near the df bounds in (a), where the errors come closest to the 10^{-5} rule; at the grid's own cells the largest was 8.3×10^{-6}. Between the measured cells larger errors cannot be ruled out, and beyond 10000 groups (20000 at infinite df) the df bound in (a) and the cuts in (d) are extrapolated, not measured. Above 10^8 error df the distribution is within 2.3×10^{-7} in probability (1.4×10^{-6} in the quantile) of the infinite-df one, and the values are those of `qtukey` and `ptukey` at infinite df, except under (c), (d), and (e), where the exact infinite-df distribution is used; at the three df measured there DMAR's values were within 1.6×10^{-6} of the exact quantile and 9.1×10^{-7} of the exact probability. The exact computation agrees with the independent computations to 2×10^{-8} in the quantile at probabilities from .01 to .99999, and less closely far in the lower tail, where the quantile is small (to 4.6×10^{-8} at .001, 1.1×10^{-7} at .0001, and 4.8×10^{-7} at .00001); with two groups it reproduces the closed form $q = \sqrt{2}\, t_{1-\alpha/2, df}$.

Value

Returns the critical value in a output style (a `data.frame` with one row per critical value, following the format used by `cv_t` and `cv_z`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Tukey, J. W. (1953). *The problem of multiple comparisons*. Unpublished manuscript, Princeton University.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 5 on the multiple-comparisons problem, where the Tukey method for all-pairwise comparisons is developed.)

See Also

cv_t, cv_z, TukeyHSD, qtukey

Other critical values: cv_bonferroni_f(), cv_bryant_paulson(), cv_chisq(), cv_dunnett(), cv_f(), cv_scheffe(), cv_smm(), cv_t(), cv_z()

Examples

```
# Following the all-pairwise comparisons setting of Maxwell, Delaney, and
# Kelley (2027, Chapter 5): every pair of k = 3 group means is compared,
# with 27 error degrees of freedom, holding the family-wise error rate at
# .05.
cv_tukey_hsd(alpha_level = .05, df = 27, n_groups = 3)

# Using DMAR's test_market data (6 marketing panels, N = 24).
fit <- aov(brand_movement ~ panel, data = test_market)
cv_tukey_hsd(
  alpha_level  = .05,
  df     = df.residual(fit),
  n_groups = nlevels(test_market$panel)
)

# A more stringent alpha with a simple (non-verbose) result.
cv_tukey_hsd(alpha_level = .01, df = 27, n_groups = 3, verbose = FALSE)

# With 20 groups and only 2 error degrees of freedom the critical value
# comes from the exact distribution, 26.830. Base R's qtukey() is not
# accurate this far out; its value would give 31.485:
cv_tukey_hsd(alpha_level = .01, df = 2, n_groups = 20)
qtukey(.99, nmeans = 20, df = 2) / sqrt(2)
```

---

cv_z *Provides the Critical Value(s) for the Standard Normal Distribution (the z-distribution, With Mean 0 and Variance 1)*

Description

Provides the Critical Value(s) for the Standard Normal Distribution (the z-distribution, With Mean 0 and Variance 1)

Usage

```
cv_z(
  alpha_level,
  alternative = "not_equal",
  alpha_lower,
  alpha_upper,
  verbose = TRUE
)
```

Arguments

alpha_level	Type I error rate (i.e., the false positive rate).
alternative	The type of alternative hypothesis of interest.
alpha_lower	The error rate on the lower (negative) side of the distribution.
alpha_upper	The error rate on the upper (positive) side of the distribution.
verbose	Provides extra information about areas under the curve.

Value

Returns the critical value(s), based on the input specifications, in a output style.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

Other critical values: cv_bonferroni_f(), cv_bryant_paulson(), cv_chisq(), cv_dunnett(), cv_f(), cv_scheffe(), cv_smm(), cv_t(), cv_tukey_hsd()

Examples

```
# A basic call for finding critical values with equal area in the two tails.
cv_z(alpha_level = .05)

# A basic call for a single-sided confidence interval (for "a greater than" alternative hypothesis)
cv_z(alpha_level = .05, alternative = "greater")

# A single-sided confidence interval (for "a greater than" alternative hypothesis); simple output.
cv_z(alpha_lower = 0, alpha_upper = .05, verbose = FALSE)

# For a nonsymmetric 95% confidence interval.
cv_z(alpha_lower = .01, alpha_upper = .04)
```

---

depression_bdi *Depression Treatment Study With a Pretest Covariate*

Description

The hypothetical three-group depression study that runs through the analysis of covariance development of Maxwell, Delaney, and Kelley (2027, Chapter 9, Table 9.7). Thirty depressive individuals are randomly assigned, ten per group, to a selective serotonin reuptake inhibitor (SSRI), a placebo, or a wait list control. The Beck Depression Inventory (BDI) is administered before the study begins and again at its end, giving a pretest that serves as the covariate and a posttest that serves as the outcome.

Usage

```
depression_bdi
```

Format

A `data.frame` with 30 rows and 3 columns:

condition Factor with levels `ssri`, `placebo`, and `wait_list`: the randomly assigned treatment.

bdi_pre Beck Depression Inventory score before the study.

bdi_post Beck Depression Inventory score at the end of the study.

Details

This is the worked example behind the ANCOVA contrast pages: the pretest group means are 17, 17.7, and 17.4; the within-groups sum of squares of the pretest is 752.5; the ANCOVA error variance is about 29; and the covariate-adjusted posttest means are approximately 7.5, 12, and 14 for the SSRI, placebo, and wait list groups. The examples of `ci_c_ancova` and `ci_sc_ancova` quote those summary values, and a test recomputes each of them from these data so the printed numbers cannot drift.

The data are hypothetical, constructed for the book; higher BDI scores mean more severe depressive symptoms, so a treatment that works pulls the posttest down. The same numeric values ship in the book's data companion, the **AMCP** package, as `chapter_9_table_7`; DMAR carries them directly so its ANCOVA examples and tests need no package beyond this one.

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on ANCOVA.)

Examples

```
data(depression_bdi)

# The fingerprints the ANCOVA contrast pages quote.
tapply(depression_bdi$bdi_pre, depression_bdi$condition, mean)
fit <- lm(bdi_post ~ bdi_pre + condition, data = depression_bdi)
anova(fit)

# The covariate-adjusted group means at the pretest grand mean.
ancova(depression_bdi, outcome = "bdi_post", treatment = "condition",
       covariates = "bdi_pre")
```

`descriptives` *Descriptive Statistics for One or More Variables*

Description

Computes a compact set of descriptive statistics useful for data screening and psychometric work (including skewness and excess kurtosis), with an optional correlation matrix for the numeric variables.

Usage

```
descriptives(x, correlations = FALSE, listwise = FALSE)
```

Arguments

`x`	A `data.frame`, tibble, or `matrix` containing the variables to summarize. A `matrix` is coerced to a `data.frame`.
`correlations`	Logical. If `TRUE`, also return a correlation matrix for the numeric variables (see Details).
`listwise`	Logical. If `TRUE`, apply listwise deletion (drop every row containing any `NA`) to `x` *before* computing any statistics. If `FALSE` (the default), each variable is summarized using its own available (non-missing) observations.

Details

Skewness and kurtosis. The reported values are those of `skewness` and `kurtosis`, the bias-corrected (Type 2) formulas

$$\text{skewness} = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^3,$$

$$\text{kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)},$$

where s is the (divisor-$n-1$) sample standard deviation; the help pages of those functions give the other two formulas in use. Kurtosis is reported as excess kurtosis, whose expected value for a sample from a normal distribution is 0.

Variable-type handling. Each column is classified as `"numeric"`, `"integer"`, `"logical"`, `"factor"`, `"character"`, or whatever its first class is otherwise. Columns of type `"numeric"`, `"integer"`, and `"logical"` receive full distributional summaries (logicals are coerced so that `mean` is the proportion of `TRUE`s). All other columns report only `n`, `n_missing`, and `prop_missing`; their remaining columns are `NA`.

Correlations. The correlation matrix, when requested, uses Pearson correlations with `use = "pairwise.complete.obs"` so that each pairwise correlation uses the maximum information available. Only numeric-type variables are included.

Value

A list with two elements, returned in a stable shape regardless of the `correlations` argument:

`descriptives` A `data.frame` with one row per input variable and columns `variable`, `type`, `n`, `n_missing`, `prop_missing`, `mean`, `median`, `sd`, `min`, `max`, `q25`, `q75`, `skewness`, and `kurtosis` (excess kurtosis). For non-numeric variables, the numeric summary columns are `NA`.

`correlations` A $p \times p$ correlation matrix among the numeric variables, or `NULL` if `correlations = FALSE` or fewer than two numeric variables are available. This is returned as a plain matrix (not a table), because its natural structure is a symmetric two-dimensional array; that format reads well and plugs directly into downstream tools such as `cov2cor` or factor analysis and SEM software.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

`cor`, `quantile`

Other descriptive statistics: `kurtosis()`, `skewness()`

Examples

```
# Four cognitive tests from the holzinger_swineford data
# (301 children in two schools).
hs_tests <- holzinger_swineford[, c("t1_visual_perception", "t2_cubes",
                                    "t4_lozenges",
                                    "t6_paragraph_comprehension")]
descriptives(hs_tests)

# Include a correlation matrix (useful during scale development).
descriptives(hs_tests, correlations = TRUE)

# Mixed-type data: numeric summaries for the test scores, and
# type = "factor" (with NA numeric columns) for school.
descriptives(holzinger_swineford[, c("school", "t1_visual_perception",
                                     "t2_cubes")])

# Data with missing values: the revised paper form board and flags tests
# were administered only in the Grant-White school, so 156 of the 301
# children have no score. Per-variable N and missingness are reported.
hs_partial <- holzinger_swineford[, c("t1_visual_perception",
                                      "t25_paper_form_board_r",
                                      "t26_flags")]
descriptives(hs_partial)

# The same data with listwise deletion applied first.
descriptives(hs_partial, listwise = TRUE)
```

`design_consequences` *Consequences of a Design: Power, Sign and Magnitude Errors, and Expected Precision*

Description

Evaluates what a design of a given precision will actually deliver, under both of the package's lenses at once. The *significance lens*: the `power` of the two-sided test, the `type_s_error` (the probability that a statistically significant estimate has the wrong sign), and the `exaggeration_ratio` (Type M: the average factor by which significant estimates overstate the true effect), following the design analysis of Gelman and Carlin (2014). The *precision lens*, in the accuracy in parameter estimation (AIPE) tradition: the expected half-width and full width of the `conf_level` confidence interval the design will produce, the spread of that realized width, and, when a target `width` is supplied, `pct_ci_less_width`, the probability that the realized interval is no wider than the target, the same quantities the `ss_aipe_*_sensitivity()` family estimates by Monte Carlo, here in closed form.

Usage

```
design_consequences(
  true_effect = NULL,
  se = NULL,
  sd = NULL,
  n_1 = NULL,
  n_2 = NULL,
  alpha_level = 0.05,
  df = NULL,
  conf_level = 0.95,
  width = NULL
)
```

Arguments

`true_effect`	The assumed true (population) effect, on the scale of the estimate (a mean difference, a regression coefficient, a standardized mean difference). May be negative. May be `NULL`, in which case the significance-lens rows are returned as `NA` and only the precision lens (which does not involve the true effect) is informative.
`se`	The standard error of the estimate the design will produce, on the same scale as `true_effect`. Supply `se` directly, or supply `sd` with `n_1` (and `n_2`) and let the function derive it.
`sd`, `n_1`, `n_2`	An alternative to `se` for the two most common cases: with `sd` and `n_1` only, the one-sample (or paired-difference) design `se = sd / sqrt(n_1)` with `df = n_1 - 1`; with `n_2` as well, the two-group design `se = sd * sqrt(1/n_1 + 1/n_2)` with `df = n_1 + n_2 - 2` (`sd` is the common within-group standard deviation). A `df` supplied explicitly overrides the derived one.
`alpha_level`	Two-sided Type I error rate of the significance test. Defaults to 0.05.

df	Degrees of freedom of the reference *t* distribution. Defaults to Inf (the normal case) unless derived from n_1 / n_2.
conf_level	Confidence level of the interval evaluated by the precision lens. Defaults to 0.95.
width	Optional target full width for the confidence interval; when supplied, pct_ci_less_width reports the probability that the realized interval is no wider than width.

Details

Together they answer the two questions a chosen design should be interrogated with before data collection: *if I run this study and filter it through a significance test, what will the published record look like?* and *how precisely will I estimate the effect regardless of significance?* An underpowered design fails both: its significant estimates are exaggerated and possibly sign-reversed, and its confidence intervals are too wide to be informative.

Significance lens. Writing $\lambda = \theta/\mathrm{se}$ and c for the two-sided critical value, the power and the Type S error follow from the two tails of the distribution of the test statistic: the noncentral *t* with noncentrality λ when df is finite (the exact distribution of the *t* statistic when the standard error is estimated from the data, the same sampling model the precision lens uses), and the normal when df = Inf. The noncentral *t* probabilities are exact, including above a noncentrality of 37.62, where R's pt() is a normal approximation. The exaggeration ratio is the expected absolute estimate conditional on significance over the absolute true effect, computed exactly: from truncated normal moments when df = Inf, and otherwise by integrating those moments over the chi distribution of the estimated standard error, so no simulation error enters. Gelman and Carlin's retrodesign() instead evaluates a central *t* shifted by λ (and simulates the exaggeration ratio under that model), an approximation that treats the standard error as known; the two agree as df grows and coincide at df = Inf, but at small df they differ: in the underpowered regime the design analysis is aimed at (power below about 0.7), the known-se approximation understates power and overstates the Type S and Type M errors, so their published finite-df values differ from the exact ones reported here. When true_effect = 0 the power equals alpha_level, the Type S error is 0.5, and the exaggeration ratio is undefined (NA).

Precision lens. The realized interval half-width is $t_{1-\alpha^*/2,\,df} \cdot \widehat{\mathrm{se}}$ with $\alpha^* = 1 - \texttt{conf_level}$, and $\widehat{\mathrm{se}} = \mathrm{se}\sqrt{W/df}$ with $W \sim \chi^2_{df}$, so the width's mean, median, and standard deviation have closed chi-distribution forms and

$$P(\mathrm{width} \le w) \;=\; P\left(W \le df\,\left[\frac{w}{2\,t\,\mathrm{se}}\right]^2\right),$$

where w is the target full width, the argument width. These are the population versions of the mean_ci_width, median_ci_width, sd_ci_width, and pct_ci_less_width terms that the ss_aipe_*_sensitivity() functions estimate by Monte Carlo. With df = Inf the standard error is treated as known, the width is deterministic, and pct_ci_less_width is a step: 1 when the fixed width is at most width and 0 otherwise.

The function complements the planners rather than replacing them: ss_power_*() chooses a sample size for detection, ss_aipe_*() chooses one for precision, and design_consequences() interrogates whatever design came out (or the design a completed study used). The two lenses are the power and accuracy in parameter estimation approaches to sample size planning reviewed by Maxwell, Kelley, and Rausch (2008).

Value

A `data.frame` (class `dmar_tbl`) with the significance-lens rows (`power`, `type_s_error`, `exaggeration_ratio`), the precision-lens rows (`expected_half_width`, `mean_ci_width`, `median_ci_width`, `sd_ci_width`, `pct_ci_less_width`, `width`; the last two are `NA` when no `width` is supplied), and the design rows (`true_effect`, `se`, `df`, `alpha_level`). The confidence level is recorded in the `"conf_level"` attribute. The schema is constant: rows that do not apply are `NA`, never dropped.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Gelman, A., & Carlin, J. (2014). Beyond power calculations: Assessing Type S (sign) and Type M (magnitude) errors. *Perspectives on Psychological Science, 9*(6), 641–651. doi:10.1177/1745691614551642 (Their accompanying `retrodesign()` function evaluates a location-shifted central *t*, the known-se approximation, and obtains the exaggeration ratio by simulation; the finite-df case here uses the exact noncentral *t* and exact moments instead, so the two differ at small df. See Details.)

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

See Also

`ss_power_smd` and `ss_aipe_smd` for choosing the sample size by detection or by precision before this function interrogates the choice; `expected_smd` for the unconditional small-sample bias of the standardized mean difference, a different bias than the significance-filter exaggeration here.

Other design utilities: `design_effect()`, `effects_coding()`, `helmert_coding()`, `is_orthogonal_set()`, `orthogonal_polynomial()`

Examples

```
# ---- Both lenses on one underpowered design --------------------------
# True effect 0.1 measured with standard error 0.3 (say, d = .1 with
# about 22 per group, 45 in total): power is 6 percent, a significant
# result has a 17 percent chance of the wrong sign and overstates the
# truth seven-fold, and the 95 percent CI is about 1.2 wide, twelve
# times the effect. Bad for detection, bad for precision.
design_consequences(true_effect = 0.1, se = 0.3)
```

```
# ---- The same effect, precisely measured ------------------------------
design_consequences(true_effect = 0.1, se = 0.03)

# ---- From a planned two-group design (sd and per-group n) -------------
# d-type effect 0.4, common sd 1, 60 per group; finite df flows through
# both lenses.
design_consequences(true_effect = 0.4, sd = 1, n_1 = 60, n_2 = 60)

# ---- Will the interval beat a target width? ---------------------------
# Same design, asking for the probability the realized 95 percent CI is
# no wider than 0.7 (the closed-form pct_ci_less_width that the
# ss_aipe_smd_sensitivity() simulation estimates).
design_consequences(true_effect = 0.4, sd = 1, n_1 = 60, n_2 = 60,
                    width = 0.7)

# ---- Precision lens alone (no effect assumption needed) ---------------
design_consequences(true_effect = NULL, sd = 1, n_1 = 60, n_2 = 60,
                    width = 0.7)

# ---- After an AIPE plan: check the detection side ---------------------
# Plan for a full width of 0.5 on the SMD at delta = .4, then ask what
# that design does under the significance filter.
n_plan <- ss_aipe_smd(true_smd = 0.4, width = 0.5)$value[1]
design_consequences(true_effect = 0.4, sd = 1,
                    n_1 = n_plan, n_2 = n_plan, width = 0.5)
```

`design_effect` *Kish's Design Effect (DEFF), DEFT, and the Effective Sample Size*

Description

Computes the design effect (DEFF) and its square root (DEFT) for a clustered or multistage sample, given a vector of per-cluster sample sizes and a value of the intraclass correlation. Returns Kish's (1965) classic formula together with the effective sample size and a description of the clustering, including the number of empty clusters (no observations) and the number of singleton clusters (one observation), which carry different amounts of within-cluster information in a mixed-effects context.

Usage

```
design_effect(cluster_sizes, icc)
```

Arguments

`cluster_sizes`	Numeric vector of per-cluster sample sizes. Each element is the number of observations in one cluster (so the length of the vector is the number of clusters). Zero values are allowed and counted as empty clusters; they do not affect the computation of DEFF.

`icc` Intraclass correlation coefficient, in $[0, 1)$. Use the population value when planning prospectively, or the sample estimate (e.g., from `icc` or `icc_lmer`) when describing an observed clustered sample.

Details

Definition (Kish, 1965). For a clustered sample with per-cluster sizes $m_1, m_2, \ldots, m_K$ and intraclass correlation ρ, the design effect on the variance of the mean is

$$\text{DEFF} \;=\; 1 + (\bar{m}^{*} - 1)\rho,$$

where $\bar{m}^{*} = \sum_k m_k^2 / \sum_k m_k$ is the design-weighted average cluster size (Kish, 1965, eq. 5.4; sometimes called the "Kish weighted average" or "effective cluster size"). For equal cluster sizes $m_k = m$, this reduces to the classroom form $1 + (m - 1)\rho$. The DEFT is the square root of DEFF and is the inflation factor on the *standard error* of the mean (whereas DEFF inflates the variance).

Effective sample size. The number of observations from a simple random sample that would yield the same standard error as the clustered sample is

$$N_{\text{eff}} \;=\; N / \text{DEFF} \;=\; N / \text{DEFT}^2,$$

where $N = \sum_k m_k$ is the total observations.

Why empty and singleton clusters are reported separately. In mixed-effects / multilevel modeling, clusters with zero observations carry no information (they should be dropped before fitting), and clusters with one observation contribute to N and to the fixed-effect estimate but contribute nothing to the estimation of the random-effect variance or to the within-cluster residual. Hox et al. (2017) note that singleton-heavy designs have a design effect close to 1 even at moderate ρ because the weighted cluster size is small. The output reports the counts of empty and singleton clusters so the user can see at a glance how much of the nominal sample size carries clustering information.

If a fitted `lmerMod` object is available, the typical workflow is to extract `icc` via `icc_lmer` and the per-cluster sample sizes via `table(cluster_id)`, then pass both to `design_effect()`; see the second example.

Value

A `data.frame` with rows for the design effect, its square root, the effective sample size, the total observation and cluster counts (including separate counts of empty and singleton clusters), the mean cluster size, Kish's design-weighted mean cluster size, and the input `icc`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Hox, J. J., Moerbeek, M., & van de Schoot, R. (2017). *Multilevel analysis: Techniques and applications* (3rd ed.). Routledge.

Kish, L. (1965). *Survey sampling*. Wiley.

Kish, L. (1992). Weighting for unequal Pi. *Journal of Official Statistics, 8*(2), 183–200.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapters 15 and 16 on mixed-effects models and nested designs.)

Snijders, T. A. B., & Bosker, R. J. (2012). *Multilevel analysis: An introduction to basic and advanced multilevel modeling* (2nd ed.). Sage.

See Also

icc, icc_lmer, var_icc, ss_aipe_icc

Other design utilities: design_consequences(), effects_coding(), helmert_coding(), is_orthogonal_set(), orthogonal_polynomial()

Examples

```
# 1. Balanced design: K = 30 clusters of size 20, ICC = 0.10.
design_effect(cluster_sizes = rep(20, 30), icc = 0.10)
# DEFF = 1 + (20 - 1) * 0.10 = 2.9; DEFT = 1.7; effective N = 600 / 2.9 = 207.

# 2. Unbalanced design with some empty and some singleton clusters.
# Suppose K = 25 schools with attendance ranging from 0 (closed)
# through 1 (single student showed up) to 30 (full class).
sizes <- c(0, 0, 1, 1, 1, 2, 3, 5, 8, 10, 12, 15, 18, 20, 22,
           22, 25, 26, 28, 28, 30, 30, 30, 30, 30)
design_effect(cluster_sizes = sizes, icc = 0.10)

# 3. From a cluster-id vector: tabulate, then call design_effect().
cluster_id <- rep(1:8, times = c(20, 15, 22, 1, 0, 30, 18, 25))
design_effect(cluster_sizes = as.numeric(table(factor(cluster_id, levels = 1:8))),
     icc = 0.15)
```

---

df_barnard_rubin *Barnard and Rubin's Small-Sample Degrees of Freedom for Multiple Imputation*

Description

Computes the degrees of freedom of the t reference distribution for an estimate pooled over m imputed data sets by Rubin's (1987) rules, using Barnard and Rubin's (1999) small-sample formula, together with the quantities it is built from: the total variance, the fraction of missing information the imputations estimate, the relative increase in variance due to the missing values, and the two components of the degrees of freedom. The function is vectorized over parameters.

Usage

```
df_barnard_rubin(between, within, m, df_complete = Inf, floor = 3)
```

Arguments

`between`	The between-imputation variance B of each parameter: the variance, with divisor $m-1$, of its *m* estimates. A numeric vector, one value per parameter, whose names label the rows. (It is not called `B` because **DMAR** keeps `B` for the number of bootstrap replications.)
`within`	The within-imputation variance $\bar{U}$ of each parameter: the mean of its *m* squared standard errors. A numeric vector of the same length as `between`.
`m`	The number of imputations, a single whole number of 2 or more.
`df_complete`	The complete data degrees of freedom, of length 1 or the length of `between`. Defaults to `Inf`, which gives Rubin's (1987) large-sample degrees of freedom; a small-sample analysis should supply its complete data error degrees of freedom.
`floor`	The lower bound on the degrees of freedom: a single number of 0 or more. Defaults to 3, von Hippel's (2016) recommendation; `0` reproduces Barnard and Rubin (1999).

Details

Rubin's rules. Each of the *m* completed data sets gives an estimate $\hat{\theta}_d$ and a squared standard error U_d. The pooled estimate $\bar{\theta}$ is their mean, the within-imputation variance is $\bar{U} = m^{-1}\sum_d U_d$, the between-imputation variance is $B = (m-1)^{-1}\sum_d(\hat{\theta}_d - \bar{\theta})^2$, and the total variance of the pooled estimate is

$$T = \bar{U} + (1 + 1/m)B.$$

The share of the total variance due to the missing values,

$$\gamma = (1 + 1/m)B/T,$$

is the multiple imputation estimate of the fraction of missing information, which **mice** reports as `lambda`, and $r = (1 + 1/m)B/\bar{U}$ is the relative increase in variance due to nonresponse (**mice**'s `riv`). **mice**'s `fmi` column is a different quantity, $(r + 2/(\nu + 3))/(r + 1)$, which adjusts γ for the degrees of freedom. Von Hippel's (2016) Eq. 25 prints the within-imputation variance in place of B in the numerator of γ, a typographical error: that ratio would exceed 1 with no missing values at all.

The degrees of freedom. Rubin's (1987) degrees of freedom,

$$\nu_m = (m-1)/\gamma^2,$$

reflect only the finite number of imputations: they treat each completed data analysis as if it had infinitely many degrees of freedom, and when γ is small they can exceed the degrees of freedom the analysis would have with no missing values at all, which cannot be right in a small sample. Barnard and Rubin (1999) combine them with observed data degrees of freedom,

$$\nu_{obs} = \nu_{com}(1-\gamma)\frac{\nu_{com}+1}{\nu_{com}+3},$$

where ν_{com}, given as `df_complete`, is the complete data degrees of freedom (for a regression coefficient, $n-k$, with k the number of regression parameters), as

$$\nu_{BR} = \left(\frac{1}{\nu_m} + \frac{1}{\nu_{obs}}\right)^{-1},$$

which never exceeds ν_{obs} and hence never exceeds ν_{com}. The formulas are as von Hippel (2016, Eqs. 23 and 24) writes them, and they agree with **mice**'s `barnard.rubin()`. With `df_complete = Inf`, the default, ν_{obs} is infinite and ν_{BR} is Rubin's ν_m. The observed data degrees of freedom are `df_missing`'s `"complete_df"` rule, and von Hippel's degrees of freedom for maximum likelihood are the limit of ν_{BR} as the number of imputations grows, where ν_m becomes infinite.

The floor. An estimate of γ near 1 drives the degrees of freedom toward 0, where the t quantile grows without bound. In von Hippel's (2016, p. 431) simulation of 25 cases with half the values of one variable missing at random, 24% of the Barnard and Rubin degrees of freedom were below 3, 11% below 2, and 2% below 1, and some intervals were orders of magnitude longer than coverage required. He bounds the degrees of freedom below at 3 (his Eq. 27), which is the default `floor = 3` here; `floor = 0` reproduces Barnard and Rubin's formula, and **mice**, exactly. The unbounded value is kept in `df_unbounded`, and `at_floor` marks the rows where the bound applied.

DMAR does not impute; this function serves analyses pooled elsewhere and the comparison of maximum likelihood with multiple imputation. For a FIML fit, `df_missing` gives the corresponding degrees of freedom from the fit's own fraction of missing information, without imputations.

Value

A `data.frame` (class `dmar_tbl`) with one row per parameter and the columns `term`, `df` (bounded below by `floor`), `df_unbounded` (ν_{BR}), `gamma`, `relative_increase` (r), `total_variance` (T), `nu_m`, `nu_obs`, and `at_floor`. The attributes `m`, `df_complete`, and `floor` record the call, and the printed table, at the console or knitted, is followed by a line naming the rule, m, `df_complete`, and the floor. A missing input value gives missing values in its row.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Barnard, J., & Rubin, D. B. (1999). Small-sample degrees of freedom with multiple imputation. *Biometrika, 86*(4), 948–955. doi:10.1093/biomet/86.4.948

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. Wiley.

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software, 45*(3), 1–67. doi:10.18637/jss.v045.i03

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

See Also

`df_missing` for the degrees of freedom of a FIML estimate.

Other missing data: `cov_fiml()`, `df_missing()`, `fraction_missing_information()`, `missing_coverage()`, `missing_patterns()`

Examples

```
# One coefficient from m = 5 imputations, with complete data error
# degrees of freedom of 23: B = 0.04 and a within-imputation variance of
# 0.10. floor = 0 gives Barnard and Rubin's formula itself.
df_barnard_rubin(between = 0.04, within = 0.10, m = 5, df_complete = 23,
                 floor = 0)

# Rubin's (1987) large-sample degrees of freedom are 38, well above the
# 23 a complete data analysis of these cases would have.
df_barnard_rubin(between = 0.04, within = 0.10, m = 5)

# Pooling by hand: five estimates of each of two coefficients and their
# squared standard errors (illustrative values), one column per
# coefficient.
est <- cbind(slope = c(0.42, 0.51, 0.38, 0.47, 0.55),
             intercept = c(1.90, 1.94, 1.88, 1.93, 1.91))
u <- cbind(slope = c(0.010, 0.011, 0.009, 0.010, 0.012),
           intercept = c(0.020, 0.021, 0.019, 0.020, 0.022))
df_barnard_rubin(between = apply(est, 2, var), within = colMeans(u),
                 m = 5, df_complete = 23)
```

---

df_missing *Small-Sample Degrees of Freedom for Maximum Likelihood Intervals With Missing Data*

Description

Computes the degrees of freedom of a *t* reference distribution for a Wald confidence interval from a full information maximum likelihood (FIML) fit, from each parameter's fraction of missing information (FMI), by either of von Hippel's (2016) two rules. The interval keeps the FIML estimate and its standard error and replaces the normal quantile with the *t* quantile on these degrees of freedom, which lengthens it by an amount that grows as the sample shrinks and as the share of the information lost to missing values grows.

Usage

```
df_missing(
  fmi,
  N = NULL,
  k = NULL,
  df_complete = NULL,
  method = c("effective_n", "complete_df"),
  floor = 3
)
```

Arguments

`fmi`	The fraction of missing information of each parameter: a numeric vector of values below 1, or the table returned by `fraction_missing_information`, whose `fmi` and `N` columns are used and whose `term` column labels the rows. The names of a named vector label the rows. A missing value gives missing degrees of freedom for that row.
`N`	The number of cases the fit used (with **lavaan**, the rows with at least one observed variable, `lavInspect(fit, "nobs")`), of length 1 or the length of `fmi`. Required by `method = "effective_n"` unless `fmi` is a table with an `N` column; not used by `"complete_df"`.
`k`	The number of parameters subtracted from the effective sample size, of length 1 or the length of `fmi`: for a regression coefficient, the number of regression parameters (slopes and intercept), so that the complete data degrees of freedom are $N-k$; for a factor loading or omega, 1, the value DMAR's calibration chose (see *Details*). Required by `method = "effective_n"`, with no default.
`df_complete`	The complete data degrees of freedom, of length 1 or the length of `fmi`: the error degrees of freedom the interval would use if no value were missing ($n-k$ for a regression coefficient, $n-1$ for a mean), not the model degrees of freedom of a structural equation model. `Inf` is allowed and gives the normal reference. Required by `method = "complete_df"`, with no default.
`method`	`"effective_n"` (the default; von Hippel's Eq. 31, the rule his Table 5 used) or `"complete_df"` (his Eq. 32).
`floor`	The lower bound on the degrees of freedom: a single number of 0 or more. Defaults to 3, von Hippel's (2016) recommendation; `0` removes it.

Details

Why a *t* reference distribution. A maximum likelihood Wald interval is the estimate plus or minus a normal quantile times the standard error; the normal quantile comes from the large-sample distribution of the estimate. With incomplete data in small samples that interval is too short. In von Hippel's (2016) simulations of bivariate normal data with half the values of one variable missing, the normal FIML intervals at *n* = 25 covered 5 to 6 points below the nominal 95% for the intercept and slope of a regression and 8 to 14 points below it for the residual variance, and at *n* = 100 they were 0 to 3 points below (pp. 432–433). What fails is the reference distribution, not the estimate or its standard error: the normal quantile treats the standard error as known, and with few cases, or few effective cases once the missing information is subtracted, the standard error is itself estimated with error, as it is in a complete data regression whose coefficients are referred to *t* on $N-k$ degrees of freedom. The *t* distribution is an approximation here, since the small-sample distribution of a FIML estimate is not exactly *t* (von Hippel, 2016, p. 431). Von Hippel's *t* intervals brought the coverage of most parameters to within 2 points of nominal (p. 434). At *n* = 25 it stayed below nominal for the parameters whose point estimates are biased, because a longer interval covers the variability of an estimate, not its bias.

The two rules. With γ the parameter's fraction of missing information, `method = "effective_n"` (von Hippel's ν^*_{ML}, his Eq. 31) is

$$\nu^* = N(1-\gamma) - k,$$

the complete data regression degrees of freedom $N-k$ with N replaced by the effective sample size $N(1-\gamma)$, the number of complete cases that would carry the same information about the parameter (Savalei & Rhemtulla, 2012, p. 488). Here k is the number of regression parameters, slopes and intercept. With no missing information it is $N-k$ exactly. `method = "complete_df"` (von Hippel's ν_{ML}, his Eq. 32) is

$$\nu = \nu_{com}(1-\gamma)\frac{\nu_{com}+1}{\nu_{com}+3},$$

where ν_{com}, given as `df_complete`, is the degrees of freedom the interval would use if no value were missing: the error degrees of freedom, not the model degrees of freedom of a structural equation model (p. 423). It is the limit of Barnard and Rubin's (1999) degrees of freedom for multiple imputation as the number of imputations grows, which von Hippel derives from the large-sample equivalence of maximum likelihood and multiple imputation with infinitely many imputations, and it is the observed data part of that formula (see `df_barnard_rubin`). With no missing information it gives $\nu_{com}(\nu_{com}+1)/(\nu_{com}+3)$ rather than ν_{com} (21.2 rather than 23 when ν_{com} = 23); the factor comes from the observed data degrees of freedom and does not vanish with the missing information. The two rules agree closely: for a simple regression with *n* = 25, *k* = 2, and γ = .5 they give 10.6 and 10.5 (p. 432). Von Hippel's Table 5 used the effective sample size rule, and he reports very similar results with the general one (p. 433).

The floor. Either rule can return a small or negative value when the FMI is estimated near 1 or, for `"effective_n"`, when the effective sample size approaches *k*. A *t* distribution with 2 or fewer degrees of freedom has no finite variance, and its quantiles grow without bound as the degrees of freedom approach 0 (a 95% interval on 1 degree of freedom uses 12.7), so von Hippel bounds the degrees of freedom below at 3 (his Eq. 34). `floor = 3` is that bound. The unbounded value is kept in `df_unbounded`, and `at_floor` marks the rows where the bound applied. `floor = 0` returns the unbounded `"complete_df"` value, which is never negative; a negative `"effective_n"` value becomes 0 there, which is not a usable reference distribution.

Where the FMI comes from. The FMI of a parameter is the share of its sampling variance due to the missing values, $\gamma = 1-SE_C^2/SE_O^2$, with SE_C its standard error had the data been complete and SE_O its standard error from the incomplete data (Savalei & Rhemtulla, 2012, Eq. 4). It differs from parameter to parameter, and so do the degrees of freedom: a parameter estimated mostly from fully observed variables keeps nearly all of its degrees of freedom however many values are missing elsewhere. `fraction_missing_information` estimates it from a FIML fit, and its table can be passed as `fmi` directly; **lavaan**'s `parameterEstimates(fit, fmi = TRUE)` gives another estimate. Some estimators return negative FMIs in small samples (Chen & Savalei, 2021); a negative FMI has no degrees of freedom interpretation, so a value below -10^{-8} is refused, and a value between that and 0, which is rounding error around a parameter with no missing information, is set to 0.

Which parameters the rules are supported for. Von Hippel's simulations used bivariate normal data with correlation .5, *n* = 25 and 100, and half the values of one variable missing, completely at random or whenever the other variable was negative (missing at random). The intervals evaluated were for the intercepts and slopes of the regression of each variable on the other, the mean of the incomplete variable, the covariance, and the variances (on a cube root scale). The rules therefore have published support for regression coefficients and for moments: means, variances, and covariances. They have none for factor loadings or for reliability coefficients such as omega, whose complete data intervals are themselves asymptotic, so that no complete data degrees of freedom are defined for them. For those, DMAR's own calibration study (in the package's source repository, `methodology/missing_data`; `fa_confirmatory` describes its design) simulated one factor models fit by FIML and found that `method = "effective_n"` with `k = 1` brought the coverage of the loadings and of omega closer to .95 than the normal reference did: no cell's coverage, averaged over

its loadings, exceeded .9502, and omega's .9550 (single loadings reached .9595 in the cells with missing values and .963 in the complete ones, where the rule is t on $N - 1$ and the normal reference reached .962); it is the rule `fa_confirmatory` and `reliability_omega` apply under `wald_df = "missing_information"`. For coefficient alpha no rule passed the study's decision rule, fixed before the run, which a coverage of .966 in one cell decided (see `reliability_alpha`).

These numerical results are conditional on the H1 eligibility rule the study applied. The affine-collapse diagnostic sets aside additional draws, and the study has not established coverage under that selection. A raw-data eligibility replay alone cannot validate the FMI refits and information diagnostics. The keyword is an approximate, opt-in rule, not a coverage guarantee.

The function has no default for `k` or `df_complete`, because the right value depends on the parameter.

Value

A `data.frame` (class `dmar_tbl`) with one row per value of `fmi`, in input order, and the columns `term`, `df` (the degrees of freedom, bounded below by `floor`), `df_unbounded`, `fmi` (the value used), `N`, `k`, `df_complete`, and `at_floor` (`TRUE` where the bound applied). A column the method does not use is `NA`: `df_complete` under `"effective_n"`, `k` under `"complete_df"`, and `N` under `"complete_df"` unless a table supplied it. The attributes `method` and `floor` record the call, and the printed table, at the console or knitted, is followed by a line naming the rule and the floor.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Barnard, J., & Rubin, D. B. (1999). Small-sample degrees of freedom with multiple imputation. *Biometrika, 86*(4), 948–955. doi:10.1093/biomet/86.4.948

Chen, L., & Savalei, V. (2021). Three sample estimates of fraction of missing information from full information maximum likelihood. *Frontiers in Psychology, 12*, 667802. doi:10.3389/fpsyg.2021.667802

Savalei, V., & Rhemtulla, M. (2012). On obtaining estimates of the fraction of missing information from full information maximum likelihood. *Structural Equation Modeling, 19*(3), 477–494. doi:10.1080/10705511.2012.687669

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

See Also

`df_barnard_rubin` for the degrees of freedom of a multiple imputation estimate; `mlmr` for FIML regression.

Other missing data: `cov_fiml()`, `df_barnard_rubin()`, `fraction_missing_information()`, `missing_coverage()`, `missing_patterns()`

Examples

```
# von Hippel's (2016, p. 432) worked example: a simple regression with
# n = 25 cases, an intercept and a slope (k = 2), and half of the
# information about the slope missing.
df_missing(0.5, N = 25, k = 2)
df_missing(0.5, df_complete = 23, method = "complete_df")

# Each parameter gets the degrees of freedom of its own FMI. With no
# missing information the effective sample size rule returns the
# complete data degrees of freedom, N - k = 23; near an FMI of 1 the
# floor of 3 takes over.
df_missing(c(slope = 0.5, intercept = 0.1, complete = 0, sparse = 0.95),
           N = 25, k = 2)

# A regression whose outcome is observed for only the 145 Grant-White
# students of 301: t26_flags is missing for every Pasteur student. mlmr()
# fits it by full information maximum likelihood (with the Wald interval,
# whose normal quantile the t reference replaces, and without the effect
# size refits, which this example does not use), and
# fraction_missing_information() gives each slope's FMI with the N behind
# it, so its rows go to df_missing() as they are; k = 3 counts the two
# slopes and the intercept.
fit <- mlmr(t26_flags ~ t1_visual_perception + t4_lozenges,
            data = holzinger_swineford, ci_method = "wald",
            effect_sizes = FALSE)
fmi <- fraction_missing_information(fit)
slopes <- fmi[fmi$op == "~", ]
dfs <- df_missing(slopes, k = 3)
dfs

# The t interval keeps the FIML estimate and standard error and replaces
# the normal quantile. With more than a hundred effective cases the
# change is small; it matters when the effective sample size is small.
est <- coef(fit)[-1]
se <- sqrt(diag(vcov(fit)))[-1]
crit <- qt(0.975, dfs$df)
data.frame(term = names(est), df = dfs$df,
           t_ci_lower = est - crit * se, t_ci_upper = est + crit * se,
           normal_ci_lower = est - qnorm(0.975) * se,
           normal_ci_upper = est + qnorm(0.975) * se, row.names = NULL)
```

`diagnosis_agreement` *Cohen's (1968) Psychiatric Diagnosis Agreement Table*

Description

The illustrative agreement matrix from Cohen's (1968) weighted kappa paper, Table 1: two judges independently assign $N = 200$ cases to three diagnostic categories (personality disorder, neurosis, psychosis). The data set reproduces the printed table cell for cell and in its original layout, Judge

B indexing the rows and Judge A the columns, one row per cell of the 3 x 3 matrix. Each cell carries the three quantities Cohen prints: the ratio-scaled disagreement weight, the chance-expected proportion (his parenthetical values), and the observed proportion; the raw frequency is the observed proportion times *N*.

Usage

```
diagnosis_agreement
```

Format

A data frame with 9 observations (one per cell of the 3 x 3 agreement matrix) on 6 variables.

`judge_b` Factor: Judge B's diagnostic category (the table's rows), with levels `Personality disorder`, `Neurosis`, `Psychosis`.

`judge_a` Factor: Judge A's diagnostic category (the table's columns), same levels.

`frequency` Number of the 200 cases jointly assigned to the cell.

`disagreement_weight` Cohen's ratio-scaled disagreement weight v_{ij} for the cell: 0 on the agreement diagonal, 1 for a personality disorder-neurosis confusion, 3 for personality disorder-psychosis, and 6 for neurosis-psychosis, the confusion the illustration treats as gravest.

`observed_proportion` Observed proportion of cases in the cell, `frequency / 200`.

`expected_proportion` Chance-expected proportion of cases in the cell, the product of the cell's row (Judge B) and column (Judge A) marginal proportions; the parenthetical values in Cohen's Table 1.

Details

Cohen built this table to show that weighted kappa is fully chance corrected and that it can be *smaller* than unweighted kappa on the same data. Here the judges disagree far less than chance expectation in the mildly weighted personality disorder-neurosis cells but at about the chance level in the heavily weighted neurosis-psychosis cells, so $\kappa = .492$ while $\kappa_W = .348$: they disagree least where it matters least. Interchanging the 6 and 1 weights reverses the conclusion ($\kappa_W = .574$).

The reconstruction was verified against every quantity Cohen computes from the table: the marginals (.50/.30/.20 for Judge B, .60/.30/.10 for Judge A), the chance-expected cell proportions, the weighted disagreement sums $q'_o = .90$ and $q'_c = 1.38$, $\kappa = .492$, $\kappa_W = .348$, and his Formula 10 and 13 standard errors (.0901 and .0916). The cohen_kappa help page replicates the full set of analyses, and the weighted kappa vignette works the illustration end to end, including the orientation of the printed weight display in the paper's asymmetric-weight validity reinterpretation.

Author(s)

Ken Kelley <kkelley@nd.edu>

Source

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin, 70*(4), 213–220 (Table 1, p. 214).

References

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin, 70*(4), 213–220. doi:10.1037/h0026256

See Also

cohen_kappa, which analyzes this table in its examples.

Other reliability: cohen_kappa(), fleiss_kappa(), icc(), reliability(), reliability_H(), reliability_alpha(), reliability_kr20(), reliability_omega(), reliability_omega_categorical()

Examples

```
data(diagnosis_agreement)

# Rebuild Cohen's Table 1 layout (Judge B in rows, Judge A in columns).
xtabs(frequency ~ judge_b + judge_a, data = diagnosis_agreement)

# Unweighted and weighted kappa, reproducing kappa = .492 and
# kappa_w = .348.
tab <- xtabs(frequency ~ judge_b + judge_a, data = diagnosis_agreement)
v   <- xtabs(disagreement_weight ~ judge_b + judge_a,
             data = diagnosis_agreement)
cohen_kappa(table = tab)
cohen_kappa(table = tab, weights = unclass(v),
            weight_scaling = "disagreement")

# Cohen's Formula 8 directly from the per-cell quantities.
with(diagnosis_agreement,
     1 - sum(disagreement_weight * observed_proportion) /
         sum(disagreement_weight * expected_proportion))
```

---

dmacs *The dMACS Effect Size of Measurement Noninvariance*

Description

Quantifies how much a violation of measurement invariance actually matters for an item, rather than only whether it is statistically detectable. A likelihood ratio or score test can flag a loading or intercept difference that is too small to change anyone's score in a meaningful way, and with a large sample size it usually will. The dMACS index of Nye and Drasgow (2011) answers the size question directly: it is the expected difference between the reference group's and the focal group's measurement equations for that item, averaged over the focal group's latent distribution and standardized by the pooled item standard deviation, so it reads on the familiar standardized mean difference scale. Input is either a fitted multiple group **lavaan** model (requires **lavaan**) or the loadings, intercepts, and pooled standard deviations a paper reports.

Usage

```
dmacs(
  fit = NULL,
  reference = NULL,
  focal = NULL,
  factor_loading_reference = NULL,
  factor_loading_focal = NULL,
  intercept_reference = NULL,
  intercept_focal = NULL,
  mean_focal = 0,
  sd_focal = 1,
  sd_pooled = NULL,
  item_names = NULL
)
```

Arguments

`fit`	A fitted multiple group **lavaan** object carrying a mean structure, with the two groups' loadings and intercepts on a common metric (see Details). Supply either `fit` or the parameter vectors, never both. A fit whose optimizer did not converge (`lavaan::lavInspect(fit, "converged")` is `FALSE`) is refused with an error of class `dmar_not_converged`, since its estimates mark where the optimizer stopped.
`reference`, `focal`	Which groups play the reference and focal roles, each given as a single group label or a single group index in `lavInspect(fit, "group.label")`. When both are `NULL` (default) and the fit has exactly two groups, the first is the reference and the second the focal; with more than two groups they must be named.
`factor_loading_reference`, `factor_loading_focal`	Numeric vectors of unstandardized factor loadings (λ_R and λ_F of Details), one per item, in the reference and focal groups. Any finite values are admissible.
`intercept_reference`, `intercept_focal`	Numeric vectors of unstandardized item intercepts (ν_R and ν_F of Details), one per item, in the reference and focal groups, in the same item order as the loadings. Any finite values are admissible.
`mean_focal`	Focal group latent mean μ_F on the common metric, a single finite number (default `0`, the usual identification in which the reference group's latent mean is fixed at zero). Used only on the parameter vector path; with a `fit` it is read from the fit.
`sd_focal`	Focal group latent standard deviation σ_F on the common metric, a single positive number (default `1`). Used only on the parameter vector path; with a `fit` it is read from the fit.
`sd_pooled`	Pooled observed standard deviation of each item across the two groups, either one positive number applied to every item or one per item. Required on the parameter vector path; with a `fit` it is computed from the group sample sizes and observed item variances.

`item_names` Optional character vector of item labels, one per item. Defaults to the names carried by the parameter vectors, to the indicator names in the fit, or to `item_1`, `item_2`, and so on.

Details

For item i, let ν_R and λ_R be the reference group's intercept and loading (`intercept_reference` and `factor_loading_reference`), let ν_F and λ_F be the focal group's (`intercept_focal` and `factor_loading_focal`), and let the focal group's latent variable be $\eta \sim N(\mu_F, \sigma_F^2)$ with density f_F. Each group's measurement equation gives an expected item score at every value of η, and dMACS is the root mean squared vertical distance between those two lines over the focal group's latent distribution, divided by the pooled item standard deviation:

$$d_{MACS,i} = \frac{1}{SD_i}\sqrt{\int \left[(\nu_R + \lambda_R\eta) - (\nu_F + \lambda_F\eta)\right]^2 f_F(\eta)\, d\eta}.$$

Writing $a = \nu_R - \nu_F$ for the intercept difference and $b = \lambda_R - \lambda_F$ for the loading difference, the integrand is $(a + b\eta)^2$ and the integral is the second moment of a linear function of a normal variate, so it has the closed form $a^2 + 2ab\mu_F + b^2(\sigma_F^2 + \mu_F^2)$. No numerical integration is performed. The standardizer is the pooled observed standard deviation of the item,

$$SD_i = \sqrt{\frac{(n_R - 1)s_R^2 + (n_F - 1)s_F^2}{n_R + n_F - 2}},$$

the same pooling used by the standardized mean difference, which puts dMACS on a scale a reader of `smd` already understands.

dMACS is a root mean square and so is nonnegative by construction; it carries no sign and is not given one here. The direction of the violation is read from the returned components: $\nu_R - \nu_F$ says which group is scored higher at the mean of the latent variable, and $\lambda_R - \lambda_F$ says in which group the item discriminates more sharply. When only the intercepts differ, the integral collapses to a^2 and dMACS reduces to $|a|/SD_i$, a plain standardized intercept difference.

The index is interpretable only when the two groups' loadings and intercepts are expressed on a common metric. In practice that means a partial invariance model in which a set of anchor items is constrained equal across groups while the suspect items are freed, and the focal group's latent mean and variance are freely estimated. A configural model, which sets each group's latent scale separately, does not put the groups on a common metric, and dMACS computed from one is not meaningful. The usual workflow is therefore `measurement_invariance` to locate where the ladder breaks, a partial invariance refit that frees the offending parameters, and then `dmacs()` on that refit to judge whether the violation is large enough to matter.

On the fit path each group's item variance is computed from the data in the fit with the usual $n - 1$ divisor, using the number of nonmissing observations for that item in that group. When the model was fitted from sample moments rather than raw data, the sample covariances in the fit are used instead, rescaled to the $n - 1$ divisor when the fit's likelihood option calls for it.

Value

A wide `data.frame` (class `dmar_tbl`) with one row per item and columns

`item` Item label.

`factor_loading_reference` Reference group unstandardized loading.

`factor_loading_focal` Focal group unstandardized loading.

`intercept_reference` Reference group unstandardized intercept.

`intercept_focal` Focal group unstandardized intercept.

`sd_pooled` Pooled observed standard deviation of the item.

`dmacs` The dMACS effect size, nonnegative.

The returned object carries four attributes: `"reference"` and `"focal"`, the two group labels, and `"mean_focal"` and `"sd_focal"`, the focal group's latent mean and standard deviation used in the integral (named by latent variable on the fit path). A fifth, `"measured_terms"`, names the numeric columns for the display, so that a loading or an intercept given as a whole number prints to the decimal places of the others (see `dmar_tbl`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Meredith, W. (1993). Measurement invariance, factor analysis and factorial invariance. *Psychometrika, 58*(4), 525–543. doi:10.1007/BF02294825

Millsap, R. E. (2011). *Statistical approaches to measurement invariance*. Routledge.

Nye, C. D., Bradburn, J., Olenick, J., Bialko, C., & Drasgow, F. (2019). How big are my effects? Examining the magnitude of effect sizes in studies of measurement equivalence. *Organizational Research Methods, 22*(3), 678–709. doi:10.1177/1094428118761122

Nye, C. D., & Drasgow, F. (2011). Effect size indices for analyses of measurement equivalence: Understanding the practical importance of differences between groups. *Journal of Applied Psychology, 96*(5), 966–980. doi:10.1037/a0022955

See Also

`measurement_invariance` for the invariance ladder that locates a violation; `smd` for the standardized mean difference whose pooling and scale dMACS borrows; `cfa_1` for the single group measurement model.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
# Reported measurement equations: the two items share loadings, and the
# second item's intercept is 0.30 higher in the reference group. With no
# loading difference, dMACS is just 0.30 divided by the pooled SD.
dmacs(factor_loading_reference = c(0.80, 0.75),
      factor_loading_focal = c(0.80, 0.75),
      intercept_reference = c(2.00, 2.30),
      intercept_focal = c(2.00, 2.00),
      sd_pooled = c(1.20, 1.10), item_names = c("optimism", "worry"))

# A partial invariance model for the four spatial tests at the two
# Holzinger and Swineford schools. The anchors are constrained equal; the
# cubes and lozenges tests are freed, so only those two can move.
data(holzinger_swineford)
items <- c("t1_visual_perception", "t2_cubes",
           "t3_paper_form_board", "t4_lozenges")
model <- paste("spatial =~", paste(items, collapse = " + "))
fit <- lavaan::cfa(model, data = holzinger_swineford, group = "school",
                   group.equal = c("loadings", "intercepts"),
                   group.partial = c("spatial =~ t2_cubes", "t2_cubes ~ 1",
                                     "spatial =~ t4_lozenges",
                                     "t4_lozenges ~ 1"))
dmacs(fit)

# The broom verbs: one row per item, and the group metadata.
generics::tidy(dmacs(fit))
generics::glance(dmacs(fit))
```

`dmar_bootstrap` *How DMAR Counts and Requires Bootstrap Replications*

Description

One rule governs every DMAR function that reads an interval, a bound, a reference quantile, a standard error, or a Monte Carlo *p*-value from `B` bootstrap or Monte Carlo replications: how many replications the request must ask for, what happens to a replication that returns no value, what the result records, and when no interval is reported. This page states the rule and its reasons; each function's page says how the rule applies to it.

How many replications an interval needs

A percentile limit is a quantile of the `B` replications, so it is located by the replications that lie beyond it. With `B` = 100 and a 95 percent interval there are two or three of them on each side, and the limit moves with every new seed. DMAR runs an interval read from quantiles only when the number of replications requested puts at least 10 beyond each limit, and never from a request of fewer than 100: 200, 400, and 2,000 replications at the 90, 95, and 99 percent levels. The count is held to the number requested, `B`; what happens when some of them return no value is decided afterward

and is described under *Replications that return no value*. In symbols, with t the smallest tail area a reported limit cuts off ($(1 - C)/2$ for a two-sided interval at level C),

$$B_{\text{required}} = \max\{100, \lceil 10/t \rceil\}.$$

With 10, the share of the bootstrap distribution a limit actually cuts off has a relative standard deviation of about 30 percent whatever the shape of that distribution (a nominal 2.5 percent is between about 1.2 and 4.5 percent in 95 of 100 runs, depending slightly on how the quantile is interpolated). How far the limit itself moves depends on the shape: with 400 replications at the 95 percent level its standard deviation from run to run is about 0.13 bootstrap standard errors for a bell-shaped distribution, about 0.20 for one with tails as heavy as a *t* distribution on 5 degrees of freedom, and about 0.29 on the long side of a skewed one. Ten is a floor and not a recommendation. The defaults (1,000 where each replication refits a model, 2,000 to 10,000 elsewhere) put 25 to 250 replications beyond each limit at the 95 percent level, and at 1,000 the standard deviation of a limit of a bell-shaped distribution is about 0.08 standard errors. A BCa interval needs more, 1,000, 2,000, and 10,000 at the three levels, for the reason given under *BCa intervals*. An interval built from the standard deviation of the replications (`"bootstrap_se"`) is reported from 100, at which that standard deviation has a coefficient of variation of about 7 percent for a bell-shaped distribution (12 to 14 percent for the heavy-tailed and the skewed shapes above), and a Monte Carlo *p*-value from 100, at which its standard error is at most 0.05. A request below the number required is refused before anything is computed, with a condition of class `dmar_count_invalid` that states the number. None of these numbers concerns the coverage of the bootstrap for a given statistic, which no number of replications repairs; they limit only the error added by drawing a finite number of resamples. The numbers in this paragraph were computed for DMAR by simulation (20,000 runs each); the statement about the share cut off follows from the Beta distribution of an order statistic (the tail area beyond the k-th of B ordered draws from a continuous distribution is distributed as Beta(k, $B - k + 1$)). No published source is cited for the counts.

A limit at quantile position p has the tail area $\min(p, 1 - p)$. For an interval with unequal tail areas (`alpha_lower` and `alpha_upper` of the ratio functions and of `ci_eta_squared_generalized`) the smaller area decides; for a one-sided bound (`htmt`) and for a reference quantile (`parallel_analysis`, `factor_count`) it is the area beyond that one quantile, so a bound at the 95th percentile needs 200 replications and one at the median needs the floor of 100.

The functions that compute their limits directly read them with `quantile` (its default, type 7); the reliability family and `ci_bifactor_indices` read them with `boot::boot.ci()`, which places a limit at the $(B + 1)\alpha$ position among the ordered replications. The two differ by a fraction of the distance between adjacent replications.

Replications that return no value

A resample can leave a statistic undefined (a constant variable, a model that does not converge, a denominator of zero). Such a replication is dropped and counted, never replaced. Every result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for) as attributes, which `dmar_metadata` returns, with `bootstrap_status`, and one warning of class `dmar_bootstrap_replications_dropped` states the counts. An interval from the replications that remain describes the resamples in which the statistic is defined. When the dropped replications number at least the share of the distribution beyond one limit (2.5 percent of `B` for a 95 percent interval), they could by themselves decide where that limit falls, and the warning says so (`caution` is `TRUE` in the condition). When more than half were dropped no interval is reported: the limits are `NA`, the estimate is returned, and the warning

has class dmar_bootstrap_insufficient. The call does not stop for this, so an estimate that took minutes to compute is kept. With more than half dropped a larger B does not help: the replications that remain are a minority of the resamples, selected by whether the statistic could be computed, and their spread describes those resamples and not the sampling distribution of the estimate.

The number required is counted on the request and not on the replications that remain. A request of exactly 400 at the 95 percent level from which 3 replications are dropped reports its interval from the 397 that remain, with the warning. The two rules about shares, in the paragraph above, then decide: more than half dropped withholds the interval, and dropped replications that reach the share beyond one limit bring the caution. The warning states how many were dropped, and when those that remain are fewer than the number required it says so and says to request more. Where replications can fail (a model refit in every resample, sparse categories, a small sample), ask for more than the minimum, so that the limits rest on at least the number the rule intends; the defaults do.

bootstrap_status is one of "complete" (nothing dropped), "dropped" (some dropped, the interval formed), "exploratory" (formed from a request below the required number, under the option described below), "insufficient_count" (no interval: a BCa interval whose adjusted positions take more replications than were requested, or fewer than two usable replications), and "insufficient_share" (more than half dropped, no interval). With several statistics in one call (the pairs of htmt, the effects of mediation_mbco, the coefficients of mlmr), B_used is the smallest count any reported limit rests on and n_dropped is B - B_used. Where a table already has rows for the counts it keeps them (mediate has rows B and B_used, krippendorff_alpha the row B_used); the reliability family keeps its fuller bootstrap attribute, which gains required and bootstrap_status.

BCa intervals

The BCa interval reads its limits from quantiles that the bias correction and the acceleration move away from the nominal ones. For a 95 percent interval the positions might be 0.9 and 99.1 percent in place of 2.5 and 97.5, and the limit at 0.9 percent has few replications beyond it. The count of 10 is applied where each limit finally sits, again to the number of replications requested, so a BCa interval can be withheld after the replications are run (warning of class dmar_bootstrap_insufficient, with the positions and the count that would have served). To make that uncommon, a BCa interval is run only with 50 replications beyond each nominal limit: 1,000, 2,000, and 10,000 at the 90, 95, and 99 percent levels. In simulations of an indirect effect and of a correlation at the 95 percent level, the adjusted positions asked for more than 2,000 replications in at most 14 percent of data sets (an indirect effect with 50 cases and small paths) and in 0 to 3 percent of the others, and for more than 10,000 in about 1 percent of the data sets of that hardest condition and in none of the others (400 data sets per condition).

When the bias correction or the acceleration is itself undefined (a leave-one-out estimate that cannot be computed, every replicate on one side of the estimate, influence values without spread), the functions differ by family. mediate, cohen_kappa, and fleiss_kappa refuse the BCa interval with an error of class dmar_bca_undefined whose message names the cause; ci_ratio_of_means and ci_ratio_of_means_paired stop with an error that names ci_method = "percentile"; the reliability family and ci_bifactor_indices report the percentile interval in its place, with a warning, and record the substitution in the ci_method attribute.

Fewer replications, for exploration

A first look at an analysis, a help page example, and a simulation that averages over many data sets do not need a reportable interval from each run. `options(dmar.bootstrap_exploratory = TRUE)` lets a function run with any `B` of 2 or more. Nothing else changes: the result still records `B_required`, its footer says that the interval is below the requirement, and every call gives a warning of class `dmar_bootstrap_exploratory`. A planner that runs `B` replications inside each of many simulated studies (`ss_aipe_indirect_effect`, `ss_aipe_indirect_effect_sensitivity`, `ss_aipe_reliability_sensitivity`) gives the warning once for the call and records the request (`B_used` is then `B`, the number each simulated study ran), and `plot_mediation_mbco`, whose result is a plot, gives the warning and records nothing. The share rules still apply. An interval obtained this way is not one to report. Set the option for the calls that need it and restore it afterward:

```
old <- options(dmar.bootstrap_exploratory = TRUE)
mediate(d, x = "x", m = "m", y = "y", B = 50, seed = 113)
options(old)
```

Monte Carlo intervals and reference quantiles

The same rule holds for limits read from the quantiles of Monte Carlo draws (`ci_method = "monte_carlo"` of `mediate` and of `mediation_mbco`, the band of `plot_mediation_mbco`) and for the reference quantile of `parallel_analysis` and `factor_count`, with "draw" or "reference data set" in place of "replication". The Bollen-Stine *p*-value of `sem_exact_fit_bootstrap` is a proportion and not a quantile: it needs a request of 100 samples, is withheld when more than half were set aside, and is reported with the Wilson interval of its own Monte Carlo error.

What is not a bootstrap

The rule does not govern `n_resamples` of `randomization_test`, `randomization_test_paired`, and `procrustes_phi`, or the number of simulated studies `G` of the sample size planning and sensitivity functions. Those counts are checked as counts and their functions report their own Monte Carlo error.

Reproducibility

Every function that draws replications has a `seed` argument (`boot_seed` in `mlmr` and `mlmr_mv`) that defaults to `NULL`: the replications are then drawn from, and advance, the current random number stream. A supplied seed is set for the call only and the caller's generator state is restored when the function returns, so a reproducible call does not disturb the random numbers of the script around it. Two runs with the same seed and the same `B` give the same limits; the same seed with a different `B` does not.

Conditions

`dmar_count_invalid` Error (a `dmar_refusal`). `B` is not a single whole number of at least the number required. Fields `arg`, `min` (the number required), `max`, and `reason` (the sentence that explains the number).

`dmar_bootstrap_replications_dropped` Warning. Replications were dropped and the interval was formed from the rest. The message states how many, and says to request more when

those that remain are fewer than `B_required`. Fields `B`, `B_used`, `n_dropped`, `B_required`, `share_dropped`, `caution`, and `status`.

`dmar_bootstrap_insufficient` Warning. No interval is reported: more than half of the replications were dropped (`status` `"insufficient_share"`), or the adjusted positions of a BCa interval take more replications than were requested, or fewer than two replications were usable (`"insufficient_count"`). The same fields; for a BCa interval withheld at its adjusted positions also `positions` (the two positions, as proportions) and `B_suggest` (one and a half times the count those positions take, rounded up to a hundred).

`dmar_bootstrap_exploratory` Warning. The result was computed from fewer replications than a reported result needs, because `options(dmar.bootstrap_exploratory = TRUE)` is set. The same fields as the dropped warning.

`dmar_bootstrap_note` Warning. Something else about the run that the reader of the interval should know, with every replication used and the interval formed: in the reliability family and `ci_bifactor_indices`, a BCa interval replaced by the percentile interval (the `ci_method` attribute records the substitution), or replicates that all have the same value, so that the interval has no width; in `average_variance_extracted` and `parallel_analysis`, the function's account of a run in which nothing was dropped (replications from an improper solution that were kept, reference data sets whose fits gave warnings). The same fields as the dropped warning where the function records them.

`dmar_bca_undefined` Error (a `dmar_refusal`). The bias correction or the acceleration of a BCa interval is undefined (`mediate`, `cohen_kappa`, `fleiss_kappa`).

The warnings of the reliability family and of `ci_bifactor_indices`, which share one bootstrap engine, begin with the name of the function that was called (`"reliability_omega(): "`) and carry that name in the field `fun`; `reliability` passes its call on, so its warning names the function that ran the bootstrap. A program can muffle or read any of the warnings by class:

```
withCallingHandlers(
  mediate(d, x = "x", m = "m", y = "y", seed = 113),
  dmar_bootstrap_replications_dropped = function(w) {
    message(w$n_dropped, " of ", w$B, " dropped")
    invokeRestart("muffleWarning")
  })
```

The functions the rule governs

"Required" is the fewest replications a request may ask for at the 95 percent level (for the reference quantiles, at the 95th percentile). Kind Q reads quantiles, S takes a standard deviation of the replications, and P is a Monte Carlo *p*-value.

Function	Kind
`mediate`, percentile and Monte Carlo	Q
`mediate`, BCa	Q, BCa
`mediation_mbco`, `plot_mediation_mbco` (Monte Carlo)	Q
`reliability_alpha`, `reliability_kr20`, `reliability_omega`, `reliability`, `reliability_H`: percentile	Q
the same functions: BCa	Q, BCa
the same functions: `"bootstrap_se"`, `"bootstrap_se_logistic"`	S
`reliability_omega_categorical`	Q, BCa, or S

ci_bifactor_indices	Q or BCa
average_variance_extracted	Q
fraction_missing_information	Q
cohen_kappa, fleiss_kappa	Q or BCa
krippendorff_alpha	Q
htmt	Q, one-sided
ci_ratio_of_means, ci_ratio_of_means_paired	Q or BCa
R2_mixed_effects	Q
ci_eta_squared_generalized	Q
mlmr, mlmr_mv	Q
parallel_analysis, factor_count	Q, reference qu
sem_exact_fit_bootstrap	P

The planning functions ss_aipe_indirect_effect, ss_aipe_indirect_effect_sensitivity, and ss_aipe_reliability_sensitivity run an interval inside each simulated study; their pages state the inner B they use.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC. (For the percentile and BCa intervals themselves, as the function pages cite it. The counts on this page are not taken from it.)

See Also

dmar_metadata for reading the counts from a result, and the pages of the functions in the table.

---

dmar_metadata *Machine-Readable Metadata of a DMAR Result, and the Classes of Its Refusals*

Description

Returns, in one list with fixed names, what a program built on DMAR needs to read beside the numbers of a result table: which estimator, standard errors, test, and missing data method produced it; which version of a fit measure was reported and what corrected it; how many cases and groups it rests on and, after two-stage estimation, how many rows each stage read; which auxiliary variables entered; whether the result is fully eligible or qualified by an unavailable unrestricted (H1) model; and the counts of a bootstrap. Every DMAR table already carries these as attributes, but under names that differ by family; this accessor reads them into one stable vocabulary, so that an application reads dmar_metadata(x)$eligibility$status rather than the attribute of the day. A field a result does not have is NULL.

Usage

```
dmar_metadata(x)
```

Arguments

x	A result table of a DMAR function (a dmar_tbl).

Details

Fields:

result The table's own class, the first element of class(x) ("dmar_sem_fit", "dmar_reliability", "dmar_fa_confirmatory", and so on), and coefficient, the reliability coefficient a reliability table reports.

estimator, se_method, test The point estimator, the method of the standard errors, and the test statistic actually used, as the function recorded them: lavaan's names for a fitted model ("ML", "robust.huber.white", "satorra.bentler (scaled)"), the reliability family's names for its own ("analytic"). The fit measure families record lavaan's point estimator ("ML" for a fit requested with estimator = "MLR", whose robustness is in se_method and test); the information criteria record the estimator as requested ("MLR").

missing The missing data method as the function recorded it: lavaan's names for a fitted model ("listwise", "ml", "two.stage", "robust.two.stage") and DMAR's for its own arguments ("fiml", "two_stage", "robust_two_stage", "pairwise", "none" or "covariance input" for a supplied matrix).

ci_method, wald_df, conf_level, rmsea_conf_level, tail_areas, denominator How an interval was formed: the method, the degrees of freedom of a Wald interval (Inf for the normal), the level, the level of the RMSEA interval, unequal tail areas when recorded, and the denominator of an omega. rmsea_conf_level is the level the table records for its RMSEA interval apart from its other intervals, 0.90 by default (the fit rows of fa_confirmatory, cfa_1, cfa_2, fa_bifactor, and fa_second_order, whose parameter intervals are at conf_level, the fit line of fa_exploratory, and the fit measures of irt_grm), or else, for a table whose RMSEA interval is its own (sem_fit_measures, factor_count, ci_rmsea, equivalence_rmsea), the table's level; it is NULL for a table with no RMSEA interval.

interval_degenerate For a table of an order-based effect size (vargha_delaney_A, cliff_delta, probability_of_superiority_paired) whose estimated variance is 0, so that its two limits equal the estimate and are not a confidence interval: "complete_separation" or "no_variation". NULL for a table with an interval.

version, version_reason, correction For a fit measure table, the version reported ("standard", "scaled", "robust"), why, and the correction behind a robust version, from the sem_fit attribute (the information criteria of sem_information_criteria record under sem_information, read the same way).

N, groups, fit_rows The cases the result rests on, where the function records them (an attribute, or the N row of a reliability table), and the number of groups; after two-stage estimation, fit_rows is the record of the rows the first stage read and lavaan counted in each group, how many were observed on an auxiliary variable alone, and whether the fit was computed again with all of them (read, nobs, extra, corrected, which is NA when no case was observed on an auxiliary variable alone, and reestimated, whether that computation estimated a model that links the groups again).

`auxiliary` The auxiliary variables of a FIML or two-stage fit, where the function recorded them.

`eligibility` A list with `status`, `code`, and `note`: `"eligible"` when nothing qualifies the result (`code` and `note` then `NULL`), and `"qualified"` with a code and the text of the warning the function gave: `"h1_unavailable"` when the unrestricted (H1) likelihood was unavailable and the constrained model's estimates were retained with their H0-based uncertainty, and `"two_stage_rows_unavailable"` when a two-stage fit whose first stage read cases observed on an auxiliary variable alone could not be computed again with every case, so that its standard errors and the values that rest on lavaan's N are qualified (see *Refusals* for the cases where nothing is returned).

`bootstrap` For a bootstrap result, a list with `B` (attempted), `used`, `failed`, `improper`, and, where the family records them, the finer counts of the reliability engine (`returned`, `undetermined`, `not_converged`, `too_few_cases`), with `required`, the fewest replications a request may ask for, and `status`, one of `"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"` (see `dmar_bootstrap`).

`simulation` For the table of a Monte Carlo sensitivity function (the `ss_aipe_*_sensitivity()`, `ss_power_*_sensitivity()`, and `ss_seq_c_sensitivity()` functions), the record of its run: a list with `attempted` (the replications asked for), `returned` (those that produced a confidence interval, or a test), `failed` (those that did not), `denominator` (the replications the proportion, width, and coverage rows are taken over, which is `returned`), `width_target_probability`, and `width_target_mc_ci`, with any element the function adds (see the section on the record of the simulation on `ss_aipe_cv_sensitivity`). For the table of a standardized cluster randomized planner (`ss_aipe_crd_es`) that simulated a width, the record of the simulation of the design reported, with the same three counts, a `denominator` that is `attempted` for an assured width, and the elements that page lists (`failure_reasons`, `width_mc_ci`, and the totals of the search). `NULL` for any other table.

`transformation`, `methods`, `rotation`, `structure` Family specifics where they exist: the transformation record of `sem_exact_fit_bootstrap`, the score methods of `factor_scores` as a plain character vector, the rotation and the fitted structure (covariance or correlation) of `fa_exploratory`.

`footer` The lines a table of the fit families prints beneath itself, which state the version, the sample, and the qualifications in words. It is `NULL` for other tables, including one that prints a line of its own (the confidence set of `nnt_from_smd` is in that table's `nnt_confidence_set` attribute).

The names are part of DMAR's interface: a field is added when a new family needs one and is not renamed or removed within a release cycle without a deprecation.

Refusals: When DMAR will not run an analysis as asked, it stops with the reason and what to do instead. Every error DMAR signals on purpose, from the check of an argument to a fit that did not converge, is a classed condition: it carries, beside its own class, the class `dmar_refusal` and a `code` element with the name of its own class without the `dmar_` prefix, so the refusals can be caught together with `tryCatch(expr, dmar_refusal = function(e) ...)` or each by its own class. An error that is not a `dmar_refusal` was not signaled by DMAR's own checks (it came from R, from another package, or from `match.arg()` for a choice that is not among an argument's values). The classes:

`dmar_h1_unavailable` The unrestricted (H1) likelihood of a FIML or two-stage fit is unavailable (its affine-collapse diagnostic fired), so a fit measure, a two-stage criterion, or an EFA uncertainty that needs it is not computed; the condition carries the diagnostic as `diagnostic`. The

same name is the class of the warning a function gives when it retains the constrained model's estimates and qualifies them instead; `inherits(cnd, "error")` tells the refusal from the qualification.

`dmar_two_stage_rows_unavailable` A two-stage fit whose first stage read cases observed on an auxiliary variable alone could not be computed again with every case, so its fit measures are not reported; the condition carries the row counts as `rows`.

`dmar_inadmissible` A fitted model given to `sem_fit_measures`, `sem_residual_fit`, `sem_information_criteria`, `compare_sem`, `ecvi`, or `sem_exact_fit_bootstrap` has an inadmissible solution: **lavaan**'s post-estimation check, made on the model's own parameters when the fit has auxiliary variables, finds a negative variance estimate or a covariance matrix that is not positive definite. The condition's `reason` element names each failure once, in the order the check makes them: `"negative_variance"`, `"cov.lv"` (the covariance matrix of the latent variables), `"theta"` (that of the residuals of the observed variables), or `"psi"` (that of the residuals of the structural part of the model), the last three the names `lavaan::lavInspect()` gives the matrices; `variables` holds the variables with a negative variance estimate; `problems` is a data frame with a row per failure and the columns `reason`, `group` (the label of the group, as **lavaan** gives it; `NA` for a fit of one group), `level` (1 or 2 in a multilevel fit; `NA` for a fit of one level), `variable` (`NA` for a matrix), and `value` (the negative estimate, or the smallest eigenvalue of the matrix); and `auxiliary` names the auxiliary variables set aside (empty without them). The group here is a label, a character string, where the refusals of `sem_exact_fit_bootstrap` below carry the group's index, a whole number counting the groups in **lavaan**'s order.

`dmar_two_stage_fixed_x` The residual based statistic of a normal theory two-stage fit, which `sem_fit_measures`, `fa_confirmatory`, and `measurement_invariance` report, is refused for a fit that conditions on observed exogenous variables (`fixed.x = TRUE` or `conditional.x = TRUE`); the condition carries those variables as `variables`, and the message says how to refit (`fixed.x = FALSE`, with `conditional.x = FALSE` too for a fit made with `conditional.x = TRUE`) and on which rows to keep the fit's estimates.

`dmar_exact_fit_bootstrap_scope` A fit that `sem_exact_fit_bootstrap` does not take. The condition's `reason` element says why: `"summary_data"` (a fit to summary statistics), `"ordered"` (ordered indicators, named in `variables`), `"estimator"` (an estimator other than maximum likelihood, given as `estimator`), `"two_level"`, `"cluster"` (a cluster variable, given as `cluster`), `"weights"` (sampling weights), `"two_stage"` (two-stage estimation), `"fixed_x"` (a fit conditional on observed exogenous variables, named in `variables`), `"missing_without_fiml"` (missing values left in the analysis data of a fit not made by FIML, as under pairwise deletion, with the group's index as `group`), or `"no_test"` (no standard test statistic).

`dmar_pattern_too_small` A missing data pattern with no more rows than observed variables, which the pattern-wise transformation of `sem_exact_fit_bootstrap` cannot use. The condition carries the pattern's index among its group's patterns, numbered in the order of their first rows (`pattern`), its rows (`n`), the number and the names of the variables it observes (`p` and `observed`), and the group's index (`group`).

`dmar_transformation_undefined` The model based transformation of `sem_exact_fit_bootstrap` is undefined for a group's data or moments. The condition's `reason` element says why: `"not_positive_definite"` (a covariance matrix, named in `matrix`, is not positive definite), `"no_mean_structure"`, `"nonpositive_variance"` (a saturated variance is not positive), `"singular_pattern"` (the model implied covariance matrix of the variables a pattern observes, the pattern given as `pattern`, is singular), or, for the Newton iteration of the common transformation, `"not_converged"`, `"singular_jacobian"`, or `"diverged"` (with the

step as step); group is the group's index.

dmar_log_ratio_undefined ci_ratio_of_means or ci_ratio_of_means_paired was asked for the log ratio interval of a ratio whose numerator or denominator is not positive, where the logarithm is undefined.

dmar_bootstrap_needs_data One of the same two functions was asked for a bootstrap interval from summary statistics, which cannot be resampled.

dmar_items_invalid The items to analyze cannot be read from the input, whether they were named by items (in fa_exploratory, factor_count, parallel_analysis, measurement_invariance, and measurement_alignment), by the factors of fa_confirmatory and of the functions that fit through it (cfa_1, cfa_2, fa_bifactor, and fa_second_order), or by the model of measurement_invariance (a named list of factors or **lavaan** syntax) or of sem_sam. The condition's reason element says which check failed: "form" (not a character vector, or a name is NA), "unknown" (a name that is not a column of data or an item of S, the empty name included), "duplicate" (a name given twice), "count" (fewer than three items), or, with raw data, "data_names" (two columns of data share a name, or, with items = NULL, which makes every column an item, a column's name is empty or NA); and items holds the names at issue, empty when the failure is not about particular names. A model's structure is checked first and without this class (too few items on a factor, an item named twice among the factors), so from fa_confirmatory, the functions that fit through it, and the model of measurement_invariance the reason is "unknown", or "form" for a name that is NA, and from sem_sam "unknown" or "data_names"; the items of measurement_invariance and measurement_alignment can give any reason but "data_names". A target naming items the input lacks is dmar_target_invalid.

dmar_htmt_blocks_invalid The blocks of htmt are not disjoint sets of distinct items under distinct names. The condition's reason element says which check failed: "construct_names" (a construct name given to two or more blocks), "repeated" (an item listed more than once in one block), "shared" (an item in two or more blocks), or "data_names" (an item whose name belongs to two or more columns of data); blocks names the blocks and items the items at issue (empty for "construct_names"). The refusal is made before any correlation is computed or any resample drawn. A block that is not two or more columns of data is dmar_argument_invalid.

dmar_target_invalid The target of a target rotation in fa_exploratory cannot be read as one (or a target was given with another rotation, or none with a target rotation); the condition's reason element says which check failed: "rotation", "form", "names", "factors", "values", "none_specified", or "nfactors".

dmar_target_unidentified The target does not identify the rotation: reason is "counts" (too few specified loadings in a column), "duplicate" (two columns specified alike), or "rank" (the unrotated loadings of a column's specified items have too low a rank), and factors names the columns at issue.

dmar_group_coverage Under full information maximum likelihood, a group of measurement_invariance never observes a variable of the model (or an auxiliary variable), or never observes two of them together, so the ladder cannot be fitted; the condition carries the variables a group never observes as unobserved (a data frame with columns group, variable, and n_group, the cases of the group) and the pairs as pairs (columns group, variable_1, and variable_2).

dmar_bifactor_unidentified A bifactor structure that fa_bifactor will not fit; the condition carries the factors and items at issue as factors and items, and the kind of refusal as reason: "unidentified" (the data cannot identify the structure), "names" (an item or factor placed

twice, or names that clash), or "scope" (a structure outside the model the function fits, such as fewer than two group factors, which need not be unidentified).

dmar_second_order_unidentified A second-order structure that fa_second_order will not fit; the condition carries the factors and items at issue as factors and items, and the kind of refusal as reason, with the values of dmar_bifactor_unidentified ("names" also for a first-order factor under no second-order factor, "scope" for an item on two first-order factors or a first-order factor under two second-order factors).

dmar_count_invalid A number of resamples, permutations, or replications that is not a single whole number in its admissible range (2.5, a missing or infinite value, a vector, or a value below the stated minimum). The condition carries the argument's name as arg and its limits as min and max (NULL when no maximum is stated).

dmar_semipartial_r_incoherent A full-model R^2 smaller than the square of the semipartial correlation it is given with, which no data can produce, in var_semipartial_r, ss_aipe_semipartial_r, and ss_aipe_semipartial_r_sensitivity. The condition carries the two values as R2_full and r_sp.

dmar_bca_undefined mediate was asked for the BCa interval of an indirect effect in data where the interval is not defined: the indirect effect is not estimable with one of the cases left out, so the jackknife acceleration has no value, or the bias correction or the acceleration is not finite. The percentile interval does not need these quantities.

dmar_scores_not_ordered The observations given to an order-based effect size (vargha_delaney_A, cliff_delta, probability_of_superiority_paired) have no common order: an unordered factor, a character vector, an ordered factor beside anything but an ordered factor with the same levels, time differences in different units, a Date beside a POSIXct, or a class or shape that is not a numeric vector. The message says which case applies, what would have been computed, and what to supply.

dmar_studentized_statistic_undefined statistic = "t" in randomization_test or randomization_test_paire for data whose observed standard error is 0 with a nonzero mean difference (two constant groups at different values; paired differences that are all the same nonzero value), where the studentized statistic has no value. The condition carries reason ("constant_groups" or "constant_differences"), alternative, the group sizes as n_1 and n_2 or the number of pairs as n, and p_value_mean, the exact p-value that statistic = "mean" gives.

dmar_hedges_correction_undefined unbiased = TRUE in smd_c with a control group of 1 or 2, or in smd with $n_1 + n_2$ of 2 or 3. On 1 degree of freedom the correction factor of Hedges (1981) is 0, so the corrected estimate would be 0 whatever the data; on 0 degrees of freedom the factor is not defined, because there is no standard deviation. The condition carries df.

dmar_assurance_invalid An assurance given to ss_aipe_cv that is neither NULL nor a single number strictly between 0 and 1. The condition carries the argument's name as arg and the value given as value.

dmar_argument_invalid A value, or a combination of values, that a function cannot use and that has no class of its own: the wrong type or length, a number outside its range, arguments that cannot be given together or that are missing together, a column that is not in the data, data or a fitted model the analysis does not cover. It is the class of every argument check in the package. The condition carries the name or names of the arguments at issue as arg. arg is NULL when the refusal is not about a named argument (a property of the data, a search that ran out of room), and in some checks whose message names the argument only in running text and not at its start. Among others it is given by ss_power_R2_change (a p_restricted that is not below p_full, squared multiple correlations that cannot belong to nested models, an

effect stated in more than one way); by the partial and semipartial correlation functions for a p below 2; by smd and smd_c for unbiased = TRUE without the sample sizes, and by smd for a group of one with its own standard deviation; by ss_aipe_sc_ancova_sensitivity when a sample size is to be planned and no width is given; and by ss_aipe_R2_sensitivity, ss_aipe_cv_sensitivity, ss_aipe_cliff_delta_sensitivity, and ss_aipe_reliability_sensitivity when both the estimate to plan from and specified_N are given.

dmar_package_required The analysis asked for needs a package that is not installed (or, for **lavaan**, a newer version than the one installed). The message names the package and how to install it, and the condition carries its name as package.

dmar_computation_failed The arguments were accepted, and a computation made with them did not give a result: a search for a sample size or for a confidence limit did not reach its target, a model fitted for the analysis failed or gave no proper solution and no more specific class applies (dmar_not_converged, dmar_no_maximum, dmar_lavaan_error), a refit or a bootstrap failed, or a distribution could not be evaluated accurately at the values reached. The message says which and what to change.

dmar_internal_error DMAR reached a state its own code should not reach. The arguments are not at fault; please report the call (see BugReports in the package's description).

dmar_desired_power_invalid A desired_power that is not a single number strictly between 0 and 1, in any power planner (the ss_power_ functions). The condition carries the argument's name as arg and the value given as value.

dmar_effect_zero A sample size was asked of ss_power_R2 or ss_power_R2_change for a planning value under which the null hypothesis is true (a squared multiple correlation, or a change in it, of 0), so that power equals alpha_level at every sample size; or of ss_power_reg_coef or ss_power_rc for a regression coefficient whose effect is 0 at the planning values, with a desired_power above alpha_level.

dmar_estimand_undetermined An estimand asked for by name (estimand = "total" or "partial") of eta_squared, omega_squared, ci_eta_squared, or ci_omega_squared that the inputs do not determine, such as the total value from an F statistic, or a total for which no interval is formed. With estimand = "all" the same estimand is reported as NA. The condition carries estimand and the reason.

dmar_cost_ratio_invalid A cost_ratio given to ss_aipe_two_phase_cost that is not a single positive number below 1. The condition carries the argument's name as arg and the value given as value.

dmar_two_phase_no_design Planning values for which ss_aipe_two_phase_cost cannot represent the sizes of a design as whole numbers (a width too narrow for the standard deviation, or a proxy too inexpensive).

dmar_no_maximum A factor analysis or structural equation model fit whose likelihood has no maximum: an error variance ran off below zero, and with it held further out (at ten times its estimate, or one model implied variance below it, whichever is further) the log likelihood is higher, or lower by no more than 10^{-6} per case, so the likelihood is still rising or flat along that ridge and **lavaan**'s optimizer stopped at a point its starting values and tolerance set; the estimates describe the optimizer rather than the data. It comes from fa_confirmatory (and cfa_k(), cfa_1, and cfa_2), fa_bifactor, fa_second_order, fa_exploratory (where it is also a dmar_efa_no_solution, with reason "no maximum"), measurement_invariance, measurement_alignment, the single-factor fits of reliability_omega and reliability_H, and the refits of ci_bifactor_indices and of reliability_H(fit =), for every output. It comes as well from the fit of the analysis model to the population moments in ss_aipe_composite_sem,

ss_power_composite_sem, ss_aipe_sem_path, and ss_aipe_sem_path_sensitivity, and from the **lavaan** fit that the **OpenMx** fits of mediation_mbco start from (DMAR's own, or one given as model). The condition carries the variables as variables (with their groups as group), their error variances where the optimizer stopped and where the check held them as estimate and fixed_at, the two log likelihoods as loglik, the check's tolerance and the number of cases as tolerance and N, and the **lavaan** code of the fit as syntax and lavaan_call, which the message prints (leaving the syntax out when it is too long to print, for R cuts an error message at 8170 bytes). Inside a bootstrap, such a replication is counted among those that raised an error, and the run goes on; in the Monte Carlo of a planner it is set aside as a replication that did not converge. A function that reads only the chi square of a fit, which does not depend on where on such a ridge the optimizer stopped, does not check it: ss_aipe_rmsea_sensitivity and sem_exact_fit_bootstrap.

dmar_null_R2_unsupported ss_power_R2 was given a nonzero null_R2. The function plans the omnibus test only; the message gives the call to ss_power_R2_change for the test of an increment. The condition carries null_R2, true_R2, true_f2, and p.

The other refusals, each a dmar_refusal as well, are dmar_fiml_unbounded and dmar_fiml_unstable (the unrestricted FIML likelihood has no finite maximum for these data, or is numerically unstable; the condition carries the certificate as diagnostic), dmar_efa_no_solution (an exploratory fit that did not converge, is improper, or left its standard errors unavailable, as its reason element says; with the reason "no maximum" it is also a dmar_no_maximum), dmar_not_converged (a fit that did not converge, from fa_confirmatory, from measurement_invariance, where it carries the rung as level and that rung's **lavaan** code as syntax and lavaan_call, and from the functions that share the check of sem_fit_measures, those under dmar_inadmissible, and from any other function that fits a model, or is given a fitted one, and reads its estimates; the refusal of a fit given to average_variance_extracted, bifactor_indices, ci_bifactor_indices, or dmacs carries the function as fn and the number of iterations **lavaan** ran as iterations), dmar_lavaan_error, dmar_too_few_cases, dmar_htmt_undefined, dmar_H_improper (an improper solution given to reliability_H as a fit), dmar_fmi_saturated_singular, and dmar_fmi_not_inverted.

A warning is a qualification of a result that is returned. Every warning DMAR signals on purpose is classed and carries its code, so it can be muffled or read by class from withCallingHandlers(). Two classes are general: dmar_result_qualified (the result is returned and the message states what to know before using it: an approximation that was used, an assumption the data do not meet, a quantity reported as NA, cases or replications that were dropped) and dmar_argument_ignored (an argument that was given has no effect on this call, and the message says why). The others name one kind of qualification: dmar_fmi_not_at_maximum, dmar_heywood_warning (an improper solution: a negative error variance or a standardized loading beyond one), dmar_H_marginal_se (an interval for coefficient H from the marginal standard errors of the loadings alone), dmar_nc_F_clamp (a lower confidence limit of a noncentrality parameter set to 0; see ci_nc_F), dmar_two_stage_aux_only (cases observed on an auxiliary variable alone, which **lavaan** leaves out of its count), dmar_negative_reliability, dmar_two_stage_n, dmar_semipartial_r_no_R2_full, dmar_semipartial_r_rounded (a full-model R^2 below the squared semipartial correlation by no more than rounding to two decimals allows; the condition carries R2_full and r_sp), dmar_nc_F_clamp_search (the sample size search of ss_aipe_R2 met the lower-limit clamp of ci_nc_F at intermediate sample sizes; given once per call, with the count as n_evaluations), dmar_replications_without_interval, dmar_replications_with_infinite_limit (replications of ss_aipe_cv_sensitivity whose interval has an infinite upper limit; given once per call, with the count as n_replications), dmar_uls_standard_statistic, dmar_bifactor_empirical, dmar_smd_undefined, dmar_meta_hk_zero_se (identical effect sizes under the Hartung-Knapp adjustment; the condition carries n_studies,

estimate, se_conventional, lower_conventional, and upper_conventional), dmar_interval_degenerate (an order-based effect size whose estimated variance is 0, with reason "complete_separation" or "no_variation", the estimate, the sample sizes n, and limit_feng_cliff, the limit Feng and Cliff (2004) give for complete separation, which the function does not report), dmar_bootstrap_replications_dropp (a bootstrap or Monte Carlo interval dropped replications in which the statistic was undefined; the condition carries B, B_used, n_dropped, B_required, share_dropped, caution, and status), dmar_bootstrap_insufficient (more than half of the replications were dropped, or the adjusted positions of a BCa interval take more replications than were requested, and the limits are NA; the same fields, and for a BCa interval withheld at its adjusted positions also positions and B_suggest), dmar_bootstrap_exploratory (the result was computed from fewer replications than a reported result needs because options(dmar.bootstrap_exploratory = TRUE) is set; the same fields), and dmar_bootstrap_note (a BCa interval replaced by the percentile interval, or replicates that all have the same value, with every replication used), all four described in dmar_bootstrap. Each function's help says when.

A refusal is never a table: when a function returns a table, its result is eligible or qualified, never refused.

The errors dmar_bootstrap_cancelled and dmar_progress_error are not refusals. A bootstrap of the reliability family or of ci_bifactor_indices stops with the first when its progress callback returns FALSE, at the request of the application that called it, and with the second when the callback itself raises an error. Both carry done and total, the replications attempted and the number requested, and the second keeps the callback's own condition, with its class, as parent, so that an application can tell a failing callback from a failing analysis and still recognize a condition of its own behind it.

Value

A named list with the fields above; a field the result does not carry is NULL.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Feng, D., & Cliff, N. (2004). Monte Carlo evaluation of ordinal d with improved confidence interval. *Journal of Modern Applied Statistical Methods, 3*(2), 322–332. doi:10.22237/jmasm/1099267560

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128. doi:10.3102/10769986006002107

See Also

dmar_tbl for the table class and its display, dmar_tidiers for tidy() and glance(), which put the same metadata into columns of a data frame.

Examples

```
x <- reliability_alpha(holzinger_swineford[c("t1_visual_perception",
                                             "t2_cubes", "t4_lozenges")])
```

```
m <- dmar_metadata(x)
m$coefficient
m$ci_method
m$eligibility$status

# Four items, so that the model has degrees of freedom to test (three
# would make it saturated, which sem_fit_measures() qualifies).
fit <- lavaan::cfa(
  "visual =~ t1_visual_perception + t2_cubes + t3_paper_form_board +
              t4_lozenges",
  data = holzinger_swineford, std.lv = TRUE, estimator = "MLR")
f <- dmar_metadata(sem_fit_measures(fit))
f$version
f$correction
```

`dmar_output_helpers` *Publication-Ready Display of DMAR Result Tables*

Description

The `dmar_tbl` print layer formats a result table for the console. These helpers carry the same formatting into the two places a researcher writes up an analysis: a knitted report and a results sentence.

Usage

```
## S3 method for class 'dmar_tbl'
knit_print(x, ...)

as_kable(x, ...)

## S3 method for class 'dmar_tbl'
as_kable(x, format = NULL, ...)

results_sentence(x, label = NULL, digits = 2, term = NULL, conf_level = NULL)
```

Arguments

`x`	A `dmar_tbl` object (a `data.frame` returned by a DMAR function). For `results_sentence`, a `dmar_tbl` that carries a confidence interval, either as `ci_lower` / `ci_upper` rows of a long table or as `ci_lower` / `ci_upper` columns of a one-row wide table, such as one row of `common_method_adjusted_r` or of a `tidy()` result, or as `<estimate>_ci_lower` / `<estimate>_ci_upper` rows beside the estimate's row, as in a `sem_fit_measures` table, or as `<estimate>_ci_lower` / `<estimate>_ci_upper` columns beside the estimate's column of a one-row wide table, as in one row of `ci_eta_squared` or `ci_omega_squared`.

`...` Additional arguments. For `knit_print.dmar_tbl` and `as_kable.dmar_tbl`, passed to `kable` (other than `caption`, which `as_kable` writes itself).

`format` Passed to `kable` as its `format` argument (for example `"html"`, `"latex"`, `"pipe"`). The default `NULL` lets **knitr** choose based on the output context.

`label` For `results_sentence`, the label that leads the sentence (for example `"Cohen's d"`). The default `NULL` uses the name of the point-estimate term: the estimate's row of a long table, the quantity a `tidy()` row names in its `estimand` column (see `dmar_tidiers`), the row's `term` beside a wide table's `estimate` or `value` column, and otherwise the estimate column's name.

`digits` For `results_sentence`, the number of decimal places for the estimate and the interval limits. Default 2. Like the `digits` of the table display (see `dmar_tbl`), it counts decimal places, but it keeps a default of its own and does not follow `options(dmar.digits = )`, so a sentence written for a manuscript does not change with the console display. As in the table, a value that rounds to zero is written without a minus sign (0.00, not -0.00).

`term` For `results_sentence`, the estimate whose interval a table holds in `<term>_ci_lower` and `<term>_ci_upper` rows, for example `"gamma_hat"` in a `sem_fit_measures` table or `"rmsea"` in a `fa_confirmatory` table, or in columns of those names, for example `"eta_squared_total"` in a row of `ci_eta_squared`. The default `NULL` reads the table's interval as described above, taking the first estimate with such rows when the table holds its interval in no other form, and, of such columns, the one estimate whose limits are not `NA` (a row with several is refused unless `term` names one). Either way the interval for the standardized mean difference that `cles` and `nnt_from_smd` echo beside their own, which theirs are computed from, is taken only when `term` is `"smd"`.

`conf_level` For `results_sentence`, the confidence level of the table's interval, for a table that records none (one assembled by hand, say). The default `NULL` reads the level the table records and refuses a table that records none, rather than assuming one; a value that differs from the level the table records is refused.

Details

`knit_print.dmar_tbl` is the **knitr** print method, so a `dmar_tbl` dropped into an R Markdown chunk renders as a formatted `kable` (values to a fixed number of decimal places, whole-number sample sizes without decimals, *p*-values to fixed decimals) rather than as unrounded doubles. `as_kable` is the explicit form of the same rendering: it returns the `knitr_kable` object so the caller can pipe it into further styling or embed it in a larger document. Both show the table the console shows, formatted by `format.dmar_tbl` (for most results the whole table; for a few, such as `fraction_missing_information`, which leaves out the columns that carry nothing for the call, and `factor_scores`, which shows the first rows of each factor, the same view the console prints), together with every line the console writes with it: the sum-of-squares type, the confidence level, how an omega's denominator was estimated and its largest residual correlation, the bootstrap replication counts, the missing data a sample size plan assumed, and the lines a result of its own adds (which version of a fit measure was reported and why, what a model comparison checked, how a fraction of missing information was estimated, which rule set a table of degrees of freedom). What a reader sees in a report therefore matches what they saw at the console, and neither rounds the stored numbers. `knit_print.dmar_tbl` sets each of those lines as its own paragraph beneath the table (a line that introduces the table, as the summary

of factor_scores, the measurement structure of fa_confirmatory, and the model summary of mediation_mbco do, comes before it, and a small table among the lines is set as a code block so its columns stay aligned), and sets the whole off with a blank line before and after it, so that two tables printed in one chunk stay apart; as_kable joins them into the table's caption. Both write those lines in the markup of the format **knitr** renders the table in, escaped for LaTeX (a document knitted from .Rnw, or format = "latex") and for HTML, so that the percent sign of a confidence level or an underscore in an item name prints as itself; **knitr** escapes a table's cells and column names but writes a caption as it is given.

results_sentence turns a table that carries a confidence interval into the one sentence an author puts in a results section, for example "smd = 0.50, 95% CI [0.10, 0.90]". It reads the estimate and its limits from the numeric columns at full precision and formats them for the sentence, so the reported numbers are exact to the requested decimals rather than transcribed from the rounded console display. In a wide table the estimate is the column the row names in an estimand column, as the glance() row of a long table names the column of the estimate its interval belongs to (see dmar_tidiers), else the estimate (or value) column when there is one, and otherwise the first numeric column that is not a limit, a standard error, a test statistic with its degrees of freedom and *p*-value, or a count; a *p*-value the table carries is appended, to the four decimals of format_p, and one below that floor as the bound it is ("p < 0.0001"). In a long table, whose interval is held in ci_lower and ci_upper rows, the estimate is the row the table names as the one the interval belongs to, when it names one (the corrected correlation of correction_for_attenuation, whose table lists the observed correlation first, and the proportion of superiority of proportion_of_superiority, listed after the standardized mean difference it is computed from), and otherwise its first row that is not a limit. Its *p*-value is likewise the row the table names as the *p*-value of its test, when it names one, and otherwise its first *p*-value row: the equivalence tables of equivalence_r, equivalence_smd, and equivalence_c list the *p*-values of the two one-sided tests first and name the TOST *p*-value, the test their $1 - 2\alpha$ interval decides. A long table may instead hold an estimate's interval in rows of its own, named for the estimate: sem_fit_measures reports rmsea beside rmsea_ci_lower and rmsea_ci_upper, and gamma_hat and adjusted_gamma_hat the same way. The sentence then reports the estimate named by term, or the first such estimate in the table (the RMSEA there), with no *p*-value, since the *p*-values of such a table belong to other tests (of exact, close, and not close fit). term also reaches the RMSEA rows of a fa_confirmatory table. The confidence level is the interval's own level when the table records it (fa_confirmatory() records the level of its RMSEA interval, its rmsea_conf_level, in the rmsea_ci_level row and in an attribute of the same name, which is read when a selection of rows leaves that row out; a table stacked with rbind() keeps the attributes of its first table only, so keep the rmsea_ci_level row in each selection when stacking tables fit at different levels), and otherwise the table's conf_level attribute, or its conf_level row or column (as the reliability coefficients and the tidy() and glance() verbs carry it). The tidy() of a long table holds the level in a conf_level column on the rows of its interval, the estimate and the two limits, and NA on the others; the sentence reads the level and the estimate from those rows, so that rows of several tidy() results stacked with rbind() give the sentence of their own table (a frame's attributes are those of the first frame of a stack). Every DMAR table that forms an interval records its level, and a sentence states no level the table does not record: a table that records none, such as one assembled by hand, is refused unless conf_level gives the level its interval was formed at, and a conf_level that differs from the level the table records is refused.

A table whose two limits are not a confidence interval gets a sentence that says so. vargha_delaney_A, cliff_delta, and probability_of_superiority_paired report both limits equal to the estimate when the estimated variance is 0 (completely separated groups, or no variation) and record the case in the table's interval_degenerate attribute. The sentence then gives the estimate and the rea-

son and no interval, "A = 1.00 (no confidence interval: the groups are completely separated, so the estimated variance is 0)" or "A = 0.50 (no confidence interval: every score is equal, so the estimated variance is 0)", with "every pair favors the same member" and "every pair is tied" for the paired function. A *p*-value is appended only when the table has one (the permutation test of `vargha_delaney_A`). The `tidy()` and `glance()` rows of these tables carry the mark in a logical column `interval_degenerate`, which says that the variance is 0 and not which case gave it, so their sentence reads "A = 1.00 (no confidence interval: the estimated variance is 0)".

The level a sentence states is the interval's coverage, which for an interval formed from tail areas `alpha_lower` and `alpha_upper` is $1 - \alpha_{lower} - \alpha_{upper}$ whatever the split. A sentence in the form "95% CI [L, U]" is read as the equal-tailed interval, so an interval whose tail areas differ says so, "smd = 0.50, 95% CI [0.03, 0.85] (unequal tail areas: 0.01 below, 0.04 above)", and an interval with a tail area of 0 is written as the one-sided confidence bound it is, "smd = 0.50, one-sided 95% upper confidence bound 0.83", since its other limit is infinite or the bound of the parameter (0 for an RMSEA or a squared multiple correlation) and carries no information. The split is read from the table's record of it, never from a limit: an equal-tailed interval whose lower limit is 0 because the statistic fell below its critical value (Steiger, 2004) is written as the two-sided interval it is. The line beneath a printed table states the same split, for example "Confidence level: 95% (unequal tail areas: 0.01 below the lower limit, 0.04 above the upper limit)", and the `tidy()` and `glance()` rows of such an interval carry it in `alpha_lower` and `alpha_upper` columns (see `dmar_tidiers`).

A confidence set that is not an interval is written as what it is. The Fieller set of `ci_ratio_of_means` and `ci_ratio_of_means_paired` can be the complement of an interval or the whole line, and with unequal tail areas an interval unbounded on one side or the union of two intervals. The sentence reads the form from the table's `confidence_set` attribute (or the `confidence_set` column of its `tidy()` row) and the ends of the excluded gap from its `excluded_lower` and `excluded_upper` rows (or columns), and writes, for example, "ratio = 0.32, 99.5% confidence set: every value except those between 0.90 and 147.05", "ratio = 0.32, 99.9% confidence set: the whole line, which excludes no value", or "ratio = 0.32, 97.8% confidence set: the union of [-0.61, 0.93] and [9.55, Inf) (unequal tail areas: 0.02 below, 0.002 above)", since "99.5% CI [-Inf, Inf]" would read as the whole line and hide the gap. An infinite end is written open and a finite one closed. A bounded Fieller interval is written as any other interval.

A table whose limits are `NA`, because no interval was formed (as from `krippendorff_alpha` without a bootstrap), has no sentence to write and gives an error.

Value

`knit_print.dmar_tbl` returns a `knit_asis` object (the rendered table) for **knitr** to place in the document. `as_kable` returns a `knitr_kable` object. `results_sentence` returns a length-one character string.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

See Also

dmar_tbl for the console print layer and format_p for the *p*-value convention these helpers reuse.

Examples

```
x <- ci_smd(smd = 0.5, n_1 = 50, n_2 = 50)

# A knitr_kable that keeps every column, ready for a report.
as_kable(x)

# The sentence an author writes in a results section.
results_sentence(x, label = "Cohen's d")

# Wide tables (one interval per row) work the same way.
results_sentence(ci_R2(R2 = 0.25, N = 100, p = 5))

# So does one row of a tidy() result, here the same interval.
results_sentence(generics::tidy(x))

# A table that holds an interval in rows named for its estimate: the
# RMSEA of a model with chi square 85.3 on 24 df and N = 301, with its
# 90 percent interval (the default level of every RMSEA interval), and
# then gamma hat.
fm <- sem_fit_measures(chisq = 85.3, df = 24, N = 301, p = 9)
results_sentence(fm, label = "RMSEA", digits = 3)
results_sentence(fm, term = "gamma_hat", digits = 3)
```

---

drinks_trial *Community Reinforcement Approach Drinking Trial With Homeless Alcohol-Dependent Individuals (Smith, Meyers, & Delaney, 1998)*

Description

Nine-month follow-up drinking outcomes for the $N = 88$ homeless alcohol-dependent participants in Smith, Meyers, and Delaney's (1998) randomized clinical trial of the Community Reinforcement Approach (CRA), published in the *Journal of Consulting and Clinical Psychology*. Participants were recruited from the Salvation Army Adult Rehabilitation Center in Albuquerque, New Mexico across two consecutive cohorts and were randomized to CRA, to CRA augmented with disulfiram, or to standard care. The outcome is the participant's average number of standard drinks per week at the nine-month follow-up, reported in both raw form (markedly right-skewed) and after a base-ten log transformation that approximately normalizes the distribution used in the original published analyses. The data are reproduced in Maxwell, Delaney, and Kelley (2027, *Designing Experiments and Analyzing Data: A Model Comparison Perspective*, 4th ed., Routledge), Chapter 3, Section 3.10.4, as the textbook's worked example of a between-subjects analysis of variance with a heavily skewed outcome.

Usage

```
drinks_trial
```

Format

A data frame with 88 observations on 5 variables.

`id` Sequential participant identifier, 1 to 88.

`cohort` Factor with levels 1 and 2. The study enrolled two consecutive cohorts. Cohort 1 compared three conditions (Standard, CRA, and CRA + Disulfiram); Cohort 2 dropped the disulfiram cell on the basis of Cohort 1 results and compared Standard against CRA only.

`treatment` Factor with levels `Standard`, `CRA`, and `CRA + Disulfiram`, the randomly assigned treatment condition.

`drinks_per_week` Average number of standard drinks per week at the nine-month follow-up. Bounded below at zero and markedly right-skewed (range 0 to 624.6, mean 36.9, median 3.8).

`log_drinks` Common-log transformation $\log_{10}(x + 1)$ of $x =$ `drinks_per_week`, the scale on which Smith, Meyers, and Delaney (1998) ran their primary between-groups analyses to obtain approximate normality. The plus-one inside the logarithm keeps the zero values finite (and mapped to zero).

Details

Per-cell sample sizes. The (`cohort`, `treatment`) crosstab is incomplete by design:

	Standard	CRA	CRA + Disulfiram
Cohort 1	17	15	19
Cohort 2	20	17	(not run)

Treatment marginals sum to 37 Standard, 32 CRA, and 19 CRA + Disulfiram; cohort marginals sum to 51 in Cohort 1 and 37 in Cohort 2.

The authors, the published article, and the textbook. The trial was conducted and reported by Jane Ellen Smith, Robert J. Meyers, and Harold D. Delaney, all then in the Department of Psychology at the University of New Mexico. Smith and Meyers were the substantive PIs of an extensive program of CRA research; Meyers (with N. H. Azrin) is widely associated with the dissemination of CRA and is the developer of the related Community Reinforcement and Family Training (CRAFT) intervention. Delaney is a quantitative psychologist who served as the trial's methodologist. In DMAR the data are included as a worked-example benchmark because they appear in Maxwell, Delaney, and Kelley (2027), Chapter 3, Section 3.10.4, as the running example for the consequences of skipping a normalizing transformation when fitting an analysis of variance to a markedly skewed outcome.

The original published article is

- Smith, J. E., Meyers, R. J., and Delaney, H. D. (1998). The community reinforcement approach with homeless alcohol-dependent individuals. *Journal of Consulting and Clinical Psychology, 66*(3), 541–548. doi:10.1037/0022006X.66.3.541

and the textbook reproduction (with Section 3.10.4 of Maxwell, Delaney, and Kelley, 2027 devoted to the worked analysis) is

- Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

The Community Reinforcement Approach (CRA). CRA is a behavioral treatment for alcohol-use disorder developed by Nathan Azrin and colleagues in the 1970s. It uses operant conditioning principles to align vocational, marital, social, and recreational reinforcers in the patient's natural environment with abstinence rather than drinking. CRA had previously been evaluated in housed populations; Smith, Meyers, and Delaney (1998) extended its evidence base to homeless alcohol-dependent individuals, the population represented in these data.

Disulfiram (Antabuse). The CRA + Disulfiram condition added disulfiram, a long-standing pharmacological adjunct that produces an aversive reaction on alcohol consumption. The Cohort 1 finding that adding disulfiram offered little incremental benefit over CRA alone motivated dropping the disulfiram cell in Cohort 2.

Design and analysis. The published analyses fit a one-way analysis of variance to `log_drinks` (the raw `drinks_per_week` variable violates the normality assumption badly enough that the omnibus inference is misleading without transformation; see Maxwell, Delaney, and Kelley, 2027, Section 3.10.4). Useful contrasts include CRA versus Standard within each cohort, CRA versus CRA + Disulfiram within Cohort 1 to estimate the incremental benefit of disulfiram, and pooling across cohorts to estimate an overall CRA-versus-Standard effect.

Use as a DMAR benchmark. This data set is the canonical DMAR example for one-way between-subjects analysis with a heavily right-skewed continuous outcome. Methods that pair naturally with the data set include the one-way analysis of variance, planned contrasts (`contrast_test`), the standardized mean difference (`smd`, `ci_smd`), Cliff's delta (`cliff_delta`), the probability of superiority, and accuracy in parameter estimation sample size planning (`ss_aipe_smd`, `ss_power_smd`). The contrast between an ANOVA fit to `drinks_per_week` and one fit to `log_drinks` is a clean teaching example for the consequences of skipping a normalizing transformation.

What is and is not here. The published paper reports drinking outcomes at multiple follow-up timepoints (2, 6, 9, and 12 months) along with several demographic and clinical covariates. The values distributed here cover the nine-month follow-up only and contain no demographic or covariate information. Anyone wanting the full longitudinal trajectories or the covariate set should consult Smith, Meyers, and Delaney (1998) directly.

Author(s)

Ken Kelley

Source

Smith, J. E., Meyers, R. J., and Delaney, H. D. (1998). The community reinforcement approach with homeless alcohol-dependent individuals. *Journal of Consulting and Clinical Psychology, 66*(3), 541–548. doi:10.1037/0022006X.66.3.541

Also reproduced in Maxwell, Delaney, and Kelley (2027), Chapter 3.

The data are distributed openly with the companion materials of Maxwell, Delaney, and Kelley (2027), *Designing Experiments and Analyzing Data: A Model Comparison Perspective*, and are redistributed here on that basis.

References

Smith, J. E., Meyers, R. J., and Delaney, H. D. (1998). The community reinforcement approach with homeless alcohol-dependent individuals. *Journal of Consulting and Clinical Psychology, 66*(3), 541–548. doi:10.1037/0022006X.66.3.541

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on between-subjects analysis of variance and Section 3.10 on transformations.)

Kelley, K. (2026). *DMAR: Methods for the design, measurement, and analysis of human-centered outcomes in R* [R package]. https://github.com/yelleKneK/DMAR

Examples

```
data(drinks_trial)
str(drinks_trial)

# Per-cell sample sizes by cohort and treatment.
with(drinks_trial, table(cohort, treatment))

# Right skew of the raw outcome versus the log-transformed scale
# used in the published analyses.
summary(drinks_trial$drinks_per_week)
summary(drinks_trial$log_drinks)

# One-way analysis of variance on the log scale, treating the
# five filled (cohort, treatment) cells as the design.
fit <- aov(log_drinks ~ cohort:treatment, data = drinks_trial)
summary(fit)

# Standardized mean difference (Cohen's d) for every pairwise
# (two-way) comparison of the three treatments, on the normalizing
# log scale. The loop assembles a table of d with its 95%
# confidence interval for each of the three possible pairs.
groups <- split(drinks_trial$log_drinks, drinks_trial$treatment)
cmp    <- combn(names(groups), 2)
smd_table <- do.call(rbind, lapply(seq_len(ncol(cmp)), function(j) {
  g1 <- groups[[cmp[1, j]]]
  g2 <- groups[[cmp[2, j]]]
  d  <- smd(group_1 = g1, group_2 = g2)$value
  ci <- ci_smd(smd = d, n_1 = length(g1), n_2 = length(g2))
  data.frame(
    comparison = paste(cmp[1, j], "vs", cmp[2, j]),
    d          = round(d, 3),
    ci_lower   = round(ci$value[ci$term == "ci_lower"], 3),
    ci_upper   = round(ci$value[ci$term == "ci_upper"], 3)
  )
}))
smd_table
```

`dunn_test` *Provides Dunn's Rank-Sum Test of All Pairwise Differences Following a Kruskal–Wallis Test*

Description

Provides Dunn's Rank-Sum Test of All Pairwise Differences Following a Kruskal–Wallis Test

Usage

```
dunn_test(x, group = NULL, method = "holm")
```

Arguments

`x`	Either (a) a numeric vector of the outcome, in which case `group` must also be supplied, or (b) a fitted `lm` or `aov` object with a single factor predictor, from which the outcome and grouping factor are taken (the model itself is not used, since the test is distribution free).
`group`	When `x` is a vector, a factor (or coercible to factor) giving the group membership of each observation.
`method`	The multiplicity adjustment applied to the pairwise *p*-values, passed to `p.adjust`. Default `"holm"`. Use `"none"` to obtain the unadjusted values.

Details

Which Dunn. Olive Jean Dunn published two different multiple comparison procedures, and both are called "Dunn's" in the literature. This function implements the *nonparametric* one of Dunn (1964), which compares mean ranks. It is not the Bonferroni procedure of Dunn (1961), which Maxwell, Delaney, and Kelley (2027, Chapter 5) call Dunn's procedure and which reaches DMAR through the `method` arguments of `contrast_adjusted` and `p.adjust`. The two are unrelated apart from their author, and the `method` argument here can apply the 1961 procedure to the 1964 procedure's *p*-values.

What it does. The Kruskal–Wallis test asks whether *any* of the groups differ; it does not say which. Dunn's test is its pairwise follow-up. All N observations are ranked together, with tied values receiving their average rank. Writing $\bar{R}_g$ for the mean rank of group g, each pair is compared with

$$z = \frac{\bar{R}_g - \bar{R}_h}{\sqrt{\left[\frac{N(N+1)}{12} - \frac{\sum_i (t_i^3 - t_i)}{12(N-1)}\right]\left(\frac{1}{n_g} + \frac{1}{n_h}\right)}},$$

where t_i is the number of observations tied at the ith distinct value; the second term in the brackets is the tie correction and vanishes when there are no ties. The statistic is referred to the standard normal distribution, and the resulting *p*-values are adjusted for multiplicity by `method`.

The pooled ranking is what makes this the right follow-up. Dunn's test ranks across all groups at once and uses the variance of the ranks implied by the Kruskal–Wallis null, so it is consistent with the omnibus test that preceded it. Running a separate Mann–Whitney test on each pair instead

re-ranks the data within every pair, which answers a different question for every comparison and is not coherent with the omnibus result.

When to use it. Use Dunn's test when the outcome is ordinal, or when it is continuous but the normality or homogeneity assumptions behind `ci_tukey_kramer` and `ci_games_howell` are untenable and a rank-based analysis is preferred to a transformation; the design is between subjects; and a significant Kruskal–Wallis test leaves the question of which groups differ. It is the rank analog of Tukey's HSD, in the sense of covering all $a(a-1)/2$ pairs.

What it does not tell you. The test compares mean *ranks*, not medians. A significant pair means one group's observations tend to be larger, that is, stochastic dominance; it does not by itself license a statement about medians unless the group distributions have the same shape. It also returns no confidence interval on any quantity in the original units, which is a real cost: where a parametric procedure is defensible, `ci_games_howell` or `ci_tukey_kramer` reports intervals on the mean difference, and Maxwell, Delaney, and Kelley (2027) emphasize interval estimation over test decisions throughout. For a distribution free effect size with a confidence interval, see `cliff_delta`.

Value

A `data.frame` with one row per pairwise comparison and columns `contrast`, `mean_rank_difference`, `se`, `z_statistic`, `p_value`, and `p_adjusted`. The table prints through the `dmar_tbl` display layer.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Dunn, O. J. (1964). Multiple comparisons using rank sums. *Technometrics, 6*(3), 241–252. doi:10.1080/00401706.1964.10490181

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association, 56*(293), 52–64. doi:10.2307/2282330 (The Bonferroni procedure; not what this function computes.)

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association, 47*(260), 583–621. doi:10.2307/2280779

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

`kruskal.test` for the omnibus test this follows, `ci_games_howell` and `ci_tukey_kramer` for the parametric all-pairs procedures, `cliff_delta` for a distribution free effect size with a confidence interval, and `p.adjust` for the `method` choices.

Examples

```
# The omnibus question first: do the three treatment arms of the
# drinks_trial data differ at all? The raw drinks_per_week outcome is
# markedly right-skewed, which is no obstacle here: ranks are invariant
# to a monotone transformation, so the raw and log scales give
```

```
# identical results.
kruskal.test(drinks_per_week ~ treatment, data = drinks_trial)

# Then which pairs, on the same pooled ranking the omnibus test used.
dunn_test(drinks_trial$drinks_per_week, drinks_trial$treatment)

# Without a multiplicity adjustment (rarely what you want).
dunn_test(drinks_trial$drinks_per_week, drinks_trial$treatment,
          method = "none")
```

`ecvi` *Expected Cross-Validation Index (ECVI) for a Covariance-Structure Model*

Description

The ECVI of Browne and Cudeck (1989) estimates how well a fitted model's implied covariance matrix would fit an independent sample of the same size from the same population. It is the single-sample estimate of the cross-validation discrepancy, so a smaller ECVI indicates a model expected to generalize better; ECVI is most useful for comparing competing models fit to the same data. A confidence interval, derived from the noncentral chi square distribution, accompanies the point estimate.

Usage

```
ecvi(
  fit = NULL,
  chisq = NULL,
  df = NULL,
  npar = NULL,
  multiplier = NULL,
  conf_level = 0.95
)
```

Arguments

`fit`	A fitted **lavaan** model. Supply this, or the summary statistics below.
`chisq`, `df`, `npar`	The model chi square, its degrees of freedom, and the number of free parameters. Used when `fit` is not supplied, so an ECVI can be obtained from a published fit table.
`multiplier`	The multiplier with which the chi square was formed from the minimized fit function, which is the divisor of the ECVI (n in Details): the total sample size N for a statistic formed as $N\hat{F}$, and $N - G$ for G groups for a statistic formed as $(N - G)\hat{F}$. Given with `chisq`, `df`, and `npar` when `fit` is not supplied; with a fitted model it is taken from the fit.
`conf_level`	Confidence level for the interval. Defaults to 0.95.

Details

With q free parameters, ECVI $= (\chi^2 + 2q)/n$, where n is the multiplier with which the chi square was formed from the minimized fit function (the argument `multiplier`), $\chi^2 = n\hat{F}$, so that ECVI $= \hat{F} + 2q/n$ (Browne & Cudeck, 1989). With a fitted model the divisor is the one lavaan uses, and the value is the one `lavaan::fitMeasures()` reports: $n = N$, the total sample size, for maximum likelihood under lavaan's default normal likelihood (including MLR, whose unscaled statistic is used, and FIML), and $n = N - G$ for G groups under `likelihood = "wishart"` and for the least squares estimators (GLS, WLS, ULS, DWLS). With summary statistics, `multiplier` is used as given: pass $N - G$ when the published chi square was formed as $(N - G)\hat{F}$, as by programs that follow the Wishart convention. Writing $\hat{\lambda} = \chi^2 - df$ for the estimated noncentrality, the ECVI is $(\hat{\lambda} + df + 2q)/n$; the confidence interval replaces $\hat{\lambda}$ by the lower and upper noncentrality limits from `ci_nc_chisq`, the same inversion used for the RMSEA interval (see `ci_rmsea`).

The noncentrality limits are at least 0, while the point estimate is the untruncated $\hat{F} + 2q/n$ of Browne and Cudeck (1989), so for a model with $\chi^2 < df$ the estimate lies below the lower limit, $(df + 2q)/n$. When the chi square lies at or below the upper $\alpha/2$ quantile of the central chi square distribution, for α = `1 - conf_level`, the lower noncentrality limit is 0 and the lower limit is that same $(df + 2q)/n$. When it lies at or below the lower $\alpha/2$ quantile, no noncentrality is consistent with it at that level, and the upper limit is `NA`, with a warning. `sem_fit_measures` reports the upper RMSEA limit as 0 in that case, so that it agrees with the test of not close fit, but the ECVI has no such test, and an upper limit of $(df + 2q)/n$ would make an interval of zero width that excludes the estimate.

A fitted model must have converged to an admissible solution, as for `sem_fit_measures`: a solution with a negative variance estimate, or with a covariance matrix of the latent variables, of the residuals of the observed variables, or of the structural residuals that is not positive definite, is refused (with an error of class `dmar_inadmissible`), since its fit describes estimates outside the parameter space; with auxiliary variables the check is made on the model's own parameters.

When the chi square is the likelihood ratio statistic $2(\ell_s - \ell_M)$, as under lavaan's default normal likelihood, $N \cdot \text{ECVI} = \text{AIC} + 2\ell_s$, where ℓ_s is the maximized log likelihood of the saturated model and ℓ_M that of the fitted model. ECVI and AIC then rank models fitted to the same data identically, and their values differ only by the saturated log likelihood constant before division by N. Under the Wishart convention the statistic is $(N - G)/N$ times the likelihood ratio statistic, and the identity holds only approximately. ECVI is reported because its metric (a discrepancy per observation) and its confidence interval are interpretable on their own.

Value

A `data.frame` (class `dmar_tbl`) with rows `ecvi`, `ci_lower`, and `ci_upper` in the `value` column.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Browne, M. W., & Cudeck, R. (1989). Single sample cross-validation indices for covariance structures. *Multivariate Behavioral Research, 24*(4), 445–455.

See Also

ci_rmsea, ci_nc_chisq.

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
# From a published fit table (no model object needed).
ecvi(chisq = 24.361, df = 8, npar = 13, multiplier = 301)

fit <- lavaan::cfa(
  "visual =~ t1_visual_perception + t2_cubes + t4_lozenges
   verbal =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning",
  data = holzinger_swineford, std.lv = TRUE)
ecvi(fit)
```

---

effects_coding *Effects-Coding Contrast Matrix for a Factor*

Description

Builds the effects-coding (also called *deviation coding* or *sum-to-zero*) contrast matrix for a factor with a levels. Each non-reference level contrasts with the grand mean (rather than with a reference category as in dummy coding). The returned matrix has rows = levels and columns named after the levels, replacing the numeric column names produced by `stats::contr.sum()`.

Usage

```
effects_coding(levels, reference = NULL)
```

Arguments

`levels`	Either an integer giving the number of levels or a character / factor vector giving the level labels. If integer, the labels default to `"L1"`, `"L2"`, ...
`reference`	Optional character name of the reference level (whose coefficients are all -1). Default is the last level.

Details

Why effects coding. Effects coding gives the regression intercept the interpretation of the grand mean (rather than the reference-category mean), and each slope coefficient becomes the deviation of that level's mean from the grand mean (rather than the difference vs the reference category). For balanced designs the effect coefficients are orthogonal to the intercept.

Equivalent to. `stats::contr.sum()` but with meaningful column names (the level labels), which is what is lost in the base-R implementation.

Value

A numeric $a \times (a-1)$ matrix with row names = the factor levels and column names = the non-reference levels. Suitable for assignment to `contrasts(factor)` or use in manual contrast construction.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Lawrence Erlbaum. (See Chapter 8.)

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapters 4, 7.)

See Also

`contr.sum`, `helmert_coding`, `is_orthogonal_set`

Other design utilities: `design_consequences()`, `design_effect()`, `helmert_coding()`, `is_orthogonal_set()`, `orthogonal_polynomial()`

Examples

```
# 1. Effects coding for a 4-level factor:
effects_coding(c("low", "med", "high", "very_high"))

# 2. With "low" as the reference category:
effects_coding(c("low", "med", "high", "very_high"),
               reference = "low")

# 3. Assign to a factor for modeling:
f <- factor(c("a", "b", "c", "a", "b", "c"))
contrasts(f) <- effects_coding(levels(f))
model.matrix(~ f)
```

`epsilon_corrections` *Greenhouse-Geisser, Huynh-Feldt, and Lower-Bound Epsilon Corrections*

Description

Computes the three standard sphericity-correction factors for the univariate within-subjects *F* test. When sphericity holds, all three equal 1; departures from sphericity reduce them, deflating the effective degrees of freedom and thus tempering the inflated Type I error rate of the unadjusted univariate test.

Usage

```
epsilon_corrections(x, id = NULL, time = NULL, outcome = NULL)
```

Arguments

`x`	Either an $n \times k$ numeric matrix or `data.frame` (rows = subjects, columns = repeated measurements); *or* a long-format `data.frame` together with `id`, `time`, and `outcome` column names.
`id`	Column name in `x` identifying the subject when `x` is in long format (`NULL` otherwise).
`time`	Column name in `x` identifying the within-subjects factor level when `x` is in long format (`NULL` otherwise).
`outcome`	Column name in `x` identifying the dependent variable when `x` is in long format (`NULL` otherwise).

Details

For an $(k-1) \times (k-1)$ covariance matrix $\hat{\Sigma}_C$ of orthonormal contrasts among the k repeated measurements (with eigenvalues $\lambda_1, \ldots, \lambda_{k-1}$):

$$\hat{\varepsilon}_{\mathrm{GG}} = \frac{(\sum \lambda_i)^2}{(k-1)\sum \lambda_i^2},$$

$$\hat{\varepsilon}_{\mathrm{HF}} = \min\Big(1,\ \frac{n(k-1)\hat{\varepsilon}_{\mathrm{GG}} - 2}{(k-1)\big(n-1-(k-1)\hat{\varepsilon}_{\mathrm{GG}}\big)}\Big),$$

$$\hat{\varepsilon}_{\mathrm{LB}} = \frac{1}{k-1}.$$

The Greenhouse-Geisser $\hat{\varepsilon}$ tends to be conservative; the Huynh-Feldt correction adjusts upward to be (approximately) unbiased; the lower bound is the worst-case adjustment.

Value

A `data.frame` with columns `epsilon_method` (`"Greenhouse-Geisser"`, `"Huynh-Feldt"`, `"lower_bound"`) and `epsilon` (the correction factor in $[1/(k-1), 1]$).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Greenhouse, S. W., & Geisser, S. (1959). On methods in the analysis of profile data. *Psychometrika, 24*(2), 95–112.

Huynh, H., & Feldt, L. S. (1976). Estimation of the Box correction for degrees of freedom from sample data in randomized block and split-plot designs. *Journal of Educational Statistics, 1*(1), 69–82.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

mauchly_test, anova_within

Other within-subjects analysis: anova_within(), anova_within_two_way(), mauchly_test(), pairwise_within(), plot_trajectories_fitted()

Examples

```
set.seed(113)
Y <- matrix(rnorm(20 * 4), nrow = 20)
epsilon_corrections(Y)
```

---

equivalence_c *Equivalence and Noninferiority Tests for a Linear Contrast via Two One-Sided Tests (TOST)*

Description

Performs the two one-sided tests procedure (Schuirmann, 1987) for equivalence, and the companion one-sided noninferiority test, for a linear contrast of group means $\psi = \sum_j c_j \mu_j$ in a fixed effects design with one pooled error term. The equivalence null hypothesis is that the population contrast lies outside the user-specified bounds, $H_0 : \psi \leq -\delta_L \cup \psi \geq \delta_U$, against the alternative $H_1 : -\delta_L < \psi < \delta_U$. The noninferiority null is $H_0 : \psi \leq -\delta_L$ against $H_1 : \psi > -\delta_L$, and the nonsuperiority null is $H_0 : \psi \geq \delta_U$ against $H_1 : \psi < \delta_U$. Following Chattopadhyay, Bandyopadhyay, Kelley, and Padalunkal (2025), the equivalence decision is read from the 100(1 - 2α)% confidence interval: equivalence is declared when the whole interval lies inside $(-\delta_L, \delta_U)$, noninferiority when the interval's lower limit exceeds $-\delta_L$, and the mirror claim, that the contrast is not relevantly superior, when its upper limit falls below δ_U.

Usage

```
equivalence_c(
  means = NULL,
  s_anova = NULL,
  c_weights = NULL,
  n_per_group = NULL,
  contrast = NULL,
  se = NULL,
  df_error = NULL,
  contrast_lower = NULL,
  contrast_upper = NULL,
  benchmark = NULL,
  alpha_level = 0.05
)
```

Arguments

means	A vector of group means. Supply together with s_anova, c_weights, and n_per_group. Alternatively, use the direct interface via contrast, se, and df_error.
s_anova	The standard deviation of the errors from the ANOVA model (the square root of the mean square error), as in ci_c.
c_weights	The contrast weights. For a mean comparison the weights must sum to zero, and, so that the bounds are on the raw scale of the response, the positive weights must sum to 1 and the negative weights to -1 (use fractional values, not integers). When benchmark is supplied, the weights must instead be nonnegative and sum to 1 (typically a single 1 selecting one group).
n_per_group	Sample sizes per group (if length 1, equal group sizes are assumed).
contrast	The estimated contrast ($\hat{\psi}$ in the formulas), for the direct interface. Supply together with se and df_error when the contrast and its standard error have already been computed (for example, from a fitted model with covariates).
se	The standard error of contrast, for the direct interface.
df_error	The error degrees of freedom. On the summary-statistic interface the default is $N - J$, with J the number of groups; it must be supplied for designs with additional factors. Required on the direct interface.
contrast_lower, contrast_upper	Equivalence bounds on the raw scale of the response. Both must be positive; the equivalence region is $(-\delta_L, +\delta_U)$, where δ_L is contrast_lower and δ_U is contrast_upper. If only contrast_upper is supplied, the bounds are symmetric. Noninferiority uses $-\delta_L$ alone.
benchmark	An optional known constant to compare against (for example, a normative or regulatory cutoff). When supplied, the contrast is $\sum_j c_j \mu_j - b$ with nonnegative weights summing to 1, and the constant contributes no sampling variability.
alpha_level	One-sided significance level for each of the two tests. Default 0.05, so the interval the decisions are read from is the 90% CI.

Details

One pooled error term. On the summary-statistic interface the standard error is $\mathrm{SE}(\hat{\psi}) = s_{\mathrm{anova}}\sqrt{\sum_j c_j^2/n_j}$, the model comparison position of Maxwell, Delaney, and Kelley (2027): every one-degree-of-freedom contrast is judged against the same yardstick, the root mean square error of one model fit to all groups.

The verdict logic. Reading the 100(1 - 2α)% CI against the bounds: an interval entirely inside $(-\delta_L, \delta_U)$ is *equivalent*; entirely above δ_U is *superior* (which implies noninferior); entirely below $-\delta_L$ is *inferior* (which implies not relevantly superior); a lower limit above $-\delta_L$ with an upper limit at or past δ_U is *noninferior only*; an upper limit below δ_U with a lower limit at or below $-\delta_L$ is *not relevantly superior only*; and an interval that contains both bounds is *inconclusive*. The labels are checked in that order, so the more informative claim wins. An inconclusive result is a statement about precision, not evidence of a difference: only an interval clearing a bound entirely licenses a directional claim.

Not relevantly superior. Noninferiority and nonsuperiority are the two halves of the TOST, each a claim in its own right. The `noninferior` flag is 1 exactly when `p_lower` is below `alpha_level`, and the `nonsuperior` flag exactly when `p_upper` is. A nonsuperior contrast is shown to be smaller than δ_U: whatever advantage the contrast measures is smaller than the smallest that would matter, and the interval may still allow a relevant disadvantage. This is the question a study asks when δ_U is the improvement that would justify a change, such as adopting a costlier treatment or procedure: an upper limit below δ_U says the change is not warranted, whether or not the new condition is worse.

Why the weights must sum to ± 1. A bound stated in raw units of the response is only meaningful if the contrast is itself a simple difference of (weighted) means on that scale, which requires the positive weights to sum to 1 and the negative weights to -1. A weight vector such as `c(2, -2)` would silently double the effective bounds, so it is rejected rather than rescaled.

Choosing the bounds. The bounds must be fixed before the data are examined, on substantive grounds: the smallest difference that would matter (Serlin & Lapsley, 1985; Lakens, Scheel, & Isager, 2018). They are never derived from a standard error, which would make the definition of "close enough" a function of the sample size.

Agreement with emmeans. The *p*-values reproduce `emmeans::test(..., side = "equivalence")` and `emmeans::test(..., side = "noninferiority")` with `adjust = "none"` to machine precision, and `p_upper` reproduces `side = "nonsuperiority"` with `delta` set to δ_U. Two settings matter when reproducing these values with emmeans: its `side = "left"` tests nonsuperiority, not noninferiority, and its `trt.vs.ctrl` families apply a Dunnett-type adjustment unless `adjust = "none"` is passed.

Value

A `data.frame` with rows for the estimated contrast (`contrast`), its standard error (`se`), the error degrees of freedom (`df`), the two one-sided test statistics (`t_lower`, `t_upper`) and their *p*-values (`p_lower`, `p_upper`), the joint TOST *p*-value (`p_tost`, the larger of the two), the noninferiority *p*-value (`p_noninferiority`, equal to `p_lower` by construction), the 100(1 - 2α)% confidence limits (`ci_lower`, `ci_upper`), the bounds (`contrast_lower`, stored as the signed lower bound, and `contrast_upper`), and five binary decision flags, in this order (1 = declared, 0 = not): `equivalent`, `noninferior` (the lower limit exceeds $-\delta_L$), `superior` (the lower limit exceeds δ_U), `inferior` (the upper limit is below $-\delta_L$), and `nonsuperior` (the upper limit is below δ_U). When all five flags are 0, the interval contains the whole region from $-\delta_L$ to δ_U and the result is inconclusive. The six-way classification is also attached as the `"verdict"` attribute, one of `"Equivalent"`, `"Superior"`,

"Inferior", "Noninferior only", "Not relevantly superior only", or "Inconclusive". The table names the TOST *p*-value, p_tost, in its p_sentence attribute, so that results_sentence pairs the interval with the test of equivalence rather than with p_lower, the first *p*-value row.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., Bandyopadhyay, T., Kelley, K., & Padalunkal, J. J. (2025). A sequential approach for noninferiority or equivalence of a linear contrast under cost constraints. *Psychological Methods, 30*(2), 425–439. doi:10.1037/met0000570

Lakens, D., Scheel, A. M., & Isager, P. M. (2018). Equivalence testing for psychological research: A tutorial. *Advances in Methods and Practices in Psychological Science, 1*(2), 259–269. doi:10.1177/2515245918770963

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means.)

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

Serlin, R. C., & Lapsley, D. K. (1985). Rationality in psychological research: The good-enough principle. *American Psychologist, 40*(1), 73–83.

Wellek, S. (2010). *Testing statistical hypotheses of equivalence and noninferiority* (2nd ed.). Chapman & Hall/CRC.

See Also

equivalence_smd, equivalence_r, ci_c, contrast_test, power_equivalence_c, ss_power_equivalence_c, plot_equivalence

Other equivalence testing: equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), plot_equivalence(), power_density_equivalence_md(), power_equivalence_c(), power_equivalence_md(), power_equivalence_md_plot(), ss_power_equivalence_c()

Examples

```
# 1. Two of five groups compared against the reference group, with a
#    pooled error term from one model across all five groups.
#    Bounds of 5 raw-scale points; alpha_level = .05, so decisions read
#    from the 90% CI.
equivalence_c(means = c(70.40, 55.61, 51.91, 65.66, 65.12),
       s_anova = 15.67,
       c_weights = c(-1, 0, 0, 0, 1),
       n_per_group = c(113, 74, 76, 80, 61),
       contrast_upper = 5)

# 2. The same contrast through the direct interface, as when the
```

```
#    estimate and standard error come from a model with covariates.
#    The 90% CI runs from about -9.39 to -1.17: its upper limit is
#    below 5, so the fifth group's mean is not relevantly above the
#    first's, but its lower limit is below -5, so a relevant shortfall
#    is not ruled out.
res <- equivalence_c(contrast = -5.28, se = 2.49, df_error = 399,
             contrast_upper = 5)
res
attr(res, "verdict")

# 3. A group mean against a fixed benchmark of 68: the constant
#    contributes no sampling variability.
equivalence_c(means = c(70.40, 55.61, 51.91, 65.66, 65.12),
       s_anova = 15.67,
       c_weights = c(1, 0, 0, 0, 0),
       n_per_group = c(113, 74, 76, 80, 61),
       benchmark = 68, contrast_upper = 5)

# 4. Asymmetric bounds: a shortfall of 3 matters, an excess of 8 does.
equivalence_c(contrast = 1.2, se = 1.1, df_error = 120,
       contrast_lower = 3, contrast_upper = 8)
```

`equivalence_proportion_paired` *Equivalence and Noninferiority Tests for Two Paired Proportions*

Description

Tests whether the difference between two proportions measured on the same units, $\psi = p_x - p_y$, lies inside equivalence bounds or above a noninferiority margin: whether a cheaper diagnostic procedure finds a condition about as often as the standard one on the same patients, or whether a short form of a scale classifies respondents about as often as the full form. The tests are the score tests of Nam (1997), which take the place of the *t* tests of `equivalence_c`, and the decisions are read from the score interval of `ci_proportion_paired` (Tango, 1998), with which they agree.

Usage

```
equivalence_proportion_paired(
  x = NULL,
  y = NULL,
  n_10 = NULL,
  n_01 = NULL,
  N = NULL,
  prop_diff_lower = NULL,
  prop_diff_upper = NULL,
  alpha_level = 0.05
)
```

Arguments

x, y	Numeric or logical vectors of the same length, one element per unit, coded 1 (success) and 0 (failure): the outcome of each unit under the first condition (x) and the second (y). A pair with a missing value in either vector is dropped. Supply either x and y or the three counts, not both.
n_10	Number of units that succeed under x and fail under y.
n_01	Number of units that fail under x and succeed under y.
N	Number of units (pairs), at least n_10 + n_01.
prop_diff_lower, prop_diff_upper	Bounds on the scale of the difference in proportions, each a single number in (0, 1); the equivalence region is $(-\delta_L, \delta_U)$, with prop_diff_lower as δ_L and prop_diff_upper as δ_U. Supply at least one. With prop_diff_upper alone the bounds are symmetric. With prop_diff_lower alone the call is a noninferiority test with margin δ_L, and the rows that would need an upper bound are NA (see Details).
alpha_level	One-sided significance level of each test. Defaults to 0.05, so the decisions are read from the 90% score interval.

Details

The tests. With the score statistic $Z(\psi_0)$ described on ci_proportion_paired, the noninferiority null hypothesis $H_0 : \psi \leq -\delta_L$ is rejected in favor of $\psi > -\delta_L$ when $Z(-\delta_L) > z_{1-\alpha}$, and the nonsuperiority null hypothesis $H_0 : \psi \geq \delta_U$ is rejected in favor of $\psi < \delta_U$ when $Z(\delta_U) < -z_{1-\alpha}$. Equivalence, $-\delta_L < \psi < \delta_U$, is claimed when both are rejected. Because the claim requires both rejections, the two one-sided tests form an intersection-union test (Berger & Hsu, 1996): each is carried out at level α and the equivalence claim has level α, with no adjustment for there being two tests. The *p*-value of the equivalence test, p_tost, is the larger of the two one-sided *p*-values.

Why the score statistic and not McNemar's. McNemar's test estimates the variance of the per-unit difference as it would be if ψ were 0, so it tests no difference and nothing else. A test against a margin needs the variance at $\psi = -\delta_L$ (or δ_U), which depends on a discordant probability that the hypothesis does not fix; the score statistic replaces that probability with its maximum likelihood estimate under the hypothesis (Nam, 1997). At $\psi_0 = 0$ the score statistic is McNemar's, so the tests here generalize it. Nam (1997) found that a Wald-type statistic, with the variance estimated from the observed proportions instead, rejects more often than its nominal level, while the score test holds close to it.

Reading the interval. Z decreases in ψ_0, and the score interval is the set of differences the score test does not reject. So the noninferiority test rejects at level α exactly when the lower limit of the $100(1 - 2\alpha)\%$ score interval exceeds $-\delta_L$, and the nonsuperiority test exactly when its upper limit is below δ_U. The flags are computed from that interval, which the table reports.

The flags and the verdict. The flags read the interval against the bounds:

- equivalent: the interval lies inside $(-\delta_L, \delta_U)$.
- noninferior: the lower limit exceeds $-\delta_L$.
- superior: the lower limit exceeds δ_U.
- inferior: the upper limit is below $-\delta_L$.

- `nonsuperior`: the upper limit is below δ_U, so a difference in favor of `x` as large as δ_U is ruled out. It is the mirror image of noninferiority.

With both bounds, the `"verdict"` attribute names the most informative statement the interval supports: `"Equivalent"`, `"Superior"`, `"Inferior"`, `"Noninferior only"` (an interval above $-\delta_L$ that contains δ_U), `"Not relevantly superior only"` (an interval below δ_U that contains $-\delta_L$), or `"Inconclusive"` (an interval that contains both bounds). An inconclusive result is a statement about precision, not evidence of a difference.

Noninferiority alone. A call with `prop_diff_lower` and no `prop_diff_upper` asks only whether $\psi > -\delta_L$, and no upper bound is assumed for it. The rows that would need one, `z_upper`, `p_upper`, `p_tost`, `prop_diff_upper`, `equivalent`, `superior`, and `nonsuperior`, are `NA`, so the table keeps its shape; the verdict is `"Noninferior"` (the lower limit exceeds $-\delta_L$), `"Inferior"` (the upper limit is below $-\delta_L$), or `"Inconclusive"`; and the *p*-value a results sentence reports is `p_noninferiority`. This is the noninferiority test of `equivalence_c`, which reads $-\delta_L$ alone. A call with `prop_diff_upper` alone gives the symmetric bounds $(-\delta_U, \delta_U)$, as `equivalence_c` does.

Choosing the bounds. The bounds are fixed before the data are examined, on substantive grounds, as the smallest difference in the two proportions that would matter, and on the scale of the difference itself: `prop_diff_lower = 0.05` is five percentage points.

Value

A `data.frame` (class `dmar_tbl`) with rows `difference` ($\hat{\psi}$); the counts `n_10`, `n_01`, and `N`; the score statistics `z_lower` ($Z(-\delta_L)$) and `z_upper` ($Z(\delta_U)$) and their one-sided *p*-values `p_lower` and `p_upper`; the TOST *p*-value `p_tost`; the noninferiority *p*-value `p_noninferiority` (equal to `p_lower` by construction); the 100(1 - 2α)% score limits `ci_lower` and `ci_upper`; the bounds `prop_diff_lower` (stored signed, as $-\delta_L$) and `prop_diff_upper`; and the flags `equivalent`, `noninferior`, `superior`, `inferior`, and `nonsuperior` (1 = declared, 0 = not). A noninferiority test alone, with no `prop_diff_upper`, has the same rows, with `NA` in those that would need an upper bound. The attributes are `conf_level` ($1-2\alpha$), the `verdict`, and `p_sentence`, which names the *p*-value `results_sentence` reports beside the interval: `p_tost`, or `p_noninferiority` for a noninferiority test alone.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Berger, R. L., & Hsu, J. C. (1996). Bioequivalence trials, intersection-union tests and equivalence confidence sets. *Statistical Science, 11*(4), 283–319. doi:10.1214/ss/1032280304

McNeil, B. J., Keeler, E., & Adelstein, S. J. (1975). Primer on certain elements of medical decision making. *New England Journal of Medicine, 293*(5), 211–215. doi:10.1056/NEJM197507312930501

Nam, J.-m. (1997). Establishing equivalence of two treatments and sample size requirements in matched-pairs design. *Biometrics, 53*(4), 1422–1430. doi:10.2307/2533508

Tango, T. (1998). Equivalence test and confidence interval for the difference in proportions for the paired-sample design. *Statistics in Medicine, 17*(8), 891–908. doi:10.1002/(SICI)10970258(19980430)17:8<891::AID-SIM780>3.0.CO;2B

See Also

ci_proportion_paired for the interval alone; equivalence_c for a linear contrast of means.

Other equivalence testing: equivalence_c(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), plot_equivalence(), power_density_equivalence_md(), power_equivalence_c(), power_equivalence_md(), power_equivalence_md_plot(), ss_power_equivalence_c()

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# Liver scan data from McNeil, Keeler, and Adelstein (1975), the
# example of Nam (1997, p. 1427): 344 patients classified by scan (x)
# and by pathology (y), 32 positive by scan only and 27 by pathology
# only. Does the scan understate the rate of disease by less than five
# percentage points? That is a noninferiority question, so only
# prop_diff_lower is given, and the rows that would need an upper bound
# are NA. z_lower is the score statistic Nam reports, 2.795; the table
# rounds it for display and stores the full value.
res <- equivalence_proportion_paired(n_10 = 32, n_01 = 27, N = 344,
                                     prop_diff_lower = 0.05)
res
res$value[res$term == "z_lower"]
attr(res, "verdict")

# Agreement within five points in both directions is a stronger
# claim. The interval rules out a shortfall of five points but not an
# excess of five, so the verdict is noninferiority only.
res <- equivalence_proportion_paired(n_10 = 32, n_01 = 27, N = 344,
                                     prop_diff_lower = 0.05,
                                     prop_diff_upper = 0.05)
attr(res, "verdict")

# A short form (x) and the full form (y) of a screening scale given to
# the same 400 respondents: 38 screen positive on the short form only
# and 52 on the full form only. A shortfall of five points is not ruled
# out, but an excess of five is.
res <- equivalence_proportion_paired(n_10 = 38, n_01 = 52, N = 400,
                                     prop_diff_lower = 0.05,
                                     prop_diff_upper = 0.05)
res
attr(res, "verdict")
```

`equivalence_r` *Equivalence Test for the Pearson Correlation via Two One-Sided Tests (TOST)*

Description

Performs a two one-sided tests procedure for equivalence of a Pearson correlation ρ to zero against user-specified equivalence bounds $[-\rho_L, \rho_U]$ (Counsell & Cribbie, 2015; Goertzen & Cribbie, 2010). Uses the Fisher's Z transformation throughout, a large-sample approximation whose accuracy under bivariate normality improves quickly with *N*. Equivalence is declared when the 100(1 - 2α)% Fisher's Z CI on ρ lies entirely inside the equivalence region.

Usage

```
equivalence_r(
  r = NULL,
  N = NULL,
  x = NULL,
  y = NULL,
  cor_lower = NULL,
  cor_upper = NULL,
  alpha_level = 0.05
)
```

Arguments

`r`, `N`	Observed sample correlation *r* and sample size. Alternatively supply `x` and `y` to compute *r* from raw data.
`x`, `y`	Numeric vectors of paired observations. If supplied, `r` and `N` are computed from the data and the `r`/`N` arguments are ignored.
`cor_lower`, `cor_upper`	Equivalence bounds on the correlation scale, both positive. The equivalence region is $[-\rho_L, +\rho_U]$, where ρ_L is `cor_lower` and ρ_U is `cor_upper`. If only `cor_upper` is supplied, the bounds are symmetric.
`alpha_level`	One-sided significance level. Default `0.05`.

Details

Fisher's Z transformation.

$$Z = \tfrac{1}{2}\log\left(\frac{1+r}{1-r}\right), \quad \mathrm{Var}(Z) = \frac{1}{N-3}.$$

The TOST is run on the Fisher's Z scale:

- Lower test: $(Z - Z_{-\rho_L})\sqrt{N-3}$ compared against the upper α of $N(0,1)$.
- Upper test: $(Z - Z_{\rho_U})\sqrt{N-3}$ compared against the lower α of $N(0,1)$.

The CI bounds are back-transformed from the Z scale via $r = \tanh(Z)$ so they remain in $[-1, 1]$.

Choosing ρ_L and ρ_U. Common choices in psychology are 0.1, or domain-specific meaningfulness thresholds (e.g., 0.2 for cognitive task correlations). The bounds must be set *before* data collection.

Value

A `data.frame` with rows for the observed *r*, the two one-sided test statistics on the Fisher's Z scale, their *p*-values, the joint TOST *p*-value, the 100(1 - 2α)% CI on ρ, the equivalence bounds, a binary equivalence flag, and the sample size (`N`). The table names the TOST *p*-value, `p_tost`, in its `p_sentence` attribute, so that `results_sentence` pairs the interval with the test of equivalence rather than with `p_lower`, the first *p*-value row.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Counsell, A., & Cribbie, R. A. (2015). Equivalence tests for comparing correlation and regression coefficients. *British Journal of Mathematical and Statistical Psychology, 68*(2), 292–309. doi:10.1111/bmsp.12045

Goertzen, J. R., & Cribbie, R. A. (2010). Detecting a lack of association: An equivalence testing approach. *British Journal of Mathematical and Statistical Psychology, 63*(3), 527–537. doi:10.1348/000711009X475853

Lakens, D. (2017). Equivalence tests: A practical primer for *t* tests, correlations, and meta-analyses. *Social Psychological and Personality Science, 8*(4), 355–362. doi:10.1177/1948550617697177

See Also

`equivalence_smd`, `ci_r`

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Other equivalence testing: `equivalence_c()`, `equivalence_proportion_paired()`, `equivalence_rmsea()`, `equivalence_smd()`, `plot_equivalence()`, `power_density_equivalence_md()`, `power_equivalence_c()`, `power_equivalence_md()`, `power_equivalence_md_plot()`, `ss_power_equivalence_c()`

Examples

```
# 1. Equivalence test that |rho| < 0.10 with N = 200 and r = 0.05:
equivalence_r(r = 0.05, N = 200, cor_upper = 0.10)

# 2. From raw data:
set.seed(113)
x <- rnorm(150); y <- 0.04 * x + rnorm(150)
equivalence_r(x = x, y = y, cor_upper = 0.15)
```

`equivalence_rmsea` *Equivalence Test for the Population RMSEA (Test of Not Close Fit)*

Description

Tests whether the population root mean square error of approximation (RMSEA) of a structural equation model is below a bound the researcher chooses in advance, so that a model can be endorsed as fitting within a stated tolerance rather than merely not rejected. This is the test of not close fit of MacCallum, Browne, and Sugawara (1996), which Yuan, Chan, Marcoulides, and Bentler (2016) present as an equivalence test: the null hypothesis $H_0 : \epsilon \geq \epsilon_0$ that the population RMSEA ϵ is at least the bound ϵ_0 is tested against $H_1 : \epsilon < \epsilon_0$, and rejecting it supports fit within the bound. It reverses the test of close fit (Browne & Cudeck, 1993), whose null hypothesis $\epsilon \leq \epsilon_0$ a well-fitting model merely fails to reject. The function works from the summary statistics a paper reports.

Usage

```
equivalence_rmsea(
  chisq,
  df,
  N,
  rmsea_bound,
  n_groups = 1,
  alpha_level = 0.05,
  sample_size_convention = c("N-1", "N")
)
```

Arguments

`chisq`	The model's chi square test statistic T, the likelihood ratio statistic of a normal theory maximum likelihood fit.
`df`	The model's degrees of freedom, a positive whole number.
`N`	The total sample size, summed over groups: a whole number greater than `n_groups`.
`rmsea_bound`	The bound ϵ_0: the largest population RMSEA the researcher is prepared to call fit within tolerance, chosen before seeing the data. No universal value is assumed.
`n_groups`	The number of independent groups of a multiple group model, whose `chisq`, `df`, and `N` are totals over the groups. Default 1.
`alpha_level`	The one-sided Type I error rate, in $(0, 0.5)$; default 0.05. The interval reported has confidence level $1 - 2\alpha$, 90 percent at the default.
`sample_size_convention`	The multiplier a in $T = a\hat{F}$ with which `chisq` was formed: `"N-1"` (the default), each group's sample size less one, so $a = N - G$ for G groups, or `"N"`, so $a = N$. lavaan's default normal likelihood and Mplus form T with N, so their statistics take `"N"`.

Details

Write T for the chi square statistic on df degrees of freedom and a for the multiplier with which it was formed. Under a model whose misfit is small relative to sampling error, T is approximately noncentral chi square with noncentrality $\lambda = a \cdot df \cdot \epsilon^2 / G$ (Steiger & Lind, 1980; Browne & Cudeck, 1993; Steiger, 1998, for G groups). At the boundary of the null hypothesis the noncentrality is $\lambda_0 = a \cdot df \cdot \epsilon_0^2 / G$ (row `null_noncentrality`), and a small T is evidence against H_0, so the p-value is the lower tail, $P\{\chi^2_{df}(\lambda_0) \leq T\}$ (MacCallum et al., 1996, Figure 2). The model is declared within tolerance (row `equivalent` equal to 1) when the p-value is below `alpha_level`. The p-value and the interval stay accurate for a statistic in the millions, where R's `pchisq()` fails (see `ci_nc_chisq`).

The decision is the same as comparing an interval with the bound. The interval reported is the two-sided $100(1-2\alpha)$ percent interval for the population RMSEA, 90 percent at the default, whose upper limit is the one-sided $100(1-\alpha)$ percent upper bound; the p-value is below α exactly when the upper limit is below ϵ_0 (MacCallum et al., 1996, Table 1). This is why the RMSEA is conventionally reported with a 90 percent interval (Browne & Cudeck, 1993): its upper limit settles the test of not close fit at level .05 and its lower limit the test of close fit. The upper limit is the T-size RMSEA of Yuan et al. (2016, Eq. 4): the data endorse the model at level α for every bound above it, so it is the quantity to report when no bound was fixed in advance. The point estimate is truncated at zero; Steiger (2000) shows that the truncation follows from a basic property of a good estimator and costs little in practice, and the interval carries the information about precision. Yuan et al. use $a = N - 1$, the default here. The function does not compute their T-size CFI (their Eqs. 6 to 10, which divide α between the model and baseline statistics) or the adjusted cutoff values for judging a T-size index, which their Tables 1 and 2 give as regression coefficients in N and df).

When T lies at or below the lower α quantile of the central chi square distribution, no noncentrality is consistent with it at this level; the upper limit is then 0, as Yuan et al. (2016, Eq. 4) define the T-size RMSEA and as lavaan reports it, and the model is within tolerance for every positive bound. That happens with probability α for a model that fits exactly and more often when the normal theory statistic is deflated, for example by data with light tails, so a statistic far below its degrees of freedom deserves a look at the data before it is read as evidence of fit.

Failing to reject H_0 does not establish poor fit; it says only that fit within the bound has not been shown at this sample size. The test rests on the noncentral chi square approximation for a normal theory statistic. A scaled or robust statistic does not follow that distribution with this noncentrality, so it must not be supplied here; for a fitted model, including a robust (MLR, MLM), categorical (WLSMV), or FIML fit, `sem_fit_measures` reports this test as `p_not_close_fit` from lavaan's corrected measures. For a multiple group model the convention is Steiger's (1998): `chisq`, `df`, and `N` are totals over the groups, and the RMSEA and its limits carry the factor $\sqrt{G}$.

Value

A `dmar_tbl` (class `dmar_sem_fit`) with columns `term` and `value` and the rows `rmsea` (the point estimate $\sqrt{G \max(T - df, 0)/(a \cdot df)}$), `ci_lower` and `ci_upper` (the $100(1 - 2\alpha)$ percent interval for the population RMSEA), `rmsea_bound`, `null_noncentrality`, `p_value`, `equivalent` (1 when the test rejects H_0 at `alpha_level`, 0 otherwise), `chi_square`, `df`, `N`, and `n_groups`. The `conf_level` attribute is $1 - 2\alpha$. The `verdict` attribute states the inference in a sentence, and the `sem_fit` attribute records the sample size convention and multiplier, the boundary convention that applied, and the lines of the printed footer.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Browne, M. W., & Cudeck, R. (1993). Alternative ways of assessing model fit. In K. A. Bollen & J. S. Long (Eds.), *Testing structural equation models* (pp. 136–162). Sage.

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130–149. doi:10.1037/1082989X.1.2.130

Steiger, J. H. (1998). A note on multiple sample extensions of the RMSEA fit index. *Structural Equation Modeling, 5*(4), 411–419. doi:10.1080/10705519809540115

Steiger, J. H. (2000). Point estimation, hypothesis testing, and interval estimation using the RMSEA: Some comments and a reply to Hayduk and Glaser. *Structural Equation Modeling, 7*(2), 149–162.

Steiger, J. H., & Lind, J. C. (1980). *Statistically-based tests for the number of common factors*. Paper presented at the annual Spring meeting of the Psychometric Society, Iowa City, IA.

Yuan, K.-H., Chan, W., Marcoulides, G. A., & Bentler, P. M. (2016). Assessing structural equation models by equivalence testing with adjusted fit indexes. *Structural Equation Modeling, 23*(3), 319–330. doi:10.1080/10705511.2015.1065414

See Also

sem_fit_measures for the test of not close fit from a fitted model (robust, categorical, and FIML fits included) beside the other fit measures, ci_rmsea for an interval from a reported RMSEA, and equivalence_r and equivalence_smd for equivalence tests of other effect sizes.

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Other equivalence testing: equivalence_c(), equivalence_proportion_paired(), equivalence_r(), equivalence_smd(), plot_equivalence(), power_density_equivalence_md(), power_equivalence_c(), power_equivalence_md(), power_equivalence_md_plot(), ss_power_equivalence_c()

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
# Yuan et al. (2016, Example 1): nine holzinger_swineford
```

```
# tests in the Grant-White school (N = 145), the three-factor model with
# one cross-loading, T = 28.098 on 23 df. The 90 percent upper limit,
# their T-size RMSEA, is .083, so fit within .05 is not established.
equivalence_rmsea(chisq = 28.098, df = 23, N = 145, rmsea_bound = 0.05)

# A bound of .09 is endorsed at alpha = .05.
equivalence_rmsea(chisq = 28.098, df = 23, N = 145, rmsea_bound = 0.09)

# The statistic lavaan reports for the two-school model of the same
# tests (formed with N, its default), with Steiger's (1998) multiple
# group convention.
equivalence_rmsea(chisq = 115.8513, df = 48, N = 301, rmsea_bound = 0.10,
                  n_groups = 2, sample_size_convention = "N")
```

`equivalence_smd` *Equivalence Test for the Standardized Mean Difference via Two One-Sided Tests (TOST)*

Description

Performs a two one-sided tests procedure (Schuirmann, 1987) for equivalence between two independent groups on the standardized mean difference (Cohen's *d*) scale. The null hypothesis is that the true δ lies outside the user-specified equivalence bounds $[-\delta_L, \delta_U]$; the alternative is that δ lies inside them. The test is the joint pair of one-sided *t* tests: $H_{0,L} : \delta \leq -\delta_L$ vs $H_{1,L} : \delta > -\delta_L$, and $H_{0,U} : \delta \geq \delta_U$ vs $H_{1,U} : \delta < \delta_U$. Equivalence is declared when *both* null hypotheses are rejected at level α.

Usage

```
equivalence_smd(
  x = NULL,
  y = NULL,
  smd = NULL,
  n_1 = NULL,
  n_2 = NULL,
  smd_lower = NULL,
  smd_upper = NULL,
  alpha_level = 0.05
)
```

Arguments

x, y	Numeric vectors of observations from the two groups. Alternatively, supply smd, n_1, n_2 via the summary-statistic interface.
smd	Observed standardized mean difference (Cohen's *d*). Required if x and y are not supplied.
n_1, n_2	Group sample sizes. Required if x and y are not supplied.

`smd_lower`, `smd_upper`
: Equivalence bounds on the d scale. Both must be positive; the equivalence region is $[-\delta_L, +\delta_U]$, with `smd_lower` as δ_L and `smd_upper` as δ_U. If only `smd_upper` is supplied, the bounds are symmetric: $[-\delta_U, +\delta_U]$.

`alpha_level`
: One-sided significance level for each of the two tests. Default `0.05`. The TOST is then equivalent to a 100(1 - 2α)% CI on δ lying inside the equivalence bounds.

Details

Schuirmann's TOST. The TOST procedure tests $H_0 : \delta \leq -\delta_L \cup \delta \geq \delta_U$ against $H_1 : -\delta_L < \delta < \delta_U$. Both component tests are rejected (and equivalence is declared) when the 100(1 - 2α)% CI on δ lies entirely inside $[-\delta_L, \delta_U]$.

Critical insight. A non-significant conventional NHST (t test of $\delta = 0$) is *not* evidence of equivalence; it only means we cannot reject $\delta = 0$. TOST inverts the testing logic so that "no meaningful effect" is the alternative, not the null.

Choosing δ_L and δ_U. The equivalence bounds (`smd_lower` and `smd_upper`) must be set *before* data collection, based on what constitutes the smallest effect size of practical interest (Lakens, Scheel, & Isager, 2018). Common choices in the literature are 0.2 or 0.3, but the bound should reflect domain-specific meaningfulness.

Connection to the CI. The TOST rejection at level α is equivalent to the 100(1 - 2α)% Cohen's-*d* CI (i.e., 90% for the default $\alpha = 0.05$) lying entirely inside the equivalence region. This is the "two-one-sided" equivalence and matches the Westlake (1972) rationale for symmetric bioequivalence CIs.

Standard error and approximation. Each one-sided component is a Wald t test, $t = (\hat{d} \mp \delta)/\mathrm{SE}(\hat{d})$, referred to a central t distribution on $n_1 + n_2 - 2$ degrees of freedom, with the Hedges and Olkin (1985) large-sample standard error $\mathrm{SE}(\hat{d}) = \sqrt{(n_1 + n_2)/(n_1 n_2) + \hat{d}^2/(2(n_1 + n_2))}$. This is the asymptotic form of Schuirmann's (1987) two one-sided tests applied on the standardized scale; it is accurate in moderate-to-large samples but is an approximation to the exact noncentral t inversion used by `ci_smd`, and the two can differ in small samples. The 100(1 - 2α)% interval reported here, $\hat{d} \pm t_{1-\alpha,\, n_1+n_2-2}\, \mathrm{SE}(\hat{d})$, is usually the wider of the two at the same $\hat{d}$, group sizes, and level. For 90% intervals from equal groups and $|\hat{d}|$ up to 3, it is wider by 18 to 30 percent with 3 per group and by less than 1 percent with 100 per group. At the 90, 95, and 99 percent levels it was the wider in every case computed ($|\hat{d}|$ up to 12, 2 to 5,000 per group); at the 80 percent level it can be the narrower, by less than 1 percent, when $|\hat{d}|$ is 5 or more.

Planning. `ss_aipe_equivalence_smd` plans the sample size for the interval reported here, and `ss_aipe_equivalence_smd_sensitivity` simulates it, so that the interval a study is planned for, the interval this function reports, and the interval the simulation scores are one interval, formed by the same code.

Value

A `data.frame` with rows for the observed d, the two one-sided test statistics (`t_lower`, `t_upper`) and their degrees of freedom (`df`), the two one-sided p-values (`p_lower`, `p_upper`), the joint TOST p-value (the larger of the two), the 100(1 - 2α)% CI on δ, the equivalence bounds (`smd_lower`, stored signed as $-\delta_L$, and `smd_upper`), and a binary decision flag (`equivalent`: 1 = equivalent, 0 = not). The table names the TOST p-value, `p_tost`, in its `p_sentence` attribute, so that

results_sentence pairs the interval with the test of equivalence rather than with p_lower, the first *p*-value row.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. Academic Press.

Lakens, D. (2017). Equivalence tests: A practical primer for *t* tests, correlations, and meta-analyses. *Social Psychological and Personality Science, 8*(4), 355–362. doi:10.1177/1948550617697177

Lakens, D., Scheel, A. M., & Isager, P. M. (2018). Equivalence testing for psychological research: A tutorial. *Advances in Methods and Practices in Psychological Science, 1*(2), 259–269. doi:10.1177/2515245918770963

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

Westlake, W. J. (1972). Use of confidence intervals in analysis of comparative bioavailability trials. *Journal of Pharmaceutical Sciences, 61*(8), 1340–1341.

See Also

equivalence_r, ss_aipe_equivalence_smd, ci_smd, smd

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Other equivalence testing: equivalence_c(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), plot_equivalence(), power_density_equivalence_md(), power_equivalence_c(), power_equivalence_md(), power_equivalence_md_plot(), ss_power_equivalence_c()

Examples

```
# 1. Two groups, raw data, symmetric equivalence bounds of 0.4:
set.seed(113)
x <- rnorm(50, 100, 15); y <- rnorm(50, 101, 15)
equivalence_smd(x = x, y = y, smd_upper = 0.4)

# 2. Summary-statistic interface: published d = 0.05, n = 40 per group.
equivalence_smd(smd = 0.05, n_1 = 40, n_2 = 40, smd_upper = 0.3)

# 3. Asymmetric bounds (acceptable -0.2 to +0.5):
equivalence_smd(smd = 0.10, n_1 = 80, n_2 = 80,
         smd_lower = 0.2, smd_upper = 0.5)
```

`eta_squared` *Eta Squared (Effect Size for ANOVA)*

Description

Computes the sample eta squared (η^2) of each effect of an analysis of variance, the proportion of the variance of the dependent variable that the effect accounts for, as two estimands: the *total* η^2, the effect's sum of squares as a proportion of the corrected total sum of squares, and the *partial* η^2, its sum of squares as a proportion of its own sum of squares and the error sum of squares. With `estimand = "all"`, the default, the function reports every estimand its inputs determine; `"total"` or `"partial"` reports one. In a one-way design the two are the same number.

Usage

```
eta_squared(
  object = NULL,
  estimand = c("all", "total", "partial"),
  SS_effect = NULL,
  SS_error = NULL,
  SS_total = NULL,
  F_value = NULL,
  df_effect = NULL,
  df_error = NULL
)
```

Arguments

`object`	Optional. A fitted model of class `aov`, `lm`, or `aovlist` (a multi-stratum fit such as `aov(y ~ A + Error(subject/A), data = d)`). The function returns one row per effect (every row of `anova(object)`, or of each stratum of `summary(object)`, other than `Residuals`), with the stratum named for an `aovlist`. Give either `object` or summary statistics, not both.
`estimand`	Which estimand to report: `"all"` (the default, every estimand the inputs determine, `NA` for the others), `"total"`, or `"partial"`. An estimand asked for by name that the inputs do not determine is refused (see Details).
`SS_effect`	The sum of squares of the effect, or a vector of the sums of squares of several effects of one analysis (named for the effects, which label the rows).
`SS_error`	The error (residual) sum of squares. With `SS_effect`, it determines the partial η^2.
`SS_total`	The corrected total sum of squares, $\sum(y - \bar{y})^2$ (see Details). With `SS_effect`, it determines the total η^2.
`F_value`	The observed F statistic of the effect, or a vector of the F statistics of several effects tested against the same error term. With `df_effect` and `df_error`, it determines the partial η^2 only.

`df_effect` The numerator degrees of freedom of the effect (one value for each effect, or one for all). Required with `F_value`; with sums of squares it is optional and, with `df_error` and `SS_error`, fills the `F_value` column.

`df_error` The error (residual) degrees of freedom. Required with `F_value`; optional with sums of squares.

Details

The function takes a fitted model (`aov`, `lm`, or an `aovlist` from `aov()` with an `Error()` term), the sums of squares of a published ANOVA table, or an F statistic with its degrees of freedom. The confidence interval is computed by `ci_eta_squared`.

Total and partial. For an effect with sum of squares SS_{effect}, error sum of squares SS_{error}, and corrected total sum of squares SS_{total},

$$\hat{\eta}^2_{\text{total}} = \frac{SS_{\text{effect}}}{SS_{\text{total}}}, \qquad \hat{\eta}^2_{\text{partial}} = \frac{SS_{\text{effect}}}{SS_{\text{effect}} + SS_{\text{error}}} = \frac{df_{\text{effect}} F}{df_{\text{effect}} F + df_{\text{error}}}$$

(Olejnik & Algina, 2000, Table 10, p. 268; Maxwell, Delaney, & Kelley, 2027, Eqs. 7.40 to 7.42). The total value is a proportion of all the variation in the data, and so depends on the other factors of the design: adding a factor with a large effect lowers it. The partial value leaves the other effects out of its denominator, so the partial values of the effects of one design do not share a denominator and can sum to more than 1 (Olejnik & Algina, 2000, pp. 268–269). Maxwell, Delaney, and Kelley (2027, Chapter 7) frame the choice by the other factors: when another factor is extrinsic to the question (a study of the effect could as well hold it constant, as a study of drugs could hold biofeedback constant) the partial value answers it, and when the factor is intrinsic (it varies wherever the effect is studied, as gender does) the total value does. In a one-way design $SS_{\text{total}} = SS_{\text{effect}} + SS_{\text{error}}$, and the two estimands are the same number.

What the inputs determine. A fitted model determines both estimands of every effect, and so do sums of squares that include both `SS_error` and `SS_total`. `SS_effect` with `SS_total` alone determines the total, and with `SS_error` alone the partial. An F statistic and its degrees of freedom determine the partial value only: the total of an effect also depends on the sums of squares of the design's other effects (from the F statistics of every effect it can be written as $df_{\text{effect}} F / (\sum df F + df_{\text{error}})$, Maxwell, Delaney, & Kelley, 2027, Eq. 7.43, which a fitted model provides). The function cannot tell from F whether the design has other effects, so the F interface reports the total as `NA` even for a one-way design, where it equals the partial value. With `estimand = "all"` an estimand the inputs do not determine is `NA`, with the reason in the `estimand_notes` attribute and beneath the printed table; asked for by name, it is refused with an error of class `dmar_estimand_undetermined` (see `dmar_metadata`). Sums of squares and F are two ways of giving the same effect, and a call that gives both, or a fitted model with summary statistics, is refused, as is a set of sums of squares in which `SS_effect` + `SS_error` exceeds `SS_total`.

The corrected total. `SS_total` is the corrected total sum of squares, $\sum (y - \bar{y})^2$, the sum of every sum of squares of an ANOVA table with an intercept, the error included. Some programs also print an uncorrected total, $\sum y^2$, which includes the grand mean: Kim (2016, Table 3) lists both, a "Total" of 53758.357 and a "Corrected total" of 5863.715, and the second is `SS_total`. A fitted model without an intercept does not partition the corrected total, and its total η^2 is `NA`.

Sums of squares of a fitted model. The sums of squares and F statistics of a fitted model are those of `anova()` (one stratum) or `summary()` (an `aovlist`), which are Type I (sequential): each effect is adjusted for the effects before it in the formula. Sequential sums of squares add up to the corrected

total, so the total η^2 of every effect and the error's share sum to 1. In a balanced design Types I, II, and III agree; in an unbalanced one they differ, and the order of the terms changes the Type I values. For Type II or III sums of squares, compute them with, for example, `car::Anova(object, type = 3)`, and give the effect's sum of squares with the error and corrected total sums of squares to the sums-of-squares interface (Type III sums of squares need not add up to the total in an unbalanced design, so the total values of the effects then need not sum with the error's share to 1). Type III tests depend on how the factors are coded. The Type III main-effect tests compare unweighted marginal means only when the factors are coded with contrasts that sum to zero over their levels, such as `contr.sum` (`contr.helmert` and `contr.poly` also qualify). Under R's default treatment contrasts, the Type III test of a factor that enters an interaction compares its levels only at the reference levels of the factors it interacts with, which is seldom the hypothesis intended. Set the coding before the model is fitted, either in the call (`lm(y ~ A * B, data = d, contrasts = list(A = contr.sum,B = contr.sum))`) or for the session (`options(contrasts = c("contr.sum", "contr.poly"))`). The help page of `car::Anova()` gives the same warning, and notes that when a model has numeric covariates and interactions, the Type III test of a factor compares the groups where the covariates are 0; Type II tests do not depend on the coding. `factorial_anova` applies sum-to-zero contrasts itself and reports partial η^2 and partial ω^2 with their confidence intervals for between-subjects factorial designs.

Interactions and covariates. An interaction is an effect like any other, with its own sum of squares and degrees of freedom, and so is a covariate of an `lm` fit: each is a row, its partial value computed against the error, its total against the corrected total.

Within-subjects designs. For an `aovlist` fit, each effect's partial η^2 is computed against the error of its own stratum, so a within-subjects factor's partial value leaves the differences among subjects out of its denominator. Its total value divides by the corrected total over all strata (the sum of every sum of squares that `summary(object)` reports), which includes the differences among subjects: in a design with one within-subjects factor the total value is the factor's sum of squares over the sum of those for subjects, the factor, and their interaction (Bakeman, 2005, p. 380), and is smaller than the partial value. Neither is comparable with the same effect in a between-subjects design; see `eta_squared_generalized` for an index that is (Olejnik & Algina, 2003; Bakeman, 2005). Other model classes (`lmerMod`, `lme`) are not taken; give their sums of squares or F statistics to the summary interfaces.

Value

A `data.frame` (class `dmar_anova_es`, printed through `dmar_tbl`) with one row per effect and the columns `effect`, `eta_squared_total` and `eta_squared_partial` (only the one asked for when `estimand` is `"total"` or `"partial"`), `stratum` (for an `aovlist` fit), `F_value`, `df_effect`, and `df_error`. An effect given through the summary interfaces without names is `"overall"` (or `effect_1`, `effect_2`, and so on, for several), and a quantity the inputs do not give (the F statistic of sums of squares given without degrees of freedom) is `NA`. The `estimand` attribute records the estimand asked for, and the `estimand_notes` attribute names each estimand column that is `NA` with the reason, which the line beneath the printed table states. `tidy()` gives one row per effect and estimand (see `dmar_tidiers`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bakeman, R. (2005). Recommended effect size statistics for repeated measures designs. *Behavior Research Methods, 37*(3), 379–384. doi:10.3758/BF03192707

Bargmann, R. E. (1970). Interpretation and use of a generalized discriminant function. In R. C. Bose, I. M. Chakravarti, P. C. Mahalanobis, C. R. Rao, & K. J. C. Smith (Eds.), *Essays in probability and statistics* (pp. 35–60). University of North Carolina Press.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40*(3), 659–670. doi:10.1177/001316448004000309

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*, 137–152. doi:10.1037/a0028086

Kim, H.-Y. (2016). Statistical notes for clinical researchers: Sample size calculation 3. Comparison of several means using one-way ANOVA. *Restorative Dentistry & Endodontics, 41*(3), 231–234. doi:10.5395/rde.2016.41.3.231

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on η^2, Chapter 7 on factorial designs, and Chapter 11 on generalized η^2 for within-subjects designs.)

Olejnik, S., & Algina, J. (2000). Measures of effect size for comparative studies: Applications, interpretations, and limitations. *Contemporary Educational Psychology, 25*(3), 241–286. doi:10.1006/ceps.2000.1040

Olejnik, S., & Algina, J. (2003). Generalized eta and omega squared statistics: Measures of effect size for some common research designs. *Psychological Methods, 8*(4), 434–447. doi:10.1037/1082989X.8.4.434

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Venables, W. (1975). Calculation of confidence intervals for noncentrality parameters. *Journal of the Royal Statistical Society, Series B, 37*(3), 406–412. doi:10.1111/j.25176161.1975.tb01554.x

See Also

`ci_eta_squared` for the confidence interval, `omega_squared`, `eta_squared_generalized`, `ci_pvaf`

Other effect size estimates: `cles()`, `cliff_delta()`, `correction_for_attenuation()`, `eta_squared_generalized()`, `expected_partial_r()`, `expected_r()`, `expected_smd()`, `nnt_from_smd()`, `omega_squared()`, `probability_of_superiority_paired()`, `proportion_of_superiority()`, `responder_analysis()`, `smd_trimmed()`

Examples

```
# 1. The sums of squares of a published two-way table: Kim (2016,
#    Table 3), light by material, 2 by 4 with 10 per cell (N = 80).
#    SS_total is the corrected total, 5863.715.
```

```
eta_squared(SS_effect = c(LIGHT = 34.716, MATERIAL = 1996.998,
                          "LIGHT:MATERIAL" = 1570.389),
            SS_error = 2261.612, SS_total = 5863.715,
            df_effect = c(1, 3, 3), df_error = 72)

# 2. An F statistic and its degrees of freedom determine the partial
#    value only. Bargmann's (1970) example, used in turn by Venables
#    (1975), Fleishman (1980), and Steiger (2004): a one-way ANOVA with
#    five groups of 11 and an observed F of 11.221 on 4 and 50 df. In a
#    one-way design the partial value is the total.
eta_squared(F_value = 11.221, df_effect = 4, df_error = 50,
            estimand = "partial")

# 3. One-way ANOVA from a fitted model (depression_bdi: three
#    treatment arms, 10 per arm, N = 30); the two estimands agree.
fit_one <- aov(bdi_post ~ condition, data = depression_bdi)
eta_squared(fit_one)

# 4. Factorial ANOVA (pygmalion: expectancy treatment by grade, 2 by 6
#    with unequal cell sizes, N = 310). The treatment is manipulated;
#    grade is a measured classification of the pupils.
fit_factorial <- aov(iq_8 ~ treatment * factor(grade), data = pygmalion)
eta_squared(fit_factorial)

# 5. Within-subjects ANOVA: 20 simulated subjects, each measured at Pre,
#    Mid, and Post. The total value includes the differences among
#    subjects in its denominator; the partial value does not.
set.seed(113)
n <- 20
rm_data <- data.frame(
  subject = factor(rep(seq_len(n), each = 3)),
  time    = factor(rep(c("Pre", "Mid", "Post"), n),
                   levels = c("Pre", "Mid", "Post")),
  y       = rnorm(n, sd = 1.5)[rep(seq_len(n), each = 3)] +
            0.7 * rep(1:3, n) + rnorm(n * 3, sd = 1.2)
)
fit_rm <- aov(y ~ time + Error(subject/time), data = rm_data)
eta_squared(fit_rm)
```

`eta_squared_generalized` *Generalized Eta Squared (Effect Size for ANOVA, Comparable Across Designs)*

Description

Computes the sample generalized eta squared (η^2_G; Olejnik & Algina, 2003; Bakeman, 2005), the proportion of variance in the dependent variable accounted for by a fixed effect after the variance attributable to *measured* (observed) factors, but not other *manipulated* factors, has been left in the

denominator. This makes η^2_G comparable across designs that differ in which factors are present, which regular η^2 and partial η^2 are not.

Usage

```
eta_squared_generalized(
  object = NULL,
  observed = NULL,
  SS_effect = NULL,
  SS_observed = NULL,
  SS_error = NULL,
  F_effect = NULL,
  df_effect = NULL,
  F_observed = NULL,
  df_observed = NULL,
  df_error = NULL
)
```

Arguments

`object`	Optional. A fitted model object of class `aov`, `lm`, or `aovlist` (multi-stratum aov fit, e.g.\ `aov(y ~ A + Error(subject/A), data = d)`).
`observed`	Character vector naming the *measured* (rather than *manipulated*) factors. Their sums of squares, and the SS of every interaction containing a listed factor, are kept in the denominator of η^2_G per Olejnik and Algina (2003, Eq. 5). Manipulated effects are excluded from the denominator when they are not the focal effect. For `aovlist` fits the names must match the effect labels visible in `summary(object)`.
`SS_effect`	Sum of squares for the focal effect (option 2).
`SS_observed`	Sums of squares for the other sources in the denominator: the measured factors, the interactions that contain one, and covariates, and in a design with an `Error()` term the residuals of the strata other than the effect's. Scalar or numeric vector (option 2).
`SS_error`	Error (residual) sum of squares of the focal effect: the sum of squares its F test divides by, which in a design with an `Error()` term is the residual of the effect's stratum (option 2). The estimate depends on `SS_observed` and `SS_error` only through their sum, but the parametric interval of `ci_eta_squared_generalized` is built on the F they imply.
`F_effect`	Observed F-value for the focal effect (option 3).
`df_effect`	Numerator degrees of freedom for the focal effect (option 3; accepted with the sums of squares of option 2, which do not use it, as Details explains).
`F_observed`	Vector of F-values for the other sources in the denominator, each the source's mean square divided by the focal effect's error mean square (option 3).
`df_observed`	Numerator degrees of freedom for those sources, aligned with `F_observed` (option 3).
`df_error`	Degrees of freedom of the focal effect's error term (option 3; accepted with the sums of squares of option 2, which do not use it, as Details explains).

Details

The function accepts *one* of three input interfaces:

1. a fitted model object (`aov`, `lm`, or `aovlist` for within-subjects / mixed designs) together with an `observed` vector listing which factors are measured (rather than manipulated);
2. raw sums of squares: `SS_effect`, `SS_observed` (a scalar, or one value per other source in the denominator; see `SS_observed`), and `SS_error`; or
3. raw *F*-values and degrees of freedom: `F_effect`, `df_effect`, `F_observed` (vector aligned with `df_observed`), and `df_error`.

The estimate from sums of squares needs no degrees of freedom, while the parametric interval that `ci_eta_squared_generalized` builds from the same sums of squares needs `df_effect` and `df_error`, so the two are accepted with the sums of squares here and leave the estimate unchanged. With sums of squares, a call is read as the *F*/df interface only when it also gives `F_effect`, `F_observed`, or `df_observed`.

If the user supplies both the SS interface (option 2) and the *F*/df interface (option 3), the function computes η^2_G from each and compares the results to within a 1e-6 tolerance. When the two interfaces agree, the SS value is returned. When they disagree, the function stops with a detailed message reporting both values. `ci_eta_squared_generalized` refuses the two together, since they can agree on the estimate and still imply different *F* statistics for the effect, and so different intervals.

Formula.

$$\hat{\eta}^2_G = \frac{SS_{\text{effect}}}{SS_{\text{effect}} + \sum_{\text{obs}} SS_{\text{measured}} + SS_{\text{error}}}.$$

For the *F*/df interface the equivalent ratio form is used, dividing through by SS_{error} so that no total-N argument is required: $SS_i/SS_{\text{error}} = F_i \cdot df_i/df_{\text{error}}$.

Designs supported.

- *Between-subjects ANOVA (single stratum).* The function reads `anova(object)`. For each focal effect, the denominator is $SS_{\text{focal}} + \sum SS_{\text{measured (others)}} + SS_{\text{error}}$. Manipulated factors that are not the focal effect contribute nothing to the denominator.
- *Within-subjects and mixed ANOVA (*`aovlist`*, multi-stratum).* The function reads `summary(object)` and walks every error stratum. For each focal effect, the denominator is $SS_{\text{focal}} + \sum SS_{\text{measured (others)}} + \sum_s SS_{\text{error}(s)}$, where the last sum runs over every error stratum (both the between-subjects "subjects" stratum and any within-subjects error strata). This is the Olejnik & Algina (2003) / Bakeman (2005) rule that makes the subject-level variance act as an "always-measured" contributor in repeated measures designs. Each effect must be tested in a single stratum, as it is when every subject is observed equally often in every within-subjects condition, whatever the sizes of the between-subjects groups, and every covariate is constant within subjects. When observations are missing within subjects, or a covariate varies within them, `aov()` divides the effect's sum of squares among strata and tests the effect in each of them; no one stratum's share is the effect's sum of squares, and Olejnik and Algina (2003, p. 437) define the estimate for balanced designs, so the function stops and names the effects and strata involved.

Focal-effect self-exclusion. If the focal effect itself is listed in `observed`, the function excludes it from the observed-sum component (the focal effect's *own* SS already appears in the numerator and the leading term of the denominator). Practically this means listing every effect as `observed` reduces to total η^2 for between-subjects designs.

Higher-order interactions. An effect is a measured source of variance when *any* factor in its term is measured (Olejnik & Algina, 2003, Eq. 5; Bakeman, 2005), so listing a factor in `observed` also places every interaction containing that factor in the denominator automatically. In a design with manipulated `A` and measured `c`, `observed = "c"` therefore puts `c` and `A:c` in the denominator, which is what the cited papers' worked examples do. An explicit interaction label in `observed` is honored as given, declaring that one term measured without marking its constituent factors.

Covariates. Under Olejnik and Algina's Eq. 5, a covariate is a measured source whose SS always belongs in the denominator, so in an ANCOVA list the covariate in `observed`. Because `anova()` on an `lm`/`aov` fit uses sequential sums of squares, enter the covariate before the treatment factors in the model formula so its SS is adjusted the way the ANCOVA decomposition intends.

Confidence intervals. See `ci_eta_squared_generalized` for the corresponding CI function; both available CI methods are approximate and require independent evaluation.

Value

A `data.frame` with one row per focal effect. With a single-stratum fit and the raw interfaces the columns are `effect` and `eta_squared_generalized`; `effect` is `"overall"` for the raw interfaces. With an `aovlist` fit a `stratum` column is added, identifying which error stratum each effect came from.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bakeman, R. (2005). Recommended effect size statistics for repeated measures designs. *Behavior Research Methods, 37*(3), 379–384. doi:10.3758/BF03192707

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*, 137–152. doi:10.1037/a0028086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on η^2, Chapter 7 on factorial designs, and Chapter 11 on generalized η^2 for within-subjects designs.)

Olejnik, S., & Algina, J. (2003). Generalized eta and omega squared statistics: Measures of effect size for some common research designs. *Psychological Methods, 8*(4), 434–447. doi:10.1037/1082989X.8.4.434

See Also

`ci_eta_squared_generalized`, `eta_squared`, `ci_omega_squared`

Other effect size estimates: `cles()`, `cliff_delta()`, `correction_for_attenuation()`, `eta_squared()`, `expected_partial_r()`, `expected_r()`, `expected_smd()`, `nnt_from_smd()`, `omega_squared()`, `probability_of_superiority_paired()`, `proportion_of_superiority()`, `responder_analysis()`, `smd_trimmed()`

Examples

```
# 1. Fitted model with `observed`. In the pygmalion expectancy
#        experiment, treatment is manipulated while grade is a measured
#        classification, so grade's variance stays in the denominator.
pyg <- pygmalion
pyg$grade <- factor(pyg$grade)
fit <- aov(iq_8 ~ treatment * grade, data = pyg)
eta_squared_generalized(fit, observed = "grade")

# 2. Raw sums of squares.
eta_squared_generalized(SS_effect = 100, SS_observed = c(40, 30),
                        SS_error = 200)

# 3. Raw F-values and degrees of freedom.
eta_squared_generalized(F_effect = 6.0, df_effect = 2,
                        F_observed = c(2.5, 1.8),
                        df_observed = c(1, 2), df_error = 50)

# 4. Within-subjects (repeated measures) ANOVA. The denominator
#        automatically includes the subject-level variance plus the
#        within-subjects error, following Bakeman (2005).
set.seed(113)
n <- 20
rm_data <- data.frame(
  subject = factor(rep(seq_len(n), each = 3)),
  time    = factor(rep(c("Pre", "Mid", "Post"), n),
                   levels = c("Pre", "Mid", "Post")),
  y       = rnorm(n, sd = 1.5)[rep(seq_len(n), each = 3)] +
            0.7 * rep(1:3, n) + rnorm(n * 3, sd = 1.2)
)
fit_rm <- aov(y ~ time + Error(subject/time), data = rm_data)
eta_squared_generalized(fit_rm)

# 5. Mixed design with a measured between-subjects factor. Treat
#        'group' as observed; its SS stays in the denominator for
#        'time' and the 'group:time' interaction.
set.seed(113)
n_per_group <- 10
n <- n_per_group * 2
mixed_data <- data.frame(
  subject = factor(rep(seq_len(n), each = 3)),
  group   = factor(rep(c("Treatment", "Control"), each = 3 * n_per_group)),
  time    = factor(rep(c("Pre", "Mid", "Post"), n),
                   levels = c("Pre", "Mid", "Post")),
  y       = rnorm(n, sd = 1)[rep(seq_len(n), each = 3)] +
            0.5 * rep(1:3, n) + rnorm(n * 3, sd = 1)
)
fit_mixed <- aov(y ~ group * time + Error(subject/time), data = mixed_data)
eta_squared_generalized(fit_mixed, observed = "group")
```

`expected_partial_r` *Exact Expected Value of the Sample Partial Correlation*

Description

Computes $\mathrm{E}[r_{XY \cdot Z_1 \cdots Z_{p-1}} \mid \rho, N, p]$, the exact expected value of the sample partial Pearson correlation coefficient under multivariate normality, applying the Olkin-Pratt (1958) / Hotelling (1953) result for the simple Pearson correlation to the partial-correlation setting, an extension Olkin and Pratt (1958, Section 2.3) give themselves. Like the simple *r*, the partial *r* is downward-biased as an estimator of the population partial correlation $\rho_{XY \cdot Z}$, with the magnitude of the bias growing in the number of predictors p and shrinking in the sample size N.

Usage

```
expected_partial_r(true_cor, N, p)
```

Arguments

`true_cor`	Population partial correlation $\rho_{XY \cdot Z_1 \cdots Z_{p-1}}$. Scalar or vector in $[-1, 1]$.
`N`	Total sample size; $N - p \geq 3$ is required for well-conditioned computation. Scalar or vector.
`p`	Number of predictors in the regression of *Y* on *X* and the variables partialed out, so `p - 1` variables are partialed out and `p` is at least 2. A categorical variable with a levels counts as a - 1 predictors. It is the `p` of `ss_power_R2`, `ci_R2`, and `ci_reg_coef`: the correlation concerns one predictor of a regression with `p` predictors. In the notation of a model comparison, q is the number of predictors removed from the full model to give the restricted model ($q = p_{full} - p_{restricted}$, the predictors being tested), and the number of variables partialed out is $p - q$; this correlation concerns one predictor, so $q = 1$ and $p - 1$ variables are partialed out. Scalar or vector.

Details

The formula is the same as for the simple *r* but with the degrees of freedom reduced from N to $N - (p - 1)$:

$$\mathrm{E}[r_{XY \cdot Z} \mid \rho_{XY \cdot Z}, N, p] \;=\; \rho_{XY \cdot Z} \cdot {}_2F_1\!\left(\tfrac{1}{2},\, \tfrac{1}{2};\, \tfrac{N-p+2}{2};\, \rho^2_{XY \cdot Z}\right) \cdot \frac{\Gamma\!\left(\frac{N-p+1}{2}\right)^2}{\Gamma\!\left(\frac{N-p}{2}\right)\,\Gamma\!\left(\frac{N-p+2}{2}\right)}.$$

`expected_partial_r()` is especially useful at the *design stage*: sample size plans that assume $\rho_{XY \cdot Z}$ as the value to be observed will under-deliver on the expected width or power of a CI on the partial correlation by an amount that grows in p.

Generalization of Olkin-Pratt. Under multivariate normality, the partial correlation $r_{XY \cdot Z}$ computed from a sample of size N has the same sampling distribution as a simple Pearson correlation from a sample of size $N - (p - 1)$ (Anderson, 2003, Theorem 4.3.5, Section 4.3.2, p. 143, who

credits the derivation to Fisher). The bias formula for the simple *r* (Hotelling, 1953; Olkin & Pratt, 1958) therefore applies directly to the partial *r* with the substitution $N \to N - (p-1)$. The Olkin-Pratt (1958) unbiased estimator extends in the same way: given an observed $r_{XY \cdot Z}$, the unbiased estimator of $\rho_{XY \cdot Z}$ is $r \cdot {}_2F_1(1/2, 1/2; (N-p-1)/2; 1-r^2)$ (Olkin & Pratt, 1958, Section 2.3, where the partial correlation is shown to have the simple correlation's density at the reduced sample size, so their estimator applies with that substitution).

Magnitude of the bias. To leading order, from Hotelling's (1953) expansion of the simple-correlation expectation applied at the reduced sample size:

$$\rho_{XY \cdot Z} - \mathrm{E}[r_{XY \cdot Z}] \;\approx\; \rho_{XY \cdot Z}(1 - \rho^2_{XY \cdot Z})/[2(N-p)],$$

matching the sign convention of the returned `bias` column, $\rho_{XY \cdot Z} - \mathrm{E}[r_{XY \cdot Z}]$, which is positive for positive $\rho_{XY \cdot Z}$. For $\rho_{XY \cdot Z} = 0.4$, $N = 30$, $p = 3$, the exact bias is about $+0.00624$; for $N = 30$, $p = 11$, about $+0.00888$.

Tuning the series convergence. The underlying ${}_2F_1$ series is summed by forward recurrence and stops when the relative contribution falls below `getOption("dmar.expected_r.tol",1e-15)`, with a maximum of `getOption("dmar.expected_r.max_iter", 5000L)` terms. These are the same options as for `expected_r`; users almost never need to change them.

Value

A `data.frame` with one row per (`true_cor`, `N`, `p`) input and the columns `true_cor`, `N`, `p`, `expected_partial_r`, `bias`, and `relative_bias`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Anderson, T. W. (2003). *An introduction to multivariate statistical analysis* (3rd ed.), Sections 4.2 and 4.3. Wiley. (Theorem 4.3.5, Section 4.3.2, p. 143: the sample partial correlation based on N observations, with k variables partialed out, has cdf $F[r \mid N-k, \rho]$; Anderson writes the count of partialed variables as $p-q$ in his notation, and here it is $p-1$.)

Hotelling, H. (1953). New light on the correlation coefficient and its transforms. *Journal of the Royal Statistical Society, Series B, 15*(2), 193–232.

Olkin, I., & Pratt, J. W. (1958). Unbiased estimation of certain correlation coefficients. *The Annals of Mathematical Statistics, 29*(1), 201–211.

See Also

`expected_r`, `var_partial_r`, `ss_aipe_partial_r`

Other effect size estimates: `cles()`, `cliff_delta()`, `correction_for_attenuation()`, `eta_squared()`, `eta_squared_generalized()`, `expected_r()`, `expected_smd()`, `nnt_from_smd()`, `omega_squared()`, `probability_of_superiority_paired()`, `proportion_of_superiority()`, `responder_analysis()`, `smd_trimmed()`

Examples

```
# 1. A single value: true_cor = 0.4, N = 30, three variables partialed out
#    (p = 4 predictors: X and the three variables).
expected_partial_r(true_cor = 0.4, N = 30, p = 4)

# 2. Bias grows with p at fixed (true_cor, N):
expected_partial_r(true_cor = 0.4, N = 30, p = c(2, 3, 6, 11, 21))

# 3. Partialing one variable (p = 2) at N = 11 is distributed exactly as
#    a simple correlation at N = 10 (Anderson, 2003, Theorem 4.3.5):
expected_partial_r(true_cor = 0.5, N = 11, p = 2)$expected_partial_r
expected_r(true_cor = 0.5, N = 10)$expected_r
```

---

expected_r *Exact Expected Value of the Sample Pearson Correlation Given ρ and N*

Description

Computes $\mathrm{E}[r \mid \rho, N]$, the exact expected value of the sample Pearson product-moment correlation coefficient under bivariate normality, using the Olkin-Pratt (1958) closed form built on Hotelling's (1953) exact density. The sample r is downward-biased as an estimator of the population ρ, a fact known since Soper (1913) and Fisher (1915). The exact expected value requires summing a Gauss hypergeometric series (see Details), which this function does, so that users can see the exact bias and decide whether to apply the Olkin-Pratt (1958) unbiased estimator of ρ (see Details).

Usage

```
expected_r(true_cor, N)
```

Arguments

`true_cor`	Population correlation coefficient ρ. A numeric scalar or vector in the interval $[-1, 1]$.
`N`	Sample size on which the Pearson r would be computed. A scalar or vector of integers with $N \geq 4$. (At $N = 3$ the bias is defined but the exact formula is numerically delicate; we require $N \geq 4$ for well-conditioned computation.) When `true_cor` and `N` are both vectors they must have the same length or recycle cleanly.

Details

`expected_r()` is especially useful at the *design stage* of a study, where sample size planning typically proceeds from an assumed value of the population correlation ρ. Because the value of r a researcher should expect to observe on average is *smaller in absolute value* than the assumed ρ, plugging ρ directly into sampling-distribution machinery (precision of r, width of a confidence interval, power of a test of $H_0 : \rho = 0$) systematically over-promises on the realized precision or

power. Substituting `expected_r(true_cor, N)` for the bare ρ in such planning calculations corrects the leading-order over-promise. See **Examples** for a design-stage walk-through.

The exact formula. Under bivariate normality, if r is the sample Pearson correlation in a sample of size N,

$$\mathrm{E}[r \mid \rho, N] \;=\; \rho \cdot {}_2F_1\!\left(\tfrac{1}{2},\, \tfrac{1}{2};\, \tfrac{N+1}{2};\, \rho^2\right) \cdot \frac{\Gamma\!\left(\frac{N}{2}\right)^2}{\Gamma\!\left(\frac{N-1}{2}\right)\,\Gamma\!\left(\frac{N+1}{2}\right)}.$$

Here ${}_2F_1(a, b; c; z) = \sum_{k=0}^{\infty} \frac{(a)_k (b)_k}{(c)_k\, k!}\, z^k$ is the Gauss hypergeometric function with Pochhammer symbols $(x)_k = x(x+1)\cdots(x+k-1)$ (Hotelling, 1953, Section 7 for the moments of r, Section 3, equation 25, for the exact density; Olkin & Pratt, 1958, equation 3.2, for this closed form). For $\rho^2 < 1$ the series converges absolutely; this implementation sums by a numerically stable forward recurrence and stops when the relative contribution of the next term falls below a tolerance. The `bias` column is computed directly, as ρ times one minus the product of the series and the gamma ratio, formed from their logarithms, and not as the difference $\rho - \mathrm{E}[r]$. At a large N the two numbers share their leading digits (about eight of them at $N = 10^8$), and the subtraction would leave the bias only the digits after those.

Tuning the series convergence (rarely needed). Two internal tuning constants control the ${}_2F_1$ series summation:

- `dmar.expected_r.tol`, relative tolerance for stopping the series (default `1e-15`). The summation stops once a term is below this fraction of the sum of the terms after the leading 1, the only part of the series on which the bias depends.
- `dmar.expected_r.max_iter`, maximum number of series terms (default `5000L`). If the last term summed is still not below `dmar.expected_r.tol` times the whole series, a non-convergence warning is issued.

Users who need different values (e.g., for very near-boundary $|\rho|$ where the series converges slowly) can set them via `options()`, for example `options(dmar.expected_r.tol = 1e-12)`. They are deliberately hidden from the function signature so as not to clutter the everyday user's view of the call.

Why the bias is present even though s^2 is unbiased for σ^2. The downward bias arises because r is a nonlinear function of unbiased sample moments. By Jensen's inequality and the concavity of the square root in the denominator of r, the expectation of the ratio is not the ratio of the expectations. The bias is largest when N is small or $|\rho|$ is moderate; as $N \to \infty$, $\mathrm{E}[r] \to \rho$.

Sign and magnitude. The bias $\rho - \mathrm{E}[r]$ has the same sign as ρ and is approximately $\rho(1 - \rho^2)/[2(N-1)]$ to leading order (Fisher, 1915; Hotelling, 1953; Ghosh, 1966); the exact formula above incorporates all higher-order corrections. For $\rho = 0.5$, $N = 10$, the bias is about $+0.021$; for $\rho = 0.5$, $N = 30$, about $+0.0065$.

Olkin-Pratt (1958) unbiased estimator of ρ. The companion to this expected-value calculation is the Olkin-Pratt unbiased estimator of ρ given an observed r:

$$\tilde{\rho}_{\mathrm{OP}}(r, N) \;=\; r \cdot {}_2F_1\!\left(\tfrac{1}{2},\, \tfrac{1}{2};\, \tfrac{N-2}{2};\, 1 - r^2\right).$$

Olkin & Pratt (1958) prove $\mathrm{E}[\tilde{\rho}_{\mathrm{OP}}(r, N) \mid \rho, N] = \rho$ exactly, for every $\rho \in (-1, 1)$ and $N \geq 4$. The commonly quoted first-order approximation $\tilde{\rho} \approx r\,[1 + (1 - r^2)/(2(N-3))]$ (Olkin, 1967) is the truncation of the OP series at $k = 1$; the full series is what makes the estimator exactly unbiased.

Size of the bias relative to sampling variability. The standard deviation of r is approximately $(1-\rho^2)/\sqrt{N-1}$, so the bias of a single r is about $\rho/(2\sqrt{N-1})$ of its standard deviation, whether

the coefficient is tested, given a confidence interval, or used as a reliability, validity, or planning value; at $\rho = 0.8$ and $N = 20$ the exact ratio is 0.09. When K correlations from samples of the same N and the same ρ are averaged, as in a meta-analysis, the bias of the average equals the bias of a single r, while the standard error of the average falls as $1/\sqrt{K}$; the bias is then about $\rho\sqrt{K}/(2\sqrt{N-1})$ standard errors of the average, and at $\rho = 0.8$, $N = 20$, and $K = 50$ the exact ratio is 0.63.

Connection to Fisher's Z transform. The variance-stabilizing transform $z = \tanh^{-1}(r)$, proposed in passing in Fisher (1915, p. 521) and developed in Fisher (1921), has approximate variance $1/(N-3)$ regardless of ρ, but $\mathrm{E}[z]$ also carries a small-sample bias of order $1/N$ (Hotelling, 1953, Section 8). Hotelling (1953, Sections 9–10) gives bias-adjusted and variance-stabilized refinements of Z, e.g. $z - (3z + r)/(4n)$.

Value

A `data.frame` with one row per (`true_cor`, `N`) input and the columns

- `true_cor`, the input population correlation,
- `N`, the input sample size,
- `expected_r`, $\mathrm{E}[r \mid \rho, N]$,
- `bias`, $\rho - \mathrm{E}[r \mid \rho, N]$ (the amount by which the sample r underestimates ρ on average),
- `relative_bias`, `bias / true_cor` when $\rho \neq 0$, and `NA` when $\rho = 0$.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Anderson, T. W. (2003). *An introduction to multivariate statistical analysis* (3rd ed.), Section 4.2. Wiley.

Fisher, R. A. (1915). Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika, 10*(4), 507–521.

Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron, 1*, 3–32.

Ghosh, B. K. (1966). Asymptotic expansions for the moments of the distribution of correlation coefficient. *Biometrika, 53*(1/2), 258–262.

Hotelling, H. (1953). New light on the correlation coefficient and its transforms. *Journal of the Royal Statistical Society, Series B, 15*(2), 193–232. (Discussion, pp.\ 225–232.)

Olkin, I. (1967). Correlations revisited. In J. C. Stanley (Ed.), *Improving experimental design and statistical analysis* (pp.\ 102–128). Rand McNally.

Olkin, I., & Pratt, J. W. (1958). Unbiased estimation of certain correlation coefficients. *The Annals of Mathematical Statistics, 29*(1), 201–211.

Soper, H. E. (1913). On the probable error of the correlation coefficient to a second approximation. *Biometrika, 9*(1/2), 91–115.

Soper, H. E., Young, A. W., Cave, B. M., Lee, A., & Pearson, K. (1917). On the distribution of the correlation coefficient in small samples. Appendix II to the papers of "Student" and R. A. Fisher: A cooperative study. *Biometrika, 11*(4), 328–413.

Stuart, A., & Ord, J. K. (1994). *Kendall's advanced theory of statistics, Vol.\ 1: Distribution theory* (6th ed.), Section 16.32. Edward Arnold.

See Also

ci_r, cor, cor.test

Other effect size estimates: cles(), cliff_delta(), correction_for_attenuation(), eta_squared(), eta_squared_generalized(), expected_partial_r(), expected_smd(), nnt_from_smd(), omega_squared(), probability_of_superiority_paired(), proportion_of_superiority(), responder_analysis(), smd_trimmed()

Examples

```
# 1. A single value: true_cor = 0.5, N = 10. The sample r is downwardly
#        biased by about 0.021, roughly 4% of true_cor.
expected_r(true_cor = 0.5, N = 10)

# 2. Bias as a function of N for fixed true_cor. The bias is roughly
#        true_cor * (1 - true_cor^2) / (2(N - 1)) to leading order; as N
#        grows it shrinks toward zero.
expected_r(true_cor = 0.5, N = c(5, 10, 20, 50, 100, 500))

# 3. Bias as a function of true_cor for fixed N. The bias is zero at
#        true_cor = 0 and true_cor = +/- 1, and largest near
#        true_cor = +/- 0.6.
expected_r(true_cor = seq(0, 0.95, by = 0.05), N = 10)

# 4. The Olkin-Pratt unbiased estimator: invert the bias for an
#        observed sample r.
set.seed(113)
x <- rnorm(20); y <- 0.4 * x + sqrt(1 - 0.4^2) * rnorm(20)
r_obs <- cor(x, y)
r_obs

# Olkin-Pratt unbiased estimator of rho:
op_unbiased <- function(r, N, tol = 1e-15, max_iter = 5000) {
  z <- 1 - r^2; c_par <- (N - 2) / 2
  s <- 1; term <- 1
  for (k in seq_len(max_iter)) {
    term <- term * ((k - 0.5)^2) / ((c_par + k - 1) * k) * z
    s <- s + term
    if (abs(term) < tol * abs(s)) break
  }
  r * s
}
op_unbiased(r_obs, N = 20)

# 5. Design-stage use: a study planned around true_cor = 0.4 with N = 30.
```

```
#        Naive plug-in says we expect to observe r = 0.40 on average,
#        but the realized expected r is smaller, and that gap matters
#        for any precision- or power-based sample size calculation that
#        plugs in true_cor as if it were the expected sample r.
cor_planned <- 0.4
n_planned   <- 30
expected_r(true_cor = cor_planned, N = n_planned)

# 6. Tuning constants are hidden from the signature but tunable
#        through options() for the rare cases that need them (e.g.,
#        very near-boundary |true_cor| where the 2F1 series converges
#        slowly). The defaults rarely need to be changed.
options(dmar.expected_r.tol = 1e-12,
        dmar.expected_r.max_iter = 20000L)
expected_r(true_cor = 0.999, N = 5)
options(dmar.expected_r.tol = NULL,
        dmar.expected_r.max_iter = NULL)   # restore defaults
```

`expected_R2` *Expected Value of the Squared Multiple Correlation Coefficient*

Description

Computes the expected value of the observed squared multiple correlation coefficient given the population squared multiple correlation coefficient, the sample size, and the number of predictors. The sample R^2 is a positively biased estimator of its population value, and the expected value quantifies how large that bias is for a particular design.

Usage

```
expected_R2(true_R2, N, p)
```

Arguments

`true_R2`	Population squared multiple correlation coefficient, ρ^2, the value at which the expected value is evaluated
`N`	Sample size
`p`	The number of predictor variables

Details

Uses the hypergeometric function as discussed in section 28 of Stuart, Ord, and Arnold (1999) in order to obtain the *correct* value for the squared multiple correlation coefficient. Many times an exact value is given that ignores the hypergeometric function. This function yields the correct value.

Value

A 1-row `data.frame` with columns `term` and `value`. The `term` value is `"expected_R2"` and `value` is the expected value of R^2 under random sampling, as `var_R2` reports its variance in the row `var_R2`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*, 524–555. doi:10.1080/00273170802490632

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Olkin, I., & Pratt, J. W. (1958). Unbiased estimation of certain correlation coefficients. *The Annals of Mathematical Statistics, 29*(1), 201–211.

Stuart, A., Ord, J. K., & Arnold, S. (1999). *Kendall's advanced theory of statistics, volume 2A: Classical inference and the linear model* (6th ed.). Arnold.

See Also

`ss_aipe_R2`, `ci_R2`, `var_R2`

Examples

```
expected_R2(.5, 10, 5)
expected_R2(.5, 25, 5)
expected_R2(.5, 50, 5)
expected_R2(.5, 100, 5)
expected_R2(.5, 1000, 5)
expected_R2(.5, 10000, 5)
```

---

expected_smd *Exact Expected Value of Cohen's* d *(and Hedges'* g *Bias Correction)*

Description

Computes $\mathrm{E}[\hat{d} \mid \delta, n_1, n_2]$, the exact expected value of the sample standardized mean difference (Cohen's *d*, with pooled variance) under normality within each group with a common variance, from the noncentral *t* sampling distribution (Hedges, 1981). The sample *d* is upward-biased as an estimator of the population δ: $\mathrm{E}[\hat{d}] = \delta / J(df)$, where

$$J(df) = \frac{\Gamma(df/2)}{\sqrt{df/2}\,\Gamma((df-1)/2)}$$

is Hedges' (1981) bias-correction factor (< 1 for finite df, tending to 1 as $n \to \infty$). Hedges' g, the unbiased estimator of δ, is then $g = J \cdot \hat{d}$. The same $J(df)$ is the workhorse of `smd` when `unbiased = TRUE`.

Usage

```
expected_smd(true_smd, n_1, n_2 = NULL)
```

Arguments

`true_smd`	Population standardized mean difference, δ in the formulas. A numeric scalar or vector.
`n_1`	Sample size in the first group. Scalar or vector.
`n_2`	Sample size in the second group. Scalar or vector. If omitted, defaults to `n_1` (balanced design).

Details

`expected_smd()` is especially useful at the *design stage* of a study, where sample size planning typically proceeds from an assumed population standardized mean difference δ. Because the value of $\hat{d}$ a researcher should expect to observe on average is *larger in absolute value* than δ, plugging δ directly into sampling-distribution machinery (precision of $\hat{d}$, width of a confidence interval, power of a test of $H_0 : \delta = 0$) over-promises on the realized precision when the precision is expressed on the $\hat{d}$ scale. Substituting `expected_smd(true_smd, n_1, n_2)` for the bare δ corrects this leading-order bias.

Derivation. When the scores are normal within each group with a common variance, the observed t-statistic $t = \hat{d}\sqrt{n_1 n_2/(n_1 + n_2)}$ follows a noncentral t distribution with $df = n_1 + n_2 - 2$ degrees of freedom and noncentrality parameter $\lambda = \delta\sqrt{n_1 n_2/(n_1 + n_2)}$. The expected value of a noncentral t variate equals $\lambda/J(df)$ (Johnson, Kotz, & Balakrishnan, 1995, Section 31.3), so $\mathrm{E}[\hat{d}] = \mathrm{E}[t]/\sqrt{n_1 n_2/(n_1 + n_2)} = \delta/J(df)$. The bias $\mathrm{E}[\hat{d}] - \delta = \delta\,(1 - J)/J$ is positive when $\delta > 0$.

Magnitude of the correction. $J(df) \approx 1 - 3/(4\,df - 1)$ to leading order (Hedges & Olkin, 1985, p.\ 81). For $\delta = 0.5$, $n_1 = n_2 = 10$ ($df = 18$), $J \approx 0.957$, so $\mathrm{E}[\hat{d}] \approx 0.522$ and the upward bias is about 4%. For $n_1 = n_2 = 50$ the bias is under 1%; for $n_1 = n_2 = 5$ (very small samples) it exceeds 10%.

Connection to Hedges' *g*. The natural inverse of this function is Hedges' g: given an observed $\hat{d}$, the unbiased estimator of δ is $g = J(df)\hat{d}$, which satisfies $\mathrm{E}[g \mid \delta] = \delta$ exactly under the same noncentral t model. `smd` with `unbiased = TRUE` returns g.

Value

A `data.frame` with one row per (`true_smd`, `n_1`, `n_2`) input and the columns

- `true_smd`, the population SMD,
- `n_1`, `n_2`, group sample sizes,
- `expected_smd`, $\mathrm{E}[\hat{d} \mid \delta]$,
- `bias`, $\mathrm{E}[\hat{d}] - \delta$ (the amount by which $\hat{d}$ overestimates δ on average),

- `j_correction`, the Hedges (1981) correction factor $J(df)$ used to compute the unbiased g, where $df = n_1 + n_2 - 2$.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. Academic Press. (See Section 5, equations 6 and 9.)

Johnson, N. L., Kotz, S., & Balakrishnan, N. (1995). *Continuous univariate distributions, volume 2* (2nd ed.), Section 31.3. Wiley.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

See Also

`smd`, `ci_smd`, `ss_aipe_smd`, `expected_r`, `expected_R2`

Other effect size estimates: `cles()`, `cliff_delta()`, `correction_for_attenuation()`, `eta_squared()`, `eta_squared_generalized()`, `expected_partial_r()`, `expected_r()`, `nnt_from_smd()`, `omega_squared()`, `probability_of_superiority_paired()`, `proportion_of_superiority()`, `responder_analysis()`, `smd_trimmed()`

Examples

```
# 1. Balanced design: true_smd = 0.5, n = 10 per group.
expected_smd(true_smd = 0.5, n_1 = 10)

# 2. Bias as a function of n at a fixed true_smd.
expected_smd(true_smd = 0.5, n_1 = c(5, 10, 20, 50, 100, 500))

# 3. Unbalanced design.
expected_smd(true_smd = 0.5, n_1 = 30, n_2 = 60)

# 4. Design-stage use: an a-priori true_smd of 0.4, planned n of 40/group.
#        The d we should expect to observe on average is slightly larger.
expected_smd(true_smd = 0.4, n_1 = 40)
```

`factorial_anova` *Between-Subjects Factorial ANOVA for Unbalanced Designs*

Description

Fits a between-subjects factorial ANOVA (two or more crossed factors) and returns each effect's sum of squares, F test, and partial η^2 and partial ω^2 with confidence intervals, using a sum-of-squares type the user chooses. It is built for the case that makes the choice matter, an *unbalanced* design (unequal cell sizes), where the Type I, II, and III sums of squares differ. For a balanced design the three types coincide and the choice is immaterial.

Usage

```
factorial_anova(formula, data, ss_type = 3L, conf_level = 0.95)
```

Arguments

`formula`	A two-sided `formula` naming the numeric response and the crossed factors, for example `y ~ A * B` or `y ~ A * B * C`. Predictors are coerced to factors. Write the full crossing you want tested; `A * B` expands to `A + B + A:B`.
`data`	A `data.frame` containing the response and the factors.
`ss_type`	The sum-of-squares type: `1`, `2`, or `3` (equivalently `"I"`, `"II"`, or `"III"`). Default `3` (Type III), the common reporting default. See Details for what each type tests.
`conf_level`	Confidence level for the effect size confidence intervals. Default `0.95`.

Details

Every sum of squares is computed as a model comparison, the increase in error sum of squares when an effect's parameters are removed from a model, following the model comparison development of Maxwell, Delaney, and Kelley (2027, Chapter 7). The computation uses only base R (`stats`); it does not depend on the **car** package, though it agrees with `car::Anova()` to numerical precision.

The types as model comparisons. A sum of squares for an effect is the increase in the error sum of squares when the effect's parameters are dropped from the model, `SS = E(restricted) - E(full)`. The three conventional types differ only in which other effects the full and restricted models hold in common (Maxwell, Delaney, and Kelley, 2027, Chapter 7; Overall and Spiegel, 1969):

- **Type I (sequential).** Each effect is adjusted only for the effects listed before it in `formula`: `SS(A)`, then `SS(B | A)`, then `SS(A:B | A, B)`. The parts sum to the model sum of squares, but the answer depends on the order of the terms.
- **Type II.** Each effect is adjusted for every other effect that does not contain it (it respects marginality): a main effect is adjusted for the other main effects but not for the interactions that contain it. Type II is the most powerful choice when the interaction is null and does not depend on how the factors are coded (Overall and Spiegel's Method 2; Appelbaum and Cramer, 1974).

- **Type III.** Each effect is adjusted for all other effects, including the higher order interactions that contain it. This tests each main effect as a contrast on the unweighted marginal means, so it is computed here with sum-to-zero contrasts (`contr.sum`), which is what makes the Type III main-effect test the intended one. It is the default reported by many programs.

For a balanced design the effects are orthogonal and the three types are identical; the distinction is a property of unbalanced (nonorthogonal) data.

Reading the main effects when an interaction is present. When an interaction is real, the marginal main-effect tests, of any type, are usually not the question of interest; examine the interaction and the simple effects instead (Maxwell, Delaney, and Kelley, 2027). See the vignette *Sums of Squares in Nonorthogonal Designs* for a worked comparison.

Effect sizes. Partial η^2 and partial ω^2 are formed for each effect from its F, its degrees of freedom, and the error degrees of freedom, with noncentral F confidence intervals (`ci_eta_squared`, `ci_omega_squared`). The confidence limits are those of the interval for the population proportion of variance the effect accounts for (Kelley, 2007). Partial η^2 and partial ω^2 are two point estimators of that same population quantity, partial ω^2 correcting the upward bias of partial η^2, so the two estimators differ but share the interval.

Estimability. All cells must be filled. If a factor combination is empty the design is rank deficient and the factorial effects are not all estimable; the function stops with a message rather than return a value that depends on an arbitrary choice.

Value

A `data.frame` (class `dmar_tbl`) with one row per effect plus a `Residuals` row. Columns are the effect label (`effect`), the sum of squares (`SS`), degrees of freedom (`df`), the F statistic (`F_value`) and its p-value (`p_value`), and partial η^2 and partial ω^2 with their lower and upper confidence limits (`eta_squared_partial`, `eta_squared_partial_ci_lower`, `eta_squared_partial_ci_upper`, and the same three columns for `omega_squared_partial`). The chosen sum-of-squares type is recorded on the object (`attr(x, "ss_type")`) and printed beneath the table. The residual row carries only `SS` and `df`. Stored values keep full precision; the display rounds (see `dmar_tbl`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Appelbaum, M. I., & Cramer, E. M. (1974). Some problems in the nonorthogonal analysis of variance. *Psychological Bulletin, 81*(6), 335–343.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 7 on higher order between-subjects designs.)

Overall, J. E., & Spiegel, D. K. (1969). Concerning least squares analysis of experimental data. *Psychological Bulletin, 72*(5), 311–322.

See Also

ancova for a covariate-adjusted one-way design, mixed_anova for mixed-model (fixed and random) F ratios, manova_split_plot for the multivariate mixed design, and ci_eta_squared / ci_omega_squared for the effect size intervals.

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# An unbalanced two-factor design: IQ gain in the pygmalion expectancy
# experiment, by treatment and by lower (grades 1 and 2) versus upper
# (grades 3 through 6) grades, where the expectancy effect concentrated
# in the lower grades. The cell sizes are unequal, so the Types differ.
pyg <- pygmalion
pyg$grade_band <- factor(ifelse(pyg$grade <= 2, "lower", "upper"))
factorial_anova(iq_gain ~ treatment * grade_band, data = pyg)

# The same design under Type II (adjusts each main effect for the other
# main effect, but not for the interaction). Here the main effect F
# statistics drop, and a warning reports that the affected noncentral F
# lower limits are clamped to 0.
factorial_anova(iq_gain ~ treatment * grade_band, data = pyg, ss_type = 2)
```

---

factor_congruence *Tucker's Congruence Coefficients Between the Factors of Two Solutions*

Description

Compares the factors of two solutions for the same items by Tucker's congruence coefficient, after matching each factor of one solution with a factor of the other. The two solutions can come from two samples, two rotations, two estimators, or two factor counts, and each can be a loading matrix, a table from fa_exploratory(), or a **lavaan** fit. The factors of y are paired one to one with those of x by the assignment that maximizes the total absolute congruence, and a factor whose match is negative is reflected. Each factor is reported with its coefficient and with the largest absolute coefficient it has with any other factor of the other solution, so that a close competitor is in view, and the full matrix of coefficients travels with the table.

Usage

```
factor_congruence(
  x,
  y,
```

```
  match = c("congruence", "order"),
  loadings = c("pattern", "structure")
)
```

Arguments

x	The reference solution, in one of three forms. A numeric matrix of loadings with items in rows and factors in columns (a numeric vector is one factor, and a `"loadings"` object, such as `psych::fa()$loadings` or `factanal()$loadings`, is used as its matrix). A table returned by `fa_exploratory()`, whose standardized pattern loadings and factor correlations are used. Or a single-group, single-level **lavaan** fit, such as `fa_confirmatory(output = "fit")`, `lavaan::cfa()`, or `lavaan::efa(output = "lavaan")`, whose standardized loadings and model implied factor correlations are used. A table of results from `fa_confirmatory()`, `cfa_1()`, or `cfa_2()` is not a fit and is refused with a pointer to `output = "fit"`. Items are named by the row names of a matrix, the names of a vector, or the item names of a fit.
y	The solution compared with `x`, in any of the same forms. Its factors are matched to those of `x`.
match	How the factors are paired: `"congruence"` (default), the one-to-one assignment that maximizes the sum of the absolute coefficients, with each matched factor of `y` reflected when its coefficient is negative; or `"order"`, column *j* of `y` with column *j* of `x`, signed and never reflected, for solutions whose factors are already aligned. See Details.
loadings	`"pattern"` (default) compares columns of the pattern matrix, the loadings themselves; `"structure"` compares columns of the structure matrix, the pattern times the factor correlations, the correlations of the items with the factors. The structure needs the factor correlations, so it is available for `fa_exploratory()` tables and **lavaan** fits, not for a matrix; a matrix is used as supplied, and for a structure comparison of matrices the structure matrices themselves are passed under the default.

Details

The coefficient. For two columns of loadings a and b on the same n items, Tucker's congruence coefficient is

$$\phi(a, b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2}},$$

the cosine of the angle between the two columns (Lorenzo-Seva & ten Berge, 2006, eq. 1). Burt suggested it in 1948, and it became known as Tucker's congruence coefficient through Tucker's report of 1951 (Lorenzo-Seva & ten Berge, 2006, p. 57). It is not the Pearson correlation of the two columns, which would subtract their means first. The coefficient is unchanged when a column is multiplied by a positive constant, so it compares the pattern of the loadings rather than their size. It changes when a constant is added, as it should, since loadings of .4, .4, and 0 and loadings of .1, .1, and $-.3$ differ by a constant and are read as different factors (their coefficient is .43, their correlation 1). It is unchanged when an item's sign is reversed in both solutions, and it reverses its sign when one of the two factors is reflected (Lorenzo-Seva & ten Berge, 2006, pp. 57–58). The columns are usually those of the pattern matrix, though those of the structure matrix can be compared as well (p.

57). In an oblique solution the two differ, and `loadings` chooses between them; in an orthogonal one they coincide.

Rescaling a whole column leaves the coefficient alone, but rescaling an item does not, and an item's loadings in the original metric scale with its standard deviation, which differs from sample to sample. A fit is therefore read in the standardized metric. A matrix is used as supplied, so two matrices should be in the same metric.

Why matching is part of the comparison. The order in which a solution lists its factors, and the sign of each, are arbitrary: a rotation can list the verbal factor first in one sample and second in another, and reflecting a factor (reversing the signs of its loadings and of its correlations with the other factors) gives an equivalent solution. Before two solutions can be compared, each factor of `y` has to be paired with a factor of `x`. Under `match = "congruence"` the pairing is the one-to-one assignment that maximizes the sum of the absolute coefficients over the pairs, found by the Hungarian method (Kuhn, 1955), and each matched factor of `y` whose coefficient is negative is then reflected, so the reported coefficients are not negative. The assignment is optimal over the whole matrix rather than factor by factor: pairing each factor of `x` with its best factor of `y` can give two factors the same partner, and taking the largest coefficient first and then the largest of what remains can give a lower total. Assignments whose totals agree to within rounding are decided in favor of the listed order: the first factor of `x` takes the earliest factor of `y` it can, then the second, and so on. Under `match = "order"` column *j* of `y` is paired with column *j* of `x`, the coefficients keep their signs, and nothing is reflected; this suits solutions whose factors are aligned by construction, such as two confirmatory fits of one model, and a negative coefficient then shows a factor whose direction differs.

When the two solutions have different numbers of factors, as many pairs are formed as the smaller solution has factors, and each surplus factor of the larger solution has a row with no partner and no coefficient. Its `next_largest` shows how close it comes to a factor already taken, which is how a factor of one solution that splits into two in the other shows itself.

Items are aligned by name when both solutions name them, and the two sets of items must then be the same; otherwise they are aligned by position, and the two numbers of items must be the same.

The matched value is the largest of several candidates. Matching chooses for each factor the partner it agrees with best, so a matched coefficient is a maximum, larger on average than the coefficient of a pairing fixed in advance. The `next_largest` column gives each factor's strongest competitor, the largest absolute coefficient it has with any factor of the other solution other than its partner. A matched value close to its `next_largest` marks a pairing that a small change in the data could reverse, and the `congruence_matrix` attribute, printed under the table, shows every candidate.

Chance congruence is not zero. Two columns of loadings unrelated to each other still have a coefficient different from zero. For two unrelated factors whose loadings point in random directions, Korth and Tucker (1975, p. 366) give the expected squared coefficient as $1/n$ for n items (for six items a root mean square of about .41). Matching raises the chance values, and rotating the two solutions toward each other raises them further. Korth and Tucker gave each of two unrelated patterns an orthogonal transformation chosen to make the two agree as closely as possible (pp. 364–365); for two factors on ten items, the first and largest coefficient after that transformation was typically about .64 (their Table 4, p. 368, a mean on the Fisher *Z* scale converted back). Reordering and reflecting factors are orthogonal transformations too, so for the same two patterns no coefficient this function reports exceeds that first coefficient. Columns whose loadings are all positive, as a general factor's often are, agree further by construction: two unrelated columns of loadings drawn independently and uniformly between 0 and 1 have a coefficient that approaches 3/4 as the number

of items grows. A coefficient is therefore read against what unrelated solutions of the same shape would give, not against zero.

Reading the coefficient. Tucker declined in his 1951 report to name a value below 1 that would count as identity of two factors until experience with the coefficient had accumulated (Lorenzo-Seva & ten Berge, 2006, p. 58). Lorenzo-Seva and ten Berge (2006) asked 56 university professors who use exploratory factor analysis in their research to rate, on a five-point scale, the similarity of pairs of loading columns constructed to have coefficients from .62 to .97. The mean ratings tracked the coefficient closely (r = .974 between the coefficient and the mean rating), and the raters' mean judgments led the authors to read values from .85 to .94 as fair similarity and values above .95 as equality. The individual ratings varied widely, and they depended on the number of items and on the judges' research field; the authors concluded that a rule of thumb based on the judgment of a single researcher is hazardous (p. 61). Their study describes how analysts judged similarity, not a property of the coefficient, and the table reports the coefficient and applies no such reading.

What the function does not do. It rotates neither solution: it compares them as they were rotated. Rotating one solution toward the other before computing the coefficients, as is often done (Lorenzo-Seva & ten Berge, 2006, p. 57), is a separate step. It reports no standard error, interval, or test for the coefficients. `procrustes_phi()` computes the same coefficient for one pair of loading vectors that the user has already paired, with a permutation p-value, and does no matching.

Value

A `dmar_tbl` data frame (classes `dmar_factor_congruence`, `dmar_tbl`, and `data.frame`) with one row per factor of `x`, in the order of `x`, followed by a row for each factor of `y` left without a partner. Its columns are `factor_x` and `factor_y`, the names of the paired factors (`NA` for a factor without a partner); `congruence`, the coefficient of the pair after reflection (signed under `match = "order"`; `NA` without a partner); `reflected`, whether the factor of `y` was reflected to match (`NA` without a partner); and `next_largest`, the largest absolute coefficient between the row's factor (its factor of `x`, or its factor of `y` in the row of a factor of `y` without a partner) and any factor of the other solution other than its partner (`NA` when there is none). The attributes are `congruence_matrix`, the signed coefficients of every factor of `x` (rows) with every factor of `y` (columns), before matching or reflection; `loadings_aligned`, the loadings of `y` that were compared, with the items in the order of `x` and the factors in the order of their partners, reflected where matched with a reflection, followed by any factor without a partner; `Phi_aligned`, the factor correlations of `y` in that order, $D\Phi D$ with D the diagonal matrix of the reflections, when `y` carries them (absent for a matrix); `items`, the item names in the order of `x` (their positions when neither solution names its items); `match` and `loadings`, the arguments; and `inputs`, a list naming the kind of each input (`"matrix"`, `"fa_exploratory"`, or `"lavaan"`) and whether the items were aligned by name or by position. A `measured_terms` attribute names the two columns of coefficients, so the display prints a coefficient of 1 as 1.000 beside the others (see `dmar_tbl`). The print method states the coefficient, the counts, and the matching rule above and below the table, and prints the full matrix, to the decimal places of the table (`print(x, digits = 5)` shows five in both). `tidy()` gives one row per row of the table, with `term` `"factor_x | factor_y"` (`"(none)"` standing in for the missing name of a factor without a partner, as in `"X3 | (none)"`), `estimate` (the coefficient), `reflected`, and `next_largest`; `glance()` gives `n_items`, `n_factors_x`, `n_factors_y`, `n_matched`, `match`, `loadings`, and `mean_congruence`, the mean coefficient over the matched pairs.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Korth, B., & Tucker, L. R. (1975). The distribution of chance congruence coefficients from simulated data. *Psychometrika, 40*(3), 361–372. doi:10.1007/BF02291763

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2*(1–2), 83–97. doi:10.1002/nav.3800020109

Lorenzo-Seva, U., & ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology, 2*(2), 57–64. doi:10.1027/16142241.2.2.57

See Also

`procrustes_phi` for the coefficient of one pair of loading vectors with a permutation *p*-value, `fa_exploratory` and `fa_confirmatory` for the solutions compared, and `dmar_tidiers` for `tidy()` and `glance()`.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
items <- c("t1_visual_perception", "t2_cubes", "t4_lozenges",
           "t6_paragraph_comprehension", "t7_sentence", "t9_word_meaning")
# Congruence compares loadings alone, so the two fits leave out the
# standard errors, tests, and intervals (point_estimates_only = TRUE).
oblimin <- fa_exploratory(holzinger_swineford, nfactors = 2, items = items,
                          seed = 113, point_estimates_only = TRUE)
geomin <- fa_exploratory(holzinger_swineford, nfactors = 2, items = items,
                         rotation = "geomin", seed = 113,
                         point_estimates_only = TRUE)

# The same two factors under two rotations, compared by their pattern
# loadings and then by their structure loadings.
factor_congruence(oblimin, geomin)
factor_congruence(oblimin, geomin, loadings = "structure")

# A loading matrix and a copy with its items in reverse order, its
# factors listed the other way around, and one factor reflected:
# matching recovers the pairing and the sign.
L <- matrix(c(0.7, 0.6, 0.8, 0.1, 0.0, 0.2,
              0.1, 0.2, 0.0, 0.7, 0.8, 0.6), ncol = 2,
            dimnames = list(paste0("item_", 1:6), c("A", "B")))
L_copy <- L[6:1, c("B", "A")]
L_copy[, "A"] <- -L_copy[, "A"]
```

```
matched <- factor_congruence(L, L_copy)
matched
attr(matched, "loadings_aligned")

# Paired by position instead, the coefficients keep their signs.
factor_congruence(L, L_copy, match = "order")
generics::glance(matched)
```

`factor_count` *Factor Count Comparison: Model Fit, Parallel Analysis, and MAP*

Description

Fits the exploratory common-factor model with one factor, two, and so on, and sets beside each count what bears on it: the fit of that model and the test of the previous count against it, its information criteria, the reduced eigenvalue at that rank against its parallel analysis reference, and Velicer's average squared partial correlation after that many components. The table applies no cutoff and chooses nothing: the lines beneath it state what each criterion shows, and the count is a judgment that combines them with theory and the interpretability of the rotated solutions, which `fa_exploratory` gives at a chosen count.

Usage

```
factor_count(
  data = NULL,
  S = NULL,
  N = NULL,
  items = NULL,
  max_factors = NULL,
  estimator = "ML",
  missing = "fiml",
  ordered = NULL,
  criterion = "BIC",
  conf_level = 0.9,
  B = 1000L,
  percentile = 0.95,
  seed = NULL
)

## S3 method for class 'dmar_factor_count'
print(x, digits = getOption("dmar.digits", 3L), ...)
```

Arguments

`data`	Numeric raw observations. Supply exactly one of `data` or `S`.
`S`	Named positive definite covariance matrix, with `N`.

`N` Sample size for covariance input, or a reference sample size for the parallel analysis of incomplete raw data (see `parallel_analysis`).

`items` The items to analyze, by name: columns of `data`, or rows and columns of `S`, which is then reduced to them, in the order given; only the reduced matrix need be finite and positive definite. `NULL`, the default, analyzes every column of `data` or of `S`. Names that cannot be read as at least three distinct items are refused with an error of class `dmar_items_invalid` (see `dmar_metadata`).

`max_factors` The largest count fitted. The default is the largest count whose model keeps at least one degree of freedom, the largest k with $(p-k)^2-(p+k) \geq 2$ for p items; a larger count is accepted up to the one with zero degrees of freedom, whose model reproduces the covariances exactly.

`estimator`, `missing`, `ordered`

As for `fa_exploratory`, which fits each count with `rotation = "none"`, since the fit of a count does not depend on the rotation. The missing data method is resolved once, as `fa_exploratory()` resolves it, and the same method serves every fit, the correlations of the eigenvalues and the MAP, and the parallel analysis. With ordered items and no item value missing, leaving `missing` out fits the items as they stand (listwise deletion with nothing to delete), exactly as `missing = "listwise"` does, and the `method` attribute records `"listwise"`; with a value missing, ordered items require an explicit `"pairwise"` or `"listwise"`, and leaving `missing` out is refused before any correlation, fit, or reference data set is computed.

`criterion` The information criterion by which the proper counts are compared, as `compare_sem` compares a set of models: `"BIC"` (the default), `"SABIC"`, `"HBIC"`, or `"AIC"`, in either case. IBIC and SPBIC, the information matrix based criteria of Bollen, Ray, Zavisca, and Harden (2012), are not offered: lavaan's exploratory fits carry singular information matrices (their parameterization counts the rotational constraints), which `sem_information_criteria` refuses for them.

`conf_level` Confidence level of the RMSEA intervals, default 0.90, the conventional level, whose limits decide the tests of close and not close fit at .05 (see `sem_fit_measures`); `conf_level = 0.95` gives 95 percent intervals. It sets nothing else: the Monte Carlo limits of the parallel analysis reference in the `parallel_analysis` attribute keep the 95 percent default of `parallel_analysis`.

`B` Number of reference datasets for the parallel analysis, a single whole number, default 1000; `0` omits the reference, and the table then carries the reduced eigenvalues alone. A positive `B` must be at least the number `parallel_analysis` requires at `percentile`: 200 at the default 95th percentile, 1000 at the 99th, 100 at the 90th or at the median (at least 10 reference datasets beyond the quantile and never fewer than 100). A smaller positive `B` is refused before any model is fit (class `dmar_count_invalid`); see `dmar_bootstrap`.

`percentile` Reference percentile of the parallel analysis, default 0.95.

`seed` Optional seed for the parallel analysis; default `NULL` uses the current random stream. A supplied seed restores the caller's stream.

`x` For `print()`, a table returned by `factor_count()`.

`digits` For `print()`, the decimal places of the display (default `getOption("dmar.digits", 3)`; see `dmar_tbl`).

`...` For `print()`, passed to `print.data.frame()`.

Details

The fit of each count. Each count is fitted by `fa_exploratory` with the same estimator and missing data method, and its `fit_measures` supply the chi square statistic with its degrees of freedom and p-value, the RMSEA with its interval, the CFI, TLI, and SRMR, under the same version policy (see `sem_fit_measures`; after FIML the indices are the corrected ones of Zhang and Savalei, 2023). The model with one factor fewer is nested in the model with one more, and `delta_chi_square`, `delta_df`, and `p_delta` give `lavaan::lavTestLRT()`'s test of the previous count against this one (the Satorra-Bentler scaled difference under `estimator = "MLR"`). The AIC, BIC, and sample size adjusted BIC are those of `sem_information_criteria`; smaller is preferred, and only the differences between rows carry meaning. Under `estimator = "ULS"` the statistic has no reference distribution, so the p-values, the RMSEA, CFI, TLI, difference tests, and information criteria are `NA` and the SRMR remains (see `fa_exploratory`).

Differences, weights, and evidence. The values of a criterion mean nothing on their own; what carries the evidence is the difference between two counts. The proper counts are therefore compared by `criterion` as `compare_sem` compares a set of models fitted to the same data: `criterion_delta` is each count's gap from the lowest value, `criterion_weight` is $\exp(-\Delta/2)$ normalized over the proper counts, and, for BIC and HBIC, which approximate -2 times the log marginal likelihood (Bollen et al., 2012), `criterion_log_evidence` is $\Delta/2$, the approximate log Bayes factor of the best count over this one, and the weights approximate posterior probabilities under equal prior odds (Burnham & Anderson, 2004); for AIC the weights are Akaike weights and for SABIC they carry no further interpretation, so the log evidence column is `NA` for both. The weights are conditional on the set of proper counts compared, and the line under the table gives the best count, its weight, and how far the nearest other count stands from it. Every count is fitted to the same cases under the same missing data method and mean structure, which `compare_sem()` checks. The comparison needs two proper likelihood fits, so under ULS, or with one proper count, these columns are `NA`.

A count whose model does not converge, or converges to an improper solution (lavaan's post-estimation check fails: a residual variance below zero, or a covariance matrix that is not positive definite), or whose likelihood has no maximum (a residual variance runs off below zero while the likelihood is flat or still rising, and the optimizer stopped somewhere along the way; see `fa_exploratory`), has no fit to report: its `solution` column says so, its fit columns are `NA`, and a line under the table names it. Such a count is informative in its own right, since the unrestricted model with that many factors drives an item's residual variance to or below zero on these data, but it is not a stopping rule: the counts beyond it are fitted and reported in the same way. `fa_exploratory` refuses an improper solution for the same reason.

Eigenvalues and their reference. The `eigenvalue` column holds the eigenvalues of the reduced correlation matrix, the item correlations with the communalities of a fixed one-factor maximum likelihood solution on the diagonal, as `parallel_analysis` forms them; the `reference_quantile` is that function's `percentile` of the same eigenvalue in `B` independent normal reference datasets, and `exceeds_reference` compares the two (Horn, 1965). The correlation matrix is the one `parallel_analysis()` uses: the full information maximum likelihood estimate of `cov_fiml` for incomplete continuous items, the polychoric correlations for ordered items, the sample correlations otherwise. The reference percentile is a null threshold, not a confidence interval for an eigenvalue or a confidence statement about the count. The parallel analysis is run when every item is ordered or none is, which is the interface `parallel_analysis()` supports.

The minimum average partial. Velicer's (1976) rule extracts principal components of the correlation matrix one at a time, partials the first m out of it, and averages the squared partial correlations that remain; no component is retained after the average reaches its minimum. The `map` column holds

the average after as many components as the row has factors; the `map` attribute holds it for every m from 0 (the average squared correlation itself) to $p - 2$, and the line under the table names the m at which it is smallest. The rule concerns components used as a first-stage description of the common variance; it is reported beside the fit of the factor models, not as a substitute for it.

Display. The printed table, and `knit_print()` and `as_kable` of it, show one row per count with its solution status, the chi square statistic, degrees of freedom, and p-value, the RMSEA with its interval (whose level the line beneath the table names), the CFI, TLI, and SRMR, the chosen criterion with its `delta` and `weight`, the p-value of the difference test against the previous count, the eigenvalue and its reference, and the MAP; columns with nothing to show (the reference when `B = 0`, the likelihood quantities under ULS, the comparison with one proper count) are left out. The table itself keeps every column. Beneath the table: the counts with the lowest AIC, BIC, and SABIC among the proper solutions, the criterion comparison, how far the eigenvalues exceed their reference, where the MAP is smallest, and the counts without a proper solution.

Value

A `dmar_tbl` data frame (class `dmar_factor_count`) with one row per count: `term`, `factors`, `solution` (`"proper"`, `"improper"`, `"no maximum"`, or `"not converged"`), `chi_square`, `df`, `p_value`, `rmsea`, `rmsea_ci_lower`, `rmsea_ci_upper`, `cfi`, `tli`, `srmr`, `aic`, `bic`, `sabic`, `hbic`, `criterion_delta`, `criterion_weight`, `criterion_log_evidence` (the comparison by `criterion`), `delta_chi_square`, `delta_df`, `p_delta`, `eigenvalue`, `reference_quantile`, `exceeds_reference`, and `map`. Its `tidy()` keeps the count as `estimate` and adds a `conf_level` column, the level of the RMSEA intervals (see `dmar_tidiers`), which `glance()` also reports. Attributes retain `fits` (the lavaan object of each proper count, `NULL` otherwise), `fit_measures` (each count's table, as `fa_exploratory()` returns it), `parallel_analysis` (the `parallel_analysis` table, when `B > 0`), `comparison` (the `compare_sem` table of the proper counts, when two or more were compared), `map` (the average squared partial correlation for $m = 0, 1, \ldots, p - 2$), `reasons` (why a count has no proper solution), `method`, and `analysis_note`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bollen, K. A., Ray, S., Zavisca, J., & Harden, J. J. (2012). A comparison of Bayes factor approximation methods including two new methods. *Sociological Methods & Research, 41*(2), 294–324. doi:10.1177/0049124112452393

Burnham, K. P., & Anderson, D. R. (2004). Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research, 33*(2), 261–304. doi:10.1177/0049124104268644

Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika, 30*, 179–185. doi:10.1007/BF02289447

Velicer, W. F. (1976). Determining the number of components from the matrix of partial correlations. *Psychometrika, 41*(3), 321–327. doi:10.1007/BF02293557

Zhang, X., & Savalei, V. (2023). New computations for RMSEA and CFI following FIML and TS estimation with missing data. *Psychological Methods, 28*(2), 263–283. doi:10.1037/met0000445

See Also

fa_exploratory for the solution at a chosen count, parallel_analysis for the eigenvalue reference with its Monte Carlo limits, compare_sem for the comparison the criterion columns reproduce, and sem_information_criteria.

Examples

```
items <- c("t1_visual_perception", "t2_cubes", "t4_lozenges",
           "t6_paragraph_comprehension", "t7_sentence", "t9_word_meaning")
# B = 200 is the fewest reference data sets the 95th percentile of the
# parallel analysis is reported from; the default is B = 1000.
factor_count(holzinger_swineford, items = items, B = 200, seed = 113)
```

---

factor_scores *Factor Scores With Standard Errors, Intervals, and Determinacy*

Description

Computes, for every row given to a fitted **lavaan** model and every latent variable, both standard factor scores: the regression score (Thomson, 1934; Thurstone, 1935) and the Bartlett score (Bartlett, 1937, 1938). Each comes with a per-case standard error and confidence interval, beside the number of the factor's items the case answered, the case's missing data pattern, and the determinacy of each score for that pattern (the correlation of the score with the factor it stands for). For ordered items the two scores are their categorical counterparts in **lavaan**, the empirical Bayes modal (EBM) score and the maximum likelihood (ML) score.

Usage

```
factor_scores(fit, data = NULL, conf_level = 0.95)

## S3 method for class 'dmar_factor_scores'
print(x, n = 3L, digits = getOption("dmar.digits", 3L), ...)

## S3 method for class 'dmar_factor_scores'
tidy(x, ...)

## S3 method for class 'dmar_factor_scores'
glance(x, ...)
```

Arguments

fit	A fitted **lavaan** model (from lavaan::cfa(), lavaan::sem(), or fa_confirmatory(output = "fit")) estimated from raw data. A fit to summary statistics (it holds no rows) and a fit that did not converge are refused, as are a two-level fit, a fit of continuous items that keeps incomplete cases without FIML (two-stage maximum likelihood, pairwise deletion or the available cases) and is conditional on exogenous covariates, a fit with regressions among the variables **lavaan** holds

as latent in which the disturbance of an observed variable regressed on others covaries with a factor (an observed mediator, for one), a fit conditional on exogenous covariates with regressions and an observed variable that **lavaan** holds among the latent variables but regresses on nothing, and, with ordered items, a fit with a direct effect of an exogenous covariate on an observed variable that also depends on a factor directly, one with a regression on an observed variable that is not an exogenous covariate when the variable predicted also depends on a factor directly, the predictor is ordered, or, in a row the fit keeps, the predictor is missing where the variable predicted is observed and depends on a factor through no variable the row has observed, one with an observed variable whose disturbance covaries with a factor and whose row **lavaan** keeps among the loadings, one with exogenous covariates that is not conditional on them, and any fit under a **lavaan** older than 0.7-3 (see *Details*). A fit with ordered items and residual covariances is scored with a warning. For continuous unconditional fits, a disturbance related only to an observed conditioning variable leaves the factor's regression conditional mean valid. Its scores and intervals are retained; Bartlett scores, SEs, limits, and determinacy are `NA` with the reason when their stronger conditional-unbiasedness requirement fails.

`data` Optional but recommended: the data frame given to the model, with the same rows in the same order (extra columns are ignored). It is checked against the data the model holds (every observed variable present, the same number of rows, and the same values in the rows the model used), and a frame that differs is refused. With it, the table has one row for every row of `data`, including rows the fit did not use (deleted listwise, or with a missing group), and their reason and items answered are filled in; for a multiple group fit it must hold the group variable. Without it, the table runs from row 1 to the last row the fit holds, so rows at the end of the data that the fit did not use are not represented, and the reason given a row the fit did not use cannot say which of its values was missing. The scores are always those of the rows the fit was estimated from; scoring new observations is not what this function does.

`conf_level` Confidence level for the per-case intervals; defaults to 0.95.

`x` A `dmar_factor_scores` table returned by `factor_scores()`.

`n` For `print()`, the number of rows shown for each factor (default 3); the pattern table shows up to `4 * n` rows. `knit_print.dmar_tbl` and `as_kable` show the same summary with the default.

`digits` For `print()`, the decimal places of the scores, their standard errors and limits, and the determinacies (default `getOption("dmar.digits", 3)`; see `dmar_tbl`).

`...` Unused.

Details

Both scores are returned because neither is better in general; which one belongs in a later analysis depends on the role the factor plays there (see *Which Score Belongs Where*). For relations among factors, the scores are rarely the right tool at all: a latent variable model, estimated in one step or by structural after measurement (`lavaan::sam()`; Rosseel & Loh, 2024), estimates them without the bias a regression among scores carries. The scores are for what only scores can do: describe

and rank individual cases, carry a factor into an analysis outside the model, or check the model's assumptions case by case. Requires **lavaan**.

The Two Scores. For the observed items o of a case, with Λ_o their rows of the loading matrix, Θ_{oo} their residual covariance matrix, Φ the model implied covariance matrix of the factors, $\Sigma_{oo} = \Lambda_o \Phi \Lambda_o' + \Theta_{oo}$, and μ_o the model implied item means,

$$\hat{\eta}_R = \mathrm{E}(\eta) + \Phi \Lambda_o' \Sigma_{oo}^{-1} (y_o - \mu_o)$$

is the regression score, the mean of the factor's distribution given the observed items under the normal factor model, and

$$\hat{\eta}_B = \mathrm{E}(\eta) + (\Lambda_o' \Theta_{oo}^{-1} \Lambda_o)^{-1} \Lambda_o' \Theta_{oo}^{-1} (y_o - \mu_o)$$

is the Bartlett score, the generalized least squares (and, under normality, maximum likelihood) estimate of the case's factor value treated as a fixed quantity. The regression score is shrunk toward the factor mean, by more when the items carry less information, and so has the smaller mean squared error; the Bartlett score is not shrunk, so its expected value given the case's factor value is that value (conditionally unbiased; under missing data this needs a condition on why values are missing, see *Missing Data*), at the price of a larger variance. Their standard errors differ in kind: that of the regression score is the posterior standard deviation, the square root of the diagonal of $\Phi - \Phi \Lambda_o' \Sigma_{oo}^{-1} \Lambda_o \Phi$, and that of the Bartlett score is its sampling standard error given the case's factor value, the square root of the diagonal of $(\Lambda_o' \Theta_{oo}^{-1} \Lambda_o)^{-1}$. The scores and their standard errors are those of `lavaan::lavPredict()` with `se = "standard"`, aligned to the rows of the data (with auxiliary variables, the same formulas applied to the indicators alone; see below).

The Bartlett score of a factor exists only when the observed items identify the factor's value, that is, when the factor's unit vector lies in the row space of Λ_o, so that its value is a function of the items' expected values. With simple structure that holds for every factor with an observed item. With items shared among factors (cross-loadings, a bifactor model) a pattern can fail it: a case that answered one item loading on two factors, and nothing else of either, has items that do not separate the two. The matrix $\Lambda_o' \Theta_{oo}^{-1} \Lambda_o$ is then singular, and **lavaan** falls back on a generalized inverse, whose minimum norm solution is not an estimate of either factor's value and whose standard error is not the error of anything; such a factor's Bartlett score, standard error, and determinacy are `NA` here with the reason. A factor the pattern does identify keeps its score, which the generalized inverse gives uniquely.

Which Score Belongs Where. Using scores in place of factors in a later regression (factor score regression) with the same type of score on both sides gives inconsistent estimates of the structural slopes. Skrondal and Laake (2001) showed that the revised scheme, the regression score for an explanatory factor and the Bartlett score for a response factor, gives consistent estimates of every parameter of a latent regression model, provided the explanatory and the response factors are fitted as separate measurement models; scores from one joint model of both are not covered by that result and should not feed a regression among the same factors, and fitting each factor alone (factorwise) is consistent only in special cases such as the independent clusters model with one explanatory factor. Even then the standard errors the second step reports do not account for the scores having been estimated (corrected ones may require resampling), no conventional test of fit is available, and factor score regression works badly for nonlinear models such as polynomial regression. Devlieger, Mayer, and Rosseel (2016) compared regression and Bartlett factor score regression, the bias avoiding method of Skrondal and Laake, and the bias correcting method of Croon (2002). The bias avoiding method was unbiased for unstandardized but not for standardized coefficients and had the largest standard error bias of the four, and Croon's method, with the standard error they

developed for it, was the only suitable alternative to structural equation modeling, comparable in bias, efficiency, mean squared error, power, and Type I error rate, though with more standard error bias. The differences among the methods disappeared when factor score indeterminacy was low, and they advise assessing it first, which the determinacy columns here allow. Croon's correction is local structural after measurement with the maximum likelihood mapping, whose matrix is the Bartlett score matrix (Rosseel & Loh, 2024), and Rosseel and Loh recommend local structural after measurement over one-step estimation when the sample is small to medium or the items have low to moderate reliability, with as many separate measurement blocks as the model permits. It is available as `lavaan::sam()`. Rosseel and Loh developed the framework for complete continuous data, so with missing data or ordered items its two-step standard errors deserve a check against a bootstrap.

Missing Data. Under full information maximum likelihood (`missing = "ml"`), **lavaan** scores each case from its own observed items, pattern by pattern, which is what *o* denotes above. For continuous items, the regression score so computed is the posterior weighted maximum likelihood estimate that Estabrook and Neale (2013) call full ML, and the Bartlett score so computed is their unweighted ML estimate. In their simulation (one factor), the two were equally reliable with complete data, and with missing values the posterior weighted score was the more reliable (with five items and values missing completely at random, a mean reliability of .780 against .767, and .752 for the scaled person mean), which they attribute to its empirical Bayes component. A case that answered none of a factor's items has no Bartlett score for that factor; its regression score is kept, but it comes only from the case's other observed variables, through the model (the items of correlated factors, a covariate that predicts the factor, an outcome the factor predicts), it is the factor's mean when none of them bears on the factor, and its determinacy says how little that is. A case with no observed variable has no score of either kind. Under listwise deletion the rows with a missing value were not used by the fit and are not scored; for continuous items, a fit with `missing = "ml"` scores them, unless the missing value is an exogenous covariate's. With `fixed.x = TRUE`, **lavaan**'s default, every fit deletes a row whose covariate is missing, FIML with `missing = "ml"` included, except one with `missing = "ml.x"` (which a fit conditional on the covariates does not allow), and the reason column names that remedy; with ordered items no fit that **lavaan** can score keeps such a row. **lavaan** has no FIML for ordered items: fitted by WLSMV, they take their parameters from pairwise or listwise deletion, which assumes the values are missing completely at random, and under pairwise deletion (`missing = "pairwise"`) the EBM and ML scores use each case's observed items, so that fit scores the rows listwise deletion leaves out.

A fit of continuous items that keeps the incomplete cases without FIML is scored the same way, each case from its own observed items, with the fit's estimates in the formulas above: a fit by two-stage maximum likelihood (**lavaan**'s `missing = "two.stage"` or `"robust.two.stage"`, as `fa_confirmatory` fits with `missing = "two_stage"`) or by pairwise deletion (`missing = "pairwise"`, or `"available.cases"`, which for continuous items gives **lavaan** the same pairwise complete moments and estimates). The two-stage estimates are consistent when the values are missing at random, as the FIML estimates are (Savalei & Bentler, 2009); the estimates from pairwise deletion require the values to be missing completely at random. The scores treat the estimates as known, as they treat the FIML estimates, so the scores, standard errors, and determinacies of such a fit mean what they mean under FIML and differ from FIML's only as the estimates differ. A fit without a mean structure (the default for one group under pairwise deletion) centers its observed variables at their sample means and its factors at the means **lavaan** gives them (`lavInspect(fit, "mean.lv")`): zero, or for a factor a covariate predicts, the prediction at the covariate's mean, as a FIML fit has it. `lavaan::lavPredict()` scores such a fit as complete data, leaving every case with a missing value without a score, so these scores are computed here, by the closed forms `lavPredict()` applies pattern by pattern under FIML. Auxiliary variables of the first stage of a

two-stage fit (**lavaan**'s `aux`) are not in the model and do not enter the scores. Such a fit conditional on exogenous covariates (`conditional.x = TRUE`) is refused.

Missing values affect the two scores differently. Conditional unbiasedness of the Bartlett score and the coverage of its Wald interval are properties of its sampling distribution given the case's factor value, and with missing values they hold when, given the factors, whether an item is observed does not depend on the residuals of the items scored: values missing completely at random, or missing because of the factors themselves or of variables related to the items only through the factors. Rubin (1976) showed that sampling distribution inferences may ignore the process that causes missing data when the missing values are missing at random and the observed values are observed at random, whereas likelihood and Bayesian inferences need only the first (with the parameters of the process distinct from those of the data). Missingness that depends on the observed values of the factor's own items, the kind of missing at random that FIML is commonly used for, fails the second: it selects each pattern on those items' residuals, so the Bartlett score of a pattern is biased given the factor value, and its interval covers the factor value less often than stated in that pattern. In a one-factor model of four items with loadings of .7 and 5,000 cases, the last three items missing whenever the first fell below -0.3, the Bartlett scores of the cases left with one item averaged 0.82 below their factor values and their 95% intervals covered 92.7% of them; with as many values missing completely at random, -0.01 and 95.3%. The regression score is a posterior quantity, so missingness that depends only on what the score conditions on leaves its posterior interval with its average coverage (94.6% in the first case). Missingness that runs through an auxiliary variable the scores set aside (see *Auxiliary Variables and Covariates*) is not of that kind for the regression score, since the pattern then carries information about the factor beyond the observed items, and the regression score of a pattern is biased. The Bartlett score is not, but only when the auxiliary is related to the items through the factors alone (independent of them given the factors); one that shares variance with an item beyond the factor, as an earlier wave of the same item does, selects the pattern on that item's residual, which fails the condition above. In the same design with the last three items missing whenever an auxiliary fell below -0.3, an auxiliary that correlated .4 with the factor and with nothing else left the cases with one item Bartlett scores 0.05 below their factor values on average (within two Monte Carlo standard errors of zero), with intervals covering 95.6%, and regression scores 0.17 above them; one that correlated .4 with the factor and .5 with the first item's residual left them Bartlett scores 0.56 below their factor values, with intervals covering 93.1%. A score whose error has mean zero given the factor leaves the slopes of a regression in which it is the response unbiased, and that is the Bartlett score's place in the revised scheme of factor score regression above, so missingness that depends on the factor's own observed items weakens it there. The reliabilities of Estabrook and Neale (2013) quoted above are for values missing completely at random.

Determinacy. The determinacy of a score is its correlation with the factor, $[A\Lambda_o\Phi]_{kk}/\sqrt{[A\Sigma_{oo}A']_{kk}\Phi_{kk}}$ for the weight matrix A of the score, the formula **lavaan** uses for `fs.determinacy` and `fs.determinacy.bartlett`, evaluated here for each missing data pattern. Its square is the reliability of the score as a measure of the factor. The regression score has the larger determinacy, since it is the linear function of the observed items that correlates most with the factor; for a one-factor model the two are equal, and with uncorrelated residuals their square is coefficient *H* (Hancock & Mueller, 2001), the reliability of the optimally weighted composite of the items. Per-pattern determinacy shows the cost of each missing item. It is computed for continuous items; with ordered items each case's standard error is given instead. Beside the regression score's determinacy ρ, the display and the `determinacy` and `patterns` attributes report $2\rho^2 - 1$ (`min_score_correlation`), the lowest correlation that two sets of scores, each as consistent with the model as the other, can have (Guttman, 1955, as Grice, 2001, p. 435, presents it). It is zero when $\rho^2 = .5$, where two such sets can be uncorrelated, and negative

below that, so it says how far the scores of different cases can be trusted to order the cases on the factor. The package reports the quantity and draws no line.

Intervals. The interval around a regression score is a posterior (empirical Bayes) interval: its coverage holds on average over cases, not for each case's factor value. The interval around a Bartlett score is a Wald interval for the case's factor value. Both treat the model's estimated parameters as known, so they do not carry the parameters' sampling variability. For ordered items the EBM interval approximates the posterior around its mode, and the ML interval is a Wald interval. The ML score of a factor exists only when the likelihood of the case's responses has a finite maximum in it. A case whose observed items of a factor all sit at the same end of their scales (all in the lowest category, or all in the highest, for positive loadings) has none, since the likelihood keeps rising as the factor's value moves out. With items shared among factors (cross-loadings, a bifactor model) the likelihood can also rise without bound along a direction that moves several factors together, or stay flat along one that the observed items cannot separate. The check is exact: a factor has a finite and unique ML score when no direction along which no observed item's probability falls moves it, which a nonnegative least squares problem decides (by Farkas' lemma), and with simple structure it reduces to the rule for the items at an end. It reads each item's loadings as **lavaan**'s likelihood does, including those of an observed variable regressed on the factors (an outcome the factor predicts, or an item with a direct effect of another factor), which **lavaan** holds among the regressions. A factor that fails it has its ML score and interval `NA` with the reason. A factor that passes it keeps its ML score, also when it shares observed items with a factor whose likelihood rises without bound: as that factor moves out, the shared items' probabilities tend to 1 whatever this factor's value, so its maximum is that of the case's other items, which **lavaan**'s optimizer reaches within its tolerance (within 1.2e-4 of a direct maximization of the likelihood for the factor of a cross-loading item in the tests, and within 1.1% of the standard error in a bifactor model of 12 items). When **lavaan**'s default optimizer stops on its iteration limit and returns no score for the case, the score comes from **lavaan**'s `nlminb` optimizer. A higher order factor, which has no items of its own, has no ML score at all: the likelihood of the responses depends on the factors it measures, not on it, and **lavaan**'s value is the center of the small ridge its optimizer adds. Its EBM score, which the factor's place in the model informs, is kept.

Ordered Items Need lavaan 0.7-3. The probability of an ordered response compares each threshold with the mean of the item's latent response, its intercept plus its loadings times the factor values. The EBM and ML scores of **lavaan** 0.7-2 counted the intercept twice in that comparison, so they were wrong in any model whose ordered indicators have intercepts other than zero (a multiple group model with equal thresholds, for one); **lavaan** 0.7-3 corrected it, and also changed how the scores treat the scale factors of the delta parameterization. A fit with ordered items is therefore refused under an older **lavaan**, with the remedy, rather than scored with numbers that may be wrong.

Multiple Groups. The scores use each group's own parameters, and the `group` column names the group. Scores from different groups are on a common metric only to the extent the measurement model is invariant across the groups (equal loadings for comparing relations that involve the scores, and equal intercepts as well for comparing their levels); see `measurement_invariance`. Rows whose group is missing are not used by the fit; with `data` they are kept, without a score.

Auxiliary Variables and Covariates. An observed variable that indicates no factor and enters the model only through covariances with the indicators is an auxiliary variable carried as a saturated correlate (Graham, 2003), as the reliability functions of this package carry them. It helps the estimates under missing data, but under that parameterization it is uncorrelated with every factor and correlated with the item residuals, so whatever it shares with the factor is carried as residual covariance, and a score that conditions on it subtracts true score variance. When such variables are present the

scores are computed from the indicators alone, with the fit's estimates, the `scored_from` attribute is `"indicators"`, and a message says so. A higher order factor, which has no items of its own, then has no Bartlett score, nor, when the data have missing values, in the scores computed here for a two-stage fit or one by pairwise deletion (`missing = "pairwise"` or `"available.cases"`), nor in a fit in which **lavaan** gives an observed variable a row of zeros among the loadings (see *Observed Variables in Regressions*): **lavaan** forms that score through the loadings of the factors it measures, a construction not repeated here, and the reason column says so; its regression score is computed as for any other factor. A covariate that predicts a factor (`f ~ x`) is different: the scores condition on it as the model specifies (as on any observed variable that enters a regression of the model), and Curran, Cole, Bauer, Hussong, and Gottfredson (2016) found that including such covariates in the scoring model improved score quality in nearly every condition of their simulation.

Observed Variables in Regressions. An observed variable regressed on others, such as an item with a direct effect of a covariate (`y1 ~ x`, the multiple indicators multiple causes model of differential item functioning) or an outcome the factor predicts (`z ~ f`), is held by **lavaan** as a latent variable measured by itself without error, with its loadings among the regressions and its residual variance among those of the latent variables. The scores, their standard errors and determinacies, the check of the ML score, and the check of the solution take its loadings and residual variance from there, as `lavaan::lavPredict()` does, so an item with a direct effect enters its factor's Bartlett score, which is then conditionally unbiased given the factor and the covariate. An observed variable that **lavaan** holds the same way but regresses on nothing, such as a criterion that covaries with a factor (`f ~~ z`) or a covariate of a factor that also covaries with another variable (so that **lavaan** does not count it among the exogenous covariates), is conditioned on, as an exogenous covariate is: the regression score is the factor's mean given it and the items, and the Bartlett score is conditionally unbiased given the factors and it. In a fit conditional on the covariates (`conditional.x = TRUE`) without regressions among the variables **lavaan** holds as latent (no B matrix), so is an observed variable regressed only on exogenous covariates (`z ~ x`): **lavaan** then keeps the variable's own row among the loadings, `lavPredict()`'s regression score is the factor's mean given the items, the variable, and the covariates, and its Bartlett score is conditionally unbiased given the factors, the variable, and the covariates, also when the variable covaries with a factor (`f ~ x`, `z ~ x`, `f ~~ z`; `lavaan::sem()` frees that covariance by default). In a model with regressions among them, `lavPredict()` gives a variable regressed on nothing a row of zeros among the loadings, which leaves its covariances with the factors out of the scores (with `f ~ x` and `f ~~ z` for three Holzinger and Swineford tests, its regression score missed the conditional mean by up to 1.9), so the scores of such a fit are computed here, with the formulas `lavPredict()` uses; one conditional on the covariates (`conditional.x = TRUE`) is refused, with the remedy. In a fit without a mean structure and without exogenous covariates, `lavPredict()` counts the mean of a variable held this way twice, as its intercept and as the mean of the latent variable that holds it, which shifts by a constant every score whose weights involve the variable: the regression score of every factor related to it and the Bartlett score of a factor that predicts it (for a factor of three Holzinger and Swineford spatial tests that predicts a fourth test and correlates .50 with a factor of three speed tests, by -0.39 and -0.61 for the first factor's regression and Bartlett scores and by -0.07 for the second factor's regression score); the scores of such a fit are computed here, with the formulas `lavPredict()` uses and the means the fit implies. A higher order factor's Bartlett score, which **lavaan** forms through the loadings of the factors it measures, is **lavaan**'s less the constant the double count adds (-1.09 for a second order factor of three factors of three Holzinger and Swineford tests each, when the spatial factor predicts the paper form board test), so it equals that of the fit with a mean structure. In a model with regressions among the variables **lavaan** holds as latent, a variable regressed on others is represented by its measurement equation, with its disturbance as the error, which is right only when that disturbance is uncorrelated

with the factors and with the variables the scores condition on, so a fit in which such a variable predicts a factor, directly or through others (an observed mediator), or has a residual covariance with a factor is refused: `lavPredict()` then gives a Bartlett score that is not conditionally unbiased and, when a factor is involved, a regression score that is not the factor's conditional mean (with `f ~ a` and `a ~ x` for three Holzinger and Swineford tests, it missed the conditional mean by up to 1.08, with a correlation of .92). Carried as a single indicator latent variable (`A =~ a` with `a ~~ 0*a`), such a variable is scored correctly: written so, **lavaan** fixes one scale constraint, the loading or, with `std.lv = TRUE`, the latent variable's variance, so the refitted model is the same model, with the refused fit's log likelihood and degrees of freedom (Bollen, 1989, Chapter 7). Written `A =~ 1*a` in a fit with `std.lv = TRUE`, it fixes both, which is a different model whose scores are not those of the refused one. In a multiple group fit, `group.equal` holds parameters equal by kind, and the variable's intercept, residual variance, and covariances become the latent variable's, so the refit is the same model only when it carries the refused fit's constraints across the groups: with equal intercepts, **lavaan** frees the latent variable's mean after the first group unless it is fixed at zero in every group (`A ~ 0*1`; for two schools with equal loadings and intercepts the refit without it had a log likelihood of -4246.42 on 15 degrees of freedom, against -4246.51 on 16), and the equalities `group.equal` places on residual variances and covariances, or on those of latent variables, must be given to the variable's parameters as the refused fit had them and to no others. The message says which apply.

With ordered items, **lavaan**'s EBM and ML scores leave a direct effect of an exogenous covariate on an observed variable out of each case's likelihood, and they take an observed variable that is not an exogenous covariate as zero where it predicts another. That leaves the scores unchanged when the variable predicted depends on no factor directly: its density given its predictors then does not involve the factors, so the term left out changes each case's likelihood by a factor constant in them. Such fits are scored: a covariate's effect on an observed outcome unrelated to the factors (`w ~ x`), and a fully mediated chain of continuous observed variables (`m ~ f` with `w ~ m`, whose EBM and ML scores agree with a direct maximization of the model's likelihood for four Holzinger and Swineford tests cut at their quartiles). A fit in which the term left out involves the factors is refused: a direct effect on a variable that also depends on a factor directly (`y1 ~ x` for an item, whose thresholds are estimated given the covariate, so the scores are those of a case with the covariate at zero; ML scores off by 0.11 to 0.44 for an item regressed on age), a regression of such a variable on an observed one (`y4 ~ y3` for local dependence between two items, ML scores off by up to 0.18 for the four tests; or `w ~ m + f`), a regression on an ordered item, whose latent response is not observed (`w ~ y3`), and a regression on an observed variable that a row the fit keeps lacks while it has the one predicted, where the missing variable depends on a factor through no variable the row has observed (pairwise deletion, where the model's likelihood of the variable predicted then runs through the missing one to the factors). A variable the row has observed on the way cuts that dependence, and the row is scored: with `m ~ f`, `w ~ m`, and `s ~ w`, a row lacking `w` but having `m` and `s` is scored, while a row lacking `m` but having `w` refuses the fit. A fit is refused, too, for an observed variable **lavaan** holds among the latent variables whose disturbance covaries with a factor, regressed on nothing or, in a fit without regressions among the latent variables, only on exogenous covariates (`f ~ x` with `w ~ x`, where `lavaan::sem()` frees the covariance of `f` and `w` by default): the EBM score takes the factor's prior from its model implied mean given the covariates, which leaves the variable's value out, so it is the posterior mode given the items and covariates alone; written as a predictor of the factor (`f ~ x`), a variable regressed on nothing is an exogenous covariate, which the EBM score conditions on. A fit with ordered items and exogenous covariates that is not conditional on them (`conditional.x = FALSE`), which `lavPredict()` cannot score, is refused as well. The EBM and ML scores of **lavaan** also take each case's likelihood as the product of its observed variables' probabilities given the factors, so they leave residual covariances among those variables out; a fit with one is scored with

a warning, since its scores are those of the model without it (with a residual correlation of -0.12 between two of the four tests, the ML scores of the 160 response patterns differed from the scores that maximize the model's own likelihood by up to 0.11, 15% of their standard error, and by 0.03 at the median).

An improper solution is scored with a warning: one with a negative residual variance (a Heywood case, also of an observed variable regressed on others), a negative factor variance, a covariance matrix of the factors that is not positive definite (a factor correlation beyond 1, for one), a residual covariance matrix of the continuous variables scored from that is not positive definite, or, when none of these holds, a residual covariance matrix of the structural part of the model that is not, the conditions **lavaan**'s post-estimation check flags. The regression weights use the model implied covariance matrix and the Bartlett weights the loadings and the inverse of the residual covariance matrix, so both scores inherit the solution. For ordered items **lavaan** computes no EBM or ML score for a solution with a negative residual variance, and the warning and the reason column say so.

Value

A `data.frame` (class `dmar_factor_scores`, `dmar_tbl`) with one row for each row of the data and each factor, ordered by factor and then row, so that `x$.derived_bartlett[x$factor == "f"]` lines up with the rows of `data`. Its columns are always the same: `row` (the row of the data), `group` (the group label; `NA` for a fit without groups), `factor`, `.derived_regression`, `.derived_regression_se`, `.derived_regression_ci_lower`, `.derived_regression_ci_upper`, `.derived_bartlett`, `.derived_bartlett_se`, `.derived_bartlett_ci_lower`, `.derived_bartlett_ci_upper`, `items_observed` and `items` (the factor's items the case answered, and its number of items; a higher order factor's items are those of the factors it measures, and a loading the model holds at zero, written `0*x` or `a*x` with `a == 0` directly or through a chain of equalities, makes no item: the scores and reasons are those of the model without it), `pattern` (the missing data pattern of the variables scored from, within the group, numbered from the most complete), `determinacy_regression`, `determinacy_bartlett` (for the case's pattern), and `reason` (empty, or why a value is missing or borrowed; reasons are separated by semicolons). The `.derived_regression` and `.derived_bartlett` columns hold the EBM and ML scores for ordered items.

The eight columns that hold the scores, their standard errors, and their confidence limits carry the prefix `.derived_`, so that a score joined to the observed data cannot be taken for an observed variable or collide with one (a data set may well have a column named `regression`). The `derived_columns` attribute names them by what each holds (`regression`, `regression_se`, `regression_ci_lower`, `regression_ci_upper`, and the same four for `bartlett`), so `x[[attr(x, "derived_columns")[["bartlett"]]]]` is the column of Bartlett scores. All are present under every supported model.

Attributes: `methods` (the **lavaan** methods behind the two columns), `conf_level`, `estimator`, `missing`, `patterns` (one row per group, pattern, and factor: the number of cases, the items observed, the variables missing, both determinacies, `min_score_correlation`, and both standard errors), `determinacy` (per group and factor, for complete data: both determinacies and `min_score_correlation`), `scored_from` (`"fit"` or `"indicators"`), `auxiliary` (the auxiliary variables set aside), `ngroups`, and `lavaan_version`. `tidy()` returns the table as a plain data frame; `glance()` returns one row with the numbers of rows, rows scored, factors, groups, and patterns, the two methods, the estimator, the missing data treatment, `scored_from`, and `conf_level`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bartlett, M. S. (1937). The statistical conception of mental factors. *British Journal of Psychology, 28*(1), 97–104.

Bartlett, M. S. (1938). Methods of estimating mental factors. *Nature, 141*(3570), 609–610.

Bollen, K. A. (1989). *Structural equations with latent variables*. Wiley.

Croon, M. (2002). Using predicted latent scores in general latent structure models. In G. Marcoulides & I. Moustaki (Eds.), *Latent variable and latent structure models* (pp. 195–223). Erlbaum.

Curran, P. J., Cole, V., Bauer, D. J., Hussong, A. M., & Gottfredson, N. (2016). Improving factor score estimation through the use of observed background characteristics. *Structural Equation Modeling, 23*(6), 827–844. doi:10.1080/10705511.2016.1220839

Devlieger, I., Mayer, A., & Rosseel, Y. (2016). Hypothesis testing using factor score regression: A comparison of four methods. *Educational and Psychological Measurement, 76*(5), 741–770. doi:10.1177/0013164415607618

Estabrook, R., & Neale, M. (2013). A comparison of factor score estimation methods in the presence of missing data: Reliability and an application to nicotine dependence. *Multivariate Behavioral Research, 48*(1), 1–27. doi:10.1080/00273171.2012.730072

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

Grice, J. W. (2001). Computing and evaluating factor scores. *Psychological Methods, 6*(4), 430–450. doi:10.1037/1082989X.6.4.430

Guttman, L. (1955). The determinacy of factor score matrices with implications for five other basic problems of common-factor theory. *British Journal of Statistical Psychology, 8*(2), 65–81. doi:10.1111/j.20448317.1955.tb00321.x

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), *Structural equation modeling: Present and future* (pp. 195–216). Scientific Software International.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

Rosseel, Y., & Loh, W. W. (2024). A structural after measurement approach to structural equation modeling. *Psychological Methods, 29*(3), 561–588. doi:10.1037/met0000503

Rubin, D. B. (1976). Inference and missing data. *Biometrika, 63*(3), 581–592. doi:10.1093/biomet/63.3.581

Savalei, V., & Bentler, P. M. (2009). A two-stage approach to missing data: Theory and application to auxiliary variables. *Structural Equation Modeling, 16*, 477–497. doi:10.1080/10705510903008238

Skrondal, A., & Laake, P. (2001). Regression among factor scores. *Psychometrika, 66*(4), 563–575. doi:10.1007/BF02296196

Thomson, G. H. (1934). The meaning of *i* in the estimate of *g*. *British Journal of Psychology, 25*, 92–99.

Thurstone, L. L. (1935). *The vectors of mind*. University of Chicago Press.

See Also

fa_confirmatory to fit the measurement model; reliability_H for coefficient H; reliability_omega for the reliability of the unit-weighted composite; measurement_invariance before comparing scores across groups; lavaan::lavPredict(), lavaan::sam().

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
data(holzinger_swineford)

# Two later spatial tests (t25, t26) were given to the Grant-White
# pupils only, so under full information maximum likelihood each
# Pasteur pupil (rows 1 to 156) is scored from four spatial tests and
# each Grant-White pupil from six.
model <- "
  spatial =~ t1_visual_perception + t2_cubes + t3_paper_form_board +
             t4_lozenges + t25_paper_form_board_r + t26_flags
  verbal  =~ t5_general_information + t6_paragraph_comprehension +
             t7_sentence + t8_word_classification + t9_word_meaning"
fit <- lavaan::cfa(model, data = holzinger_swineford, std.lv = TRUE,
                   missing = "ml")
fs <- factor_scores(fit, data = holzinger_swineford)
fs

# One pupil from each school: the spatial scores of the Pasteur pupil
# rest on four tests and have the wider intervals.
fs[fs$row %in% c(1, 157),
   c("row", "factor", ".derived_regression",
     ".derived_regression_ci_lower", ".derived_regression_ci_upper",
     ".derived_bartlett", ".derived_bartlett_ci_lower",
     ".derived_bartlett_ci_upper", "items_observed", "items")]

# Determinacy and standard errors for each missing data pattern.
attr(fs, "patterns")

# The table is ordered by factor and then row, so one factor's column
# lines up with the data.
spatial_bartlett <- fs$.derived_bartlett[fs$factor == "spatial"]
length(spatial_bartlett) == nrow(holzinger_swineford)

# When scores must stand in for factors in a regression, Skrondal and
# Laake (2001) pair the regression score of the predictor with the
# Bartlett score of the outcome, each from its own measurement model.
# With both factor variances fixed at 1 the slope estimates the latent
# correlation; pairing the regression scores attenuates it. The lm()
```

```
# standard errors treat the scores as error free and are not valid.
fit_v <- lavaan::cfa("verbal =~ t5_general_information +
  t6_paragraph_comprehension + t7_sentence + t8_word_classification +
  t9_word_meaning", data = holzinger_swineford, std.lv = TRUE,
  missing = "ml")
fit_s <- lavaan::cfa("spatial =~ t1_visual_perception + t2_cubes +
  t3_paper_form_board + t4_lozenges + t25_paper_form_board_r +
  t26_flags", data = holzinger_swineford, std.lv = TRUE,
  missing = "ml")
fs_v <- factor_scores(fit_v)
fs_s <- factor_scores(fit_s)
c(revised = unname(coef(lm(fs_s$.derived_bartlett ~
                           fs_v$.derived_regression))[2]),
  regression_both = unname(coef(lm(fs_s$.derived_regression ~
                                   fs_v$.derived_regression))[2]),
  latent = lavaan::lavInspect(fit, "cor.lv")["spatial", "verbal"])

# The broom verbs: the table as a plain data frame, and one summary row.
head(generics::tidy(fs))
generics::glance(fs)
```

`fa_bifactor` *Bifactor Confirmatory Factor Analysis Model*

Description

Fits the bifactor model, in which a general factor accounts for what every item of a scale has in common and two or more group factors account for what the items of each subscale share beyond it, with all of the factors uncorrelated (Holzinger & Swineford, 1937; Reise, 2012). The model is described as in `fa_confirmatory`, by naming the items of each group factor, and is fit by maximum likelihood to raw data (by full information maximum likelihood, the default, or after listwise deletion) or to a covariance matrix. The table reports the loadings, error variances, and intercepts with their confidence intervals; the indices that describe how much of the total score and of each subscale score the general factor accounts for, each with a delta method standard error and a confidence interval (Rodriguez, Reise, & Haviland, 2016): omega and omega hierarchical (omega_H) for the total score, omega and omega hierarchical subscale (omega_HS) for each subscale score, the explained common variance (ECV), coefficient *H*, and the maximal reliability of each factor, and the percentage of uncontaminated correlations (PUC); and the fit of the model.

Usage

```
fa_bifactor(
  data = NULL,
  factors,
  S = NULL,
  N = NULL,
  M = NULL,
  general = "general",
```

```
  general_only = NULL,
  equal_loading = FALSE,
  equal_intercept = FALSE,
  equal_error = FALSE,
  meanstructure = NULL,
  estimator = "ML",
  missing = "fiml",
  se = NULL,
  conf_level = 0.95,
  rmsea_conf_level = 0.9,
  output = c("verbose", "measurement", "summary", "standardized", "fit"),
  ...,
  wald_df = Inf
)

## S3 method for class 'dmar_fa_bifactor'
tidy(x, ...)

## S3 method for class 'dmar_fa_bifactor'
glance(x, ...)
```

Arguments

`data`	A raw data matrix or data frame, rows are respondents and columns include the items named in `factors`. Supply exactly one of `data` or `S`.
`factors`	Named list of the group factors. Each element names a group factor and gives the character vector of its items; each item is listed under one group factor, and the general factor loads on all of them. Two or more group factors are needed, each with three or more items (two when their loadings are equated with `equal_loading`); see *Details* for the structures that are refused. An item, here or in `general_only`, that is not a column of `data` or an item of `S` is refused with an error of class `dmar_items_invalid` (see `dmar_metadata`).
`S`	A symmetric covariance matrix of the items, with dimnames naming the items; `N` is then required. Supply exactly one of `data` or `S`.
`N`	Total sample size. Required with `S`; ignored (inferred from the rows) with `data`.
`M`	Optional named numeric vector of item means, used with a covariance matrix to model the mean structure (required there when `equal_intercept` is used). Ignored for raw data.
`general`	The name of the general factor, `"general"` by default. It names the general factor's rows (`lambda_general_1`, `omega_H_general`).
`general_only`	`NULL` (the default), or a character vector of items on the general factor alone, with no group factor: an item written for the construct as a whole, say, or the items of a subscale too small to carry a group factor of its own. They count in the general factor's rows (its loadings, omega, omega_H, ECV, *H*, maximal reliability, and PUC) and in no subscale.
`equal_loading`	Logical, or a named logical vector with one element for the general factor and one for each group factor. `TRUE` constrains the loadings within a factor to a single

value. Defaults to FALSE.

equal_intercept
: Logical, or a named logical vector with one element per group factor. TRUE constrains the intercepts of a group factor's items to a single value; this requires the mean structure. The items on the general factor alone keep their own. Defaults to FALSE.

equal_error
: Logical, or a named logical vector with one element per group factor. TRUE constrains the error variances of a group factor's items to a single value; the items on the general factor alone keep their own. Defaults to FALSE.

meanstructure
: Logical or NULL. NULL (default) models the mean structure exactly when it is needed: under FIML (the default with raw data, so the item intercepts, the nu rows, join the table), when any equal_intercept is TRUE, or when M is supplied. TRUE models the intercepts regardless (raw data or M required); FALSE suppresses them, which FIML does not allow.

estimator
: "ML" (default), maximum likelihood, or "MLR", maximum likelihood with standard errors and a test statistic robust to nonnormality (raw data required). Ordered items are not available; see *Details*.

missing
: The missing data treatment for raw data: "fiml" (default; full information maximum likelihood, which uses every case with an observed item; **lavaan**'s name "ml" is accepted) or "listwise". The default is resolved as in fa_confirmatory: with se = "robust.sem" or a test statistic FIML cannot compute, the items are fit as they stand when no item value is missing, and otherwise the call stops and names the option to choose. Ignored with S.

se
: Standard error method passed to **lavaan**. The default, NULL, follows the estimator: "standard" for ML and "robust.huber.white" for MLR. "robust.sem" requires complete data and is refused with FIML, and "none" requests point estimates only.

conf_level
: Confidence level of the intervals for the parameters and the indices. Defaults to 0.95. It does not reach the RMSEA interval, whose level is rmsea_conf_level.

rmsea_conf_level
: Confidence level of the RMSEA interval of the fit rows, which the rmsea_ci_level row records. Defaults to 0.90, the conventional level, whose limits decide the tests of close and not close fit at .05 (see sem_fit_measures); rmsea_conf_level = 0.95 gives a 95 percent interval. It sets the level in the table DMAR returns and in its glance(); output = "summary" returns the **lavaan** summary, which shows lavaan's 90 percent interval whatever this argument is.

output
: Format of the returned object:

  "verbose" (default) The loadings, factor variances (fixed at 1), error variances, and intercepts, then the indices, then the fit rows.

  "measurement" The indices alone.

  "summary" The summary() of the **lavaan** fit (not a data frame).

  "standardized" lavaan::standardizedSolution() of the fit; see *Details*. Under se = "none" its se, z, pvalue, ci.lower, and ci.upper columns are NA.

  "fit" The **lavaan** fit, which bifactor_indices, ci_bifactor_indices, factor_scores, and lavaan::lavTestLRT() take.

`...` For `fa_bifactor()`, additional arguments forwarded to `lavaan` (for example `cluster`); a `group` argument is an error, as in `fa_confirmatory()`, and so is `ordered`. Unused by `tidy()` and `glance()`.

`wald_df` The reference distribution of the Wald intervals and *p*-values: `Inf` (the default), the normal distribution, or a single positive number, the degrees of freedom of a *t* distribution for every parameter and index row (for the indices, on the logit scale), as in `fa_confirmatory`. The keyword `"missing_information"` is refused: the calibration study behind it simulated one factor models, and nothing calibrates it for the loadings of a bifactor model or for its indices. It follows `...`, so it must be named in full.

`x` A `dmar_fa_bifactor` table returned by `fa_bifactor()`.

Details

The model. An item i of group factor s is

$$x_i = \nu_i + \lambda_{gi} g + \lambda_{si} s + e_i,$$

and an item on the general factor alone lacks the group term. Every factor's variance is fixed at 1 (and its mean at 0 when the mean structure is modeled), every pair of factors is uncorrelated, and the first loading of each factor is freed, so each loading is the covariance of its item with the factor. The general factor carries what all of the items share; a group factor carries only what its subscale's items share beyond that, so its loadings are smaller than those of a factor of the subscale alone, and they can be near zero or change sign when the general factor takes up what the subscale measures. Every loading, error variance, and intercept carries a label, as in `fa_confirmatory`, and the indices are **lavaan** defined parameters of those labels, so the fit itself gives each one a delta method standard error. The standardized solution is derived from this fit of the covariances (or of the raw data), with the factor variances fixed at 1, not from a fit to correlations.

Scope and identification. A structure outside the model, or one the data cannot identify, is refused before anything is fit, with an error of class `dmar_bifactor_unidentified` (which inherits `dmar_refusal`; see `dmar_metadata`) that names the factors or items at issue and gives the kind of refusal as its `reason` element. Fewer than two group factors, and an item listed under two group factors, are outside the model the function fits (`reason "scope"`): a general factor beside a single group factor, with items on the general factor alone, can be identified, but omega_HS, ECV, and PUC describe how the common variance divides among subscales, which takes two or more. Names placed twice or that coincide are refused with `reason "names"`: an item listed twice under one group factor, or both under a group factor and in `general_only`; the general factor and a group factor of one name, a factor and an item of one name, or labels made from the names that coincide. A group factor of one item, or of two whose loadings are free (they enter the model only through their product), and two group factors whose general loadings are fixed only up to a scale are not identified (`reason "unidentified"`). With two group factors and no item on the general factor alone, the covariances between the items of the two groups fix the general loadings of one group only up to a factor c and those of the other up to $1/c$, and c must come from inside a group, whose item variances and covariances must outnumber the group loadings and error variances it estimates. Two groups each of three items (or of two items with equal loadings) leave c undetermined, and so the model is not identified; an item on the general factor alone, a group of four or more items, equal loadings or equal error variances within a group of three, or equal general loadings each identify it. Every structure with exactly two group factors that is fit gives a warning of

class `dmar_bifactor_empirical`: such a model can be empirically underidentified, since when the item covariances are nearly equal within each group and between the groups, different general and group loadings reproduce them equally well (Green & Yang, 2018). Items on the general factor alone, or items whose loadings differ in strength, protect against it. The identification checks run once the other arguments and the data have passed theirs, so the warning comes only with a fit. A count of parameters cannot see empirical underidentification, and when **lavaan**'s information matrix cannot be inverted, or is singular (scaled to a unit diagonal, its smallest eigenvalue is below 10^{-6}), every standard error, test, and interval is `NA`, the maximal reliability's included, with a warning. The singularity is checked, as in `fa_confirmatory`, for the observed information of a FIML fit and for the expected information of a listwise fit or of a covariance matrix alike, since **lavaan** can invert either when it is singular.

The indices. Write λ_{gi} for the general loading of item i, λ_{si} for its loading on its group factor s (zero for an item on the general factor alone), ψ_i for its error variance, and $v_i = \lambda_{gi}^2 + \lambda_{si}^2 + \psi_i$ for its model implied variance. The variance of the total score, the unweighted sum of the items, is

$$\sigma_X^2 = (\textstyle\sum_i \lambda_{gi})^2 + \sum_s (\sum_{i \in s} \lambda_{si})^2 + \sum_i \psi_i,$$

and the rows are

`omega_<general>` Omega for the total score, $[(\sum_i \lambda_{gi})^2 + \sum_s (\sum_{i \in s} \lambda_{si})^2]/\sigma_X^2$: its reliability with every factor counted as true score (McDonald, 1999).

`omega_H_<general>` Omega hierarchical, $(\sum_i \lambda_{gi})^2/\sigma_X^2$: the share of the total score's variance that is due to the general factor alone, the group factors' variance counted as error. Its square root is the correlation of the total score with the general factor.

`omega_<s>` Omega for the subscale score of group factor s, the sum of its items, $(\sum_{i \in s} \lambda_{gi})^2 + (\sum_{i \in s} \lambda_{si})^2$ over that score's variance, which adds $\sum_{i \in s} \psi_i$: its reliability with both factors counted as true score.

`omega_HS_<s>` Omega hierarchical subscale, $(\sum_{i \in s} \lambda_{si})^2$ over the same variance: the share of the subscale score's variance that is due to its group factor alone, what the subscale score says beyond the total.

`ECV_<general>`, `ECV_<s>` The explained common variance, formed from the standardized loadings $\lambda_{gi}/\sqrt{v_i}$ and $\lambda_{si}/\sqrt{v_i}$, as `bifactor_indices` forms it: for the general factor, its share of the common variance of all of the items, $\sum_i \lambda_{gi}^2/v_i$ over $\sum_i (\lambda_{gi}^2 + \lambda_{si}^2)/v_i$; for a group factor's row, the general factor's share of the common variance of that factor's items, the same sums over them, so one minus it is the group factor's own share.

`H_<general>`, `H_<s>` Coefficient *H* in the form of Rodriguez, Reise, and Haviland (2016): the closed form of Hancock and Mueller (2001) over the factor's loadings λ_{fi}, with every other source of an item's variance (the other factors' included) counted as its error, $1/(1+1/\sum_i \lambda_{fi}^2/(v_i - \lambda_{fi}^2))$.

`maximal_reliability_<general>`, `maximal_reliability_<s>` The maximal reliability of the factor under the fitted model, $\lambda_f' \Sigma^{-1} \lambda_f$, with λ_f the factor's loadings (zero for an item off it) and Σ the model implied covariance matrix of the items: the squared multiple correlation of the factor with all of the items, the largest reliability any weighted sum of them reaches as a measure of that factor, and the square of the determinacy of the factor's regression score. *H* treats the variance an item shares with the other factors as uncorrelated error, which in a bifactor model it is not, so the two differ, in either direction; `bifactor_indices` says when.

`PUC_<general>` The percentage of uncontaminated correlations, as a proportion: the share of the pairs of items that do not share a group factor, whose correlations reflect the general factor alone. It depends on the structure, not on the data, so it has no standard error or interval.

Rodriguez, Reise, and Haviland (2016) describe how each is read. In brief, omega_H against omega for the total score says how much of the total score's reliable variance is general; omega_HS against omega for a subscale says how much of the subscale score's reliable variance is its own; ECV and PUC describe how close the items are to measuring one thing; and *H* and the maximal reliability describe how well a factor could be measured by an optimally weighted composite. The table reports the numbers with their uncertainty; whether a total or subscale score is worth reporting is the researcher's judgment, and the package draws no line.

The metric of omega. Omega, omega_H, and omega_HS here are the reliabilities of unit-weighted sums of the items as they are scored, the coefficients that `fa_confirmatory` and `semTools::compRelSEM()` compute. `bifactor_indices` computes them from the standardized solution, for sums of the standardized items, as Rodriguez, Reise, and Haviland (2016) write them. The two agree when the items have equal variances and differ otherwise: on the example below, whose tests' variances range from 8.0 to 153.3, omega_H is 0.552 for the sum of the scores and 0.577 for the sum of the standardized scores, and on the simulated example of `bifactor_indices`, whose items have nearly equal variances, 0.67305 against 0.67348. ECV is formed from the standardized loadings, and *H*, the maximal reliability, and PUC do not depend on the metric of the items, so those rows equal `bifactor_indices`'s with its model implied denominator. With `output = "standardized"`, the defined indices are evaluated at the standardized parameters, so the omega rows there are those of the standardized items. **lavaan** evaluates a defined parameter at the standardized values of the labels it names, and a label that `equal_loading` or `equal_error` gives several items has one unstandardized value but a standardized value for each item, which differ when the items' variances differ; **lavaan** uses the first item's. Each index that uses such a label is computed here from every item's own standardized loadings and error variance, as `fa_confirmatory` computes its rows, with a delta method standard error from their derivatives with respect to the free parameters and **lavaan**'s covariance matrix of the estimates (under `se = "bootstrap"`, from **lavaan**'s bootstrap draws); the other indices are **lavaan**'s. ECV and *H* then equal the rows of the other tables.

Intervals. The standard errors of omega, omega_H, omega_HS, ECV, and *H* are **lavaan**'s delta method standard errors of the defined parameters. The maximal reliability needs a matrix inverse, which **lavaan**'s defined parameters cannot express, so its standard error is the delta method's with the derivatives taken numerically over the free parameters and **lavaan**'s covariance matrix of their estimates, as `fa_exploratory` forms those of its determinacies. For every index the limits are formed on the logit scale from that standard error and transformed back (Browne, 1982; the confidence interval transformation principle of Kelley, 2007), as `fa_confirmatory` forms the limits of omega and *H*, so they lie within (0, 1); the `se` column is the standard error of the index itself. The sample values of *H* and of the maximal reliability overstate their population values on average, the more so for a weakly determined factor (Aguirre-Urreta, Rönkkö, & McIntosh, 2019); `ci_bifactor_indices` on the fit of `output = "fit"` gives bootstrap intervals for the indices of `bifactor_indices` and the bootstrap estimate of that bias.

Improper solutions. A bifactor model often gives an improper solution, an error variance estimated below zero, and the estimate is one of two kinds. It can be a finite maximum of the likelihood: on `holzinger_swineford`, nine tests in three groups of three (visual perception, cubes, and lozenges; paragraph comprehension, sentence completion, and word meaning; addition, counting groups of dots, and straight and curved capitals) give visual perception an error variance of -41, and the likelihood falls on either side of that value. Such a solution is reported. The maximal reliability

is not defined for it, and neither is *H* for a factor on which an item's standardized loading reaches one, nor an omega whose items' error variances sum below zero; those rows are NA, with a warning that names the items (beside the warning of class dmar_heywood_warning that fa_confirmatory gives), and the improper attribute holds the negative error variances. The other rows are reported as estimated, and an improper solution's indices are better read as a sign that the model asks more of the data than they hold. Or the likelihood can have no maximum at all. With a fourth group factor on the Woody-McCall mixed fundamentals, revised paper form board, and flags tests beside those three groups, the residual covariances of the fourth group's tests, beyond the general factor, are all negative, which no group factor can reproduce (a single factor implies three covariances whose product is not negative), and the likelihood keeps rising, toward a limit, as the error variance of the paper form board falls without bound. **lavaan**'s optimizer stops at -1163 under FIML, where its steps stop gaining, a point set by its starting values and tolerance rather than by the data, and every index read there would describe the optimizer. So when an error variance is negative, the function fits the model again with it held further out (at ten times its estimate, or one model implied variance below it, whichever is further), and when the log likelihood there is higher, or lower by no more than 10^{-6} per case (a likelihood that flat leaves the estimate as much to where the optimizer stopped as a ridge does), it stops, for every output ("fit" included), with an error of class dmar_no_maximum (see dmar_metadata) that names the test and prints the **lavaan** code of the fit; there, bounds = "pos.var" (which holds every error variance at zero or above, and which ... passes on from here as well) or a different model, such as one with the covariances of the group's items freed in place of the group factor, are the choices.

Ordered items. fa_bifactor() fits continuous items, and ordered is refused. Fit the model to ordered items with lavaan::cfa() (ordered = TRUE, orthogonal = TRUE, std.lv = TRUE) and pass the fit to bifactor_indices, which reports the indices on the metric of the latent responses behind the categories; the bifactor coefficients of a sum of the observed categories are not implemented.

The other bifactor functions. output = "fit" returns the **lavaan** fit, so the same model feeds bifactor_indices, which reports the indices on the standardized metric, without intervals, and can take the observed composite variance as the denominator, and ci_bifactor_indices, which gives them bootstrap intervals, an alternative to the delta method intervals here that does not rest on the normal approximation and that sets improper replications aside.

Row names under term: lambda_<factor>_<j> (loadings, the general factor's numbered over every item, the items of the group factors in their order and then those of general_only; lambda_<factor> when equated), phi_<factor> (the factor variances, fixed at 1), psi_<factor>_<j> and nu_<factor>_<j> (the error variances and intercepts of a group factor's items; psi_<general>_<j> and nu_<general>_<j> for an item on the general factor alone, numbered as its general loading), the indices above, and fa_confirmatory's fit rows chi_square through H1, whose versions and the lines under the table are as there.

Value

For output = "verbose" (default) and "measurement", a data.frame (classes dmar_fa_bifactor, dmar_tbl) with columns syntax, term, estimate, se, test_statistic, p_value, ci_lower, and ci_upper. Its attributes record the model: general, factors, and general_only; equal, a list naming the factors whose loadings, error variances, and intercepts are equated; has_means; omega_metric ("raw_item_sum"); improper, the negative error variances named by item (empty for a proper solution); and wald_df. The verbose table also carries fa_confirmatory()'s test, estimator, missing, fit_indices, and fit_rows. The printed header describes the model, and the lines under the table say what the indices describe. For "summary", the **lavaan** summary; for

`"standardized"`, the standardized solution with the `"wald_df"` attribute; for `"fit"`, the **lavaan** fit. `tidy()` gives the parameter and index rows and `glance()` one row of the fit rows and what produced them, as for `fa_confirmatory`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Aguirre-Urreta, M. I., Rönkkö, M., & McIntosh, C. N. (2019). A cautionary note on the finite sample behavior of maximal reliability. *Psychological Methods, 24*(2), 236–252. doi:10.1037/met0000176

Browne, M. W. (1982). Covariance structures. In D. M. Hawkins (Ed.), *Topics in applied multivariate analysis* (pp. 72–141). Cambridge, UK: Cambridge University Press.

Green, S., & Yang, Y. (2018). Empirical underidentification with the bifactor model: A case study. *Educational and Psychological Measurement, 78*(5), 717–736. doi:10.1177/0013164417719947

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), *Structural equation modeling: Present and future* (pp. 195–216). Scientific Software International.

Holzinger, K. J., & Swineford, F. (1937). The bi-factor method. *Psychometrika, 2*(1), 41–54. doi:10.1007/BF02287965

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

McDonald, R. P. (1999). *Test theory: A unified treatment*. Erlbaum.

Reise, S. P. (2012). The rediscovery of bifactor measurement models. *Multivariate Behavioral Research, 47*(5), 667–696. doi:10.1080/00273171.2012.715555

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016). Evaluating bifactor models: Calculating and interpreting statistical indices. *Psychological Methods, 21*(2), 137–150. doi:10.1037/met0000045

See Also

`fa_confirmatory` for the correlated factors model of the same items; `fa_second_order` for the second-order model whose first-order factors are the group factors, which is nested in the bifactor model; `bifactor_indices` and `ci_bifactor_indices` for the indices of a fitted bifactor model on the standardized metric and their bootstrap intervals; `factor_scores` for the factor scores and their determinacy; `reliability_omega` and `reliability_H`.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
data(holzinger_swineford)
hs_groups <- list(
  spatial = c("t1_visual_perception", "t2_cubes", "t3_paper_form_board",
              "t4_lozenges"),
  verbal  = c("t5_general_information", "t6_paragraph_comprehension",
              "t7_sentence", "t8_word_classification", "t9_word_meaning"),
  memory  = c("t14_word_recognition", "t15_number_recognition",
              "t16_figure_recognition", "t17_object_number",
              "t18_number_figure", "t19_figure_word"))

# A general factor on the 15 tests and three group factors: the
# indices, each with a delta method interval.
fa_bifactor(holzinger_swineford, hs_groups, output = "measurement")

# The fit statistics of the same model, one row.
generics::glance(fa_bifactor(holzinger_swineford, hs_groups))

# The lavaan fit feeds bifactor_indices(), whose omega, omega_H, and
# omega_HS describe sums of the standardized tests (0.577 for omega_H,
# against 0.552 for the sum of the scores above).
fit <- fa_bifactor(holzinger_swineford, hs_groups, output = "fit")
bifactor_indices(fit)
```

`fa_confirmatory` *Multiple-Factor Confirmatory Factor Analysis Model*

Description

Fits a confirmatory factor analysis model with one or more factors, where each factor is specified by naming its indicator variables and the measurement structure is specified by describing what is constrained (`equal_loading`, `equal_intercept`, `equal_error`) rather than by more technical terms. The function then reports which classical measurement structure the description implies (congeneric, essentially tau-equivalent, tau-equivalent, essentially parallel, or parallel), the parameter estimates with confidence intervals, fit information, and, per factor, coefficient omega, the average variance extracted (AVE), and coefficient *H*, each with a delta method standard error and confidence interval computed by **lavaan** from defined parameters (no additional packages are involved). `cfa_k()` is an alias of this function.

Usage

```
fa_confirmatory(
  data = NULL,
  factors,
  S = NULL,
  N = NULL,
  M = NULL,
```

```
  equal_loading = FALSE,
  equal_intercept = FALSE,
  equal_error = FALSE,
  correlated_factors = TRUE,
  meanstructure = NULL,
  estimator = "ML",
  missing = "fiml",
  ordered = NULL,
  se = NULL,
  conf_level = 0.95,
  output = c("verbose", "measurement", "summary", "standardized", "fit"),
  ...,
  wald_df = Inf,
  rmsea_conf_level = 0.9
)

cfa_k(
  data = NULL,
  factors,
  S = NULL,
  N = NULL,
  M = NULL,
  equal_loading = FALSE,
  equal_intercept = FALSE,
  equal_error = FALSE,
  correlated_factors = TRUE,
  meanstructure = NULL,
  estimator = "ML",
  missing = "fiml",
  ordered = NULL,
  se = NULL,
  conf_level = 0.95,
  output = c("verbose", "measurement", "summary", "standardized", "fit"),
  ...,
  wald_df = Inf,
  rmsea_conf_level = 0.9
)
```

Arguments

data	A raw data matrix or data frame, rows are respondents and columns include the items named in factors. Supply exactly one of data or S.
factors	Named list. Each element names a factor and gives the character vector of its indicator columns (two or more per factor; three or more when only one factor is specified). Each item loads on exactly one factor (simple structure). An item that is not a column of data or an item of S is refused with an error of class dmar_items_invalid (see dmar_metadata).
S	A symmetric covariance matrix of the items, with dimnames naming the items;

N is then required. Supply exactly one of data or S.

N — Total sample size. Required with S; ignored (inferred from the rows) with data.

M — Optional named numeric vector of item means, used with a covariance matrix to model the mean structure (required there when equal_intercept is used). Ignored for raw data.

equal_loading — Logical, or a named logical vector with one element per factor. TRUE constrains the loadings within a factor to a single value (across factors nothing is equated). Defaults to FALSE.

equal_intercept — Logical, or a named logical vector with one element per factor. TRUE constrains the item intercepts within a factor to a single value; this requires the mean structure (raw data, or M with a covariance matrix). Defaults to FALSE.

equal_error — Logical, or a named logical vector with one element per factor. TRUE constrains the error variances within a factor to a single value. Defaults to FALSE.

correlated_factors — Logical. If TRUE (default) the factors covary freely; because each factor variance is fixed to 1, the phi terms for factor pairs are the latent correlations. If FALSE the factor covariances are fixed to zero.

meanstructure — Logical or NULL. NULL (default) models the mean structure exactly when it is needed: under FIML or two-stage estimation of raw data (the default, so the item intercepts, the nu rows, join the table), when any equal_intercept is TRUE, or when M is supplied. Set TRUE to model intercepts regardless (raw data or M required), or FALSE to suppress them (an error if equal_intercept is used).

estimator — Character; estimator passed to **lavaan**. Must be one of "ML" (default; maximum likelihood, fully efficient under multivariate normality), "MLR" (robust maximum likelihood, raw data required: maximum likelihood estimates with the test statistic corrected for nonnormality, the scaled statistic of Satorra and Bentler, 1994, in the form Yuan and Bentler, 2000, gave it for incomplete data; its matching Huber-White sandwich standard errors are used when se is omitted), "WLS" (the asymptotic distribution free estimator of Browne, 1984; raw data and a large sample required), "WLSMV" (diagonally weighted least squares with mean- and variance-adjusted test statistic; Muthén, 1984; Muthén, du Toit, & Spisic, 1997; the standard choice for ordered categorical items, and what ordered switches to), or "GLS" (generalized least squares; Browne, 1974). With a robust estimator the reported fit indices are the robust versions.

missing — Character; missing-data handling when raw data are supplied: "fiml" (default; full information maximum likelihood, which uses every case with an observed item; **lavaan**'s name "ml" is accepted), "listwise" (listwise deletion), "pairwise" (passed to **lavaan**, for ordered items), "two_stage" (two-stage maximum likelihood with normal theory standard errors and test), or "robust_two_stage" (two-stage maximum likelihood with standard errors and test robust to nonnormality); lavaan's spellings "two.stage" and "robust.two.stage" are accepted for the last two. See *Details* for two-stage estimation. Ignored with S, except that a two-stage request with S is an error, since a covariance matrix has no incomplete cases. With ordered items, "ml"/"fiml" and the two-stage options are not available; use "pairwise" or "listwise". FIML needs a maximum

likelihood estimator of continuous items with normal theory or Huber-White standard errors. When the default meets ordered items, a least squares estimator (`"WLS"`, `"WLSMV"`, `"GLS"`), `se = "robust.sem"`, or a Satorra-Bentler kind of test passed through `...`, the items are fitted as they stand if no item value is missing (listwise deletion then deletes nothing), and otherwise the call stops and names the option to choose, rather than deleting cases unasked. With complete data FIML and listwise deletion give the same estimates and the same chi square, and for `estimator = "ML"` the same fit indices; the standard errors differ slightly, since FIML takes them from the observed information, as **lavaan** does and as Savalei (2010) recommends with incomplete data, and listwise deletion from the expected information.

`ordered` Ordered categorical items: `NULL` (none, the default), `TRUE` (every item), or a character vector of item names. Requires raw data. Each factor must be all ordered or all continuous. Declaring ordered items switches the estimator to `"WLSMV"` (with a message, unless a categorical estimator was requested), fits thresholds in place of intercepts, and reports each ordered factor's omega on the categorical sum score metric (categorical omega with the unrestricted polychoric denominator); see *Details*.

`se` Standard error method passed to **lavaan**. The default, `NULL`, follows the estimator: model-based `"standard"` errors for ML, GLS and WLS; Huber-White sandwich `"robust.huber.white"` errors for MLR; and `"robust.sem"` errors for WLSMV. Explicit compatible choices are retained. The ML and MLR point estimates use the same likelihood; their standard errors and tests use different reference assumptions. This difference does not depend on whether optimization is iterative or an equivalent solution is computed analytically. `"robust.sem"` requires complete data and is refused with FIML; use `"robust.huber.white"` to retain incomplete respondents. No standard-error request silently changes the missing-data treatment. `"none"` requests point estimates only. Under two-stage estimation the matching corrected two-stage errors are used; `se` must be `NULL`, `"standard"`, or `"none"`. MLR and Huber-White errors require raw data. Robust SEM errors from covariance input require ML and a matching `NACOV` in `...`.

`conf_level` Confidence level for the parameter confidence intervals, including the delta method intervals for omega, AVE, and *H*. Defaults to 0.95. It does not reach the RMSEA interval, whose level is `rmsea_conf_level`.

`output` Format of the returned object:

`"verbose"` (default) Parameter estimates with confidence intervals, the per-factor defined parameters (`loading_sum`, `error_sum`, `omega`, `ave`, `H`), and fit information.

`"measurement"` The measurement-property rows only: per factor `omega`, `ave`, and `H` (with delta method standard errors and confidence intervals), the latent correlation `phi` for every factor pair (with its confidence interval), and, for raw data, the heterotrait-monotrait ratio `htmt` for every factor pair via `htmt`.

`"summary"` The raw `summary()` output from **lavaan** (not a data frame).

`"standardized"` The standardized parameter estimates from `lavaan::standardizedSolution()`, whose defined rows describe the standardized items: `omega` there is the reliability of the unit-weighted sum of the standardized items, while `ave` and

H do not depend on the metric of the items and equal the rows of the other tables. For a factor with equal_loading or equal_error these rows are computed from each item's own standardized loading and error variance (see *The standardized solution* under *Details* of fa_confirmatory). Under se = "none" the se, z, pvalue, ci.lower, and ci.upper columns are NA.

"fit" The raw **lavaan** fit object. Escape hatch for direct lavaan access, including likelihood ratio tests between two fa_confirmatory() fits via lavaan::lavTestLRT().

... Additional arguments forwarded to lavaan (e.g., cluster or bootstrap). A group argument is an error: see *Details* for why, and measurement_invariance for the model in several groups.

wald_df The reference distribution of the Wald intervals and *p*-values of the parameter rows: Inf (the default), the normal distribution, which leaves **lavaan**'s intervals and *p*-values as they are; a single positive number, the degrees of freedom of a *t* distribution for every parameter row, the intervals and *p*-values recomputed from the same estimates and standard errors; or "missing_information", von Hippel's (2016) degrees of freedom $\max\{3, N(1-\gamma_j)-1\}$ for each loading, from its fraction of missing information γ_j ($N-1$ with complete data), with every other row left on the normal reference. A number applies to the tables of output = "verbose", "measurement", and "standardized", the keyword to that of "verbose" only (the "measurement" table has no loading), and either needs standard errors other than "none" and "bootstrap"; the keyword also needs the fits its calibration covers, maximum likelihood (estimator = "ML" or "MLR") of continuous items by FIML or with complete data, with normal theory or Huber-White standard errors. Anything else is an error that says why. It follows ..., so it must be named in full. See *Details*.

rmsea_conf_level Confidence level of the RMSEA interval of the fit rows, which the rmsea_ci_level row records. Defaults to 0.90, the conventional level, whose limits decide the tests of close and not close fit at .05 (see sem_fit_measures); rmsea_conf_level = 0.95 gives a 95 percent interval. It sets the level in the table DMAR returns and in its glance(); output = "summary" returns the **lavaan** summary, which shows lavaan's 90 percent interval whatever this argument is.

Details

With ordered items, the model is fit by WLSMV to polychoric correlations with thresholds. A sum score of ordered items lives on the metric of the observed categories, not on the latent response metric of the polychoric loadings, so for a factor whose items are ordered the reported omega is categorical omega, the reliability of the sum of the observed categories (Green & Yang, 2009), computed from the same fit with the total variance built from the unrestricted polychoric correlations (Kelley & Pornprasertmanit, 2016, Eq. 18), the default denominator of reliability_omega_categorical; the substitution is announced in a message and recorded in the omega_metric attribute, and the delta method interval columns are NA for those rows because the delta method interval describes the latent response metric. When the data do not determine a polychoric correlation among a factor's items (a bivariate table with empty cells, as two items skewed in opposite directions produce), that factor's omega is NA with a warning naming the pair. AVE and coefficient *H* concern the latent

response correlations themselves and are reported unchanged on that metric. An improper solution (a negative error variance) leaves some of these properties undefined, and their rows are then NA with a warning; see *Details*. For a bootstrap confidence interval on a categorical omega, use reliability_omega_categorical on the factor's items.

cfa_1 and cfa_2 are convenience wrappers around this function for the one and two factor cases: cfa_1() takes a vector of items and fits one factor over them, and cfa_2() takes the items of each of two factors. Both forward every argument here, so their results are this function's results, with the factors named f1 (and f2).

Describing the model instead of naming it. The classical measurement structures are nested patterns of within-factor equality constraints (Lord & Novick, 1968; Graham, 2006):

congeneric loadings, intercepts, and error variances all free.

essentially tau-equivalent equal loadings; intercepts and error variances free.

tau-equivalent equal loadings and equal intercepts; error variances free.

essentially parallel equal loadings and equal error variances; intercepts free.

parallel equal loadings, equal intercepts, and equal error variances.

The caller states the constraints; the function reports the implied name, per factor, in the printed header and in the "model" attribute of the returned table. The distinction between tau-equivalent and essentially tau-equivalent (and between parallel and essentially parallel) lives entirely in the mean structure: the covariance structure of the two members of each pair is identical, so without intercepts in the model only the "essentially" form can be claimed. That is why equal_intercept requires raw data or M: covariances alone cannot speak to it. A constraint pattern outside the classical list (for example equal error variances with free loadings) is fit as requested and labeled descriptively, since it has no conventional name.

Identification fixes each factor variance to 1 (and each factor mean to 0 when the mean structure is modeled), so all loadings are estimated and within-factor equality constraints are meaningful. Two fa_confirmatory() fits that differ only in descriptor settings are nested, so output = "fit" feeds lavaan::lavTestLRT() directly (with estimator = "MLR", lavaan applies the scaled difference test).

Measurement properties. For factor f with unstandardized loadings λ_j and error variances ψ_j (factor variance 1): coefficient omega $= (\sum_j \lambda_j)^2/((\sum_j \lambda_j)^2 + \sum_j \psi_j)$ (McDonald, 1999), the reliability of the unit-weighted composite; the average variance extracted $= J^{-1}\sum_j \lambda_j^2/(\lambda_j^2 + \psi_j)$ (Fornell & Larcker, 1981), the mean proportion of item variance the factor accounts for; and coefficient $H = (1 + (\sum_j \lambda_j^2/\psi_j)^{-1})^{-1}$ (Hancock & Mueller, 2001), the reliability of the optimally weighted composite, which no single item can drag below its value for any subset. All three are computed as **lavaan** defined parameters, so each carries a delta method standard error and a conf_level confidence interval in the same table as the model parameters. The limits of all three are formed on the logit scale from that standard error and transformed back to the scale of the coefficient (Browne, 1982; the confidence interval transformation principle of Kelley, 2007: the limits of a monotone transformation of a parameter are the transformation of its limits), as reliability_H and fa_exploratory form those of *H* by default, so they lie within (0, 1) and are asymmetric near one, as the sampling distribution of a high reliability is; for *H* with one factor they are the limits those two functions give the same fit. The se column is the standard error of the coefficient itself. **lavaan**'s own limits for these defined parameters, the estimate plus or minus the normal quantile times the standard error, can exceed one.

The standardized solution. With `output = "standardized"` the defined rows are evaluated at the standardized values: the standardized loading $s_j = \lambda_j/(\lambda_j^2 + \psi_j)^{1/2}$ and error variance $e_j = \psi_j/(\lambda_j^2 + \psi_j)$ of each item take the places of λ_j and ψ_j in the formulas above (for an ordered item, those of its latent response). Omega is then the reliability of the unit-weighted sum of the standardized items, which differs from the omega of the sum of the items as they are scored whenever the items' variances differ; the average variance extracted and *H* are the same on either scale. **lavaan** evaluates a defined parameter at the standardized values of the labels it names, and a label that `equal_loading` or `equal_error` gives several items has one unstandardized value but a standardized value for each item, which differ when the items' variances differ; **lavaan** uses the first item's. For such a factor the rows are computed here from every item's own s_j and e_j, with delta method standard errors from their derivatives with respect to the free parameters and **lavaan**'s covariance matrix of the estimates (under `se = "bootstrap"`, from **lavaan**'s bootstrap draws), and with the test statistic and *p*-value, as **lavaan** forms them for every other row. `fa_bifactor` does the same for its indices.

Improper solutions. When the error variance ψ_j of an item is estimated below zero (a Heywood case; for an ordered item, the residual variance of its latent response, which the theta parameterization estimates), the item's communality $\lambda_j^2/(\lambda_j^2 + \psi_j)$ exceeds one. The average variance extracted averages the communalities, and coefficient *H*, the largest reliability that a weighted sum of the items attains, is at least that communality, since the item weighted alone attains it; for such a factor neither is a proportion of variance, so its `ave` and `H` rows are `NA`, with a warning naming the items and their communalities. Omega, the reliability of the unit-weighted sum, depends on the error variances only through their sum, and while that sum is positive it remains a proportion and is reported, as `reliability_omega` reports it, beside the warning; when the sum is negative the model implies a negative error variance for the sum score and an omega above one, and the `omega` row is `NA` as well. Categorical omega is not defined for an improper ordered solution, so an ordered factor's `omega` row is `NA` whenever one of its items is improper. On `holzinger_swineford`, a factor measured by visual perception and lozenges alone, beside one measured by general information, paragraph comprehension, and sentence completion, gives visual perception an error variance below zero and the first factor an *H* of 1.32; that row is `NA`, while its omega, .71, is reported. A negative error variance can also mean that the likelihood has no maximum, the variance falling without bound while the likelihood still rises and **lavaan**'s optimizer stopping where its steps stop gaining; the function checks for this by fitting the model again with the variance held further out (see `fa_bifactor`) and, when the log likelihood there is higher or lower by no more than 10^{-6} per case, stops for every `output` with an error of class `dmar_no_maximum` that prints the **lavaan** code of the fit, where `bounds = "pos.var"` (also accepted through `...` here) or a different model are the choices. The **lavaan** fit (`output = "fit"`) keeps the defined parameters as **lavaan** computes them.

Discriminant validity. Three complementary readings come from `output = "measurement"`: (a) the latent correlation `phi` for a factor pair, with a confidence interval whose upper limit near 1 means the data cannot distinguish the two factors; (b) the Fornell and Larcker (1981) comparison, which asks whether each factor's `ave` exceeds the squared `phi` of its pairs (the factor should share more variance with its own items than with the other factor); and (c) for raw data, the model-free `htmt` ratio (Henseler, Ringle, & Sarstedt, 2015). A pair whose ratio the data leave undefined, as when some items of two factors are never observed together in a planned missing data design whose forms share other items, has an `htmt` row of `NA`, with a warning that names it, while the model based rows are reported: FIML (or two-stage estimation) estimates the latent correlation from the pairs of their items that are observed together, through the factor structure. When no item of one factor is ever observed with an item of the other, as when the two factors were given to different respondents, the latent correlation enters the likelihood only through covariances the data never observe, so it is not identified

(the likelihood is flat in it, and **lavaan** leaves it at its starting value with no standard error, or under two-stage estimation fits it to an arbitrary covariance), and `fa_confirmatory()` refuses the fit, naming the pair: fit each factor alone, whose `omega`, `ave`, and `H` are identified, or link the forms with items given on both (`missing_coverage` reports the cases behind every pair of items). Orthogonal factors (`correlated_factors = FALSE`) are not refused: their correlation is fixed at 0 rather than estimated, so nothing is left unidentified, and since the likelihood then separates into the parts of the factors' items, each factor's `omega`, `ave`, and `H`, with their standard errors, are those of its own one factor fit (to the accuracy of the optimizer). The fit rows of such a fit are **lavaan**'s, whose degrees of freedom count the covariances no case observes among the moments tested (**lavaan** warns that their coverage is zero), though the data cannot test them; each factor's fit is better judged from its own fit. More generally, when **lavaan** cannot compute the standard errors of a fit (its information matrix cannot be inverted, which suggests a parameter that is not identified at the estimates), every standard error, test, and interval is `NA`, with a warning of class `dmar_result_qualified`, rather than the 0 **lavaan** reports for the defined quantities. So they are when **lavaan** computes them from an information matrix that is singular in the directions the equality constraints leave free (scaled to a unit diagonal, its smallest eigenvalue is below 10^{-6}), which it inverts, sometimes with a warning that the covariance matrix of the estimates does not appear to be positive definite and sometimes with none, reporting finite values that do not estimate the sampling variability of the estimates: the observed information of a FIML fit and the expected information of a listwise fit or of a covariance matrix alike, as in `fa_bifactor`, `fa_second_order`, and `reliability_H`. The rows report the numbers and their uncertainty; the judgment is the researcher's.

Two-stage estimation with incomplete data. With `missing = "two_stage"` the model is estimated in two stages (Yuan & Bentler, 2000; Savalei & Bentler, 2009). The first stage estimates the means and covariances of the items by maximum likelihood under the saturated normal model (the EM estimates), from every case with an observed item and from any auxiliary variables named through **lavaan**'s `aux` argument, which enter this stage only, so the model itself is untouched by them. A case observed on an auxiliary variable but on no item is used as well: the first stage estimates the items and the auxiliaries jointly, so its auxiliary values inform the item moments, and its likelihood is summed over every case (Savalei & Bentler, 2009, Eq. 10). That is what makes the estimates consistent when the items go missing together at random given the auxiliary variables, the case an auxiliary is named for. **lavaan**'s two-stage estimator reads such a case in its first stage but leaves it out of its sample size, and that sample size turns the per case quantities of the two stages into every standard error (each too large by the square root of the ratio of the two counts), the test statistics, the fit indices, and the information criteria. The fit is therefore computed again with its sample size set to the cases the first stage read, from the same estimates (in one group they do not depend on the sample size), so that every number in the table, and in the **lavaan** fit of `output = "fit"`, counts them: **lavaan**'s `summary()` of that fit counts them as used, and `lavTestLRT` warns that a difference test between it and a fit **lavaan** made of the same data on its own sample size is invalid. A message says how many such cases there are, and a line under the table names the sample size. When the fit cannot be computed again, a warning says what rests on **lavaan**'s sample size instead. On six Holzinger and Swineford tests with the whole block missing for 77 of 301 pupils, at random given general information (the auxiliary), the standard error of a verbal loading of 2.840 is .190 rather than **lavaan**'s .220. Cases observed only on auxiliary variables are kept, as under FIML. Such cases still contribute to the observed-data likelihood through their auxiliary measurements. The second stage fits the model to those moments with the complete data maximum likelihood discrepancy function. The estimates are consistent when the data are missing at random, as FIML's are, and Savalei and Bentler (2009) found them only marginally less efficient. What the second stage cannot supply alone is a standard error or a test, because it treats the EM moments as

the moments of complete data, which makes its naive chi square and standard errors too optimistic. The standard errors are therefore the two-stage sandwich of Savalei and Bentler (2009, Eq. 6), built from the observed information of the saturated model under missing data, and the `chi_square` row is their residual based statistic (Eq. 8; Eq. 14 with auxiliary variables), which is referred to chi square on the model's degrees of freedom. In their simulations with normal data and 15 or 30 percent of the values missing it rejected a correct model in no more than 9 percent of the tests at the .05 level with samples of 200, where the FIML chi square reached 21.9 percent. With `missing = "robust_two_stage"` the standard errors are the sandwich that is robust to nonnormality (Yuan & Bentler, 2000) and the `chi_square` row is lavaan's scaled statistic, the pairing Savalei and Falk (2014) found to outperform robust FIML with incomplete nonnormal data. Under both, `cfi` and `rmsea` are lavaan's robust indices, which correct the complete data indices of the second stage for the missing data as Zhang and Savalei (2023) propose: each subtracts from the naive statistic T an estimate c of its expectation under the model, built from the asymptotic covariance matrix of the stage one moments. With `missing = "robust_two_stage"` that matrix is the sandwich of the observed and first order information of the saturated model, and the indices are their TS V2 (Table 2), the version they recommend after two-stage estimation, for incomplete nonnormal data as well. With `missing = "two_stage"` it is the inverse of the observed information, the form the sandwich takes under normality, so the indices are the normal theory analog of TS V2, which Zhang and Savalei did not study: it estimates the same population values and agrees closely with TS V2 when the data are normal, but it departs from TS V2 as the data depart from normality, where `"robust_two_stage"` gives TS V2 itself. On nine Holzinger and Swineford tests with 90 scores on each of two deleted at random, the two RMSEAs are .0869 and .0859. Zhang and Savalei define only the RMSEA and the CFI; the `tli` and `nnfi` rows apply the same correction to the Tucker-Lewis index, and the RMSEA interval inverts the scaled statistic T/c and rescales its limits by c, both of them **lavaan**'s extensions. Indices from a FIML fit are never paired with two-stage estimates. The `AIC` and `BIC` rows are the two-stage information criteria of Lai (2021), built, like `H0` and `H1`, from the complete data normal log likelihood evaluated at the EM estimates; Lai shows that, unlike the FIML criteria, they select in large samples the model that complete data would select, and they are comparable only with other two-stage fits of the same variables (`sem_information_criteria` reports the same values). The difference $-2(H0-H1)$ is the naive statistic, which is not chi square distributed. Two nested two-stage fits from `output = "fit"` are compared with `lavaan::lavTestLRT()`, which applies the scaled difference test (Satorra & Bentler, 2001). The `test` attribute names the statistic in the `chi_square` row, and the lines under the printed table say which statistic and indices it reports. Two-stage estimation is a normal theory method for raw continuous data: ordered items, covariance input, and an estimator other than `"ML"` are refused, robustness coming from `"robust_two_stage"` rather than from `estimator = "MLR"`. Under `"two_stage"` the residual based statistic is refused, with a message that points to robust two-stage estimation, for a fit with sampling weights or clustered cases, inequality constraints, or a parameter on a bound passed through `...`, since the statistic assumes independent, equally weighted cases and a projection on the parameters left free. The residual based statistic already sums the information of the saturated model over every case the first stage read, so the correction for cases observed on an auxiliary variable alone leaves it as it was.

One group. The model is fit in one group. Every loading, error variance, intercept, and factor correlation carries a label, which is how the equality descriptors and the defined parameters refer to it, and **lavaan** holds a label shared by several groups equal across them. With `group` passed through `...` the groups would therefore share every measurement parameter, with each factor's variance fixed at 1 (and its mean at 0) in every group: a model of complete invariance rather than a model fit by group, whose omega, `ave`, and H belong to the pooled model rather than to any one group. A `group` argument is therefore an error. `measurement_invariance` fits the model in several

groups, beginning with the configural model, which leaves every parameter free in each group; `reliability_H` reads *H* for one factor in one group from its fit, and `reliability_omega` on one group's rows gives that group's omega.

Confidence interval conventions. Parameter rows (including omega, `ave`, *H*, and `phi`) use `conf_level`, 0.95 by default. The RMSEA interval uses `rmsea_conf_level`, 0.90 by default, the conventional level for the RMSEA, because a $100(1-2\alpha)$ percent interval decides the one-sided tests of close and not close fit at level α (MacCallum, Browne, & Sugawara, 1996, p. 136); the `rmsea_ci_level` row records that level, and the line beneath the printed table names it when it differs from `conf_level`.

The parameter intervals are Wald intervals, the estimate plus or minus a normal quantile times its standard error, and the *p*-values refer the ratio of the two (`test_statistic`) to the normal distribution. The normal reference treats the standard error as known, which it is only in the limit, and in small samples, more so with missing values, the intervals are too short (von Hippel, 2016). `wald_df` refers both to the *t* distribution on that many degrees of freedom instead (von Hippel's Eq. 29): the limits become the estimate plus or minus the *t* quantile times the same standard error (for the omega, AVE, and *H* rows, on the logit scale) and the *p*-value is $2P(T_\nu > |z|)$; the `test_statistic` column holds the same ratio under either reference, and a line under the printed table names the reference distribution. One number applies to every parameter row, a sensitivity check of the analyst's own choosing. The fit rows, the RMSEA interval, categorical omega, and the HTMT rows are untouched.

Degrees of freedom from the missing information. With `wald_df = "missing_information"` each loading gets its own degrees of freedom by von Hippel's (2016, Eqs. 31 and 34) effective sample size rule with $k = 1$,

$$\nu_j = \max\{3, N(1 - \gamma_j) - 1\},$$

where N is the number of cases the fit used and γ_j the parameter's fraction of missing information, estimated from the fit as `fraction_missing_information` estimates it (the unstructured estimator of Chen and Savalei, 2021, which cannot be negative); `df_missing` describes the rule and its floor. A loading of an item that is rarely missing keeps nearly $N - 1$ degrees of freedom, and with complete data (listwise deletion, or a covariance matrix) every γ_j is 0 and every ν_j is $N - 1$. The `"wald_df"` attribute holds each parameter's degrees of freedom, the `"wald_df_table"` attribute the `df_missing` table they came from, and the line under the printed table names the rule and the rows it applied to.

The rule, and $k = 1$ rather than k equal to the number of items or von Hippel's complete data rule (his Eq. 32) on $N - 1$ or N less the number of items, come from a calibration study of these intervals under full information maximum likelihood (in the package's source repository, `methodology/missing_data`). It simulated one factor models as this function writes them (loadings from .8 to .5, the factor variance fixed at 1), with six items at *N* = 30, 50, 100, and 200 and with 12 items at *N* = 50 and 100, normal data fit by ML with normal theory standard errors and nonnormal data (the factor and the errors drawn from a standardized chi square on 4 degrees of freedom, skewness 1.41 and excess kurtosis 3) fit by MLR with Huber-White standard errors, with no missing values or with 15 or 30 percent of the values of half the items missing completely at random or at random, in 2000 replications of each of 60 cells; a repetition of von Hippel's own design reproduced the coverage he published to within .012. Under a decision rule fixed before the run (mean absolute distance of the coverage from .95, over the cells with $N \leq 100$ and missing values, below the normal interval's; coverage above .96 in no cell; a mean distance within .005 of the best rule's; the simplest passing rule chosen, a cell's coverage being the mean over the target's parameters), $k = 1$ passed for the loadings, with a mean distance of .0171 against .0247 for the normal reference and no cell's coverage, averaged over its loadings, above .9502 (single loadings, the

largest of 384 with missing values, reached .9595, with a Monte Carlo standard error of .0044, and none exceeded .96), and for omega, .0180 against .0242 with no coverage above .9550. It passed again within the normal theory cells and within the robust ones taken alone. With nonnormal data the intervals still covered below .95 (between .908 and .937 for the loadings, averaged over a cell's loadings), since the robust standard errors themselves are too small at these sample sizes: the *t* reference narrows the shortfall and does not remove it.

These coverage figures rest on the data sets that the study's eligibility rule for the unrestricted (H1) estimates admitted. The affine-collapse diagnostic sets aside some of those data sets, and the coverage over the data sets that remain has not been established. Replaying the eligibility rule on the raw data cannot by itself validate the refits of the fraction of missing information and the information diagnostics. The keyword is an approximate, opt-in rule, not a coverage guarantee.

The keyword is restricted to the study's estimator and parameter types. The other parameter rows keep the normal reference: the study evaluated the unstandardized loadings and omega, not factor variances or correlations, intercepts, error variances, or the sums, AVE, and *H* built from them, and nothing shows that the rule suits them; omega passed with its limits formed on its own scale, but its limits here are formed on the logit scale, where the study found the *t* reference overcovered (.9645 with normal data), so it keeps the normal reference as `reliability_omega`'s logistic intervals do. It is an error with the `"measurement"` table, which has no loading, with the standardized solution, with an estimator other than ML and MLR or ordered items, with pairwise deletion or two-stage estimation (for which no fraction of missing information is computed), and with standard errors other than the normal theory and Huber-White ones (`se = "standard"` and `"robust.huber.white"`; `"robust.sem"` only under listwise deletion, or from a covariance matrix with the asymptotic covariance matrix of its moments passed to **lavaan** as `NACOV` through `...`, without which **lavaan** refuses it, since **lavaan** computes that sandwich from raw data or a supplied `NACOV`, and under FIML falls back to listwise deletion, so that `missing = "fiml"` would not be honored). More generally, it is an error wherever **lavaan** replaces the FIML fit that `missing = "fiml"` asks for with a fit to the complete cases, which it also does for a test statistic of the Satorra-Bentler kind passed through `...` (`test = "satorra.bentler"`, `"mean.var.adjusted"`, `"scaled.shifted"`, and their corrected forms), whether or not a value is missing: `estimator = "MLR"`, whose scaled statistic is computed under FIML, or the test with `missing = "listwise"` is the alternative. It is an error, too, where `fraction_missing_information` cannot compute the fraction for the data: a pair of items that no case observes together, or an unrestricted estimate of the means and covariances that does not exist (see its *Fits that are refused*). Under FIML the standard errors must come from the observed information computed from the Hessian of the fitted model, **lavaan**'s default there and the one the study used; the first order information and the observed information of the unrestricted model (`observed.information = "h1"`), which reach **lavaan** only through `...`, are an error. Under listwise deletion (or from a covariance matrix) **lavaan**'s normal theory standard errors use the expected information where the study's cells without missing values used the observed; at $N = 30$ and 50 they are 1 to 2 percent shorter for the loadings and less than 1 percent shorter for omega, so the complete data intervals here are, if anything, slightly shorter than those the study evaluated, whose coverage with normal data, averaged over a cell's loadings, rose from between .936 and .946 to between .945 and .949 on $N - 1$ degrees of freedom (single loadings between .934 and .963, where the normal reference reached .962). Several factors, equal loadings, and auxiliary variables are accepted, and the rule is applied to each loading as it stands, which the one factor models of the study support by analogy rather than directly.

Which version of each fit row. With complete continuous data and `estimator = "ML"` the fit rows are the standard ones, under FIML (the default) as under listwise deletion: with no value missing the FIML corrected indices described below equal them, and no line under the table names a ver-

sion. With a robust or categorical estimator, `chi_square` is the scaled statistic and `cfi`, `tli`, `nnfi`, `rmsea`, and the RMSEA interval are **lavaan**'s robust versions, the corrections `sem_fit_measures` describes. One version holds all of those rows. For the older mean and variance adjusted statistic (`test = "mean.var.adjusted"` passed through `...`), **lavaan** computes no robust versions, and they are computed here as `sem_fit_measures` computes them: Savalei (2018) shows that the robust RMSEA and CFI do not depend on which corrected statistic accompanies them, so they equal those of the Satorra-Bentler statistic of the same fit. Where **lavaan** did not compute the robust indices of a maximum likelihood fit of continuous items with complete data, the rows hold the standard indices, which estimate the population values of the ML indices with the small sample bias corrected as if the data were normal, and not the scaled ones, which put the scaled statistic into the standard formulas and estimate population values that change with the kurtosis of the data (Savalei, 2018); after FIML or two-stage estimation, and for ordered items or a least squares estimator, they hold its scaled indices, or else its standard ones. A model with no degrees of freedom (one factor of three items, say) fits exactly in every version: its `cfi` is 1, the model term $T - c \cdot df$ of every version being zero, and its `tli`, `nnfi`, and `rmsea`, which divide by the degrees of freedom and are 0/0, take **lavaan**'s conventions for a model that fits exactly, 1 and 0, whatever the version, and no line under the table names a version for them, under FIML and after two-stage estimation as well (the `srmr` line and the two-stage lines on the test, the criteria, and the cases are kept). A model with no degrees of freedom whose statistic is not zero does not fit exactly, as when an inequality bound binds (lavaan's count of the degrees of freedom leaves it out): its `cfi` is formed from its statistic in the version its CFI alone decides, and its `tli`, `nnfi`, `rmsea`, and RMSEA limits, which divide by the degrees of freedom, are undefined and `NA`, as `sem_fit_measures` reports them, with a line under the table that says so. One factor of three variables correlated .8, .8, and .5, with `bounds = "pos.var"` passed through `...`, has a statistic of 32.8 on 0 degrees of freedom (N = 200) and a `cfi` of .925; its `tli`, `nnfi`, `rmsea`, and RMSEA limits are `NA`, where **lavaan** reports the values of exact fit, 1 and 0, whenever the degrees of freedom are zero. Under FIML (`missing = "ml"` or `"fiml"`) the `chi_square` row is the FIML likelihood ratio statistic (its scaled version with `estimator = "MLR"`), and those rows are **lavaan**'s robust versions, which for a FIML fit are the corrected indices of Zhang and Savalei (2023): they evaluate the complete data fit function at the FIML estimates, with the saturated FIML estimates in place of the sample moments, and so approximate the indices complete data would give, where the usual indices computed from the `chi_square` row change with the amount of missing data. **lavaan** uses their version 6, which assumes normality, and their version 3, which does not, with `estimator = "MLR"`. Zhang and Savalei define only the RMSEA and the CFI; the TLI, the NNFI, and the RMSEA interval are **lavaan**'s extension of the same correction. **lavaan** cannot combine FIML with the Satorra-Bentler or the mean and variance adjusted statistic (`test = "satorra.bentler"` or `"mean.var.adjusted"` passed through `...`); an explicit FIML request with one of them stops rather than returning a fit to the complete cases, and the default fits the items as they stand when none is missing (see `missing`). The `srmr` row is the SRMR of the covariance residuals alone (**lavaan**'s `srmr_bentler_nomean` when the model has a mean structure), as `sem_fit_measures` reports it, so that the intercepts a FIML fit adds, whose residuals are zero, do not lower it (**lavaan**'s `srmr` of a model with a mean structure averages the mean residuals in). With auxiliary variables (**lavaan**'s `aux` under FIML) it is over the items alone, for the reason `sem_fit_measures` gives. A line under the printed table names the version of the indices after FIML with incomplete data, with `estimator = "MLR"`, and after two-stage estimation, when they fall back from the robust ones, and when DMAR computed them.

Common row names under `term`: `lambda_<factor>_<j>` (loadings; `lambda_<factor>` when equated), `psi_<factor>_<j>` (error variances; `psi_<factor>` when equated), `nu_<factor>_<j>` (intercepts, when the mean structure is modeled; `nu_<factor>` when equated), `phi_<factor>` (factor vari-

ance, fixed to 1), `phi_<factor1>_<factor2>` (latent correlation), the per-factor defined parameters (`loading_sum_<factor>`, `error_sum_<factor>`, `omega_<factor>`, `ave_<factor>`, `H_<factor>`), and the fit rows `chi_square`, `df`, `p_chi_square`, `cfi`, `tli`, `nnfi`, `rmsea`, `rmsea_ci_lower`, `rmsea_ci_upper`, `rmsea_ci_level`, `srmr`, `AIC`, `BIC`, `H0`, `H1`.

Value

For `output = "verbose"` (default) and `output = "measurement"`, a `data.frame` (classes `dmar_fa_confirmatory`, `dmar_cfa_k`, and `dmar_tbl`) with columns `syntax`, `term`, `estimate`, `se`, `test_statistic`, `p_value`, `ci_lower`, `ci_upper`. The `"model"` attribute is a named character vector giving, per factor, the implied classical structure; the printed header displays it. For `output = "verbose"` the `"test"` attribute names the statistic in the `chi_square` row: `"standard"`, `"scaled"` (a robust or categorical estimator), or, under the two-stage options, the two-stage residual based or scaled statistic; a two-stage fit also carries the `"two_stage"` attribute, a list with the method, the lines printed under the table, and, in `rows`, the cases the first stage read and **lavaan**'s count of them when an auxiliary variable was named, and the `"fit_rows"` attribute holds the lines that name the version of the indices after FIML estimation and the SRMR of a model with a mean structure (`NULL` when there are none). The tables of `"verbose"` and `"measurement"` also carry the `"wald_df"` attribute, the degrees of freedom of the *t* reference of their intervals and *p*-values (`Inf` for the normal one); under `wald_df = "missing_information"` it holds the degrees of freedom of each loading in the table, named by term (the other rows keep the normal reference), the `"wald_df_table"` attribute is the `df_missing` table they came from, and `"wald_df_rows"` names what they are. For `output = "summary"`, the **lavaan** summary object; for `"standardized"`, the standardized solution, with the same `"wald_df"` attribute; for `"fit"`, the **lavaan** fit object.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Brosseau-Liard, P. E., & Savalei, V. (2014). Adjusting incremental fit indices for nonnormality. *Multivariate Behavioral Research, 49*(5), 460–470. doi:10.1080/00273171.2014.933697

Brosseau-Liard, P. E., Savalei, V., & Li, L. (2012). An investigation of the sample performance of two nonnormality corrections for RMSEA. *Multivariate Behavioral Research, 47*(6), 904–930. doi:10.1080/00273171.2012.715252

Browne, M. W. (1974). Generalized least squares estimators in the analysis of covariance structures. *South African Statistical Journal, 8*, 1–24.

Browne, M. W. (1982). Covariance structures. In D. M. Hawkins (Ed.), *Topics in applied multivariate analysis* (pp. 72–141). Cambridge, UK: Cambridge University Press.

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*, 62–83.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika, 49*(1), 115–132.

Muthén, B., du Toit, S. H. C., & Spisic, D. (1997). *Robust inference using weighted least squares and quadratic estimating equations in latent variable modeling with categorical and continuous outcomes*. Unpublished technical report.

Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis. In A. von Eye & C. C. Clogg (Eds.), *Latent variables analysis: Applications for developmental research* (pp. 399–419). Thousand Oaks, CA: Sage.

Chen, L., & Savalei, V. (2021). Three sample estimates of fraction of missing information from full information maximum likelihood. *Frontiers in Psychology, 12*, 667802. doi:10.3389/fpsyg.2021.667802

Fornell, C., & Larcker, D. F. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of Marketing Research, 18*(1), 39–50.

Graham, J. M. (2006). Congeneric and (essentially) tau-equivalent estimates of score reliability: What they are and how to use them. *Educational and Psychological Measurement, 66*(6), 930–944. doi:10.1177/0013164406288165

Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika, 74*(1), 155–167. doi:10.1007/s11336008-90993

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), *Structural equation modeling: Present and future* (pp. 195–216). Scientific Software International.

Henseler, J., Ringle, C. M., & Sarstedt, M. (2015). A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the Academy of Marketing Science, 43*(1), 115–135. doi:10.1007/s1174701404038

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Lai, K. (2021). Using information criteria under missing data: Full information maximum likelihood versus two-stage estimation. *Structural Equation Modeling, 28*(2), 278–291. doi:10.1080/10705511.2020.1780925

Lord, F. M., & Novick, M. R. (1968). *Statistical theories of mental test scores*. Addison-Wesley.

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130–149. doi:10.1037/1082989X.1.2.130

McDonald, R. P. (1999). *Test theory: A unified treatment*. Erlbaum.

Satorra, A., & Bentler, P. M. (2001). A scaled difference chi-square test statistic for moment structure analysis. *Psychometrika, 66*(4), 507–514. doi:10.1007/BF02296192

Savalei, V. (2010). Expected versus observed information in SEM with incomplete normal and nonnormal data. *Psychological Methods, 15*(4), 352–367. doi:10.1037/a0020143

Savalei, V. (2018). On the computation of the RMSEA and CFI from the mean-and-variance corrected test statistic with nonnormal data in SEM. *Multivariate Behavioral Research, 53*(3), 419–429. doi:10.1080/00273171.2018.1455142

Savalei, V., & Bentler, P. M. (2009). A two-stage approach to missing data: Theory and application to auxiliary variables. *Structural Equation Modeling, 16*, 477–497. doi:10.1080/10705510903008238

Savalei, V., & Falk, C. F. (2014). Robust two-stage approach outperforms robust full information maximum likelihood with incomplete nonnormal data. *Structural Equation Modeling, 21*(2), 280–302. doi:10.1080/10705511.2014.882692

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*, 165–200.

Zhang, X., & Savalei, V. (2023). New computations for RMSEA and CFI following FIML and TS estimation with missing data. *Psychological Methods, 28*(2), 263–283. doi:10.1037/met0000445

See Also

`cfa_1` and `cfa_2` for the one and two factor convenience wrappers; `fa_bifactor` for the bifactor model, a general factor beside group factors, and `fa_second_order` for second-order factors over the factors, both fit by the same engine; `plot_fa_confirmatory` to display the estimates and the equality question visually; `reliability_omega`, `reliability_H`, `average_variance_extracted`, and `htmt` for the measurement properties as standalone functions; `measurement_invariance` for the across-group analog of these within-factor constraints; `lavaan`, `lavTestLRT`.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
data(holzinger_swineford)
hs_factors <- list(
  verbal    = c("t6_paragraph_comprehension", "t7_sentence",
                "t9_word_meaning"),
  deduction = c("t20_deduction", "t22_problem_reasoning",
                "t23_series_completion"))

# Congeneric measurement model for both factors (the default:
# nothing is constrained, and the header names the structure).
fa_confirmatory(holzinger_swineford, hs_factors)

# Equal loadings: equal_loading = TRUE equates the loadings within
# every factor, and a descriptor can also differ by factor, as here,
# where the verbal loadings are equated and the deduction loadings are
# left free. Because only the covariance structure identifies this
# constraint, the implied structure of the verbal factor is
# essentially tau-equivalent, and the header names each factor's
# structure separately.
fa_confirmatory(holzinger_swineford, hs_factors,
      equal_loading = c(verbal = TRUE, deduction = FALSE))
```

```
# Measurement properties: omega, ave, and H per factor (each with a
# delta method standard error and confidence interval), the latent
# correlations, and the htmt ratios.
fa_confirmatory(holzinger_swineford, hs_factors, output = "measurement")

# Equal loadings and intercepts make the items tau-equivalent, and
# equal error variances as well make them parallel, the structure
# the header names here. The mean structure is added because
# equal_intercept asks about it, so the nu terms join the table.
fa_confirmatory(holzinger_swineford, hs_factors, equal_loading = TRUE,
      equal_intercept = TRUE, equal_error = TRUE)

# Ordered categorical items: two correlated factors of three
# four-category items each, generated here. The model is fit by WLSMV
# to polychoric correlations, and each ordered factor's omega is
# reported on the categorical sum score metric (categorical omega),
# which is why the interval columns of those two rows are NA.
set.seed(113)
n <- 200
eta_a <- rnorm(n)
eta_b <- 0.5 * eta_a + sqrt(1 - 0.5^2) * rnorm(n)
lambda <- c(0.6, 0.7, 0.8)
lat <- cbind(outer(eta_a, lambda), outer(eta_b, lambda)) +
  matrix(rnorm(n * 6), n, 6) %*% diag(sqrt(1 - c(lambda, lambda)^2))
likert <- as.data.frame(apply(lat, 2, function(x)
  as.integer(cut(x, breaks = c(-Inf, -1, 0, 1, Inf)))))
names(likert) <- paste0("item_", 1:6)
fa_confirmatory(likert,
      list(scale_a = paste0("item_", 1:3),
           scale_b = paste0("item_", 4:6)),
      ordered = TRUE, output = "measurement")

# Two-stage maximum likelihood with incomplete data, here 60 scores on
# one verbal test deleted at random. The chi_square row is the residual
# based statistic of Savalei and Bentler (2009), the cfi and rmsea rows
# are the normal theory analog of the two-stage indices of Zhang and
# Savalei (2023), and the lines under the table say which statistic,
# indices, and criteria are reported.
hs_incomplete <- holzinger_swineford
set.seed(113)
hs_incomplete$t7_sentence[sample(301, 60)] <- NA
fa_confirmatory(hs_incomplete, hs_factors, missing = "two_stage")

# Does the equal-loadings description hold? Two fits that differ only
# in a descriptor are nested, so output = "fit" hands them straight to
# lavaan's likelihood ratio test.
fit_free  <- fa_confirmatory(holzinger_swineford, hs_factors, output = "fit")
fit_equal <- fa_confirmatory(holzinger_swineford, hs_factors,
                   equal_loading = TRUE, output = "fit")
lavaan::lavTestLRT(fit_free, fit_equal)
```

`fa_exploratory` *Exploratory Common-Factor Analysis With Confidence Intervals*

Description

Fits a common-factor model, with rotation and uncertainty for its rotated loadings. This is an exploratory measurement model: every item can load on every factor. The estimates and rotation-aware standard errors are computed by **lavaan**, which is required; DMAR adds the table and its intervals, the display, the refusals, and the fit measure policy. The function does not compute principal components. It carries a name of its own so that attaching **lavaan**, whose `efa()` it calls, masks nothing. `efa_k()` is the same function under a shorter name, as `cfa_k()` is for `fa_confirmatory`.

Usage

```
fa_exploratory(
  data = NULL,
  nfactors = 1L,
  S = NULL,
  N = NULL,
  items = NULL,
  estimator = "ML",
  missing = "fiml",
  ordered = NULL,
  rotation = "oblimin",
  target = NULL,
  conf_level = 0.95,
  rmsea_conf_level = 0.9,
  seed = NULL,
  point_estimates_only = FALSE
)

efa_k(
  data = NULL,
  nfactors = 1L,
  S = NULL,
  N = NULL,
  items = NULL,
  estimator = "ML",
  missing = "fiml",
  ordered = NULL,
  rotation = "oblimin",
  target = NULL,
  conf_level = 0.95,
  rmsea_conf_level = 0.9,
  seed = NULL,
  point_estimates_only = FALSE
)
```

```
## S3 method for class 'dmar_fa_exploratory'
print(x, digits = getOption("dmar.digits", 3L), ...)
```

Arguments

data	Numeric raw observations. Supply exactly one of data or S.
nfactors	Number of common factors, a positive integer giving an identified exploratory model. With a target, its number of factors when nfactors is not supplied; a supplied value must agree with it.
S	Named positive definite covariance matrix, with N. A correlation matrix is a covariance matrix with unit variances, and the original metric then coincides with the standardized one.
N	Sample size for covariance input.
items	The items to analyze, by name: columns of data, or rows and columns of S, which is then reduced to them, in the order given; only the reduced matrix need be finite and positive definite. NULL, the default, analyzes every column of data or of S, or, with a target, the items the target names. Names that cannot be read as at least three distinct items are refused with an error of class dmar_items_invalid (see dmar_metadata).
estimator	"ML" (default) or "MLR" for continuous items; "ULS" for the minimum residual (minres) solution of Harman and Jones (1966), unweighted least squares fitted to the correlation matrix (see Details); explicitly request "WLSMV" for ordered items. MLR uses the same likelihood estimates with Huber-White sandwich errors.
missing	"fiml" (default), "ml", or "listwise" for continuous items under ML or MLR. With ordered items the default fits the items as they stand when no item value is missing, as fa_confirmatory does; with missing values ordered items require an explicit "pairwise" or "listwise" choice, and FIML is not substituted with deletion. ULS has no likelihood, so it takes complete raw data, an explicit "listwise" choice, or a correlation matrix estimated elsewhere through S (for example the maximum likelihood estimate of cov_fiml); with incomplete data the default "fiml" is refused rather than replaced. Covariance input has no missing-data route.
ordered	NULL, TRUE for all items, or their names.
rotation	The rotation criterion, by name. Oblique, the factors free to correlate: "oblimin" (default; direct oblimin with gamma 0, Jennrich & Sampson, 1966), "quartimin" (Carroll, 1953), "geomin" (Yates, 1987; epsilon 0.001), "simplimax" (Kiers, 1994), "oblimax" (Saunders, 1961), "infomax" (McKeon, 1968, as Browne, 2001, presents it), and "bentler" (Bentler, 1977). Orthogonal, the factors kept uncorrelated: "varimax" (Kaiser, 1958; with Kaiser row weights, as lavaan applies it) and "quartimax" (Neuhaus & Wrigley, 1954). "target" and "target_orthogonal" rotate to the partially specified target given in target, the factors free to correlate (Browne, 1972b) or kept uncorrelated (Browne, 1972a); see Details. "none" returns the unrotated solution lavaan identifies, whose factors are uncorrelated. Each criterion is applied in the form named here by lavaan's gradient projection algorithm from 30 random starts; Browne (2001) reviews the family.

`target` For `rotation = "target"` or `"target_orthogonal"`, the partially specified target, in either of two forms (`NULL`, the default, for every other rotation). The usual one is a named list of item keys, the form `fa_confirmatory` takes for `factors`: each element names a factor and the items expected to load on it. An item keyed to a factor is targeted at 0 on every factor it is not keyed to and left unspecified on the factors it is keyed to (an item keyed to two factors is unspecified on both), and an analyzed item keyed to no factor is unspecified on every factor. The other is a numeric matrix of items by factors, with the items as row names in any order, `NA` for each unspecified loading, and elsewhere a target value for the standardized pattern loading, strictly between -1 and 1 (see Details); column names, if given, name the factors.

`conf_level` Confidence level of the intervals for the parameters, the communalities, and the determinacy and *H*, default 0.95.

`rmsea_conf_level` Confidence level of the RMSEA interval on the fit line and in the `fit_measures` attribute (with the gamma hat and adjusted gamma hat intervals there, which come from the same noncentrality limits), default 0.90, the conventional level, whose limits decide the tests of close and not close fit at .05 (see `sem_fit_measures`); `rmsea_conf_level = 0.95` gives a 95 percent interval. It is separate from `conf_level`, which does not reach the RMSEA.

`seed` Optional seed for rotation starts; default `NULL` uses the current random stream. A supplied seed restores the caller's stream.

`point_estimates_only` `FALSE` (default) for the full analysis; `TRUE` for the same estimates without their standard errors, tests, and intervals, which are then not computed (see Details), for a program that needs only the estimates and wants them sooner.

`x` For `print()`, a table returned by `fa_exploratory()`.

`digits` For `print()`, the decimal places of the display (default `getOption("dmar.digits", 3)`; see `dmar_tbl`).

`...` For `print()`, passed to `print.data.frame()`.

Details

The model is fitted to the covariance matrix of the items, with their means under FIML, and the standardized solution is derived from it (Cudeck, 1989, on why a correlation matrix is not analyzed as if it were a covariance matrix); the table carries both metrics. Two routes are correlation structures by construction: the minimum residual solution, which is defined on the correlation matrix and is not invariant to the scale of the items, and the WLSMV analysis of ordered items, which fits their polychoric correlations and thresholds.

FIML fits the raw incomplete observations directly, under a normal working likelihood and missingness at random given the included items. It does not treat an estimated unrestricted covariance matrix as if it were an ordinary complete-data sample covariance. Factor signs and order are arbitrary. Wald intervals describe the selected rotated solution, conditional on the requested factor count; they do not include uncertainty from choosing that count or rotation.

Display. The printed result, and `knit_print()` and `as_kable` of it, show the solution rather than the parameter list: one row per item with its standardized pattern loading on each factor and that

loading's interval, its communality with its interval, and its complexity; above the table the sample, estimator, missing data method, rotation and its form, and standard error method; beneath it the variance each factor accounts for, the factor correlations with their intervals (or the statement that the rotation fixed them at zero), each factor's determinacy and coefficient *H* with their intervals, and the fit. The table itself keeps every parameter in both metrics, which `tidy()` returns, and a subset of it prints as an ordinary DMAR table.

Hofmann's (1978) complexity of an item is the square of the sum of its squared pattern loadings divided by the sum of its loadings to the fourth power: 1 for an item that loads on one factor alone, and the number of factors for an item that loads equally on all of them. The variance a factor accounts for is its diagonal element of Phi Lambda' Lambda, with Lambda the standardized pattern loadings and Phi the factor correlation matrix (the column sum of squared loadings when the factors are uncorrelated); these sum to the sum of the communalities, and each is also given as a proportion of the total variance of the items (their number, in the standardized metric) and of the common variance. No cumulative column is shown, since the order of the factors is arbitrary. The `variance_accounted` attribute holds this table. The fit line reads the `fit_measures` attribute: the chi square statistic with its degrees of freedom and p-value, the RMSEA with its interval at `rmsea_conf_level` (90 percent by default), which the line names, the CFI, TLI, and SRMR, and the version lines that say which indices they are (see `sem_fit_measures`); under ULS it gives the discrepancy statistic, the NFI, and the SRMR (below). The measures are computed from the unrotated solution of the same model: the chi square statistic and the standard indices do not depend on the rotation, but lavaan's FIML-corrected indices of a rotated object do, since its rotated parameterization counts more free parameters than the model has, so the fit of a rotated solution is taken from the unrotated fit, and every rotation reports the same fit.

Rotation to a partially specified target. A partially specified target states what is expected of some loadings and nothing of the rest: each loading is either given a target value or left unspecified. The rotation minimizes the sum of squared differences between the specified loadings and their target values, in the standardized metric, and asks nothing of the unspecified ones (Browne, 2001, p. 124). Zeros are the usual specification, since which loadings should be near zero is easier to say than how large the others should be (Browne, 2001, p. 124); the list form of `target` specifies exactly those, a zero for each item on each factor it is not keyed to. Unlike a confirmatory model, which forces the specified loadings to their values, the rotation brings them only as close to their targets as the data allow, so a specified zero whose rotated loading is far from zero signals a misspecified zero (Browne, 2001, pp. 124–125), and its interval in the table shows by how much. A specified value must lie strictly between -1 and 1, and a target with a value outside that range is refused: the target is stated in the standardized metric, and such a value is more likely a loading taken from an item's original metric than an expected standardized loading. No row weights are applied (Browne, 2001, p. 125, finds the Kaiser and Cureton-Mulaik row weights unsuited to a partially specified target), and the rotation is iterated to a gradient tolerance of 1e-8 rather than lavaan's 1e-5, so that zeros the rotation attains exactly (below) are attained to about that precision. The factors keep lavaan's names (f1, f2, ...) in the order of the target's columns, and the display prints the target beneath the table with the names it gave. Rotation does not change the model, so the fit is that of every other rotation.

When every specified value is zero, the sign of each factor is arbitrary, and each factor is reflected, as lavaan does by default, so that its loadings have a positive sum. Since lavaan reflects all the factors or none, no factor is reflected when any specified value is nonzero. A factor whose column holds a nonzero value then takes its sign from the target (a negative target keeps its sign), but a factor whose specified values are all zero takes the sign of the random start from which the rotation reached its minimum. A supplied `seed` fixes that sign; with `seed = NULL` the loadings of such a factor, and its

correlations with the other factors, can change sign from one run to the next. Either sign gives the same solution, so such a factor may be read with its loadings and its correlations reversed together.

The target must identify the rotation, and lavaan does not check that it does: an unidentified rotation comes back with finite standard errors and no warning. DMAR checks the conditions of Algina (1980) before reporting. In oblique rotation each factor's column needs at least m - 1 specified zeros, for m factors, and the standardized unrotated loadings of the items specified in it must have rank m - 1 (Browne, 2001, p. 115; Algina, 1980, Theorem 2). A column with a nonzero target value needs one entry more, m entries with rank m (Theorem 1): with the factor variances at one, m - 1 restrictions that include a nonzero value need not identify the rotation, as Algina (1980, pp. 393–394) shows by example, so DMAR asks for the larger count. In orthogonal rotation fewer suffice: taken in increasing order of their specified entries, the columns need at least 0, 1, ..., m - 1 zeros, and again a column with a nonzero value one entry more (Theorem 3), and the rank of a column's rows is checked against the same numbers. Two columns with the same specified entries and values would leave their factors indistinguishable. A target that fails is refused with an error of class `dmar_target_unidentified`, and one that cannot be read as a target with class `dmar_target_invalid`; both inherit `dmar_refusal` and carry a `reason` element (see `dmar_metadata`). Only a set of rows that is rank deficient to within the precision of the fitted loadings is refused (a singular value below 1e-6 times the largest); one that is nearly deficient shows itself in wide intervals.

When a column carries only the minimum number of zeros, the rotation can attain them exactly. Those loadings are then fixed by the target rather than estimated (lavaan reports a standard error of zero and a statistic that means nothing). The table shows each at its target value itself, in place of the rotated estimate, which differs from it only by the rotation's convergence error, with `se`, `statistic`, and `p_value` `NA` and both limits at the target, in both metrics. When every specified loading is attained this way, as in an oblique rotation with m - 1 zeros in every column, the rotated solution is the confirmatory factor model with those loadings fixed at their targets and the factor variances at one (the factors uncorrelated in an orthogonal rotation).

Myers, Jin, Ahn, Celimli, and Zopluoglu (2015) studied target rotation by simulation. Three factors were measured by 10 or 20 items (p. 497). The communalities were low (.20 to .40), high (.60 to .80), or wide (.20 to .80), and minor factors added error to the model (both p. 498). The target values were in error by up to two standard errors (p. 500). The errors in the target values had a negligible effect on the accuracy and the stability of the rotated loadings (pp. 502–503). Specifying more targets per factor improved both somewhat, the accuracy most when the communalities were low and the stability most when they were high (p. 502).

In their comparison with geomin, every target value was in error by two standard errors (pp. 500–501). Target rotation recovered the population loadings more accurately in all 27 comparisons, one for each number of targets at each level of communality, though by small amounts (p. 502). Their measure of accuracy was the absolute value of the bias of a replication, the signed error averaged over all the loadings (p. 501); its mean was at or below 0.10 for both rotations, and the largest difference between them was 0.08 (p. 502). Geomin gave the more stable solution in 20 of the 27, among them every comparison with low communalities; target rotation was the more stable with high communalities and six or more targets per factor, and with wide communalities and nine or more (pp. 499, 502–503). Their advice on the number of targets is to specify the target as fully as the theory allows, particularly when the items are not uniformly highly reliable indicators of the factors (p. 503). Comparing the rotations, they conclude that the extra work of specifying a target may be worthwhile for accuracy; for stability, they expect geomin to do better when the items are less than uniformly reliable indicators, and target rotation to do better when the items are uniformly

reliable and several targets are specified (p. 504). Less stability is more sampling variability, which the intervals of a given analysis show.

The intervals condition on a target fixed before the data were seen. Browne (2001, pp. 125, 148) describes revising a target in sequence, leaving unspecified the entries that proved misspecified and specifying as zeros the loadings that proved near zero; such a sequence is exploratory, capitalizes on chance, and calls for a confirmatory analysis of new data. The `method` attribute holds the target (`target`, items by factors, `NA` where unspecified, with the target's factor names) and the number of loadings it specifies (`n_specified`).

Determinacy and H. The determinacy of a factor is the correlation of its regression factor score with the factor, the multiple correlation of the factor with the items (Grice, 2001; the factor determinacy index of Ferrando & Lorenzo-Seva, 2018): $\sqrt{[\Phi\Lambda' R^{-1}\Lambda\Phi]_{kk}}$, with Λ the standardized pattern loadings, Φ the factor correlations, and R the item correlation matrix the model implies. Its square is the largest reliability any weighted sum of the items reaches as a measure of the factor, which Ferrando and Lorenzo-Seva call the generalized *H* and `bifactor_indices` the maximal reliability. Coefficient *H* is Hancock and Mueller's (2001) closed form on the factor's standardized loadings l_j alone, $H = (1 + (\sum_j l_j^2/(1 - l_j^2))^{-1})^{-1}$, applied to one factor of a multidimensional solution as Rodriguez, Reise, and Haviland (2016) apply it, so that each item's variance from the other factors counts as error. With one factor *H* is the squared determinacy; otherwise the two can differ in either direction, and the squared determinacy is the quantity to read beside *H*. Both are reported for every factor with a delta method standard error over lavaan's covariance matrix of the standardized estimates (the matrix behind the standardized loading and communality intervals) and Wald limits formed on the logit scale and transformed back (Browne, 1982; the confidence interval transformation principle of Kelley, 2007: the limits of a monotone transformation of a parameter are the transformation of its limits), as `reliability_H` forms them by default, so they lie within (0, 1); the `se` column is the standard error of the index itself, and with one factor it is the standard error `fa_confirmatory` gives its *H* row for the same model fitted the same way (by FIML). For ordered items both describe the latent responses behind the categories, not composites of the observed scores (the G-H-latent of Ferrando and Lorenzo-Seva), and the display says so. Beside each determinacy ρ the display and the `factor_quality` attribute give $2\rho^2 - 1$ (`min_score_correlation`, with its delta method standard error and the determinacy's limits mapped through it), the lowest correlation two sets of factor scores equally consistent with the model can have (Guttman, 1955, as Grice, 2001, p. 435, presents it): zero when $\rho^2 = .5$, negative below. No line is drawn. `factor_scores` gives the determinacy of the Bartlett score as well, and for each missing data pattern. The `factor_quality` attribute holds this table.

`estimator = "ULS"` is the minimum residual method: the loadings and unique variances minimize the sum of squared residuals of the correlation matrix, the unique variances absorbing the diagonal, so that the off-diagonal correlations are reproduced as closely as the common factors allow (Harman & Jones, 1966; `psych::fa(fm = "minres")` gives the same loadings and communalities, up to the order and signs of the factors). The estimates need no distributional assumption, and the correlation matrix is the structure fitted (lavaan's `correlation = TRUE`), so rescaling an item does not change them. With raw data the standard errors are lavaan's sandwich for a least squares estimator with the normal theory covariance matrix of the correlations (its `se = "robust.sem.nt"`; Browne, 1984), and the rotation enters them as for ML. A supplied correlation or covariance matrix carries no such covariance matrix, so with `S` the ULS estimates and communalities are returned with their standard errors and limits `NA`, and the note under the table says so. The ULS statistic is in the units of the squared correlations and has no chi square reference distribution, so in the `fit_measures` attribute the `chi_square` row holds N - 1 times the ULS discrepancy with no p-value, the RMSEA rows and the CFI, TLI, RNI, and IFI are `NA`, and the NFI and the SRMR, which do not rest on that

distribution, are reported (see sem_fit_measures); a maximum likelihood fit of the same items gives the fit indices and their inference.

The table contains original-unit parameters, standardized loadings and factor correlations, and communalities with intervals. A communality is one minus the standardized residual variance, so its interval reverses the limits of that variance. Wald limits are not clipped to [0, 1]. Improper or unidentified solutions are refused, with an error of class dmar_efa_no_solution whose reason element says whether the model did not converge, converged to an improper solution (lavaan's post-estimation check failed: a residual variance below zero, or a covariance matrix that is not positive definite), or left the rotation-aware standard errors unavailable; factor_count reads it to report such a count as a row. When a residual variance is below zero and the likelihood has no maximum at all (the likelihood flat or still rising as the variance falls, as when too many factors are extracted), the refusal says so instead (reason "no maximum", and the class dmar_no_maximum as well; see fa_bifactor for the check) and prints the lavaan::efa() code of the unrotated fit, which selects items from the data or covariance matrix given, where bounds = "pos.var", **lavaan**'s own default, or fewer factors are the choices. When the unrestricted FIML likelihood diagnostic fails but the constrained model and its H0-based errors remain eligible, parameter results are qualified and H1-dependent fit indices are withheld. Absence of a diagnosed obstacle does not certify a unique global maximum.

Point estimates only. With point_estimates_only = TRUE the table has the rows and the estimates of the full analysis (the loadings, factor correlations, and residual variances in both metrics, the communalities, Hofmann's complexity, and each factor's determinacy and H), and so do the variance_accounted, factor_quality, and fit_measures attributes, but no standard error, test of a parameter, or interval is computed: the se, statistic, p_value, ci_lower, and ci_upper columns are NA, as are the standard errors and limits of factor_quality. lavaan is asked for no standard errors (its se = "none"), so neither the information matrix behind them (with its sandwich under MLR, WLSMV, and ULS) nor the rotation-aware errors of the rotated loadings are formed, nor the delta method errors of the standardized solution, the communalities, the determinacy, and H; and the rotated fit is made without a test statistic, which DMAR never reads, since the fit measures come from the unrotated fit. That fit keeps its test, so every fit measure is the one the full analysis reports, the robust statistic and the FIML-corrected indices under MLR and the scaled and shifted statistic under WLSMV included. conf_level then sets nothing, since the only interval left is the RMSEA interval of the fit line, at rmsea_conf_level, and the display states no other level. A loading of a target rotation that is attained exactly is recognized by lying within 1e-6 of its target, in place of its standard error of zero, and shown at the target as in the full analysis; in rare cases (a chance of the order of 1e-5 for each specified loading) an estimated loading falls that close to its target by chance and is shown at it, which the full analysis would estimate, within 1e-6 of the same value. The input and target checks, and the refusals of a solution that did not converge or is improper and of a likelihood without a maximum (whose code reproduces the full fit), are made as before. The refusal of a solution whose rotation-aware standard errors are unavailable is the full analysis's guard against a solution that is not identified at its estimates, whose estimates are then not unique; it is kept by inverting the expected information of the unrotated fit, which is cheap, and, only when that fails, the information its standard errors would use (the observed information under FIML); a solution whose information cannot be inverted is refused with the same class and reason. A failure confined to the rotation's part of the rotation-aware errors is not detected, so in that rare case the estimates are returned for a solution the full analysis refuses. A FIML fit whose standard errors would need the unavailable unrestricted (H1) information is not refused, since none are computed; its fit measures are withheld as in the full analysis, with a warning of the same class whose note speaks of the estimates alone. The display says that standard errors and intervals were

not computed, the result carries the setting as its `point_estimates_only` attribute, and the lavaan object in `fit` has no standard errors (and, when rotated, no test). The time saved depends on the estimator and the data. In one set of timings with three factors under oblimin, the nine classic Holzinger and Swineford tests took about a third less time under ML and MLR (0.79 s against 0.51 s under ML) and half or more less under ULS and WLSMV (0.70 s against 0.27 s for the tests cut into five ordered categories), and the rotated solution of 18 items with 1,000 cases and 410 patterns of missing values took 29 s against 17 s under MLR with FIML, and 18 s against 9 s under ML. What remains is the estimation itself and the fit measures, chiefly the robust statistic and the FIML-corrected indices of the unrotated fit, which are kept; an unrotated analysis therefore saves little or nothing (16.5 s against 16.7 s on those 18 items), and a program that also fits the unrotated model (for its own fit measures, say) saves on the rotated fit alone: the two together took 45 s against 34 s.

Value

A `dmar_tbl` data frame (classes `dmar_fa_exploratory` and `dmar_factor_analysis`) with `term`, the parameter labels `lhs`, `op`, and `rhs`, `estimate`, `se`, `statistic`, `p_value`, `ci_lower`, `ci_upper`, `conf_level`, and `metric`: the parameters in the original metric, the standardized solution, the communalities (op `"communality"`), Hofmann's complexity (op `"complexity"`, with no interval), and each factor's determinacy and coefficient *H* (ops `"determinacy"` and `"H"`, the factor as both `lhs` and `rhs`). Attributes retain `fit` (the lavaan object), `method` (sample size, estimator, standard error and missing data methods, the rotation and whether it is orthogonal, the fitted structure, and for a target rotation the target and the number of loadings it specifies), `variance_accounted`, `factor_quality`, eligible `fit_measures` (a `sem_fit_measures` table, whose RMSEA interval is at `rmsea_conf_level`), `rmsea_conf_level`, and `point_estimates_only` (`TRUE` or `FALSE`, as called). `tidy()` preserves the parameter rows; `glance()` reports the analysis method and sample size, and both carry `point_estimates_only` as a column; `print()` shows the display described in Details.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J. (1980). A note on identification in the oblique and orthogonal factor analysis models. *Psychometrika, 45*(3), 393–396. doi:10.1007/BF02293911

Bentler, P. M. (1977). Factor simplicity index and transformations. *Psychometrika, 42*(2), 277–295. doi:10.1007/BF02294054

Browne, M. W. (1972a). Orthogonal rotation to a partially specified target. *British Journal of Mathematical and Statistical Psychology, 25*(1), 115–120. doi:10.1111/j.20448317.1972.tb00482.x

Browne, M. W. (1972b). Oblique rotation to a partially specified target. *British Journal of Mathematical and Statistical Psychology, 25*(2), 207–212. doi:10.1111/j.20448317.1972.tb00492.x

Browne, M. W. (1982). Covariance structures. In D. M. Hawkins (Ed.), *Topics in applied multivariate analysis* (pp. 72–141). Cambridge, UK: Cambridge University Press.

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*(1), 62–83. doi:10.1111/j.20448317.1984.tb00789.x

Browne, M. W. (2001). An overview of analytic rotation in exploratory factor analysis. *Multivariate Behavioral Research, 36*(1), 111–150. doi:10.1207/S15327906MBR3601_05

Carroll, J. B. (1953). An analytical solution for approximating simple structure in factor analysis. *Psychometrika, 18*(1), 23–38. doi:10.1007/BF02289025

Cudeck, R. (1989). Analysis of correlation matrices using covariance structure models. *Psychological Bulletin, 105*(2), 317–327. doi:10.1037/00332909.105.2.317

Ferrando, P. J., & Lorenzo-Seva, U. (2018). Assessing the quality and appropriateness of factor solutions and factor score estimates in exploratory item factor analysis. *Educational and Psychological Measurement, 78*(5), 762–780. doi:10.1177/0013164417719308

Grice, J. W. (2001). Computing and evaluating factor scores. *Psychological Methods, 6*(4), 430–450. doi:10.1037/1082989X.6.4.430

Guttman, L. (1955). The determinacy of factor score matrices with implications for five other basic problems of common-factor theory. *British Journal of Statistical Psychology, 8*(2), 65–81. doi:10.1111/j.20448317.1955.tb00321.x

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), *Structural equation modeling: Present and future* (pp. 195–216). Scientific Software International.

Harman, H. H., & Jones, W. H. (1966). Factor analysis by minimizing residuals (minres). *Psychometrika, 31*(3), 351–368. doi:10.1007/BF02289468

Hofmann, R. J. (1978). Complexity and simplicity as objective indices descriptive of factor solutions. *Multivariate Behavioral Research, 13*(2), 247–250. doi:10.1207/s15327906mbr1302_9

Jennrich, R. I. (1973). Standard errors for obliquely rotated factor loadings. *Psychometrika, 38*(4), 593–604. doi:10.1007/BF02291497

Jennrich, R. I., & Sampson, P. F. (1966). Rotation for simple loadings. *Psychometrika, 31*(3), 313–323. doi:10.1007/BF02289465

Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika, 23*(3), 187–200. doi:10.1007/BF02289233

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kiers, H. A. L. (1994). Simplimax: Oblique rotation to an optimal target with simple structure. *Psychometrika, 59*(4), 567–579. doi:10.1007/BF02294392

McKeon, J. J. (1968). *Rotation for maximum association between factors and tests*. Unpublished manuscript, Biometric Laboratory, George Washington University.

Myers, N. D., Jin, Y., Ahn, S., Celimli, S., & Zopluoglu, C. (2015). Rotation to a partially specified target matrix in exploratory factor analysis in practice. *Behavior Research Methods, 47*(2), 494–505. doi:10.3758/s1342801404867

Neuhaus, J. O., & Wrigley, C. (1954). The quartimax method: An analytic approach to orthogonal simple structure. *British Journal of Statistical Psychology, 7*(2), 81–91. doi:10.1111/j.2044-8317.1954.tb00147.x

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016). Evaluating bifactor models: Calculating and interpreting statistical indices. *Psychological Methods, 21*(2), 137–150. doi:10.1037/met0000045

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

Saunders, D. R. (1961). The rationale for an "oblimax" method of transformation in factor analysis. *Psychometrika, 26*(3), 317–324. doi:10.1007/BF02289800

Yates, A. (1987). *Multivariate exploratory data analysis: A perspective on exploratory factor analysis*. State University of New York Press.

See Also

`parallel_analysis` for the factor count, `fa_confirmatory` for the confirmatory model, `factor_scores`, which takes the lavaan object in the `fit` attribute, and `reliability_H` for coefficient *H* of one scale with bootstrap intervals.

Examples

```
items <- c("t1_visual_perception", "t2_cubes", "t4_lozenges",
           "t6_paragraph_comprehension", "t7_sentence", "t9_word_meaning")
two <- fa_exploratory(holzinger_swineford, nfactors = 2, items = items,
                      seed = 113)
two                       # the solution: loadings with intervals,
                          # communalities, complexity, variance, fit
attr(two, "variance_accounted")
attr(two, "factor_quality")  # determinacy and H with intervals
generics::tidy(two)[1:6, ]   # every parameter, both metrics

# An orthogonal rotation, its estimates alone: the standard errors,
# tests, and intervals are not computed, which saves time.
fa_exploratory(holzinger_swineford, nfactors = 2, items = items,
               rotation = "varimax", seed = 113, point_estimates_only = TRUE)

# Rotation to a partially specified target, through efa_k(), the same
# function under a shorter name, here of the minimum residual (minres)
# solution: unweighted least squares on the correlation matrix, with
# sandwich standard errors from the raw data. Each item is targeted at
# zero on the factor it is not keyed to, and the items and the number of
# factors are read from the target.
efa_k(holzinger_swineford, rotation = "target", estimator = "ULS",
      seed = 113, target = list(spatial = items[1:3], verbal = items[4:6]))
```

---

fa_second_order *Second-Order Confirmatory Factor Analysis Model*

Description

Fits the second-order factor model, in which first-order factors account for what the items of each subscale share and one or more second-order factors account for what the first-order factors share. The model is described as in `fa_confirmatory`, by naming the items of each first-order factor, together with the first-order factors of each second-order factor, and is fit by maximum likelihood to raw data (by full information maximum likelihood, the default, or after listwise deletion) or to a covariance matrix. The table reports the first-order and second-order loadings, the error variances, and the intercepts with their confidence intervals; the indices that describe how much of the total

score and of each subscale score the second-order factor accounts for: omega and omega hierarchical (omega_H) for the total score, omega and omega hierarchical subscale (omega_HS) for each subscale score, the explained common variance (ECV), coefficient *H*, and the maximal reliability, each with a delta method standard error and a confidence interval, and the percentage of uncontaminated correlations (PUC), which depends on the structure alone and so has neither; and the fit of the model. The Schmid-Leiman solution the estimates imply (Schmid & Leiman, 1957), the loadings of every item on the second-order factor and on what its first-order factor measures beyond it, is returned as an attribute.

Usage

```
fa_second_order(
  data = NULL,
  factors,
  second_order,
  S = NULL,
  N = NULL,
  M = NULL,
  equal_loading = FALSE,
  equal_intercept = FALSE,
  equal_error = FALSE,
  equal_second_order_loading = FALSE,
  meanstructure = NULL,
  estimator = "ML",
  missing = "fiml",
  se = NULL,
  conf_level = 0.95,
  rmsea_conf_level = 0.9,
  output = c("verbose", "measurement", "summary", "standardized", "fit"),
  ...,
  wald_df = Inf
)

## S3 method for class 'dmar_fa_second_order'
tidy(x, ...)

## S3 method for class 'dmar_fa_second_order'
glance(x, ...)
```

Arguments

`data`	A raw data matrix or data frame, rows are respondents and columns include the items named in `factors`. Supply exactly one of `data` or `S`.
`factors`	Named list of the first-order factors. Each element names a first-order factor and gives the character vector of its items, two or more; each item is listed under one first-order factor. An item that is not a column of `data` or an item of `S` is refused with an error of class `dmar_items_invalid` (see `dmar_metadata`).
`second_order`	Named list of the second-order factors. Each element names a second-order factor and gives the character vector of its first-order factors, named as `factors`

names them; each first-order factor is listed under one second-order factor. One second-order factor needs three or more first-order factors, or two with `equal_second_order_loading`; with two or more second-order factors, which are always correlated (uncorrelated second-order factors are not offered), two first-order factors each are enough. See *Details* for the structures that are refused.

`S` A symmetric covariance matrix of the items, with dimnames naming the items; `N` is then required. Supply exactly one of `data` or `S`.

`N` Total sample size. Required with `S`; ignored (inferred from the rows) with `data`.

`M` Optional named numeric vector of item means, used with a covariance matrix to model the mean structure (required there when `equal_intercept` is used). Ignored for raw data.

`equal_loading` Logical, or a named logical vector with one element per first-order factor. `TRUE` constrains the loadings of a first-order factor's items to a single value. Defaults to `FALSE`.

`equal_intercept` Logical, or a named logical vector with one element per first-order factor. `TRUE` constrains the intercepts of a first-order factor's items to a single value; this requires the mean structure. Defaults to `FALSE`.

`equal_error` Logical, or a named logical vector with one element per first-order factor. `TRUE` constrains the error variances of a first-order factor's items to a single value. Defaults to `FALSE`.

`equal_second_order_loading` Logical, or a named logical vector with one element per second-order factor. `TRUE` constrains the loadings of a second-order factor's first-order factors to a single value, so that they correlate equally with it. Defaults to `FALSE`.

`meanstructure` Logical or `NULL`. `NULL` (default) models the mean structure exactly when it is needed: under FIML (the default with raw data, so the item intercepts, the `nu` rows, join the table), when any `equal_intercept` is `TRUE`, or when `M` is supplied. `TRUE` models the intercepts regardless (raw data or `M` required); `FALSE` suppresses them, which FIML does not allow.

`estimator` `"ML"` (default), maximum likelihood, or `"MLR"`, maximum likelihood with standard errors and a test statistic robust to nonnormality (raw data required). Ordered items are not available.

`missing` The missing data treatment for raw data: `"fiml"` (default; full information maximum likelihood, which uses every case with an observed item; **lavaan**'s name `"ml"` is accepted) or `"listwise"`. The default is resolved as in `fa_confirmatory`: with `se = "robust.sem"` or a test statistic FIML cannot compute, the items are fit as they stand when no item value is missing, and otherwise the call stops and names the option to choose. Ignored with `S`.

`se` Standard error method passed to **lavaan**. The default, `NULL`, follows the estimator: `"standard"` for ML and `"robust.huber.white"` for MLR. `"robust.sem"` requires complete data and is refused with FIML, and `"none"` requests point estimates only.

`conf_level` Confidence level of the intervals for the parameters and the indices. Defaults to 0.95. It does not reach the RMSEA interval, whose level is `rmsea_conf_level`.

`rmsea_conf_level`
: Confidence level of the RMSEA interval of the fit rows, which the `rmsea_ci_level` row records. Defaults to 0.90, the conventional level, whose limits decide the tests of close and not close fit at .05 (see `sem_fit_measures`); `rmsea_conf_level = 0.95` gives a 95 percent interval. It sets the level in the table DMAR returns and in its `glance()`; `output = "summary"` returns the **lavaan** summary, which shows lavaan's 90 percent interval whatever this argument is.

`output`
: Format of the returned object:

  `"verbose"` (default) The first-order loadings, disturbance variances (fixed at 1), error variances, and intercepts, the second-order loadings, variances (fixed at 1), and correlations, then the indices, then the fit rows.

  `"measurement"` The indices alone.

  `"summary"` The `summary()` of the **lavaan** fit (not a data frame).

  `"standardized"` `lavaan::standardizedSolution()` of the fit, whose rows for the second-order loadings are the correlations of the first-order factors with their second-order factor; the indices are not among its rows (see *Details*). Under `se = "none"` its `se`, `z`, `pvalue`, `ci.lower`, and `ci.upper` columns are `NA`.

  `"fit"` The **lavaan** fit, which `lavaan::lavTestLRT()` takes to test the model against the bifactor model (see *Details*).

`...`
: For `fa_second_order()`, additional arguments forwarded to `lavaan` (for example `cluster`); a `group` argument is an error, as in `fa_confirmatory()`, and so is `ordered`. Unused by `tidy()` and `glance()`.

`wald_df`
: The reference distribution of the Wald intervals and *p*-values: `Inf` (the default), the normal distribution, or a single positive number, the degrees of freedom of a *t* distribution for every parameter and index row (for the indices, on the logit scale), as in `fa_confirmatory`. The keyword `"missing_information"` is refused: the calibration study behind it simulated one factor models, and nothing calibrates it for the loadings of a second-order model or for its indices. It follows `...`, so it must be named in full.

`x`
: A `dmar_fa_second_order` table returned by `fa_second_order()`.

Details

The model is also called the higher-order factor model, and sometimes the hierarchical factor model. The function is named for the structure it fits because Schmid and Leiman (1957) and Yung, Thissen, and McLeod (1999) use "hierarchical factor model" for the other model, the one with uncorrelated factors in layers, of which the bifactor model fit by `fa_bifactor` is the two-layer case.

The model. An item i of first-order factor f is

$$x_i = \nu_i + \lambda_i f + e_i,$$

and the first-order factor f of second-order factor g is

$$f = \gamma_f g + u_f,$$

where the disturbance u_f is the part of f that g does not account for. The model is identified as **lavaan**'s `std.lv = TRUE` identifies it: the variance of every second-order factor is fixed at 1 (and its

mean at 0 when the mean structure is modeled), and so is the variance of every disturbance, so every first-order loading λ_i and every second-order loading γ_f is free. Both constraints are linear. The variance of f is then $1 + \gamma_f^2$, and its correlation with g, the standardized second-order loading, is $\gamma_f / \sqrt{1 + \gamma_f^2}$. Two or more second-order factors are correlated with each other (the `phi` rows), and the function offers correlated second-order factors only: uncorrelated ones would hold every first-order factor under one second-order factor uncorrelated with every first-order factor under another, and would leave a second-order factor of two first-order factors unidentified (`lavaan::cfa()` with `std.lv = TRUE` and `orthogonal = TRUE` fits that model). A factor fits as well reflected: a first-order factor with every one of its loadings and its second-order loading reversed in sign, a second-order factor with its second-order loadings and its correlations reversed. Each first-order factor is reported in the direction in which its loadings sum to a positive value, and then each second-order factor in the direction in which its second-order loadings do, so a subscale whose items are scored in reverse shows positive loadings and a negative second-order loading. A first-order factor under a second-order factor with equal loadings is not reflected alone, which would break the equality, and keeps the direction the common loading gives it. Should a reflection fail, a warning says so and the factor is reported as the optimizer left it, which changes the signs of its loadings and nothing else. Every loading, error variance, and intercept carries a label, as in `fa_confirmatory`, and the indices are **lavaan** defined parameters of those labels, so the fit itself gives each one a delta method standard error. The standardized solution is derived from this fit of the covariances (or of the raw data), not from a fit to correlations.

Scope and identification. A structure outside the model, or one the data cannot identify, is refused before anything is fit, with an error of class `dmar_second_order_unidentified` (which inherits `dmar_refusal`; see `dmar_metadata`) that names the factors or items at issue and gives the kind of refusal as its `reason` element, as in `fa_bifactor`. Not identified (`"unidentified"`): a first-order factor of one item, whose loading cannot be told apart from its error variance; a second-order factor of one first-order factor, which cannot be told apart from it; and one second-order factor over two first-order factors with free loadings, whose one correlation is the product of the two standardized loadings (equal loadings identify them). Outside the model (`"scope"`): an item under two first-order factors, or a first-order factor under two second-order factors. Names (`"names"`): an item or a first-order factor listed twice under one factor, a first-order factor under no second-order factor, and names that coincide. The identification checks run once the other arguments and the data have passed theirs. With two or more second-order factors, a second-order factor of two first-order factors is identified through the correlations of its first-order factors with those of the others, and is empirically underidentified when the second-order factors barely correlate. With incomplete raw data under FIML, first-order factors that no case observes together (no case has an item of one and an item of the other) are refused too, as `"unidentified"`, when the pairs of first-order factors the data do observe together do not identify the second-order loadings: three first-order factors under one second-order factor with one pair never observed together leave two correlations for three loadings, while four with one such pair leave five for four and are fit. When **lavaan**'s information matrix cannot be inverted, or is singular, whether it is the observed or the expected information, every standard error, test, and interval is `NA`, the maximal reliability's included, with a warning, as in `fa_bifactor`.

A just identified structural part. The second-order loadings and correlations account for the correlations of the first-order factors, $k(k-1)/2$ of them for k first-order factors. Exactly three first-order factors under one second-order factor have three correlations and three second-order loadings, and two with equal loadings have one of each, so the second-order structure is just identified: it is another parameterization of the correlated factors model of the same first-order factors, which

`fa_confirmatory` fits. Whenever that model's three correlations have a positive product and imply standardized second-order loadings below one in absolute value (its one correlation is positive, for two factors), the two models fit exactly alike, with the same chi square on the same degrees of freedom, and the test of fit says nothing about the second-order structure; a line under the table says so, and the `structural_df` attribute, the correlations less the second-order parameters, is 0. On `holzinger_swineford`, nine tests in three groups of three (visual perception, cubes, and lozenges; paragraph comprehension, sentence completion, and word meaning; addition, counting groups of dots, and straight and curved capitals) give a chi square of 85.31 on 24 degrees of freedom under both models. Four or more first-order factors leave the second-order structure something to test.

The Schmid-Leiman solution. Substituting the second line of the model into the first gives

$$x_i = \nu_i + \lambda_i \gamma_f g + \lambda_i u_f + e_i,$$

a model in which every item loads on the second-order factor and on its first-order factor's disturbance, all of them uncorrelated with variance 1: the transformation of Schmid and Leiman (1957), which turns a second-order solution into one with uncorrelated factors in layers. In the standardized metric an item's loading on g is $\lambda_i^* \gamma_f^*$ and its loading on u_f is $\lambda_i^* \sqrt{1 - \gamma_f^{*2}}$, with λ_i^* its standardized first-order loading and γ_f^* the standardized second-order loading of its factor (Yung, Thissen, & McLeod, 1999, Eqs. 4 and 10). These are the `general_loading` and `group_loading` columns of the `schmid_leiman` attribute, beside `item`, `factor` (the first-order factor), `second_order`, and `uniqueness`, one minus the sum of their squares. Within a first-order factor the ratio of the two loadings is $\gamma_f^* / \sqrt{1 - \gamma_f^{*2}}$ for every item: the proportionality constraint by which the second-order model differs from a bifactor model with the same groups, whose general and group loadings are free (Yung, Thissen, & McLeod, 1999; Chen, West, & Sousa, 2006).

The indices. The indices are those of `fa_bifactor`, computed from the Schmid-Leiman loadings of the items of each second-order factor: in the unstandardized metric, the general loading $\lambda_i \gamma_f$ and the group loading λ_i, beside the error variance ψ_i, with model implied item variance $v_i = \lambda_i^2 (1 + \gamma_f^2) + \psi_i$. The variance of the total score, the unweighted sum of the items of a second-order factor g, is

$$\sigma_X^2 = (\textstyle\sum_i \lambda_i \gamma_{f(i)})^2 + \sum_f (\sum_{i \in f} \lambda_i)^2 + \sum_i \psi_i,$$

and the rows are

`omega_<g>` Omega for the total score, the share of σ_X^2 that is not error variance: its reliability with every factor counted as true score.

`omega_H_<g>` Omega hierarchical, $(\sum_i \lambda_i \gamma_{f(i)})^2 / \sigma_X^2$: the share of the total score's variance that is due to the second-order factor alone.

`omega_<f>` Omega for the subscale score of first-order factor f, the sum of its items, $(\sum_{i \in f} \lambda_i)^2 (1 + \gamma_f^2)$ over that score's variance, which adds $\sum_{i \in f} \psi_i$: the omega that `fa_confirmatory` reports for a factor, here at the estimates of the second-order model.

`omega_HS_<f>` Omega hierarchical subscale, $(\sum_{i \in f} \lambda_i)^2$ over the same variance: the share of the subscale score's variance that is due to the disturbance of f, what the subscale score says beyond the second-order factor.

`ECV_<g>`, `ECV_<f>` The explained common variance, formed from the standardized Schmid-Leiman loadings as `fa_bifactor` forms it: for g, its share of the common variance of its items; for f, the share of the common variance of the items of f that is due to g, which in this model is γ_f^{*2}, the squared standardized second-order loading, since every item of f divides its common variance in that proportion.

`H_<g>`, `H_<f>` Coefficient *H* in the form of Rodriguez, Reise, and Haviland (2016), the closed form of Hancock and Mueller (2001) over the factor's Schmid-Leiman loadings with every other source of an item's variance counted as its error.

`maximal_reliability_<g>`, `maximal_reliability_<f>` The maximal reliability under the fitted model, $c'\Sigma^{-1}c$, with c the covariances of all of the items with the factor and Σ their model implied covariance matrix: the largest reliability any weighted sum of the items reaches as a measure of g, or of the disturbance of f. With several second-order factors, the items of the others count too, through the correlations of the second-order factors.

`PUC_<g>` The percentage of uncontaminated correlations, as a proportion: the share of the pairs of the items of g that do not share a first-order factor. It depends on the structure, not on the data, so it has no standard error or interval.

With several second-order factors each one's rows describe the sum of its own items. Rodriguez, Reise, and Haviland (2016) describe how each index is read; the table reports the numbers with their uncertainty, and whether a total or subscale score is worth reporting is the researcher's judgment, on which the package draws no line. Omega, omega_H, and omega_HS describe unit-weighted sums of the items as they are scored, as in `fa_bifactor`; ECV, *H*, the maximal reliability, and PUC do not depend on the metric of the items. With `output = "standardized"`, **lavaan** would evaluate the defined indices at the standardized values of their labels, and a standardized first-order loading is scaled by the whole variance of its factor, not by that of the disturbance the indices need, so the index rows are left out of that output, and **lavaan**'s warning that under `equal_loading` or `equal_error` their standardized values are ambiguous is not passed on.

Intervals. As in `fa_bifactor`: the standard errors of omega, omega_H, omega_HS, ECV, and *H* are **lavaan**'s delta method standard errors of the defined parameters, that of the maximal reliability is the delta method's with the derivatives taken numerically, and the limits of every index are formed on the logit scale from that standard error and transformed back (Browne, 1982; the confidence interval transformation principle of Kelley, 2007), so they lie within (0, 1); the `se` column is the standard error of the index itself.

The test against the bifactor model. The second-order model is nested in the bifactor model whose group factors are its first-order factors: the bifactor model frees the proportionality constraint above (Yung, Thissen, & McLeod, 1999; Chen, West, & Sousa, 2006). With `output = "fit"` from both functions, `lavaan::lavTestLRT(fa_second_order(..., output = "fit"),fa_bifactor(..., output = "fit"))` gives the likelihood ratio test of the constraint, on as many degrees of freedom as there are items less first-order factors (Yung, Thissen, & McLeod, 1999), as the example shows. When the second-order structure is just identified the second-order fit is that of the correlated factors model, and the test compares the bifactor model with it. The comparison is for one second-order factor; the bifactor model of `fa_bifactor` has one general factor.

Finding the maximum. In the parameterization above a standardized second-order loading reaches one only as γ_f runs off to infinity, the loadings λ_i of its factor shrinking toward zero beside it, and **lavaan**'s optimizer can stop on the way at a point that is not the maximum of the likelihood, with other second-order loadings drifted far from their values there, or fail to converge. The function therefore fits the model a second time, in a parameterization that writes each first-order factor on the scale of its second-order factor, $f = g + u_f$, with the variance of u_f, which is $1/\gamma_f^2$, free and bounded below at zero, so that a standardized loading of one is a point the optimizer reaches; the equality constraints of the call hold exactly in it, as in the first. That fit starts from the first and, when the first did not converge, came near the boundary, or fell short, also from **lavaan**'s own starting values; the largest log likelihood is carried back to the parameterization above, so the estimates

and the fit rows are those of the maximum, and **lavaan** accepts each fit only where its gradient is near zero. Log likelihoods are compared per case (two within 1e-6 per case are taken to be the same maximum), since the log likelihood is a sum over the cases and the comparison should not depend on their number. On `holzinger_swineford`, one second-order factor over the spatial, verbal, and speed tests of the example and the three tests with missing values (`t24_woody_mccall`, `t25_paper_form_board_r`, and `t26_flags`) stopped at a chi square of 251.77 in the first parameterization, with three second-order loadings between 43 and 233; the maximum is 169.77. When the second fit cannot be made, a warning says that the maximum could not be confirmed and names any standardized second-order loading of .995 or more as possibly a point where the optimizer stopped; when neither fit converges, the error says so and names the standardized second-order loadings the first fit was taking toward one.

Improper solutions. As in `fa_bifactor`, a negative error variance makes the maximal reliabilities undefined, and with them the H of a factor on which an item's standardized Schmid-Leiman loading reaches one and an omega whose items' error variances sum below zero; those rows are `NA`, with a warning that names the items (beside the warning of class `dmar_heywood_warning` that `fa_confirmatory` gives), and the `improper` attribute holds the negative error variances. When the likelihood has no maximum at all, flat or still rising as an error variance falls, the function stops for every `output`, as `fa_bifactor` does, with an error of class `dmar_no_maximum` that prints the **lavaan** code of the fit (with a standardized second-order loading at one, that of the second fit described above, in which the check is made); there, `bounds = "pos.var"` or a different model are the choices. The second order has improper solutions of its own, handled in the same way. A standardized second-order loading of one, a Heywood case at the second order, leaves the first-order factor's disturbance without variance (a parameterization that let that variance be negative would estimate it at or below zero). The loading is at one when the second fit above puts the variance of u_f at its bound of zero (within 1e-6 of it, a standardized loading of .9999995 or more). The parameterization above has no point there, so the fit holds the loading within 5e-9 of one, where the log likelihood is within 1e-6 per case of its maximum, and that factor's second-order loading and its loadings on its disturbance have no finite estimate and are `NA`, as are the rows of its disturbance, omega_HS, ECV, H, and the maximal reliability of that first-order factor. The other rows are those of the maximum, their standard errors computed with the loading held at one (with `se = "bootstrap"` they are `NA`, with a warning, since the fit at that point is computed without the optimizer and a bootstrap of it would not refit the samples); the fit rows are those of the model at its maximum, on its own degrees of freedom; and the factor's items have group loadings of zero in the Schmid-Leiman attribute. With `output = "fit"`, `"summary"`, or `"standardized"` the object holds the loading at that point, and a warning says so; with `se = "bootstrap"` that object carries no standard errors, tests, or intervals (lavaan's `se = "none"`). Under FIML, lavaan cannot form its robust CFI, TLI, RMSEA, and GFI for a fit computed at that point, so the `"summary"` output shows them as `NA`, with one warning in place of lavaan's. A correlation of two second-order factors beyond one in absolute value, which leaves the covariance matrix of the latent variables short of positive definite, leaves every maximal reliability `NA`, since each reads that correlation. A warning names the factors, and the `improper_second_order` attribute holds the standardized loadings at one and the correlations beyond one, named by their terms. Two second-order factors over two or three first-order factors each are prone to both: of the ten ways of placing five first-order factors of `holzinger_swineford` (spatial, verbal, speed, memory, and reasoning tests), two under one second-order factor and three under the other, every one puts the standardized second-order loading of the reasoning factor at one, and seven of them also give a correlation of the second-order factors beyond one.

Ordered items. `fa_second_order()` fits continuous items, and `ordered` is refused.

Row names under `term`: `lambda_<factor>_<j>` (first-order loadings, numbered within the fac-

tor; lambda_<factor> when equated), zeta_<factor> (the disturbance variances, fixed at 1), psi_<factor>_<j> and nu_<factor>_<j> (the error variances and intercepts), gamma_<second>_<factor> (second-order loadings; gamma_<second> when equated), phi_<second> (the second-order variances, fixed at 1), phi_<second>_<second> (the correlations of the second-order factors), the indices above, and fa_confirmatory's fit rows chi_square through H1, whose versions and the lines under the table are as there.

Value

For output = "verbose" (default) and "measurement", a data.frame (classes dmar_fa_second_order, dmar_tbl) with columns syntax, term, estimate, se, test_statistic, p_value, ci_lower, and ci_upper. Its attributes record the model: factors and second_order; equal, a list naming the factors whose loadings, error variances, intercepts, and second-order loadings are equated; has_means; structural_df, the degrees of freedom of the second-order structure (0 when it is just identified); omega_metric ("raw_item_sum"); improper, the negative error variances named by item (empty for a proper solution); improper_second_order, the standardized second-order loadings at one and the second-order correlations beyond one, named by their terms (empty for a proper solution); schmid_leiman, the Schmid-Leiman solution described above; and wald_df. The verbose table also carries fa_confirmatory()'s test, estimator, missing, fit_indices, and fit_rows. The printed header describes the model, and the lines under the table say what the indices describe. For "summary", the **lavaan** summary; for "standardized", the standardized solution with the "wald_df" attribute; for "fit", the **lavaan** fit. tidy() gives the parameter and index rows and glance() one row of the fit rows and what produced them, as for fa_confirmatory.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Browne, M. W. (1982). Covariance structures. In D. M. Hawkins (Ed.), *Topics in applied multivariate analysis* (pp. 72–141). Cambridge, UK: Cambridge University Press.

Chen, F. F., West, S. G., & Sousa, K. H. (2006). A comparison of bifactor and second-order models of quality of life. *Multivariate Behavioral Research, 41*(2), 189–225. doi:10.1207/s15327906mbr4102_5

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), *Structural equation modeling: Present and future* (pp. 195–216). Scientific Software International.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016). Evaluating bifactor models: Calculating and interpreting statistical indices. *Psychological Methods, 21*(2), 137–150. doi:10.1037/met0000045

Schmid, J., & Leiman, J. M. (1957). The development of hierarchical factor solutions. *Psychometrika, 22*(1), 53–61. doi:10.1007/BF02289209

Yung, Y.-F., Thissen, D., & McLeod, L. D. (1999). On the relationship between the higher-order factor model and the hierarchical factor model. *Psychometrika, 64*(2), 113–128. doi:10.1007/BF02294531

See Also

fa_bifactor for the bifactor model of the same groups, in which the second-order model is nested; fa_confirmatory for the correlated factors model of the first-order factors; reliability_omega and reliability_H.

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
data(holzinger_swineford)
hs_factors <- list(
  spatial = c("t1_visual_perception", "t2_cubes", "t4_lozenges"),
  verbal  = c("t6_paragraph_comprehension", "t7_sentence",
              "t9_word_meaning"),
  speed   = c("t10_addition", "t12_counting_groups_of_dots",
              "t13_straight_and_curved_capitals"),
  memory  = c("t14_word_recognition", "t15_number_recognition",
              "t17_object_number"))
hs_second <- list(general = c("spatial", "verbal", "speed", "memory"))

# One second-order factor over four first-order factors: the indices,
# each with a delta method interval.
so <- fa_second_order(holzinger_swineford, hs_factors, hs_second,
                      output = "measurement")
so

# The Schmid-Leiman loadings the estimates imply.
attr(so, "schmid_leiman")

# The second-order model is the bifactor model of the same groups with
# the general loadings proportional to the group loadings; the
# likelihood ratio test of that constraint, on 12 - 4 = 8 degrees of
# freedom.
second_order_fit <- fa_second_order(holzinger_swineford, hs_factors,
                                    hs_second, output = "fit")
bifactor_fit <- fa_bifactor(holzinger_swineford, hs_factors,
                            output = "fit")
print_anova(lavaan::lavTestLRT(second_order_fit, bifactor_fit))
```

---

fleiss_kappa *Fleiss's Kappa for Inter-Rater Agreement Among Multiple Raters*

Description

Computes Fleiss's (1971) kappa coefficient of agreement among $m \geq 2$ raters who classify each of N subjects into one of k nominal categories. Returns the point estimate together with the asymptotic standard error and the Wald confidence interval, plus a test of $H_0 : \kappa = 0$.

Usage

```
fleiss_kappa(
  ratings,
  conf_level = 0.95,
  ci_method = c("wald", "percentile", "bca"),
  B = 10000L,
  seed = NULL
)
```

Arguments

`ratings`	A numeric $N \times k$ matrix or `data.frame`: row i gives the counts of raters who assigned each of the k categories to subject i. Each row must sum to the same value m (the common number of raters per subject). Missing values are not supported, and SPSS user-defined missing values (a `haven_labelled_spss` column) are missing values, so a count declared missing is refused rather than counted.
`conf_level`	Confidence level for the interval (default `0.95`).
`ci_method`	Interval method: `"wald"` (the default, the asymptotic interval from the Gwet (2008) linearization variance), `"percentile"` (bootstrap percentile), or `"bca"` (bootstrap bias-corrected and accelerated). The multirater kappa variance literature is unsettled, and Zapf, Castell, Morawietz, and Karch (2016) recommend bootstrap intervals in this setting: the subjects (rows) are resampled with replacement `B` times, kappa is recomputed on each resample, and the interval is read off the bootstrap distribution; the BCa variant additionally adjusts the quantile positions for median bias and acceleration (Efron & Tibshirani, 1993). The `se`, `z_value`, and `p_value` columns keep their asymptotic definitions under every `ci_method`; only the interval changes.
`B`	Number of bootstrap replications when `ci_method` is `"percentile"` or `"bca"`, a single whole number (ignored for `"wald"`). Default `10000`, because kappa is cheap to recompute on a resample. The fewest accepted at the default `conf_level` is 400 for the percentile interval (200 at 0.90, 2000 at 0.99) and 2000 for the BCa interval (1000 at 0.90, 10000 at 0.99), whose adjustment moves the working quantiles into the tails; a smaller or fractional `B` is refused before anything is computed (class `dmar_count_invalid`). See *Bootstrap replications* under *Details* and `dmar_bootstrap`.
`seed`	Optional integer seed for the bootstrap. The default `NULL` uses the current state of the random number generator; a supplied seed is set internally and the prior state restored on exit.

Details

For n_{ij} = the number of raters who assigned subject i to category j, with $\sum_j n_{ij} = m$ for every i, define the marginal proportion of category j as $p_j = \sum_i n_{ij}/(Nm)$, and the per-subject agreement

$$P_i = \frac{1}{m(m-1)}\Big(\sum_j n_{ij}^2 - m\Big).$$

Then Fleiss's kappa is

$$\hat\kappa_F = \frac{\bar P - P_e}{1 - P_e}, \qquad \bar P = \frac{1}{N}\sum_i P_i, \qquad P_e = \sum_j p_j^2.$$

Standard error. Two variances are involved, because the variance of $\hat\kappa_F$ under $H_0 : \kappa = 0$ is not its variance at a nonzero value. The test of no agreement uses the null variance of Fleiss, Nee, and Landis (1979, Equation 12), who corrected the standard errors given in Fleiss (1971),

$$\mathrm{Var}_0(\hat\kappa_F) = \frac{2\left(P_e + P_e^2 - 2\sum_j p_j^3\right)}{N\,m\,(m-1)\,(1-P_e)^2},$$

and the reported z statistic and p-value come from it. On the Fleiss (1971) Table 1 example below this gives $z = 17.65$, matching `irr::kappam.fleiss`. The confidence interval instead uses the linearization variance of Gwet (2008, Section 6), which is consistent at the estimated $\hat\kappa_F$: each subject i contributes an influence value κ_i^* (Gwet's Equations 34 and 35), and $\mathrm{Var}(\hat\kappa_F) = \sum_i(\kappa_i^* - \hat\kappa_F)^2/\{N(N-1)\}$, Gwet's Equation 33 with the sampling fraction set to zero. (Gwet derives the variance for the multiple-rater pi statistic, which is the same estimator as Fleiss's kappa.) The Wald confidence interval is $\hat\kappa_F \pm z_{1-\alpha/2}\,\widehat{\mathrm{SE}}$, with the upper limit truncated at 1. Using the null variance for the interval would understate the standard error and give a spuriously narrow interval.

Bootstrap interval. The variance of multirater kappa is unsettled in the literature, and Zapf, Castell, Morawietz, and Karch (2016) recommend a bootstrap interval in this setting. With `ci_method` = `"percentile"` or `"bca"` the subjects (the rows of `ratings`) are resampled with replacement `B` times, kappa is recomputed on each resample, and the interval is read off the bootstrap distribution: the percentile interval takes the empirical quantiles, and the BCa interval adjusts the quantile positions for median bias and for acceleration (Efron & Tibshirani, 1993). Ask for it when N is small or $\hat\kappa_F$ is near a boundary, where the Wald interval's coverage is least dependable. The `se`, `z_value`, and `p_value` columns keep their asymptotic definitions under every `ci_method`; only the interval changes. Bootstrap results vary from run to run; supply `seed` for reproducibility.

Bootstrap replications. The percentile interval reads its limits from the quantiles of the `B` replicated values of kappa and is run only when `B` puts at least 10 replications beyond each limit (400 at the 95 percent level); the BCa interval, whose limits move farther into the tails, only with 50 beyond each nominal limit (2000). A replication in which kappa is undefined (chance agreement of 1 in the resample, as when every resampled rating falls in one category) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that

limit falls. When more than half were dropped, both limits are NA, the estimate is returned, and the warning has class dmar_bootstrap_insufficient. For the BCa interval the count of 10 is applied again where the bias correction and the acceleration place the limits, and when fewer than 10 replications lie beyond one of them no BCa interval is reported (limits NA, the same class, with the two positions as positions and a suggested B as B_suggest). When kappa is undefined with some case left out (chance agreement 1), when every bootstrap estimate falls on one side of the sample estimate, or when the acceleration is not finite, the BCa interval is refused with an error of class dmar_bca_undefined. options(dmar.bootstrap_exploratory = TRUE) lets a smaller B (2 or more) run for a first look or a demonstration; every such call gives a warning of class dmar_bootstrap_exploratory and the result is not one to report. See dmar_bootstrap for the rule and its reasons.

Fleiss's kappa is purely nominal (no weighting). For ordinal categories with two raters, use cohen_kappa with quadratic weights; for ordinal categories with three or more raters, an extension based on the intraclass correlation (icc) is more appropriate.

Value

A one-row data.frame (class dmar_tbl) with columns kappa, se (asymptotic standard error of $\hat{\kappa}_F$ used for the interval), ci_lower, ci_upper, z_value, p_value (Wald test of $H_0 : \kappa = 0$), n_subjects, n_raters (m), and n_categories (k). With a bootstrap interval the result carries the attributes B, B_used, n_dropped, B_required, and bootstrap_status (see *Bootstrap replications* under Details), which dmar_metadata reads.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin, 76*(5), 378–382.

Fleiss, J. L., Nee, J. C. M., & Landis, J. R. (1979). Large sample variance of kappa in the case of different sets of raters. *Psychological Bulletin, 86*(5), 974–977.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Gwet, K. L. (2008). Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology, 61*(1), 29–48. doi:10.1348/000711006X126600

Zapf, A., Castell, S., Morawietz, L., & Karch, A. (2016). Measuring inter-rater reliability for nominal data: Which coefficients and confidence intervals are appropriate? *BMC Medical Research Methodology, 16*, 93. doi:10.1186/s1287401602009

See Also

cohen_kappa, icc

Other reliability: cohen_kappa(), diagnosis_agreement, icc(), reliability(), reliability_H(), reliability_alpha(), reliability_kr20(), reliability_omega(), reliability_omega_categorical()

Examples

```
# Fleiss (1971) Table 1 example: 30 subjects rated by 6 raters into
# 5 diagnostic categories (Depression, Personality Disorder,
# Schizophrenia, Neurosis, Other). Each row of `ratings` gives, for one
# subject, the count of raters who chose each category (rows sum to 6).
# kappa = 0.430, matching Fleiss (1971).
fleiss_1971 <- matrix(c(
  0, 0, 0, 6, 0,
  0, 3, 0, 0, 3,
  0, 1, 4, 0, 1,
  0, 0, 0, 0, 6,
  0, 3, 0, 3, 0,
  2, 0, 4, 0, 0,
  0, 0, 4, 0, 2,
  2, 0, 3, 1, 0,
  2, 0, 0, 4, 0,
  0, 0, 0, 0, 6,
  1, 0, 0, 5, 0,
  1, 1, 0, 4, 0,
  0, 3, 3, 0, 0,
  1, 0, 0, 5, 0,
  0, 2, 0, 3, 1,
  0, 0, 5, 0, 1,
  3, 0, 0, 1, 2,
  5, 1, 0, 0, 0,
  0, 2, 0, 4, 0,
  1, 0, 2, 0, 3,
  0, 0, 0, 0, 6,
  0, 1, 0, 5, 0,
  0, 2, 0, 1, 3,
  2, 0, 0, 4, 0,
  1, 0, 0, 4, 1,
  0, 5, 0, 1, 0,
  4, 0, 0, 0, 2,
  0, 2, 0, 4, 0,
  1, 0, 5, 0, 0,
  0, 0, 0, 0, 6
), nrow = 30, byrow = TRUE)
fleiss_kappa(fleiss_1971)

# A percentile bootstrap interval, which resamples the subjects (rows)
# with replacement and recomputes kappa on each resample. Compare its
# limits with the Wald interval above; z_value and p_value keep their
# asymptotic definitions. B = 2000 keeps the example quick; a reported
# interval deserves the default B = 10000.
fleiss_kappa(fleiss_1971, ci_method = "percentile", B = 2000,
             seed = 113)
```

---

format_p *Format* p*-Values for Display at a Fixed Number of Decimal Places*

Description

Render a vector of *p*-values as character strings at a fixed number of decimal places (default 4) with a floor label "< 10^(-digits_p)" for values too small to express. Never uses scientific notation. This is the package-wide convention for displaying *p*-values used by the `dmar_tbl` print layer, by `correlations_test`, and by the helper display functions `print_anova` and `print_summary`. Exposing it as a user-facing function lets analysts apply the same convention to ad hoc *p*-values that they want to report in prose or in a manually constructed table.

Usage

```
format_p(p, digits_p = 4L)
```

Arguments

`p`	Numeric vector of *p*-values. `NA` values pass through as `NA_character_`. The function only formats: it shows each value as given and does not check that it is a probability. A value below 0 (and `-Inf`) is below the floor and shows as the floor label, a value above 1 is shown at `digits_p` decimal places (1.1 as "1.1000"), `Inf` is shown as Inf (with the leading spaces `formatC` gives it), and `NaN` is returned as `NA_character_`.
`digits_p`	Number of decimal places, a single whole number from 1 to 324, the most decimal places `formatC` writes (`3` and `3L` are accepted; `3.9` is refused with a condition of class `dmar_argument_invalid`, rather than cut to 3). Default `4L`. Values strictly below $10^{-\text{digits_p}}$ render as the floor label "< 10^(-digits_p)" (for example "< 0.0001" when `digits_p = 4`).

Details

The function does not modify the input value; the underlying numeric *p*-value retains full precision and can still be indexed out of whatever object holds it. Only the returned display string is rounded.

Value

A character vector of the same length as `p`.

Author(s)

Ken Kelley

See Also

`print_anova`, `print_summary`, `dmar_tbl`.

Examples

```
# Round to four decimals with a "< 0.0001" floor:
format_p(c(0.5, 0.0234, 0.0001234, 1e-10, NA))

# Six decimals when more precision is wanted:
format_p(0.0001234, digits_p = 6)
```

```
# Use inline in prose for a publication-style summary:
fit <- lm(weight ~ Time + Diet, data = ChickWeight)
p_time <- summary(fit)$coefficients["Time", "Pr(>|t|)"]
paste0("The Time coefficient was significant (p = ", format_p(p_time), ").")
```

`fraction_missing_information` *Fraction of Missing Information for Each Parameter of a FIML Fit*

Description

Estimates the fraction of missing information (FMI) for every free and every defined parameter of a model fit by full information maximum likelihood (FIML), and states each in the sample size and interval width terms of accuracy in parameter estimation (AIPE). The FMI of a parameter is the proportion of the sampling variance of its estimate that is due to the missing values (Savalei & Rhemtulla, 2012): zero when nothing is missing, and closer to one the less the observed data say about the parameter. Unlike the percentage of missing values, it depends on the parameter, the model, the correlations among the variables, and the missing data mechanism, so it is the number that says how much the missing values cost the estimate a reader cares about, and how many more cases a study with the same missing data would need to estimate it as precisely as complete data would.

Usage

```
fraction_missing_information(
  fit,
  information = c("unstructured", "structured", "hessian"),
  ci_method = c("none", "percentile"),
  conf_level = 0.95,
  B = 1000L,
  seed = NULL
)
```

Arguments

`fit`	A converged full information maximum likelihood fit: a **lavaan** fit with `missing = "ml"` (or `"ml.x"`), for example from `fa_confirmatory(..., missing = "fiml", output = "fit")` or `lavaan::cfa(..., missing = "ml")`, or a fit returned by `mlmr` or `mlmr_mv`, whose **lavaan** fit is used. Requires the **lavaan** package.
`information`	The estimate of the information matrix behind both standard errors: `"unstructured"` (the default), `"structured"`, or `"hessian"`, the three estimators of Chen and Savalei (2021). `"hessian"` is **lavaan**'s default: for a fit made with **lavaan**'s default information settings it agrees with `lavaan::parameterEstimates(fit, fmi = TRUE)` to about 10^{-7} when the fit runs **lavaan**'s EM algorithm as far as the computation here does (`em_h1_args = list(tol = 1e-10)`), and otherwise

to the precision of **lavaan**'s EM, which stops earlier; at extreme scales **lavaan**'s own values are the ones that are off. See *Details*.

`ci_method` `"none"` (the default) or `"percentile"`, a percentile bootstrap interval for each FMI. See *Details*.

`conf_level` Confidence level of the percentile interval (default `0.95`); used when `ci_method = "percentile"`.

`B` Number of bootstrap replications when `ci_method = "percentile"`, a single whole number. Default `1000`, smaller than the package's usual `10000` because every replication refits the model and then makes the two refits behind each FMI. The fewest accepted is 400 at the default `conf_level` (200 at 0.90, and 2000 at 0.99, which the default does not meet); a smaller `B` is refused before anything is computed (class `dmar_count_invalid`). See *Details* and `dmar_bootstrap`.

`seed` Optional integer seed for the bootstrap. The default `NULL` uses the current state of the random number generator; a supplied seed is set internally and the prior state restored on exit.

Details

What the fraction of missing information is. The missing information principle of Orchard and Woodbury (1972) splits the information that complete data would carry about the parameters, J_X, into the information the observed data carry, J_Y, and the information lost with the missing values, $J_{X|Y}$: $J_X = J_Y + J_{X|Y}$. In large samples the inverse of each information matrix is the covariance matrix of the corresponding estimates, so for parameter j

$$\mathrm{FMI}_j = 1 - \frac{\{J_X^{-1}\}_{jj}}{\{J_Y^{-1}\}_{jj}} = 1 - \frac{\mathrm{SE}_{C,j}^2}{\mathrm{SE}_{O,j}^2},$$

where $\mathrm{SE}_{O,j}$ is the standard error of the FIML estimate and $\mathrm{SE}_{C,j}$ is the standard error it would have had if the same cases had been observed completely (Savalei & Rhemtulla, 2012, Eq. 4). One minus the FMI is the asymptotic efficiency of the FIML estimate relative to the complete data estimate. The FMI is not the proportion of missing values. When the values are missing completely at random, the FMI of a parameter is bounded by the largest univariate and pairwise rates of missing values, and it falls below them when the observed variables carry information about the missing ones; when the values are missing at random it can be well above the missing rate (Savalei & Rhemtulla, 2012, pp. 489–490). Savalei and Rhemtulla (2012, p. 489) recommend reading the FMI as an effect size for the impact of the missing values on each parameter rather than against a cutoff.

The FMI in planning terms. A Wald interval's width is proportional to the standard error, and the standard error to $1/\sqrt{N}$, so the precision the missing values cost a parameter can be stated as sample sizes and widths (Savalei & Rhemtulla, 2012, p. 488). With *N* the number of cases behind the parameter, the table reports

- `N_effective` $= N(1-\mathrm{FMI})$, the number of complete cases that would estimate the parameter as precisely as the *N* incomplete cases did;
- `N_target` $= N/(1-\mathrm{FMI})$, the number of cases with the same pattern and mechanism of missing values that would match the precision of *N* complete cases; and
- `width_inflation` $= 1/\sqrt{1-\mathrm{FMI}} = \mathrm{SE}_O/\mathrm{SE}_C$, how many times wider the confidence interval is than it would have been with complete data at the same *N*. An FMI of .5 widens the interval by a factor of 1.41, and an FMI of .75 doubles it.

These are the statements of accuracy in parameter estimation, which plans a sample size for the width of a confidence interval rather than for power (Maxwell, Kelley, & Rausch, 2008), applied to the cost of the missing values. A sample size planned for a confidence interval width under complete data (by `ss_aipe_reliability`, `ss_aipe_sem_path`, or another `ss_aipe_*` planner) is, to a first approximation, divided by one minus the FMI when the study is expected to have the same missing data as the pilot or comparable study whose FMI is used. The approximation is asymptotic: it treats the FMI as known and the interval as a Wald interval.

How the two standard errors are computed. Both come from refits made at the estimates of `fit` with the optimizer switched off, under normal theory maximum likelihood standard errors and one type of information for both, which is how **lavaan** computes the FMI since version 0.6-7 (Chen & Savalei, 2021, p. 3). SE_O comes from the fit's own data under its missing data option. SE_C comes from a fit to the unrestricted FIML estimates of the means and covariances (the saturated model's estimates, found by the EM algorithm), with the same *N* and no rescaling of the covariance matrix. Savalei and Rhemtulla (2012) evaluated the complete data standard errors at the model implied moments, with expected information for the complete data and observed information for the incomplete data; Chen and Savalei (2021, p. 3) report that the mismatch produced negative estimates, particularly in small samples, and the published recipe is not zero for complete data when the model misfits. The matched computation used here gives an FMI of zero, to rounding error, for every parameter when no value is missing. The unrestricted moments are the FIML refit's own EM estimates, computed in its own order of the variables, so a fit made with **lavaan**'s `ov_order = "data"`, which orders its variables as the data do, gets the FMIs of the same fit made in the model's order (but see *Estimates that are not at the maximum* for such a fit that **lavaan** rescales). Both refits are made with the variables in units in which their standard deviations are near 1, and their standard errors are carried back to the original units exactly (see *Units of measurement*).

Units of measurement. The FMI is a ratio of two variances of the same estimate, so it does not depend on the units in which the variables are measured, and neither should its estimate. Three steps of the computation do depend on the units when they are carried out in the variables' own metric: **lavaan**'s EM estimate of the unrestricted moments, which stops when no mean or covariance changes by more than a fixed amount, too coarse a rule for variables with small variances; the inversion of the information matrix, which fails in floating point when the variances span many orders of magnitude; and the Hessian estimator, which differentiates the log likelihood numerically with a step suited to standard deviations near 1. Both refits are therefore made with every observed variable divided by the power of two nearest its standard deviation (over all groups), the factors **lavaan** itself divides by when it rescales a badly scaled fit, and with the parameter table carried through the same map: every estimate and fixed value is divided by the factor its units imply, every defined parameter and constraint is rewritten in terms of the parameters in their original units, and the standard errors are mapped back. Division by a power of two is exact in floating point, so the refits hold the fit's estimates exactly, and with no missing values their standard errors equal **lavaan**'s own to about ten digits. The EM is run until no mean or covariance changes by more than 10^{-10} times the smallest variance, about 10^{-10} in that metric, where **lavaan**'s default is an absolute 10^{-5}: EM converges slowly where much information is missing, and stopped at **lavaan**'s tolerance the FMIs of a factor model of **lavaan**'s nine Holzinger and Swineford tests, with values missing at random on two of them, moved by up to 2×10^{-5} when the EM was run further. It is run for at most 10,000 iterations (or the `max_iter` of the fit's `em_h1_args`, if that is larger), and the computation warns when it stops there short of its tolerance, which happens rarely and only where nearly all the information about some combination of the means and covariances is missing: in a one factor model of 12 items at *N* = 50 with 30 percent of the values of half of them missing, one data set needed about 12,500 iterations, and its FMIs at the 10,000th were within 3×10^{-7} of the converged ones

(and within .002 at **lavaan**'s tolerance). Where the estimate does not exist the computation stops (see *Fits that are refused*). With the fit's estimates held fixed, the three estimators' FMIs then agreed to within 2×10^{-9} when every variable was multiplied by a constant from .001 to 1,000,000, in that factor model and in a regression of one variable on three with values missing at random. Computed from fits to the rescaled data, the regression's FMIs agreed with those at a standard deviation of 1 to within 1.3×10^{-7} when one variable or every variable was multiplied by .001, 10,000, or 100,000, and when the outcome was multiplied by 100,000,000. **lavaan**'s own `parameterEstimates(fit, fmi = TRUE)`, which refits in the variables' own metric, was off by up to .93 (by .04 to .07 for the slopes) with every standard deviation near 100,000, and returned `NA` with every standard deviation near 300,000, or with one near 10,000,000.

Estimates that are not at the maximum. The FMI is computed at the fit's estimates, so it is the FMI of the maximum likelihood estimates only when they are at the maximum. **lavaan** rescales its own fit only when a standard deviation is .001 or less or 1000 or more, or two differ by a factor of 1000, and a fit to variables measured in small or large units between those limits can stop short of the maximum, although **lavaan** reports it as converged and its check of the gradient, made in the variables' own metric, passes it. With every standard deviation of that regression near .003 the slopes were off by .14 percent and the FMIs computed at them by up to .001 (by .00006 near .01 and .03). The factor model of the nine tests, identified by a marker indicator for each factor (the default of `lavaan::cfa()`), with every variable multiplied by 100 had a log likelihood .11 below the maximum and FMIs off by up to .011, and multiplied by 500 a log likelihood 287 below it and FMIs off by .12; identified by fixing each factor's variance (`std.lv = TRUE`) it reached the maximum at every scale from .01 to 500. In **lavaan** 0.7-3 a fit made with `ov_order = "data"` and rescaled by **lavaan** is not the maximum likelihood fit when the columns of the data are not in the model's order: in such a regression the log likelihood was 50 below the maximum, and the slopes' FMIs were .28, .22, and .70 in place of .06, .19, and .70. The computation therefore takes one Newton step from the fit's estimates in the metric of the refits. Where the step moves an estimate by more than 10^{-4} of its size (or of 1, for an estimate smaller than 1; fits at the maximum took steps below 6×10^{-6} in the examples studied here), it computes the FMIs again one step on, at the maximum to working precision, and warns when they differ from those at the fit's estimates by more than 10^{-4}, naming the largest change of an estimate, the gain in the log likelihood, and the largest change of an FMI. The FMIs returned are those at the fit's estimates. For variables in small or large units, or in an order other than the model's, the remedy is to fit the model again with **lavaan**'s `rescale_data = TRUE`, which rescales whatever the units, and with the columns of the data in the model's order: the regression near .003 and the factor model multiplied by 100 then gave the FMIs of the variables in their own units to within 10^{-7} and 2×10^{-6}. The warning names units as the cause, with that remedy, only for a fit **lavaan** did not rescale with a standard deviation of .01 or less or 100 or more (with every variable multiplied by one constant, the factor model and the regression reached the maximum with standard deviations from .01 to 80), and the order of the variables only for a fit **lavaan** rescaled; a fit made with `rescale_data = TRUE`, or one **lavaan** rescaled itself, is not told to take it again. A fit can also stop short where the likelihood is flat in some direction, as in a small sample with most of the outcome missing (60 cases, 16 of them observing the outcome of a regression on four predictors), where **lavaan**'s optimizer stopped about 5×10^{-4} standard errors short of the maximum and neither rescaling, a tighter tolerance, nor a restart from the estimates moved it; the FMIs were then within 3×10^{-5} of the maximum's, which passes, or in another such data set 6×10^{-4} off, which the warning reports without naming units as the cause. The check is not made for a model with an inequality constraint or a parameter at a bound, where the gradient need not be zero at the maximum.

The refits stay in the variables' own metric for a model with composites (`<~`) or product terms, which

lavaan does not rescale either, and for one that shares a label between parameters of variables in different units under `ceq.simple = TRUE` (one common factor for every variable is tried first). For such a model the Hessian estimator warns when a standard deviation falls outside .01 to 1000, an information matrix that cannot be inverted stops the computation with a message that says so, and the check for a maximum is not made; dividing the variables by constants that bring their standard deviations nearer 1, which leaves the FMI unchanged, is the remedy for all three.

The three estimators and the default. Chen and Savalei (2021) studied three versions of the observed information, which coincide for a saturated model such as a regression and estimate the same quantity when the model is correct.

- `"hessian"` (their first estimator) uses the numeric Hessian of the log likelihood, **lavaan**'s default. `information = "hessian"` agrees with `lavaan::parameterEstimates(fit, fmi = TRUE)` to about 10^{-7} for a fit made with `em_h1_args = list(tol = 1e-10)`, so that **lavaan** runs its EM as far, and otherwise to the precision of **lavaan**'s EM, about 10^{-5} in the examples studied here (see *Units of measurement* above).
- `"structured"` (their second) keeps the first, dominant term of the observed information and evaluates it at the model implied means and covariances.
- `"unstructured"` (their third) has the same analytic form but uses the unrestricted estimates of the means and covariances wherever the computation needs them.

In their simulations the unstructured estimator was much less likely than the others to break down and more precise than the structured one, while the Hessian estimator was the most likely to break down when the sample was small or the missing rate high (Chen & Savalei, 2021, p. 15). It is therefore the default here, which differs from **lavaan**'s. It also cannot be negative, apart from rounding error: both of its informations are evaluated at the saturated maximum likelihood estimates, where the observed data information of the saturated model is bounded above by its complete data information, and the bound carries through the model Jacobian to every parameter, defined parameters included, so $0 \leq \mathrm{FMI} < 1$. The structured and Hessian estimators evaluate the information away from that point and can fall below zero, far below when the model is misspecified: with the three-factor model for **lavaan**'s nine Holzinger and Swineford tests and values missing at random on six of them, the Hessian estimates for the speed loadings are near -2 and the unstructured ones between .17 and .34. When the model is wrong only the Hessian estimator is consistent for its population value, an advantage that Chen and Savalei (2021, p. 15) note has yet to be shown to matter in practice. Under misspecification the three estimate different quantities; the unstructured one is the information about the parameters carried by the unrestricted moments.

MLR and other standard errors. For a maximum likelihood fit with other standard errors (estimator `"MLR"`, whose standard errors are robust to nonnormality; `"MLF"`, whose standard errors use the outer product of the casewise gradients; or a fit with standard errors robust to clustering and no multilevel structure) the estimates are the maximum likelihood estimates, and only the standard errors differ. The FMI is defined through the information of the likelihood, and the information of the working normal likelihood is the bread of the sandwich, so the FMI reported for such a fit is the normal theory FMI at its estimates, the same number the ML fit gives. The source fit's standard errors are named in the `source_se` attribute and under the printed table. A robust counterpart would need the complete data meat of the sandwich, which involves fourth moments of the values that were not observed and is not identified from incomplete data without a distributional assumption. The ratio of a normal theory complete data variance to a robust FIML variance is not an FMI, because it attributes the inflation due to nonnormality and misfit to the missing values. The robust and the

normal theory fractions coincide when nonnormality inflates the variances by the same factor with and without the missing values, which is an assumption, not a result.

Defined parameters. A parameter defined with `:=`, such as the `omega_`, `H_`, and `ave_` rows that `fa_confirmatory` defines for each factor or an indirect effect, has a delta method standard error under each refit, so its FMI is the same ratio. For a reliability coefficient this is the FMI that a sample size plan for its confidence interval needs.

Multiple groups and the *N* column. Multiple group fits are supported. The `N` of each row is the number of cases behind that parameter: the group's *N* for a parameter of one group; the summed *N* of the groups involved for a parameter constrained equal across groups (by a shared label, an `==` constraint written with labels or with defined parameters, or `group.equal`), whose FMI is then the same in every group; and, for a defined parameter, the summed *N* of the groups its expression reaches. *N* counts the cases the fit used. **lavaan** sets aside a case with no observed value on any analyzed variable, as Savalei and Rhemtulla (2012, p. 487) did in their example, and the number set aside is the `N_excluded` attribute. In a regression fit directly with **lavaan**, `missing = "ml"` with `fixed.x = TRUE` also drops every case missing a predictor before anything is estimated; `missing = "ml.x"`, or `mlmr`, which models the predictors jointly by default, keeps those cases.

Auxiliary variables. A fit with auxiliary variables entered as saturated correlates (Graham, 2003), as **lavaan**'s `aux` argument enters them (and `fa_confirmatory(..., aux = )` passes it on), is supported, and the table includes the rows of the auxiliaries' variances, means, and covariances. An auxiliary that shares a factor with the indicators covaries with them through their residuals only, so the residual covariance matrix that includes it is not positive definite, and the covariances can imply correlations above 1 with the residuals, although every variance is positive and the model implied covariance matrix is positive definite. **lavaan** flags both, and a refit that took its checks at face value would replace the estimates; the refits here keep them.

Precision of the FMI itself. A sample FMI is a noisy estimate of the population value. In the simulations of Chen and Savalei (2021, p. 14) it was nearly unbiased but varied greatly from sample to sample at small *N*; with 10 to 20 percent of all values missing, samples of several hundred gave reasonably precise estimates, and with 30 percent missing it took more than 1,000. `ci_method = "percentile"` resamples the cases the fit used (within groups for a multiple group fit), refits the model with its own options, recomputes every FMI, and takes as the limits the quantiles of the replications at `(1 - conf_level) / 2` and `(1 + conf_level) / 2` (Efron & Tibshirani, 1993). The properties of this interval have not been studied (Chen and Savalei, 2021, p. 15, list bootstrap intervals for the sample FMI as future work); it shows the sampling variability of the estimate, and its upper limit is the conservative value to carry into a sample size plan.

Bootstrap replications. The two limits of each parameter's interval are quantiles of the `B` replicated FMI values, so the interval is run only when `B` puts at least 10 replications beyond each limit (400 at the 95 percent level). A replication in which the FMI of some parameter is undefined (the refit fails, does not converge, or returns an FMI that is not a finite number) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the limits of every parameter are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. `options(dmar.bootstrap_exploratory`

= TRUE) lets a smaller B (2 or more) run for a first look or a demonstration; every such call gives a warning of class dmar_bootstrap_exploratory and the result is not one to report. See dmar_bootstrap for the rule and its reasons.

Each replication costs a model fit and two refits, so the interval takes about a minute at the default B for a model the size of the one in the examples; the call is fraction_missing_information(fit, ci_method = "percentile", seed = 113).

Fits that are refused. The computation needs a converged and unweighted maximum likelihood fit with one level, to continuous variables, that used every observed value. A fit with ordered indicators (and so WLSMV or another least squares estimator), with an estimator outside the maximum likelihood family, with listwise deletion (**lavaan**'s default, which leaves no missing information to measure), with the pairwise, available cases, or two-stage missing data options, with conditional.x = TRUE, with more than one level, with sampling weights, or that did not converge is refused with a message that says why, as are a lavaanList or pooled multiple imputation object and a rotated exploratory (efa) solution, whose loadings **lavaan** rotates after estimation. As a last guard, the computation stops if the refit at the estimates moves them. Besides a rotated solution, this happens with an improper one: a Heywood case whose negative residual variance exceeds the variable's observed variance in size is a solution **lavaan** (0.7-3) reads as having run away, so it will not take those estimates as the starting point of a refit and starts again from its own simple starting values. The standard errors of such a refit belong to those values, not to the fit, and lavaan::parameterEstimates(fit, fmi = TRUE), which makes the same refit, then reports FMIs computed at them without saying so. Every other solution **lavaan** returns is refit at its estimates, whatever its checks of a solution say of them, and its FMIs are computed there: a Heywood case whose residual variance is less far below zero, two factors correlated above 1 (a covariance matrix of the latent variables that is not positive definite), and a covariance between residuals that implies a correlation above 1. Before it refits, lavaan::parameterEstimates(fit, fmi = TRUE) shrinks such a covariance to a correlation below 1 (its check of starting values), so it reports the FMIs of other estimates: for two factors correlated 1.08 in a sample of 60, it gave the covariance an FMI of -1.62, where the FMI at the estimates is .25. These are the FMIs of an improper solution, and they deserve the same caution as the solution itself, of which **lavaan** warns when it fits the model. Finally, the computation stops when the information matrix of a refit cannot be inverted, with a message that names the likeliest cause: variables too badly scaled for a model the refits keep in their own metric (see *Units of measurement*); a predictor (an observed variable regressed on) that lies so far from zero, relative to its standard deviation, that an intercept and its coefficients are nearly collinear, which no division by a power of two changes (a predictor of a regression with a standard deviation of 1 did so from about 8000 from zero, and gave the FMIs of the unshifted data at 7000; an indicator's or an outcome's location enters only its own intercept and does not); a model that is not identified; or a structured or Hessian information that is singular at the estimates. A predictor 1000 or more standard deviations from zero is named only when the same model fit to the data with it centered has an information that can be inverted, since centering, which changes no FMI but those of intercepts, cannot repair a model that is not identified; where centering would change a fixed or constrained intercept, so that this cannot be checked, both causes are given.

The computation also stops when the data define no unrestricted estimate of the means and covariances at which to evaluate the fraction (all three estimators need it: the unstructured information is evaluated there, and the complete data side of each is fitted to it). A pair of variables that no case observes together, as in a planned missing design, leaves their covariance undetermined: **lavaan**'s EM leaves it at its starting value and only warns, and in a two-form design on seven of the Holzinger and Swineford tests the FMIs moved by up to .12 across values of it that the data could not tell apart. The pairs are named, as cov_fiml and the reliability family name them. The computation also checks

for a supported affine-collapse direction: all rows observing a subset must satisfy the same affine relation, with a nonzero coefficient on every member of the subset. Every partially overlapping row is included when a candidate support is reduced. A subset with no more observed rows than variables and a verified full-support null direction supplies a sufficient unbounded-likelihood diagnosis, even when EM stopped at a stable interior point. The check considers the full variable set, observed-pattern sets, single variables, and reductions of their null supports; it is not an exhaustive existence test.

Affine differences are column scaled. A singular-value tolerance of $100\epsilon \max(n, p) \max(1, d_{\max})$, with machine precision ϵ, locates candidate nullspaces. A deterministic normalized direction must have every coefficient larger in absolute value than $p\sqrt{\epsilon}$ and a residual below that tolerance. A relation inferred only numerically in an overdetermined subset is reported as numerical instability, without claiming mathematical nonexistence. A near-singular iterate at EM's iteration limit is a fitting failure. No diagnosed obstacle means that no implemented diagnostic failed; it does not certify finite global existence or uniqueness. These checks concern the unrestricted likelihood needed for FMI. The constrained substantive model's own estimates and standard errors do not use the unrestricted estimate (under **lavaan**'s default information); its chi-square test and the fit indices built on the unrestricted model's likelihood do. These stops are errors of classes `dmar_fmi_never_together` and `dmar_fmi_saturated_singular`, so that a caller can catch them.

Value

A `data.frame` (class `dmar_fmi`, `dmar_tbl`) with one row per free parameter and per defined (`:=`) parameter, in the order of the **lavaan** parameter table, and the same columns for every call: `term` (**lavaan**'s `lhs` `op` `rhs`, or the name of a defined parameter); `op` (the **lavaan** operator, for filtering); `group` (the group label; `NA` for a single group fit and for defined parameters); `fmi`; `fmi_ci_lower` and `fmi_ci_upper` (the percentile limits; `NA` when `ci_method = "none"`); `width_inflation`; `N`; `N_effective`; `N_target`; `estimate` (the parameter estimate of `fit`); and `se_observed` and `se_complete` (SE_O and SE_C, normal theory standard errors under `information`). The attributes record `information` and `information_description`; `source_estimator`, `source_se`, and `source_missing`, the settings of `fit`; `basis`; `N_used` (per group); `N_excluded`; `n_patterns` (per group); `fmi_max`; `ci_method`; and, with the percentile interval, `conf_level`, `B`, `B_used`, `n_dropped`, `B_required`, and `bootstrap_status` (see *Bootstrap replications* under Details). The print method leaves out the columns that carry nothing for the call at hand (`op`; `group` for a single group; the interval limits without an interval), shows an FMI within 10^{-10} of zero (a fully observed variable's intercept, say) as 0, and states the information, the sample size, and the imprecision of the estimate under the table. The object keeps every column at full precision. `tidy.dmar_fmi` and `glance.dmar_fmi` give the one row per parameter and the one row summary views.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chen, L., & Savalei, V. (2021). Three sample estimates of fraction of missing information from full information maximum likelihood. *Frontiers in Psychology, 12*, 667802. doi:10.3389/fpsyg.2021.667802

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Orchard, T., & Woodbury, M. A. (1972). A missing information principle: Theory and applications. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 697–715). Berkeley, CA: University of California Press.

Savalei, V., & Rhemtulla, M. (2012). On obtaining estimates of the fraction of missing information from full information maximum likelihood. *Structural Equation Modeling, 19*(3), 477–494. doi:10.1080/10705511.2012.687669

See Also

`missing_patterns` and `missing_coverage` for the description of the missing values that comes before any model; `fa_confirmatory` and `mlmr` for FIML fits; `ss_aipe_reliability` for planning the sample size of a reliability interval.

Other missing data: `cov_fiml()`, `df_barnard_rubin()`, `df_missing()`, `missing_coverage()`, `missing_patterns()`

Examples

```
# The six spatial tests of the Holzinger and Swineford data. The revised
# paper form board (t25) and the flags test (t26) were given only at
# Grant-White, so both are missing for the 156 Pasteur pupils.
spatial <- c("t1_visual_perception", "t2_cubes", "t3_paper_form_board",
             "t4_lozenges", "t25_paper_form_board_r", "t26_flags")
missing_patterns(holzinger_swineford, items = spatial)

fit <- fa_confirmatory(data = holzinger_swineford,
             factors = list(spatial = spatial),
             missing = "fiml", output = "fit")
fmi <- fraction_missing_information(fit)
fmi

# The loadings of t25 and t26 have an FMI near .51, close to their 52
# percent of values missing, so each is estimated about as precisely as
# 147 complete cases would estimate it. Omega for the six tests loses
# less, about a quarter of its information: its interval is 1.15 times
# as wide as complete data would give, and the same precision with the
# same missing values would take about 398 cases.
fmi[fmi$term == "omega_spatial",
    c("term", "fmi", "N_effective", "N_target", "width_inflation")]

# The Hessian estimator, lavaan's default, for comparison.
fraction_missing_information(fit, information = "hessian")$fmi
```

glance.dmar_fa_confirmatory *Glance at a Multiple-Factor CFA Fit*

Description

Returns a one-row data.frame of model-level summaries from a fa_confirmatory table, in the column convention used by the **broom** ecosystem.

Usage

```
## S3 method for class 'dmar_fa_confirmatory'
glance(x, ...)
```

Arguments

x	A dmar_fa_confirmatory object returned by fa_confirmatory.
...	Unused.

Value

A one-row data.frame with columns chi_square, df, p_value, cfi, tli, rmsea, rmsea_low, rmsea_high, srmr, AIC, BIC, logLik, and the columns that say what produced them, so that rows of different fits can be stacked and still read: estimator; missing, the missing data method of the fit ("listwise", "pairwise", "fiml", "two_stage", "robust_two_stage", or "none" for a covariance matrix); test, the statistic in chi_square ("standard", "scaled", or a two-stage statistic); fit_indices, the version of cfi, tli, and rmsea ("robust", "scaled", or "standard"); and rmsea_ci_level, the level of the RMSEA interval (rmsea_conf_level, 0.90 by default, where the conf_level column of tidy() is the level of the parameter intervals). After two-stage estimation logLik is the complete data normal log likelihood at the EM estimates and AIC and BIC are the two-stage criteria of Lai (2021), comparable only with two-stage fits of the same variables, not with a FIML or listwise row; and the difference of two rows' chi_square or logLik is not a chi square test of nested models under a scaled or two-stage statistic (lavaan::lavTestLRT() on output = "fit" gives the right one).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K. (2021). Using information criteria under missing data: Full information maximum likelihood versus two-stage estimation. *Structural Equation Modeling, 28*(2), 278–291. doi:10.1080/10705511.2020.1780925

`glance.dmar_mediation_mbco`

Glance at an MBCO Mediation Fit

Description

Returns a one-row `data.frame` of full-model summaries from a `mediation_mbco` table, in the column convention used by the **broom** ecosystem.

Usage

```
## S3 method for class 'dmar_mediation_mbco'
glance(x, ...)
```

Arguments

`x`	A `dmar_mediation_mbco` object returned by `mediation_mbco`.
`...`	Unused.

Value

A one-row `data.frame` with columns `nobs`, `npar`, `deviance`, `AIC`, `BIC`, `logLik`.

Author(s)

Ken Kelley <kkelley@nd.edu>

`glance.dmar_reliability`

Glance at a Reliability Coefficient Estimate

Description

Returns a one-row `data.frame` of model-level summaries in the column convention used by the **broom** ecosystem (`coefficient`, `estimate`, `se`, `ci_lower`, `ci_upper`, `conf_level`, `nobs`, `n_items`, `ci_method`, `wald_df`, the degrees of freedom of the *t* reference of a Wald interval, `Inf` for the normal reference or another interval).

Usage

```
## S3 method for class 'dmar_reliability'
glance(x, ...)
```

Arguments

x	A dmar_reliability object.
...	Unused.

Value

A one-row data.frame.

Author(s)

Ken Kelley <kkelley@nd.edu>

---

glance.mlmr *Glance at an Mlmr Fit*

Description

Returns a one-row data.frame of model-level summaries in the column convention used by the **broom** ecosystem (R2, adj_R2, sigma, statistic, p_value, df, logLik, AIC, BIC, deviance, df_residual, nobs). The statistic and p_value columns report the omnibus likelihood ratio test of all slopes equal to zero (the FIML analog of the lm omnibus *F*-test).

Usage

```
## S3 method for class 'mlmr'
glance(x, ...)
```

Arguments

x	An object of class "mlmr".
...	Unused.

Value

A one-row data.frame.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
fit <- mlmr(t6_paragraph_comprehension ~ t5_general_information +
              t9_word_meaning,
            data = holzinger_swineford, ci_method = "wald")
generics::glance(fit)
```

gwet_ac *Gwet's AC1 and AC2 Chance-Corrected Agreement Coefficients*

Description

Computes Gwet's AC1 (nominal data; Gwet, 2008) and AC2 (ordinal data with user-supplied weights; Gwet, 2014) chance-corrected agreement coefficients for two or more raters. When one category is rare or very common, Cohen's κ can be near zero even though the raters agree on almost every unit (the kappa paradox; Feinstein & Cicchetti, 1990). The chance-agreement term of AC1 and AC2 shrinks rather than grows as the prevalence becomes extreme, so the two coefficients stay near the observed agreement in that case (see Details).

Usage

```
gwet_ac(
  ratings,
  weights = c("unweighted", "linear", "quadratic"),
  conf_level = 0.95
)
```

Arguments

`ratings`	A units $\times$ raters matrix or `data.frame`. Rows = units; columns = raters. `NA` entries are allowed.
`weights`	One of `"unweighted"` (default; computes AC1) or `"linear"` / `"quadratic"` (compute AC2 with weight matrices used by weighted-κ conventions).
`conf_level`	Confidence level. Default `0.95`.

Details

Coefficient.

$$\widehat{\mathrm{AC}} \;=\; \frac{p_a - p_e}{1 - p_e},$$

identical to Cohen's κ in structure but with a different chance-correction p_e:

$$p_e \;=\; \frac{T_w}{Q(Q-1)} \sum_{k=1}^{Q} \pi_k (1 - \pi_k),$$

where π_k is the mean within-unit proportion of category k, Q is the number of categories, and T_w is the sum of all entries of the weight matrix. For nominal data with unit weights (AC1), $T_w = Q$ and p_e reduces to $(1/(Q-1)) \sum_k \pi_k(1 - \pi_k)$.

Why AC over κ. κ can be near zero even when raters agree on almost every unit if the trait is rare or very common (the "kappa paradox"; Feinstein & Cicchetti, 1990). Gwet's AC keeps the same chance-correction logic but uses a less extreme reference distribution.

Variance. The SE is Gwet's (2008) linearization variance,

$$\mathrm{Var}(\widehat{\mathrm{AC}}) \;=\; \frac{1-f}{n(n-1)} \sum_i (\widehat{\mathrm{AC}}_i^{*} - \widehat{\mathrm{AC}})^2,$$

where $\widehat{\mathrm{AC}}_i^{*}$ is the ith unit's influence value, combining its agreement and chance-term contributions, n is the number of units, and f is the sampling fraction (0 for an infinite target population). The interval is $\widehat{\mathrm{AC}} \pm t_{1-\alpha/2,\,n-1} SE$, with the upper limit truncated at 1. These quantities match Gwet's (2014) reference software.

Value

A `data.frame` with rows for the point estimate $\widehat{\mathrm{AC}}$, the standard error, the CI lower and upper limits, the percent agreement p_a, and the chance-agreement term p_e.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Feinstein, A. R., & Cicchetti, D. V. (1990). High agreement but low kappa: I. The problems of two paradoxes. *Journal of Clinical Epidemiology, 43*(6), 543–549. doi:10.1016/08954356(90)90158L

Gwet, K. L. (2008). Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology, 61*(1), 29–48. doi:10.1348/000711006X126600

Gwet, K. L. (2014). *Handbook of inter-rater reliability* (4th ed.). Advanced Analytics, LLC.

See Also

`cohen_kappa`, `fleiss_kappa`, `krippendorff_alpha`

Other agreement and measurement: `R2_mixed_effects()`, `content_validity_index()`, `icc_lmer()`, `krippendorff_alpha()`, `limits_of_agreement()`, `lin_ccc()`, `variance_components_mls()`

Examples

```
# 1. Unweighted AC1, two raters, nominal:
set.seed(113)
r1 <- sample(c("A", "B", "C"), 50, replace = TRUE)
r2 <- ifelse(runif(50) < 0.8, r1, sample(c("A", "B", "C"), 50, TRUE))
gwet_ac(cbind(r1, r2))

# 2. AC2 with linear weights, ordinal scale 1-5:
set.seed(113)
r1 <- sample(1:5, 60, replace = TRUE)
r2 <- pmin(5, pmax(1, r1 + sample(-1:1, 60, replace = TRUE)))
gwet_ac(cbind(r1, r2), weights = "linear")
```

`helmert_coding` *Helmert-Coding Contrast Matrix for a Factor*

Description

Builds the Helmert-coding contrast matrix for a factor with a levels. The k-th column contrasts the $(k+1)$-th level against the average of all preceding levels, giving a fully orthogonal set under equal sample sizes. The returned matrix has columns named after the contrasted level rather than the numeric column names produced by `stats::contr.helmert()`.

Usage

```
helmert_coding(levels)
```

Arguments

`levels`	Either an integer giving the number of levels or a character / factor vector giving the level labels. If integer, the labels default to `"L1"`, `"L2"`, ...

Details

Why Helmert. Helmert contrasts are the canonical "sequential" orthogonal contrast set: under equal-n, every column is orthogonal to every other column and to the intercept. They are useful when the factor has a natural ordering and the research questions are "does the k-th level differ from the average of the preceding levels?"

Equivalent to. `stats::contr.helmert()` but with interpretable column names.

Value

A numeric $a \times (a-1)$ matrix with row names = the factor levels and column names of the form `"L2_vs_prior"`, `"L3_vs_prior"`, ...

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Lawrence Erlbaum.

See Also

`contr.helmert`, `effects_coding`, `is_orthogonal_set`

Other design utilities: `design_consequences()`, `design_effect()`, `effects_coding()`, `is_orthogonal_set()`, `orthogonal_polynomial()`

Examples

```
# 1. Helmert coding for a 4-level factor:
helmert_coding(c("baseline", "week1", "week2", "week3"))

# 2. Confirm orthogonality:
M <- helmert_coding(4)
is_orthogonal_set(M)
```

---

holzinger_swineford *Holzinger and Swineford (1939) Factor Analysis Study*

Description

The complete data set from Holzinger and Swineford's (1939) *A study in factor analysis: The stability of a bi-factor solution*. Scores on 26 ability tests for 301 seventh and eighth grade pupils at two Chicago elementary schools, Pasteur (n = 156) and Grant-White (n = 145). The data have been used over the subsequent decades as a benchmark in factor analysis, confirmatory factor analysis, structural equation modeling, and reliability research (see Details).

Usage

```
holzinger_swineford
```

Format

A data frame with 301 observations and 34 variables.

`id` Case identifier as in the original monograph. The numbering is not strictly consecutive; a small number of cases were dropped during data preparation and the original numbering was preserved (hence the visible skips).

`sex` Factor with levels `Female` and `Male`.

`grade` Grade in school, 7 or 8.

`age` Age in completed years, ignoring months past the most recent birthday.

`month_since_birthday` Completed months since the most recent birthday.

`age_months` Age in completed months, computed as `12 * age + month_since_birthday`.

`age_years` Age in fractional years, computed as `age + month_since_birthday / 12`.

`school` Factor with levels `Grant-White` and `Pasteur`, naming the two Chicago elementary schools from which pupils were drawn.

`t1_visual_perception` Visual perception test (spatial).

`t2_cubes` Cubes test (spatial).

`t3_paper_form_board` Paper form board test (spatial).

`t4_lozenges` Lozenges test (spatial).

`t5_general_information` General information test (verbal).

`t6_paragraph_comprehension` Paragraph comprehension test (verbal).

`t7_sentence` Sentence completion test (verbal).

`t8_word_classification` Word classification test (verbal).

`t9_word_meaning` Word meaning test (verbal).

`t10_addition` Addition test (mental speed).

`t11_code` Code test (mental speed).

`t12_counting_groups_of_dots` Counting groups of dots test (mental speed).

`t13_straight_and_curved_capitals` Straight and curved capitals test (mental speed).

`t14_word_recognition` Word recognition test (memory).

`t15_number_recognition` Number recognition test (memory).

`t16_figure_recognition` Figure recognition test (memory).

`t17_object_number` Object-number test (memory).

`t18_number_figure` Number-figure test (memory).

`t19_figure_word` Figure-word test (memory).

`t20_deduction` Deduction test (reasoning).

`t21_numerical_puzzles` Numerical puzzles test (reasoning).

`t22_problem_reasoning` Problem reasoning test (reasoning).

`t23_series_completion` Series completion test (reasoning).

`t24_woody_mccall` Woody-McCall mixed fundamentals, form I (arithmetic).

`t25_paper_form_board_r` Revised paper form board, administered only to the Grant-White pupils as an experimental substitute for `t3_paper_form_board`. `NA` for the 156 Pasteur pupils.

`t26_flags` Flags test, administered only to the Grant-White pupils as an experimental substitute for `t4_lozenges`. `NA` for the 156 Pasteur pupils.

Details

Karl John Holzinger (1893 to 1954) was a quantitative psychologist at the University of Chicago and one of the central figures in the first generation of factor analysis. He spent the 1922 to 1923 academic year working with Charles Spearman at University College London, absorbing Spearman's two-factor theory of intelligence, and later developed the *bi-factor model* as an extension of that theory. The bi-factor model posits a single general intelligence factor that runs through all tests, plus several group factors that capture residual correlation among substantively related subgroups of tests, uncorrelated with the general factor. The bifactor model of later psychometric work (see `fa_bifactor`) has this structure. Frances Swineford was Holzinger's research collaborator at the University of Chicago and a coauthor on much of his applied work.

The 1939 monograph reports a study of pupils in seventh and eighth grade classrooms at two Chicago elementary schools, Pasteur and Grant-White. Two schools were used deliberately, so that the stability of a bi-factor solution could be assessed by fitting the same model in each school and comparing the results. The 26 tests were designed to span five hypothesized ability domains:

- **Spatial**: tests 1 to 4 (visual perception, cubes, paper form board, lozenges), with tests 25 (a revised paper form board) and 26 (flags) administered only to the Grant-White sample as experimental substitutes for tests 3 and 4.

- **Verbal**: tests 5 to 9 (general information, paragraph comprehension, sentence completion, word classification, word meaning).
- **Mental speed**: tests 10 to 13 (addition, code, counting groups of dots, straight and curved capitals).
- **Memory**: tests 14 to 19 (word recognition, number recognition, figure recognition, object-number, number-figure, figure-word).
- **Reasoning and arithmetic**: tests 20 to 24 (deduction, numerical puzzles, problem reasoning, series completion, Woody-McCall mixed fundamentals).

Holzinger and Swineford concluded that the bi-factor solution was reasonably stable across the two schools, supporting the substantive interpretation of a general factor together with group factors.

The data have far outlived their original purpose. Jöreskog (1969) used a 9-test subset drawn from the Grant-White sample (n = 145) to introduce confirmatory maximum likelihood factor analysis; the same nine tests, for both schools and in per-item-rescaled form, are shipped as `HolzingerSwineford1939` in the **lavaan** package, whose confirmatory factor analysis example (`?lavaan::cfa`) uses them. The complete 26-test data shipped here support a wider range of analyses, including comparisons of the spatial, verbal, speed, memory, and reasoning ability blocks and multiple group analyses across the Pasteur and Grant-White schools.

The values in `holzinger_swineford` are the corrected version of the data, identical on all 26 test cells to `MBESS::HS` from MBESS version 4.9.3 onward and to `psychTools::holzinger.raw`. An older version of the data, with approximately 53 cell values that were later corrected, continues to circulate as `HS.data` in the **sem** package and as `HS.ability.data` in the **OpenMx** package; both are byte-identical snapshots taken from **MBESS** version 4.6.0 prior to the correction. The corrections are concentrated on the memory and reasoning tests, with the largest cluster on `t20_deduction` (15 cells, including a number of sign flips that reflect a corrected guessing-penalty adjustment).

Author(s)

Ken Kelley

Source

Holzinger, K. J., and Swineford, F. (1939). *A study in factor analysis: The stability of a bi-factor solution* (Supplementary Educational Monographs, No. 48). University of Chicago Press.

References

Holzinger, K. J., and Swineford, F. (1939). *A study in factor analysis: The stability of a bi-factor solution* (Supplementary Educational Monographs, No. 48). University of Chicago Press.

Jöreskog, K. G. (1969). A general approach to confirmatory maximum likelihood factor analysis. *Psychometrika, 34*, 183–202.

Holzinger, K. J. (1944). A simple method of factor analysis. *Psychometrika, 9*, 257–262.

Examples

```
data(holzinger_swineford)
str(holzinger_swineford)

# School and grade breakdown.
table(holzinger_swineford$school, holzinger_swineford$grade)

# Jöreskog (1969) drew nine tests from the Grant-White sample; lavaan
# ships the same nine tests, for both schools, as HolzingerSwineford1939.
joreskog_subset <- subset(
  holzinger_swineford,
  school == "Grant-White",
  select = c(t1_visual_perception, t2_cubes, t4_lozenges,
             t6_paragraph_comprehension, t7_sentence,
             t9_word_meaning, t10_addition,
             t12_counting_groups_of_dots,
             t13_straight_and_curved_capitals)
)
dim(joreskog_subset)
```

---

htmt *Heterotrait-Monotrait Ratio of Correlations (HTMT)*

Description

Computes the HTMT discriminant-validity index of Henseler, Ringle, and Sarstedt (2015) for every pair of constructs: the average correlation between items of *different* constructs, divided by the geometric mean of the average correlations among items *within* each construct. Two constructs whose HTMT approaches 1 are empirically indistinguishable however cleanly the model draws them; the customary red flags are 0.85 (strict) or 0.90 (liberal). An optional bootstrap gives a one-sided upper confidence bound, the quantity actually compared against the cutoff in the validity literature.

Usage

```
htmt(data, blocks, B = 0, conf_level = 0.95, seed = NULL)
```

Arguments

`data`	A `data.frame` of item responses.
`blocks`	Named list of character vectors: each element names a construct and gives its item columns (two or more per construct, two or more constructs). The blocks are disjoint: each item is listed once, in one block, and each construct has a name of its own (see *Disjoint blocks* under *Details*).

B	Number of bootstrap resamples for the upper confidence bound, a single whole number; 0 (default) skips the bootstrap and reports the point estimates only. When positive it must be at least 200 at the default conf_level (100 at 0.90, 1000 at 0.99): the bound is one quantile of the resamples, reported only when at least 10 of them lie beyond it and never from fewer than 100. A smaller positive B is refused before anything is computed (class dmar_count_invalid). See *Bootstrap replications* under *Details* and dmar_bootstrap.
conf_level	Confidence level for the one-sided upper bound. Defaults to 0.95.
seed	Optional integer seed for the bootstrap, used locally (the caller's random number generator state is restored on exit).

Details

All correlations are Pearson, computed on pairwise-complete observations, and every one of them enters its average as an absolute value, the monotrait correlations among the items of a block as well as the heterotrait correlations between blocks. This is the ratio of arithmetic means that **semTools**' htmt() computes with htmt2 = FALSE and its default absolute = TRUE; on incomplete data the two agree only when **semTools** is given the same pairwise-complete correlations (through its sample.cov argument), since by default it deletes incomplete rows listwise, and its missing = "pairwise" option estimates the correlations through **lavaan**, whose values differ from stats::cor(use = "pairwise.complete.obs"). Absolute values make the ratio indifferent to the direction in which an item is keyed, since reversing an item changes the signs of its correlations and not their sizes. With signed correlations, a reverse keyed item that was not recoded lowers its block's average within-construct correlation, which raises the ratio, while its negative correlations with the items of the other block partly cancel the positive ones in the numerator, so the ratio would depend on how the items happen to be scored. Values near or above 1 indicate that the two item sets correlate across constructs about as strongly as within them. HTMT is a correlation-based screen, deliberately model-free; the confirmatory companion is the latent correlation between the two factors (see correction_for_attenuation and its factor-model discussion, and the phi rows of fa_confirmatory with output = "measurement").

Disjoint blocks. The ratio compares two sets of items, so the blocks are disjoint sets of distinct items under distinct names. An item listed twice in one block would enter that block's average with its correlation of 1 with itself, and its other correlations would be counted twice; an item placed in two blocks would enter the average correlation between them with its correlation of 1 with itself; and two blocks under one name could not be told apart. Each of these, and an item whose name belongs to two or more columns of data, is refused with an error of class dmar_htmt_blocks_invalid (see dmar_metadata) before any correlation is computed or any resample drawn, whatever B. The condition's reason element names the check that failed ("construct_names", "repeated", "shared", or "data_names"), and its blocks and items elements the names at issue. A ratio for item sets that overlap would be a different quantity, which this function does not compute.

A pair's ratio is undefined when a correlation it averages cannot be computed: an item with no variance, or two items never observed together, as in a planned missing data design whose forms give the items of two blocks to different respondents. That pair's htmt (and ci_upper) is NA, with a warning that names it, and the other pairs are reported; the call stops only when no pair is defined.

The bootstrap, when requested (B > 0), resamples the rows of data with replacement B times and recomputes every pairwise HTMT on each resample; the reported ci_upper is the conf_level empirical quantile of each pair's bootstrap distribution, a one-sided upper percentile bound (Efron

& Tibshirani, 1993). That bound is the only interval offered, matching how the validity literature uses HTMT (the question is whether the ratio credibly exceeds the cutoff); no two-sided or bias-corrected and accelerated (BCa) variant is provided. The ratio of each resample is the one reported, with every correlation an absolute value, so a resample that draws a negative correlation within a block enters with its size. Bootstrap results vary from run to run; supply `seed` for reproducibility.

Bootstrap replications. The bound of each pair is the `conf_level` quantile of its `B` resampled ratios, so it is run only when `B` puts at least 10 resamples beyond it (200 at the 95 percent level). A replication in which the HTMT of some pair defined on the data is undefined (an item that is constant in the resample, or two items never observed together) is dropped and counted, never replaced, and the bound is computed from the rest. The result records `B` (requested), `B_used` (the number the bound rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the bound of every pair is `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the result is not one to report. See `dmar_bootstrap` for the rule and its reasons.

Value

A `data.frame` (class `dmar_tbl`) with one row per construct pair and the same four columns whatever `B`: `construct_1`, `construct_2`, `htmt`, and `ci_upper`, the one-sided `conf_level` bootstrap percentile bound, which is `NA` when `B = 0` (no bootstrap was run). Both are `NA` for a pair whose ratio is undefined on the data (see Details). With `B > 0` the table also carries the attributes `conf_level`, `B` (requested), `B_used` (the number of resamples the bound was computed from), `n_dropped`, `B_required`, and `bootstrap_status` (see *Bootstrap replications* under Details).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Henseler, J., Ringle, C. M., & Sarstedt, M. (2015). A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the Academy of Marketing Science, 43*(1), 115–135. doi:10.1007/s1174701404038

See Also

`average_variance_extracted` for the convergent side of the validity ledger; `reliability_omega` for the composite reliability of each block; `correction_for_attenuation` for the latent correlation route.

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
# Two clean constructs and their six items.
set.seed(113)
n <- 300
f1 <- rnorm(n); f2 <- 0.3 * f1 + sqrt(1 - 0.09) * rnorm(n)
d <- data.frame(
  a1 = .8 * f1 + rnorm(n, 0, .6), a2 = .7 * f1 + rnorm(n, 0, .7),
  a3 = .6 * f1 + rnorm(n, 0, .8),
  b1 = .8 * f2 + rnorm(n, 0, .6), b2 = .7 * f2 + rnorm(n, 0, .7),
  b3 = .6 * f2 + rnorm(n, 0, .8))
h <- htmt(d, blocks = list(A = c("a1", "a2", "a3"),
                           B = c("b1", "b2", "b3")))
h

# The broom verbs: one row per construct pair.
generics::tidy(h)
generics::glance(h)

# The upper confidence bound, which is the quantity the validity
# literature compares against 0.85 or 0.90, comes from a bootstrap
# that recomputes every pairwise ratio on each of B resamples and
# fills the ci_upper column, NA above. B = 500 keeps the example
# quick; a claim about discriminant validity deserves the bound at
# B = 10000 rather than the point estimate alone.
htmt(d, blocks = list(A = c("a1", "a2", "a3"),
                      B = c("b1", "b2", "b3")),
     B = 500, seed = 113)
```

---

icc *Intraclass Correlation Coefficients With Confidence Intervals*

Description

Computes one or more of the six standard intraclass correlation coefficients (ICC) of Shrout and Fleiss (1979) for an $n \times k$ matrix of ratings (rows = subjects, columns = raters/measurements), along with the F-distribution-based confidence interval for each.

Usage

```
icc(x, type = "ICC(2,1)", conf_level = 0.95)
```

Arguments

x	An $n \times k$ numeric matrix or `data.frame`: each row is a subject (target) and each column is a rater (or repeated measurement). Missing values are not supported; remove or impute first. SPSS user-defined missing values, which `haven::read_sav(user_na = TRUE)` keeps as values of a `haven_labelled_spss` column, are missing values too (as SPSS and **haven**'s `is.na()` treat them), so a column carrying them is refused in the same way rather than analyzed as ratings.
type	Which ICC variant(s) to return. Aliases include `"1"`, `"2"`, `"3"` (single rater versions of types 1, 2, 3) and `"1k"`, `"2k"`, `"3k"` (average-of-k-raters versions). The full names `"ICC(1,1)"`, `"ICC(2,1)"`, `"ICC(3,1)"`, `"ICC(1,k)"`, `"ICC(2,k)"`, `"ICC(3,k)"` are also accepted, as is `"all"`. Vectors are accepted; a row is returned for each type.
conf_level	Confidence level for the interval (default `0.95`).

Details

The six variants follow Shrout and Fleiss's (1979) classification:

- `ICC(1,1)`: one-way random model, single rater. Each subject is rated by a different (random) rater; appropriate when raters are not crossed with subjects.
- `ICC(1,k)`: one-way random model, average of k raters.
- `ICC(2,1)`: two-way random model, single rater, absolute agreement. Subjects $\times$ raters fully crossed; both effects random.
- `ICC(2,k)`: two-way random model, average of k raters, absolute agreement.
- `ICC(3,1)`: two-way mixed model, single rater, consistency. Raters are fixed (the only ones of interest); rates differences in mean across raters are not penalized.
- `ICC(3,k)`: two-way mixed model, average of k raters, consistency.

Confidence intervals follow the F-distribution-based formulas of Shrout and Fleiss (1979, pp.\ 425–426); the `ICC(2,1)` interval uses their asymmetric approximate-*df* formulation.

Value

A `data.frame` (class `dmar_tbl`) with one row per requested ICC type and columns `type`, `value` (point estimate), `ci_lower`, `ci_upper`, `F_value`, `df_1`, `df_2`, and `p_value` (for the implicit null $H_0 : \mathrm{ICC} = 0$).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Shrout, P. E., & Fleiss, J. L. (1979). Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin, 86*(2), 420–428.

McGraw, K. O., & Wong, S. P. (1996). Forming inferences about some intraclass correlation coefficients. *Psychological Methods, 1*(1), 30–46. doi:10.1037/1082989X.1.1.30

See Also

ci_r, descriptives

Other reliability: cohen_kappa(), diagnosis_agreement, fleiss_kappa(), reliability(), reliability_H(), reliability_alpha(), reliability_kr20(), reliability_omega(), reliability_omega_categor

Examples

```
# Shrout & Fleiss (1979), Table 1: 4 raters, 6 targets.
shrout_fleiss <- matrix(
  c(9, 2, 5, 8,
    6, 1, 3, 2,
    8, 4, 6, 8,
    7, 1, 2, 6,
    10, 5, 6, 9,
    6, 2, 4, 7),
  nrow = 6, byrow = TRUE,
  dimnames = list(paste0("Target_", 1:6),
                  paste0("Judge_",  1:4))
)

icc(shrout_fleiss, type = "all")

# Just the two-way mixed-model single-rater consistency ICC:
icc(shrout_fleiss, type = "ICC(3,1)")
```

---

icc_lmer *Intraclass Correlation From a Fitted* lme4 *Mixed-Effects Model*

Description

Reads the variance-component decomposition off a fitted lmer model and returns the implied intraclass correlation (ICC) for the specified grouping factor along with a Bonett (2002) Fisher-L-transform confidence interval, in tidy long form. It is the mixed-effects model counterpart of the ANOVA-based variance_components_mls and var_icc.

Usage

```
icc_lmer(fit, group = NULL, conf_level = 0.95)
```

Arguments

fit	A fitted lmer model (class lmerMod) with at least one random-effects grouping factor.
group	Character name of the grouping factor whose ICC is wanted. If NULL (default), the first random-effects grouping factor is used. For three-level models, supply the target level explicitly.
conf_level	Confidence level. Default 0.95.

Details

Definition. For a two-level model with random intercept by group j and within-group residual,

$$\rho \;=\; \frac{\sigma_b^2}{\sigma_b^2 + \sigma_w^2},$$

read directly from `VarCorr(fit)`. For three-level models, the user specifies which level (`group`) provides the variance contribution; the denominator is the total variance summed across all variance components.

Bonett (2002) CI. The CI is built on the Fisher-style L-transformation

$$L \;=\; \tfrac{1}{2}\log\left(\frac{1+(k-1)\rho}{1-\rho}\right),$$

with variance $k/(2(k-1)(n-2))$, where k is the average cluster size and n is the number of clusters. Back- transformation keeps the bounds in $[0, 1]$.

Limitations. The Bonett CI is built on a balanced / approximately balanced approximation; for severely unbalanced designs the profile likelihood CI from `confint(fit, ...)` is preferable.

Value

A `data.frame` with rows for the point estimate of the ICC, the variance components (between-group and residual), the implied total variance, the Bonett (2002) CI lower/upper limits, and the cluster-level effective sample size used in the CI.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G. (2002). Sample size requirements for estimating intraclass correlations with desired precision. *Statistics in Medicine, 21*(9), 1331–1335. doi:10.1002/sim.1108

Donner, A. (1986). A review of inference procedures for the intraclass correlation coefficient in the one-way random effects model. *International Statistical Review, 54*(1), 67–82.

Snijders, T. A. B., & Bosker, R. J. (2012). *Multilevel analysis: An introduction to basic and advanced multilevel modeling* (2nd ed.). Sage.

See Also

`icc`, `var_icc`, `variance_components_mls`, `ss_aipe_icc`, `lmer`

Other agreement and measurement: `R2_mixed_effects()`, `content_validity_index()`, `gwet_ac()`, `krippendorff_alpha()`, `limits_of_agreement()`, `lin_ccc()`, `variance_components_mls()`

Other mixed models: `R2_mixed_effects()`, `R2_mixed_effects_decomposition()`, `manova_split_plot()`, `mixed_anova()`, `ss_aipe_mixed_effects()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_power_mixed_effects()`, `ss_power_split_plot_anova()`

Examples

```
# Twenty groups of six, with a group-level standard deviation of 0.7 on
# top of within-group noise with a standard deviation of 1, so the data
# generating ICC is 0.7^2 / (0.7^2 + 1) = 0.329. The estimate below comes
# with a wide interval: 20 clusters is not many, and the interval is what
# keeps that fact visible.
set.seed(113)
n_grp <- 20; n_per <- 6
grp <- factor(rep(1:n_grp, each = n_per))
y   <- rnorm(n_grp * n_per, 0, 1) + rep(rnorm(n_grp, 0, 0.7), each = n_per)
d <- data.frame(y, grp)
fit <- lme4::lmer(y ~ 1 + (1 | grp), data = d)
icc_lmer(fit)
```

---

irt_grm *Graded Response Model for Ordered Categorical Items*

Description

Estimates Samejima's (1969) graded response model for a set of ordered categorical items (Likert items, symptom severity ratings, rubric scored performance items) and reports the item parameters in two columns: `a`, the item's discrimination, in the normal ogive metric by default or the logistic metric on request (`metric`), and `b`, the boundary location (the threshold, in item response theory terms) of each of the item's category boundaries, the value of the latent trait θ, on the scale of a standard normal trait, at which a response above that boundary has probability .5. The model is fitted as the single-factor categorical factor analysis model it provably is (Takane and de Leeuw, 1987), using **lavaan**'s categorical estimator on the polychoric correlations, and the solution is then converted to the normal ogive (or logistic) item response theory parameterization. A researcher who already fits confirmatory factor analysis models therefore gets item response theory item parameters without adopting a second estimation engine, and the two analyses of the same items stay in one modeling tradition.

Usage

```
irt_grm(
  data,
  items = NULL,
  estimator = "WLSMV",
  metric = c("normal_ogive", "logistic"),
  rmsea_conf_level = 0.9
)
```

Arguments

`data`	A `data.frame` or matrix of ordered item responses, one column per item, coded with integer category values (for example 1, 2, 3, 4, 5). Rows with any missing value on the analyzed items are listwise-deleted. Each item must have at least 2

and at most 20 distinct observed categories; a column with more than 20 distinct values, or with non-integer values, is treated as continuous and rejected.

`items` Optional character vector naming the columns of `data` to analyze. Defaults to `NULL`, which uses every column. At least 3 items are required.

`estimator` Character; the **lavaan** estimator for categorical data. The choices differ in the weight matrix applied to the polychoric correlations and in whether the test statistic is corrected. `"WLSMV"` (the default) is diagonally weighted least squares with a mean- and variance-adjusted test statistic (Muthén, 1984; Muthén, du Toit, & Spisic, 1997), the standard estimator for ordered categorical items; `"WLSM"` applies the mean adjustment only. `"DWLS"` is the same diagonal-weight estimator with no correction. `"WLS"` uses the full weight matrix (the asymptotic distribution free approach; Browne, 1984), which is unstable unless the sample is large relative to the number of thresholds. `"ULS"`, unweighted least squares, uses an identity weight matrix, an option worth considering in small samples where even the diagonal weights are noisy; `"ULSMV"` and `"ULSM"` add the corrected test statistics. When in doubt keep the default.

`metric` Which discrimination metric to report in the `a` column: `"normal_ogive"` (default) or `"logistic"`. Both are always computed and the one not reported in `a` is attached as an attribute, so the returned table has the same columns and the same number of rows either way. The boundary locations `b` are identical under the two metrics.

`rmsea_conf_level`
Confidence level of the RMSEA interval in the `"fit_measures"` attribute (its `rmsea.ci.lower` and `rmsea.ci.upper`, with the level in `rmsea.ci.level`). Defaults to 0.90, the conventional level, whose limits decide the tests of close and not close fit at .05 (see `sem_fit_measures`), and the level lavaan uses; `rmsea_conf_level = 0.95` gives a 95 percent interval. The table itself has no intervals.

Details

One model, two parameterizations. Samejima's (1969) graded response model and the single-factor categorical factor analysis model of Muthén (1984) are the same model written in different parameterizations; Takane and de Leeuw (1987) proved the equivalence, and Kamata and Bauer (2008) give the algebra item by item. Each observed response X_i is a categorization of a latent continuous response variate X_i^* at thresholds τ_{ik}, and $X_i^* = \lambda_i\theta + \varepsilon_i$ with θ standard normal and X_i^* standardized. Fitting that model on the polychoric correlations and converting the solution gives the normal ogive graded response model directly. For item i with standardized loading λ_i and standardized thresholds τ_{ik},

$$a_i = \frac{\lambda_i}{\sqrt{1-\lambda_i^2}}, \qquad b_{ik} = \frac{\tau_{ik}}{\lambda_i}.$$

The boundary response function is

$$P_{ik}^*(\theta) = \Phi[a_i(\theta - b_{ik})]\,,$$

the probability of responding above boundary k, with $P_{i0}^*(\theta) \equiv 1$ and $P_{iK}^*(\theta) \equiv 0$; the probability of the individual category is the difference of adjacent boundary functions, $P_{ik}(\theta) = P_{i,k-1}^*(\theta) - P_{ik}^*(\theta)$. When $a_i > 0$, that is, for an item keyed in the same direction as the rest of the scale, P_{ik}^*

is monotone increasing in θ, the boundary locations of the item are ordered, $b_{i1} < b_{i2} < \cdots$, and b_{ik} is the value of θ at which the probability of responding above boundary k reaches 0.50. An item keyed in the opposite direction has $\lambda_i < 0$, hence $a_i < 0$ and boundary locations that run from high to low; see the two paragraphs on direction below.

The direction of the latent variable. A single-factor model fixes θ only up to its direction. Relabeling θ as $-\theta$ changes the sign of every loading and leaves the fitted model, the thresholds, and every fit measure exactly as they were, so it is a renaming of the latent direction rather than a different model. The thresholds are untouched because τ_{ik} cuts the item's own latent response variate X_i^*, which the relabeling does not move; the sign change therefore passes straight through to $a_i = \lambda_i/\sqrt{1-\lambda_i^2}$ and to $b_{ik} = \tau_{ik}/\lambda_i$, both of which change sign. **lavaan** returns whichever direction its starting values point toward, and for a scale that contains a reverse-keyed item that direction can turn on something as incidental as the order of the columns. The solution is therefore put in a fixed direction before it is converted: if the standardized loadings sum to a negative number the whole factor is flipped, so that θ runs in the direction the scale as a whole measures. The result is the same table no matter how the columns are ordered. Whether the flip was applied is recorded on the `"factor_sign_flipped"` attribute. The **lavaan** object on the `"fit"` attribute is the fit as **lavaan** produced it, so when a flip was applied its loadings carry the opposite sign to the `lambda` column.

Reverse-keyed items. An item whose loading is still negative after the direction is fixed is keyed opposite to the rest of the scale, which is a property of the item rather than an artifact of the sign indeterminacy. Its discrimination is negative and its boundary locations run from high to low, so it does not satisfy the graded response model as written above and its parameters do not belong on the same scale as the others. Such items are named in a warning. Reverse score them (for example `x <- (min(x) + max(x)) - x`) and refit; that puts the item in the direction the rest of the scale measures and restores $a_i > 0$ and the ordering $b_{i1} < b_{i2} < \cdots$.

The two discrimination metrics and the constant 1.702. The conversion above puts a_i in the normal ogive metric, where the boundary function is a normal cumulative distribution function. The item response theory literature more often writes the graded response model with a logistic boundary function, and the two agree closely once the logistic argument is stretched by a scaling constant: $|\Phi(x) - \Psi(1.702x)| < 0.01$ for every x, where Ψ is the standard logistic cumulative distribution function. The value 1.702 is the constant that minimizes that maximum discrepancy (Haley, 1952; see Camilli, 1994, for the history), so $a_i(\text{logistic}) = 1.702\, a_i(\text{normal ogive})$ and software that reports logistic slopes (for example **mirt** and the classical two parameter logistic tradition) gives values about 1.7 times larger for the same items. The scaling multiplies the slope and leaves the location alone, so b_{ik} does not depend on the metric.

Estimation and what to expect. lavaan estimates the thresholds and the polychoric correlations, then fits the single-factor model to those correlations by (diagonally) weighted least squares. This is limited information estimation: it uses the univariate and bivariate margins of the response table, whereas marginal maximum likelihood (the usual item response theory approach, as in **mirt**) uses the full response pattern likelihood. The two are consistent for the same population parameters and agree closely in practice, but they are different estimators and will not return identical numbers on a finite sample. Limited information estimation scales well to many items and brings the whole apparatus of factor analysis fit assessment (CFI, TLI, RMSEA) along with it; the fit measures, the RMSEA interval at `rmsea_conf_level` among them, are returned on the `"fit_measures"` attribute and the **lavaan** object itself on `"fit"`, so any **lavaan** accessor can be applied to the result.

The model is unidimensional by construction. A standardized loading at or beyond one is an improper (Heywood) solution: the implied discrimination is infinite and the conversion is not inter-

pretable. That case is flagged with a warning rather than silently returned as a number.

This function requires **lavaan** to be installed.

Value

A `data.frame` (class `dmar_tbl`) with one row per item and category boundary and the columns

`item` Item name, taken from the column name.

`factor` Name of the latent variable, the same for every row in this unidimensional model.

`category` Boundary index k, running from 1 to one fewer than the item's number of categories.

`lambda` Standardized factor loading λ_i of the item's latent response variate on the factor, repeated across the item's boundaries.

`tau` Standardized threshold τ_{ik} of the factor analysis parameterization: the cut on the item's latent response variate X_i^*, not the item response theory threshold `b`.

`a` The item discrimination a_i, the slope parameter of the item's boundary response functions (the slope of each function at its boundary location is proportional to it), repeated across the item's boundaries. It is in the normal ogive metric under the default `metric = "normal_ogive"` and in the logistic metric, 1.702 times the normal ogive value, under `metric = "logistic"`; the `"metric"` attribute records which.

`b` The boundary location b_{ik} of category boundary k of the item, the item response theory threshold of that boundary: the value of θ, in the metric of a standard normal latent trait (mean 0, standard deviation 1), at which the probability of responding above boundary k is .5. It is the same under either `metric`.

The attributes are `"fit"` (the fitted **lavaan** object), `"fit_measures"` (the full named numeric vector from `lavaan::fitMeasures`, unrounded, with the RMSEA interval at `rmsea_conf_level`), `"rmsea_conf_level"` (that level), `"metric"` (the reported discrimination metric), `"estimator"`, `"n_categories"` (named integer vector of the number of observed categories per item), `"N"` (the analyzed sample size), `"factor_sign_flipped"` (a single logical recording whether the direction of the latent variable was reversed to satisfy the sign convention described in Details), and whichever of `"a_logistic"` or `"a_normal_ogive"` was not reported in the `a` column (a named numeric vector, one element per item).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*(1), 62–83. doi:10.1111/j.20448317.1984.tb00789.x

Camilli, G. (1994). Teacher's corner: Origin of the scaling constant *d* = 1.7 in item response theory. *Journal of Educational and Behavioral Statistics, 19*(3), 293–295. doi:10.3102/10769986019003293

Haley, D. C. (1952). *Estimation of the dosage mortality relationship when the dose is subject to error* (Technical Report No. 15). Applied Mathematics and Statistics Laboratory, Stanford University.

Kamata, A., & Bauer, D. J. (2008). A note on the relation between factor analytic and item response theory models. *Structural Equation Modeling, 15*(1), 136–153. doi:10.1080/10705510701758406

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika, 49*(1), 115–132.

Muthén, B., du Toit, S. H. C., & Spisic, D. (1997). *Robust inference using weighted least squares and quadratic estimating equations in latent variable modeling with categorical and continuous outcomes*. Unpublished technical report.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph Supplement, 34*(4, Pt. 2), 1–97.

Takane, Y., & de Leeuw, J. (1987). On the relationship between item response theory and factor analysis of discretized variables. *Psychometrika, 52*(3), 393–408.

Wirth, R. J., & Edwards, M. C. (2007). Item factor analysis: Current approaches and future directions. *Psychological Methods, 12*(1), 58–79. doi:10.1037/1082989X.12.1.58

See Also

cfa_1 (the same single-factor model reported in the factor analysis parameterization), reliability_omega_categorical (reliability for the same class of items), cfa.

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
# Six five-category items generated from a known graded response model.
set.seed(113)
n <- 800
a_pop <- c(1.2, 0.9, 1.5, 1.0, 1.3, 1.1)
b_pop <- rbind(c(-1.6, -0.6, 0.3, 1.2), c(-1.4, -0.4, 0.5, 1.5),
               c(-1.8, -0.7, 0.2, 1.1), c(-1.2, -0.2, 0.7, 1.6),
               c(-1.5, -0.5, 0.4, 1.3), c(-1.3, -0.3, 0.6, 1.4))
theta <- rnorm(n)
responses <- vapply(seq_along(a_pop), function(i) {
  p_star <- outer(theta, b_pop[i, ], function(z, b) pnorm(a_pop[i] * (z - b)))
  as.integer(1 + rowSums(runif(n) < p_star))
}, integer(n))
colnames(responses) <- paste0("item", seq_along(a_pop))
responses <- as.data.frame(responses)

# Item parameters in the normal ogive metric.
grm <- irt_grm(responses)
grm

# The generating discriminations, for comparison.
a_pop
```

```
# Model fit travels with the item parameters, the RMSEA with its 90
# percent interval.
attr(grm, "fit_measures")[c("cfi", "tli", "rmsea", "rmsea.ci.lower",
                            "rmsea.ci.upper", "rmsea.ci.level", "srmr")]

# The same fit reported with logistic slopes, about 1.702 times the
# normal ogive slopes above; the boundary locations do not change.
irt_grm(responses, metric = "logistic")

# A subset of the items, selected by name, is fit on its own.
irt_grm(responses, items = c("item1", "item3", "item5"))

# The boundary response function of the first item at theta = 0.
first <- grm[grm$item == "item1", ]
pnorm(first$a * (0 - first$b))
```

`irt_information` *Item and Test Information for the Graded Response Model*

Description

Evaluates the item information functions and the test information function of a graded response model on a grid of latent trait values, together with the standard error of the latent trait estimate, $SE(\theta) = 1/\sqrt{I(\theta)}$. Reliability is a single number that describes a scale at one place on the latent continuum; the information function is the same idea expressed as a function of where the respondent sits, so an item pool can be judged on where it measures precisely rather than on one global summary. Because information is additive across items, the curve also shows which items carry the precision, and over what range, which is what makes it useful for building and trimming a scale.

Usage

```
irt_information(
  discrimination,
  threshold = NULL,
  item = NULL,
  trait_value = seq(-4, 4, length.out = 81),
  grm = NULL
)
```

Arguments

`discrimination`	Discriminations (the a_i of Details), a numeric vector of positive values. Supply either one value per item (named with the item names, or in the order the items first appear in `item`) or one value per boundary row (constant within an item). Not used when `grm` is supplied.

`threshold` Boundary locations (category thresholds, the b_{ik} of Details), a numeric vector with one element per category boundary. An item with m categories has $m - 1$ boundaries, which the model requires to be in ascending order within the item. Not used when `grm` is supplied.

`item` Item labels, a character or factor vector the same length as `threshold` naming the item each boundary belongs to. When `NULL` (default), each element of `threshold` is treated as its own dichotomous item, named `item_1`, `item_2`, and so on, and `discrimination` must then have the same length as `threshold`. Not used when `grm` is supplied.

`trait_value` Latent trait values (the θ of Details) at which to evaluate the information functions. Any finite numeric vector; the default, `seq(-4, 4, length.out = 81)`, covers the range in which almost all of a standard normal trait distribution falls, in steps of 0.1.

`grm` Optionally, the result of `irt_grm()`: a `data.frame` with one row per item and category boundary and columns `item`; `a`, the item discrimination, read as `discrimination`; and `b`, the boundary location (the item response theory threshold) of each category boundary on the θ scale, read as `threshold` (a `category` column, when present, orders the boundaries within an item). The column `a` is taken to be in the normal ogive metric, unless the table's `"metric"` attribute is `"logistic"`, as `irt_grm(metric = "logistic")` records it; that `a` is divided by 1.702 before it is used. The column `b` is the same in either metric. Supply this or the parameters, not both.

Details

For item *i* with discrimination a_i (`discrimination`) and ordered boundary locations $b_{i1} < b_{i2} < \cdots < b_{i,m-1}$ (`threshold`) for *m* categories, at latent trait value θ (`trait_value`), the normal ogive graded response model of Samejima (1969) defines the boundary response function

$$P^*_{ik}(\theta) = \Phi[a_i(\theta - b_{ik})],$$

the probability of responding *above* boundary *k*, that is, in any category higher than the *k*th, with the conventions $P^*_{i0} = 1$ and $P^*_{im} = 0$. The category response function is the difference of adjacent boundary functions,

$$P_{ik}(\theta) = P^*_{i,k-1}(\theta) - P^*_{ik}(\theta),$$

and differentiating with respect to θ gives

$$P'_{ik}(\theta) = a_i\{\phi[a_i(\theta - b_{i,k-1})] - \phi[a_i(\theta - b_{ik})]\},$$

where ϕ is the standard normal density and the density terms vanish at the two extreme categories (there is no b_{i0} and no b_{im}). Item information is

$$I_i(\theta) = \sum_{k=1}^{m} \frac{[P'_{ik}(\theta)]^2}{P_{ik}(\theta)},$$

test information is $I(\theta) = \sum_i I_i(\theta)$, and the standard error of the maximum likelihood estimate of θ is $SE(\theta) = 1/\sqrt{I(\theta)}$.

Two properties make the curve worth reading. Information is additive across items, so an item's contribution can be read off directly and a pool can be assembled to cover a targeted range. And

the reciprocal relation to the squared standard error means the peak of the curve locates where the scale estimates the trait most precisely, reported here as the `"trait_value_max_information"` attribute.

For a dichotomous item the model reduces to the two parameter normal ogive, whose information has the closed form

$$I_i(\theta) = \frac{a_i^2 \phi[a_i(\theta - b_i)]^2}{\Phi[a_i(\theta - b_i)]\{1 - \Phi[a_i(\theta - b_i)]\}},$$

which the general expression above reproduces; that identity is one of the tests of this function.

The category probabilities underflow to zero for θ far from every boundary, where the ratio $(P')^2/P$ would be $0/0$. A category whose probability is not strictly positive contributes zero to the sum, which is the limit the ratio approaches, so the returned information is finite and nonnegative on any grid, however extreme, and is never `NaN`. In the regime where $(P')^2$ underflows but P does not, the ratio is formed as $\exp[2 \log |P'| - \log P]$ so the contribution is kept rather than flushed to zero. Where two boundaries of an item coincide, the category between them has probability zero everywhere and, by the same guard, contributes nothing.

The parameters are in the normal ogive metric, which is what `irt_grm()` returns by default. The logistic metric used by much of the item response theory software scales the discrimination by approximately 1.702 (Camilli, 1994); a logistic a is put on the normal ogive scale by dividing by that constant. The two metrics give information functions that are proportional in shape but not equal in value, so a cross-software comparison is a comparison of curves, not of numbers.

Value

A `data.frame` (class `dmar_tbl`) with one row per value of `trait_value` and columns:

`trait_value` The latent trait value, as supplied.

`test_information` Test information at that value, the sum of the item information functions.

`se` The standard error of the latent trait estimate, $1/\sqrt{I(\theta)}$. It is `Inf` where test information is zero, which is the correct statement that the items carry no information there.

The result carries these attributes:

`"item_information"` A numeric matrix of item information with `trait_value` in the rows (row names are the `trait_value` values) and items in the columns (column names are the item names). Its row sums are `test_information`.

`"item"` The item names, in the order they appear in the columns of `"item_information"`.

`"discrimination"` The discrimination used for each item, a numeric vector named by item.

`"threshold"` The boundary locations used, a numeric vector in item order and, within an item, in ascending order, named by the item each boundary belongs to.

`"trait_value_max_information"` The value of `trait_value` at which test information peaks on the supplied grid (the first such value if there are ties). It is a grid value, not the result of an optimization, so a finer `trait_value` locates the peak more sharply.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Baker, F. B., & Kim, S.-H. (2004). *Item response theory: Parameter estimation techniques* (2nd ed.). Marcel Dekker.

Camilli, G. (1994). Teacher's corner: Origin of the scaling constant $d = 1.7$ in item response theory. *Journal of Educational and Behavioral Statistics, 19*(3), 293–295. doi:10.3102/10769986019003293

Embretson, S. E., & Reise, S. P. (2000). *Item response theory for psychologists*. Lawrence Erlbaum.

Lord, F. M. (1980). *Applications of item response theory to practical testing problems*. Lawrence Erlbaum.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph Supplement, 34*(4, Pt. 2), 1–97.

See Also

`plot_irt_information` for the curve, `reliability_omega` for the single-number companion.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
# Three items: a five-category rating item and two dichotomous items.
# The discriminations are named, so they are matched to the item labels.
info <- irt_information(
  discrimination = c(mood_1 = 1.4, mood_2 = 0.9, mood_3 = 1.1),
  threshold = c(-1.5, -0.5, 0.5, 1.5, 0.0, 0.8),
  item = c(rep("mood_1", 4), "mood_2", "mood_3")
)
head(info)

# Where does this three-item set measure most precisely?
attr(info, "trait_value_max_information")

# Each item's contribution; the rows sum to the test information.
head(attr(info, "item_information"))

# A dichotomous item matches the two parameter normal ogive closed form.
one <- irt_information(discrimination = 1.5, threshold = 0.25,
                       trait_value = c(-1, 0, 1))
z <- 1.5 * (c(-1, 0, 1) - 0.25)
1.5^2 * dnorm(z)^2 / (pnorm(z) * (1 - pnorm(z)))
one$test_information
```

`is_orthogonal_set` *Check Whether a Set of Contrasts Is Mutually Orthogonal*

Description

Tests whether every pair of columns in a contrast-coefficient matrix is orthogonal under either the equal-n convention $\sum_i c_{ik}c_{ij} = 0$ or the unequal-n convention $\sum_i c_{ik}c_{ij}/n_i = 0$ (Maxwell, Delaney, & Kelley, 2027, Sec. 4.10; Kirk, 2013). Also checks that each column sums to zero (the contrast property).

Usage

```
is_orthogonal_set(contrasts, n_per_group = NULL, tol = 1e-08)
```

Arguments

`contrasts`	A numeric $a \times m$ matrix or `data.frame`, where a is the number of groups and m is the number of contrasts.
`n_per_group`	Optional integer vector of length a giving the per- group sample sizes. If supplied, the unequal-n convention is used; otherwise the equal-n convention is assumed.
`tol`	Numerical tolerance for declaring orthogonality. Default `1e-8`.

Details

Equal-n. Two contrasts $\mathbf{c}, \mathbf{d}$ on a groups of equal size are orthogonal iff $\sum_{i=1}^{a} c_i d_i = 0$.

Unequal-n. With sample sizes $n_1, \ldots, n_a$, the orthogonality condition that yields uncorrelated sample contrasts is $\sum_{i=1}^{a} c_i d_i / n_i = 0$.

Useful for design checks. Before performing planned comparisons or partitioning the omnibus sums of squares, the user typically wants confirmation that the chosen contrast set is orthogonal so that its component SS sum to the omnibus SS.

Value

A `data.frame` with rows for the overall orthogonality flag (`1` = all pairs orthogonal, `0` = not), the contrast-sum-to-zero flag, the number of contrasts tested, and one row per pairwise dot-product, named by contrast pair.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kirk, R. E. (2013). *Experimental design: Procedures for the behavioral sciences* (4th ed.). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Sec. 4.10.)

See Also

effects_coding, helmert_coding, ci_scheffe

Other design utilities: design_consequences(), design_effect(), effects_coding(), helmert_coding(), orthogonal_polynomial()

Examples

```
# 1. Two orthogonal contrasts on a 4-group design (equal n):
cmat <- cbind(
  c_linear  = c(-3, -1,  1,  3),
  c_quad    = c( 1, -1, -1,  1)
)
is_orthogonal_set(cmat)

# 2. Same contrasts under unequal sample sizes:
is_orthogonal_set(cmat, n_per_group = c(10, 8, 12, 9))

# 3. Non-orthogonal pair:
cmat_bad <- cbind(
  c_diff_1  = c( 1, -1,  0,  0),
  c_diff_2  = c( 1,  0, -1,  0)
)
is_orthogonal_set(cmat_bad)
```

---

krippendorff_alpha *Krippendorff's α Inter-Rater Agreement*

Description

Computes Krippendorff's (1980, 2004, 2011) α, a chance-corrected inter-rater agreement coefficient. Unlike cohen_kappa (two raters; nominal categories, or ordinal ones through weights) or fleiss_kappa (multiple raters, nominal categories, no missing ratings), Krippendorff's α supports any number of raters, missing values, and any of four levels of measurement (nominal, ordinal, interval, ratio) via a user-specified distance metric.

Usage

```
krippendorff_alpha(
  ratings,
  level = c("nominal", "ordinal", "interval", "ratio"),
  conf_level = 0.95,
  boot = FALSE,
  B = 1000L,
  seed = NULL
)
```

Arguments

`ratings`	A units × raters matrix (or `data.frame`). Rows = units of analysis; columns = raters. `NA` entries are allowed.
`level`	One of `"nominal"` (default), `"ordinal"`, `"interval"`, or `"ratio"`; controls the distance metric used to compute disagreement.
`conf_level`	Confidence level for the bootstrap CI. Default `0.95`.
`boot`	Logical. If `TRUE`, fills the interval rows with a bootstrap percentile CI. The default `FALSE` runs no bootstrap and leaves those rows `NA` (much faster).
`B`	Number of bootstrap resamples when `boot = TRUE`, a single whole number. Default `1000L`, because every resample recomputes the coincidences over all pairable values. The fewest accepted is 400 at the default `conf_level` (200 at 0.90, and 2000 at 0.99, which the default does not meet); a smaller `B` is refused before anything is computed (class `dmar_count_invalid`). See *Bootstrap replications* under *Details* and `dmar_bootstrap`.
`seed`	Optional integer seed for reproducibility of the bootstrap. Default `NULL`, under which the resamples are drawn from, and advance, the current random number stream; supply an integer for reproducibility, which is set for the call only, the caller's random number state being restored when it returns.

Details

Coefficient. Krippendorff's α is

$$\alpha \;=\; 1 - \frac{D_o}{D_e},$$

where D_o is the observed disagreement (average squared distance over all within-unit pairs of ratings, scaled by the number of pairable values), and D_e is the expected disagreement (average squared distance over all between-unit pairs). The metric used in the squared distance depends on `level`:

- *nominal*: $d(a, b) = \mathrm{I}(a \neq b)$
- *ordinal*: distance based on cumulative rank counts
- *interval*: $d(a, b) = (a - b)^2$
- *ratio*: $d(a, b) = ((a - b)/(a + b))^2$

CI. The CI is by case-resampling bootstrap over units (rows): the rows of `ratings` are resampled with replacement `B` times and α is recomputed on each resample, so units are the sampling unit and the rater panel is treated as fixed. Only the percentile interval is offered: the limits are the empirical quantiles of the bootstrap estimates (Efron & Tibshirani, 1993); there is no bias-corrected and accelerated (BCa) variant. When the coefficient is undefined for the data themselves (every value alike), no bootstrap is run and the interval rows stay `NA`. No closed-form sampling variance is in general use for Krippendorff's alpha across its measurement levels and missing data patterns, so the bootstrap is the interval Krippendorff recommends (Krippendorff, 2011; Hayes & Krippendorff, 2007). `B = 1000L` typically gives a stable CI to two decimal places. The bootstrap is opt-in (`boot = FALSE` by default, which returns the point estimate alone and is much faster); ask for it whenever the coefficient is being reported rather than explored, since a point estimate on its own says nothing about how precisely α is determined.

Bootstrap replications. The two limits are quantiles of the `B` replicated values of α, so the interval is run only when `B` puts at least 10 resamples beyond each limit (400 at the 95 percent level). A replication in which α is undefined (a resample without enough pairable values, or one whose values are all alike, so that the expected disagreement is zero) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, and `B_used` also as a row, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the two limit rows are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the result is not one to report. See `dmar_bootstrap` for the rule and its reasons. Bootstrap results vary from run to run; supply `seed` for reproducibility.

Interpretation. α ranges from $-D_e/D_o$ (perfect disagreement) through 0 (chance level) to 1 (perfect agreement). Report the coefficient with its confidence interval and judge it against the reliability the application requires; Krippendorff (2004) discusses how that judgment depends on the cost of acting on unreliable data.

Value

A `data.frame` (class `dmar_tbl`) with the same seven rows whatever `boot`: the point estimate $\hat{\alpha}$ (`krippendorff_alpha`), the observed disagreement D_o (`D_observed`), the expected disagreement D_e (`D_expected`), the number of pairable values (`n_pairable`), the lower and upper bootstrap CI limits (`ci_lower`, `ci_upper`), and `B_used`, the number of resamples that returned a finite value and so entered the interval. The last three are `NA` when `boot = FALSE`, since no bootstrap was run. With `boot = TRUE` the table also carries the `conf_level` attribute (when an interval was formed) and the attributes `B`, `B_used`, `n_dropped`, `B_required`, and `bootstrap_status` (see *Bootstrap replications* under Details).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Hayes, A. F., & Krippendorff, K. (2007). Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures, 1*(1), 77–89. doi:10.1080/19312450709336664

Krippendorff, K. (1980). *Content analysis: An introduction to its methodology*. Sage.

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology* (2nd ed.). Sage.

Krippendorff, K. (2011). Computing Krippendorff's alpha-reliability. *Departmental Papers (ASC)*, Annenberg School for Communication, University of Pennsylvania.

See Also

cohen_kappa, fleiss_kappa, icc

Other agreement and measurement: R2_mixed_effects(), content_validity_index(), gwet_ac(), icc_lmer(), limits_of_agreement(), lin_ccc(), variance_components_mls()

Examples

```
# 1. Nominal ratings, 4 raters, 12 units (from Krippendorff 2011 Tab. 1):
ratings <- matrix(c(
  1, 2, 3, 3, 2, 1, 4, 1, 2, NA, NA, NA,
  1, 2, 3, 3, 2, 2, 4, 1, 2, 5,  NA, 3,
  NA, 3, 3, 3, 2, 3, 4, 2, 2, 5,  1,  NA,
  1, 2, 3, 3, 2, 4, 4, 1, 2, 5,  1,  NA
), nrow = 12, ncol = 4)
krippendorff_alpha(ratings, level = "nominal")

# 2. Interval ratings:
set.seed(113)
r1 <- rnorm(30, 0, 1)
r2 <- r1 + rnorm(30, 0, 0.3)
krippendorff_alpha(cbind(r1, r2), level = "interval")

# The percentile bootstrap interval for the same ratings, which
# recomputes alpha on each of B resamples of the units and fills the
# ci_lower, ci_upper, and B_used rows. B = 400 is the fewest
# resamples a 95 percent interval is reported from and keeps the
# example quick; a reported interval deserves the default B = 1000.
krippendorff_alpha(cbind(r1, r2), level = "interval",
                   boot = TRUE, B = 400L, seed = 113)
```

---

kurtosis *Bias-Corrected Sample Excess Kurtosis*

Description

Computes the sample excess kurtosis of a numeric vector using the bias-corrected (SAS/SPSS Type 2) formula. Excess kurtosis measures tailedness relative to the normal distribution: zero matches a normal, positive values indicate heavier tails ("leptokurtic"), negative values indicate lighter tails ("platykurtic").

Usage

```
kurtosis(x, na_rm = TRUE)
```

Arguments

x	A numeric vector.
na_rm	Logical. If TRUE (the default), missing values are removed before computation. If FALSE, the result is NA when x contains any NA.

Details

The reported value is

$$\hat{\gamma}_2^{(2)} = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)},$$

where s is the (divisor-$n-1$) sample standard deviation. Subtracting $3(n-1)^2/((n-2)(n-3))$ makes the expected value of the statistic 0 for a sample from a normal distribution; that is, *excess* kurtosis is reported (rather than "raw" kurtosis, whose value for a normal distribution is 3).

Three formulas. Three measures of sample kurtosis are in use, as there are three of sample skewness (see *Three formulas* in `skewness`). With $m_r = \sum_{i=1}^{n}(x_i - \bar{x})^r/n$, they are $g_2 = m_4/m_2^2 - 3$ (Type 1), $G_2 = ((n+1)g_2 + 6)(n-1)/((n-2)(n-3))$ (Type 2, the value above), and $b_2 = (g_2 + 3)(1 - 1/n)^2 - 3$ (Type 3). G_2 and b_2 are each g_2 multiplied by a factor and shifted by a constant, both of which depend only on n, and the three agree as n grows. `kurtosis()` reports G_2, which is also the kurtosis that `descriptives` reports.

Value

A single numeric value: the bias-corrected sample excess kurtosis, or `NA_real_` when fewer than four non-missing observations are available or when the sample standard deviation is zero.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Joanes, D. N., & Gill, C. A. (1998). Comparing measures of sample skewness and kurtosis. *The Statistician, 47*(1), 183–189. doi:10.1111/14679884.00122

See Also

`skewness`, `descriptives`

Other descriptive statistics: `descriptives()`, `skewness()`

Examples

```
# Normal data: excess kurtosis near zero.
set.seed(113)
kurtosis(rnorm(1000))

# Heavy-tailed data: positive excess kurtosis.
kurtosis(rt(1000, df = 4))

# The classic 1:5 example: bias-corrected excess kurtosis = -1.2.
kurtosis(1:5)
```

`limits_of_agreement` *Limits of Agreement (Bland-Altman) With Confidence Intervals on the Limits*

Description

Computes the limits of agreement (LoA) of Bland and Altman (1986, 1999) between two methods of measurement applied to the same units, along with the Carkeet (2015) exact confidence intervals on the LoA themselves. The CIs treat the two limits either as a pair (the default), so that the confidence statement holds for both limits jointly, or individually, one limit at a time. Both constructions replace the approximate normal CIs originally given by Bland and Altman (1999), which are too narrow at small n. The short alias `loa()` calls the same function.

Usage

```
limits_of_agreement(
  x,
  y,
  coverage = 0.95,
  conf_level = 0.95,
  method = c("pair", "individual")
)

loa(x, y, coverage = 0.95, conf_level = 0.95, method = c("pair", "individual"))
```

Arguments

`x`, `y`	Paired numeric vectors of equal length (e.g., method A and method B applied to the same units).
`coverage`	Probability content of the limits of agreement. Default `0.95` (the conventional 95% LoA). The mean difference is bracketed by $\pm z_{(1+\text{coverage})/2}$ standard deviations of the differences.
`conf_level`	Confidence level for the CIs on the LoA themselves. Default `0.95`.
`method`	How the CIs on the LoA are constructed, following Carkeet (2015): `"pair"` (the default) treats the two limits as a pair, so that the confidence statement holds for both limits jointly; `"individual"` treats each limit separately. See Details for when each is appropriate.

Details

Definition. For paired observations (x_i, y_i), the Bland-Altman limits of agreement are

$$\text{LoA}_{\pm} \;=\; \bar{d} \pm k \cdot s_d,$$

where $d_i = y_i - x_i$, $\bar{d}$ is the mean of the differences, s_d is their SD, and $k = z_{(1+\text{coverage})/2}$ (for 95% coverage, $k = 1.96$). The LoA are population intervals: they describe the range within

which approximately *coverage%* of *individual* differences are expected to lie if the differences are normally distributed.

CIs on the LoA themselves. The sample LoA are random variables, and Carkeet (2015) derived exact CIs for them in two forms, selected by `method`. The choice turns on what the agreement claim is about.

The pair method (the default). A Bland-Altman analysis is usually read as a statement about the range of agreement as a whole, the span from the lower to the upper LoA within which about *coverage%* of individual differences lie. That claim involves both limits at once, so the confidence statement should hold for the two limits jointly; this is the treatment Carkeet (2015) recommends for most situations. Writing $k_t(F)$ for the exact two-sided normal tolerance factor with confidence F and content equal to `coverage` (Odeh, 1978), the CI on the upper LoA is

$$\left[\, \bar{d} + k_t(\alpha/2)\, s_d,\;\; \bar{d} + k_t(1-\alpha/2)\, s_d \,\right],$$

with α equal to one minus `conf_level`, and the CI on the lower LoA is its mirror image about $\bar{d}$. The joint confidence statement runs through the probability content of the two symmetric intervals: with confidence `conf_level`, the interval between the inner pair of bounds, $\bar{d} \pm k_t(\alpha/2)\, s_d$, captures less than *coverage%* of the population of differences, while the interval between the outer pair, $\bar{d} \pm k_t(1-\alpha/2)\, s_d$, captures more, so the pair of population limits is bracketed simultaneously. In the Bland and Altman (1986) example that Carkeet reanalyzes ($n = 17$, $\bar{d} = -2.1$, $s_d = 38.8$), the pair bounds are -2.1 ± 57.81 (inner) and -2.1 ± 119.60 (outer).

The individual method. When a single limit carries the substantive question (for example, only the upper limit matters because only differences in one direction are clinically consequential), each limit can be treated on its own. Writing $t_{p,\,n-1}(\delta)$ for the p quantile of the noncentral *t* distribution with $n-1$ degrees of freedom and noncentrality parameter $\delta = k\sqrt{n}$, the exact CI on the upper LoA is

$$\left[\, \bar{d} + \frac{s_d}{\sqrt{n}}\, t_{\alpha/2,\,n-1}(\delta),\;\; \bar{d} + \frac{s_d}{\sqrt{n}}\, t_{1-\alpha/2,\,n-1}(\delta) \,\right],$$

and the CI on the lower LoA uses $-\delta$ in place of δ. These intervals are asymmetric about the sample LoA, wider on the side away from the mean difference. In the worked example above, the individual CI on the upper LoA is $[48.9,\ 120.0]$. The confidence statement is per limit: each limit is covered with `conf_level` confidence separately, not both at once.

Numerical accuracy. The pair tolerance factors are computed by numerical integration of the Odeh (1978) chi square by normal integral, which reproduces Carkeet's Table 2 to all four printed decimals. The individual quantiles are those of the noncentral *t* distribution, computed exactly. R's `qt()` with a noncentrality parameter is a normal approximation once $\delta = k\sqrt{n}$ passes 37.62 (from $n = 369$ at 95% coverage, 524 at 90%, and 214 at 99%), and limits from it are then too far from the mean difference by as much as 0.0011 standard deviations of the differences at 95% coverage (0.0030 at 99%), less as n grows (0.00036 at $n = 1000$). There the quantiles are computed by integrating over the sampling distribution of the standard deviation; below it they are `qt()`'s.

Caveats. The LoA construction assumes (i) the differences d_i are approximately normally distributed, and (ii) the difference does not systematically depend on the magnitude of the measurement (proportional bias). Both should be checked, the second by plotting d_i against $(x_i + y_i)/2$; a non-flat relationship indicates that a single set of LoA is inappropriate.

Value

A `data.frame` with rows for the mean difference, the SD of differences, the lower and upper LoA (`loa_lower`, `loa_upper`), and the lower / upper CI bounds on each LoA. The rows are the

same under both methods; the construction that produced the CI bounds is recorded in the `method` attribute.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bland, J. M., & Altman, D. G. (1986). Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet, 327*(8476), 307–310.

Bland, J. M., & Altman, D. G. (1999). Measuring agreement in method comparison studies. *Statistical Methods in Medical Research, 8*(2), 135–160. doi:10.1191/096228099673819272

Carkeet, A. (2015). Exact parametric confidence intervals for Bland-Altman limits of agreement. *Optometry and Vision Science, 92*(3), e71–e80. doi:10.1097/OPX.0000000000000513

Odeh, R. E. (1978). Tables of two-sided tolerance factors for a normal distribution. *Communications in Statistics - Simulation and Computation, 7*(2), 183–201.

See Also

`lin_ccc`

Other agreement and measurement: `R2_mixed_effects()`, `content_validity_index()`, `gwet_ac()`, `icc_lmer()`, `krippendorff_alpha()`, `lin_ccc()`, `variance_components_mls()`

Examples

```
# 1. Two methods that agree well; the CIs treat the limits as a
#    pair (the default):
set.seed(113)
method_a <- rnorm(40, mean = 100, sd = 15)
method_b <- method_a + rnorm(40, mean = 0, sd = 3)
limits_of_agreement(method_a, method_b)

# 2. Each limit treated individually, for when a single limit
#    carries the substantive question:
limits_of_agreement(method_a, method_b, method = "individual")

# 3. 90% LoA with 95% CIs on the limits, each limit again treated
#    individually (coverage and conf_level work the same way under
#    the pair method, whose tolerance factors take longer to compute):
limits_of_agreement(method_a, method_b, method = "individual",
                    coverage = 0.90, conf_level = 0.95)
```

`lin_ccc` *Lin's Concordance Correlation Coefficient*

Description

Computes Lin's (1989) concordance correlation coefficient (CCC) for a pair of vectors of paired observations, together with a confidence interval built on Lin's z-transformed standard error (Lin, 1989; see also the note in Lin, 2000). The CCC measures agreement (not merely correlation) between two methods of measurement: a CCC of 1 means perfect agreement ($y_i = x_i$ for all i), while Pearson's r would still be 1 for any straight-line relationship, even one with non-unit slope.

Usage

```
lin_ccc(x, y, conf_level = 0.95, method = "lin")
```

Arguments

`x, y`	Paired numeric vectors of equal length (e.g., the two measurement methods).
`conf_level`	Confidence level for the CI. Default `0.95`.
`method`	The confidence interval method. Currently the only option is `"lin"`, the Lin (1989) z-transformed standard error.

Details

Definition. Lin (1989) defined the CCC as

$$\rho_c \;=\; \frac{2\rho\,\sigma_x\sigma_y}{\sigma_x^2+\sigma_y^2+(\mu_x-\mu_y)^2},$$

where $\rho = \mathrm{Cor}(X,Y)$ is the Pearson correlation and the denominator is inflated by the squared mean difference and by any inequality of the two variances, so disagreement in location or scale pulls ρ_c below ρ. ρ_c factors as $\rho_c = \rho \cdot C_b$, where $C_b \in [0,1]$ is the "bias correction factor" that captures location and scale agreement, and $C_b = 1$ iff $\mu_x = \mu_y$ and $\sigma_x = \sigma_y$.

Confidence interval. The Fisher-style z-transform of the CCC, $z = \frac{1}{2}\log\{(1+\rho_c)/(1-\rho_c)\}$, has approximate variance (Lin, 1989, as corrected in Lin, 2000)

$$\mathrm{Var}(z) \;\approx\; \frac{1}{n-2}\left[\frac{(1-\rho^2)\rho_c^2}{(1-\rho_c^2)\rho^2} + \frac{2\rho_c^3(1-\rho_c)u^2}{\rho(1-\rho_c^2)^2} - \frac{\rho_c^4u^4}{2\rho^2(1-\rho_c^2)^2}\right],$$

where $u = (\mu_x-\mu_y)/\sqrt{\sigma_x\sigma_y}$. The CI is built on the z-scale and back-transformed via $\tanh$.

This variance is derived under bivariate normality, and the interval inherits that assumption. Under normality its coverage is modestly below the nominal rate in small samples (roughly 0.92 to 0.94 at n of 10 to 20 for a nominal 0.95) and approaches the nominal rate as n grows (about 0.94 at $n = 50$ for a moderate CCC). With clearly skewed data the situation is worse and more data do not repair it: with heavy-tailed or log-normal style measurements the interval can cover far below the nominal rate at any sample size (Carrasco, Jover, King, & Chinchilli, 2007). With such data, transform toward symmetry before computing the CCC, or use a bootstrap interval on $\hat{\rho}_c$.

Value

A `data.frame` with rows for the CCC point estimate, the lower and upper CI limits, and decomposition components (Pearson r, accuracy C_b, location-shift u, scale-shift v).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Carrasco, J. L., Jover, L., King, T. S., & Chinchilli, V. M. (2007). Comparison of concordance correlation coefficient estimating approaches with skewed data. *Journal of Biopharmaceutical Statistics, 17*(4), 673–684. doi:10.1080/10543400701329463

Lin, L. I.-K. (1989). A concordance correlation coefficient to evaluate reproducibility. *Biometrics, 45*(1), 255–268.

Lin, L. I.-K. (2000). A note on the concordance correlation coefficient. *Biometrics, 56*(1), 324–325. doi:10.1111/j.0006341X.2000.00324.x

See Also

`limits_of_agreement`

Other agreement and measurement: `R2_mixed_effects()`, `content_validity_index()`, `gwet_ac()`, `icc_lmer()`, `krippendorff_alpha()`, `limits_of_agreement()`, `variance_components_mls()`

Examples

```
# 1. Two methods of measuring the same quantity:
set.seed(113)
method_a <- rnorm(40, mean = 100, sd = 15)
method_b <- method_a + rnorm(40, mean = 2, sd = 5)
lin_ccc(method_a, method_b)

# 2. Compare CCC with Pearson r when there is a systematic offset:
lin_ccc(method_a, method_a + 5)$value[1:2]   # CCC < r
cor(method_a, method_a + 5)                  # Pearson r = 1
```

`manova_split_plot` *Mixed-Design Multivariate ANOVA With All Four Test Statistics*

Description

Computes the multivariate analysis of variance for a mixed-design (one between-subjects factor and one within-subjects factor), returning Wilks's Λ, Pillai's trace, Hotelling-Lawley trace, and Roy's largest root, each with the associated F-approximation, degrees of freedom, and p-value. The three effects, between-subjects (A), within-subjects (B), and the interaction ($A \times B$), are tested separately, following the multivariate approach to the split-plot design of Maxwell, Delaney, and Kelley (2027, Section 14.6). Wraps `Anova` and returns the result in the tidy DMAR style.

Usage

```
manova_split_plot(data, within, between, ss_type = 3L)
```

Arguments

`data`	A `data.frame` in wide format, one row per subject, with the within-subjects measurements in separate columns and the between-subjects factor as one additional column.
`within`	Character vector of column names holding the repeated measures values (one column per level of the within- subjects factor). Must be in the canonical level order.
`between`	Character name of the between-subjects factor column in `data`.
`ss_type`	The sum-of-squares type, passed through to `Anova`. Accepts the integers `1`, `2`, or `3` or the equivalent Roman-numeral strings `"I"`, `"II"`, or `"III"`, and defaults to `3L` (Type III). With one between-subjects factor the type matters only for the within-subjects main effect when the groups differ in size: Type III tests the unweighted grand means of the within-subjects contrasts and Type II the weighted grand means (see **Details**). The chosen type is recorded in the returned table (see **Value**). Type I (`1` or `"I"`) is not available for this mixed design, because `Anova` computes only Type II and Type III sums of squares for the multivariate repeated measures path; requesting it raises an error.

Details

The three tests. The multivariate approach replaces each subject's b repeated measures with transformed variables: their mean M, and $b - 1$ contrasts among them, the D variables. The between-subjects effect (*A*) is the one-way ANOVA of M across the a groups (Maxwell, Delaney, & Kelley, 2027, Equation 14.40), which is also the univariate approach's test of *A*. The interaction ($A \times B$) asks whether the groups differ in their means on the D variables, a one-way MANOVA of the D variables (their Equations 14.62 and 14.63). The within-subjects main effect (*B*) asks whether the grand means of the D variables are all zero, averaging over the groups (their Equation 14.54). Both within-subjects tests take as their error matrix the pooled within-group sums of squares and cross-products of the D variables, and no test depends on which $b - 1$ contrasts are formed. A test of *B* within a single group is a simple effect (their Section 14.7.4), a different question from the main effect.

Weighted or unweighted grand means. When the groups differ in size, the grand mean of a D variable can be the unweighted average of its a group means or the average weighted by group size, which is its mean over all *N* subjects; the two agree when the groups are of equal size, and the choice affects only the test of *B*. Maxwell, Delaney, and Kelley (2027, Section 14.6.9, p. 796, and Chapter 14, note 6, p. 854) generally prefer the unweighted means, which they note SAS and SPSS test by default, while allowing that the weighted means suit some studies. The sum-of-squares type selects between them. Type III (the default) tests the unweighted grand means: the between-subjects factor is coded with sum-to-zero contrasts, whatever `options("contrasts")` says, so that the intercept of the regression of each D variable on the groups is the unweighted mean of its group means, and the hypothesis matrix is $a\tilde{n}\bar{D}_U\bar{D}_U^\top$, where $\bar{D}_U$ holds the unweighted grand means and $\tilde{n}$ is the harmonic mean of the group sizes (their notes 10 and 14). Type II tests the weighted grand means $\bar{D}$, with the hypothesis matrix $N\bar{D}\bar{D}^\top$ (their note 11). Under R's default treatment coding,

the intercept would instead be the first group's means, and its Type III test the simple effect of *B* in the first group (their Equation 14.66). With one between-subjects factor, the two types give the same tests of *A* and of the interaction, and with equal group sizes the same test of *B* as well. Type I, the sequential decomposition, is not available here: `Anova` computes only Type II and Type III for the multivariate repeated measures path. The type in force is reported in the returned table so the analysis is self-documenting.

The four statistics. For an effect with H and E hypothesis- and error-cross-products matrices:

- Wilks's $\Lambda = \det(E)/\det(E+H)$
- Pillai's trace $V = \mathrm{tr}(H(E+H)^{-1})$
- Hotelling-Lawley trace $T_0^2 = \mathrm{tr}(HE^{-1})$
- Roy's largest root $\theta = \lambda_1(HE^{-1})$

When to use which. For many data sets the four statistics are likely to reach the same conclusion about statistical significance, but that agreement is not guaranteed, and without more information about the population than is usually available none of them can be said to be best. Wilks's Λ is historically the most widely used, and there is some evidence that Pillai's trace is the most robust of the four (Maxwell, Delaney, & Kelley, 2027, Section 14.6.11). Roy's largest root uses only the largest eigenvalue of HE^{-1}, the largest ratio of hypothesis to error variation over linear combinations of the transformed variables. The four statistics give the same *F* whenever $s = 1$, where s is the smaller of the number of transformed variables and the effect's degrees of freedom for each of them (Maxwell, Delaney, & Kelley, 2027, Section 14.6.11): always for *A* and *B*, and for the interaction when either factor has two levels.

Dependency. Requires the **car** package on CRAN.

Value

A `data.frame` (class `dmar_tbl`) with rows for each of the three effects crossed with each of the four multivariate statistics, plus one trailing row recording the sum-of-squares type. Columns: `effect`, `statistic_name`, `statistic_value`, `F_approx`, `df_1`, `df_2`, `p_value`. The final row has `effect == "sum_of_squares_type"` and carries the chosen type (`2` or `3`) in its numeric `statistic_value`; its remaining numeric columns are `NA`, so the `statistic_value` column stays numeric. The table prints through the `dmar_tbl` display layer: whole numbers such as the degrees of freedom without a decimal part, the statistics to 3 decimal places, and *p*-values to 4 decimal places (one below 0.0001 prints as "< 0.0001"); the stored values keep full precision. `tidy()` gives one row per row of the table, its `term` joining `effect` and `statistic_name` and its `estimate` the `statistic_value`, so the final row is the term `"sum_of_squares_type:NA"` (its `statistic_name` is `NA`) with the type as its estimate; `glance()` gives one row, the number of rows (`n_terms`) with `conf_level` and `B_used` `NA`, since no interval or bootstrap is formed (see `dmar_tidiers`). `as.data.frame()` returns the table as a plain `data.frame`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 14, Section 14.6, on the multivariate approach to split-plot designs, and Chapter 7, Sections 7.9 and 7.10, on weighted and unweighted marginal means.)

Rencher, A. C., & Christensen, W. F. (2012). *Methods of multivariate analysis* (3rd ed.). Wiley. doi:10.1002/9781118391686

See Also

Anova, manova, anova_within_two_way

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Other mixed models: R2_mixed_effects(), R2_mixed_effects_decomposition(), icc_lmer(), mixed_anova(), ss_aipe_mixed_effects(), ss_aipe_mixed_effects_sensitivity(), ss_power_mixed_effects(), ss_power_split_plot_anova()

Examples

```
# Two groups of ten measured at three times. Both groups start at the
# same place and rise over time, and group B rises twice as fast, so the
# data generating means differ in slope as well as in level.
set.seed(113)
n_per <- 10
d <- data.frame(
  subject = factor(1:(2 * n_per)),
  group   = factor(rep(c("A", "B"), each = n_per)),
  t1      = c(rnorm(n_per, 0,   1), rnorm(n_per, 0,   1)),
  t2      = c(rnorm(n_per, 0.4, 1), rnorm(n_per, 0.8, 1)),
  t3      = c(rnorm(n_per, 0.8, 1), rnorm(n_per, 1.6, 1))
)

# Rows are the between-subjects effect (labeled A), the within-subjects
# main effect (labeled B), and their interaction, each with all four
# multivariate criteria, followed by a row recording the sum-of-squares
# type. The multivariate tests make no sphericity assumption, which is
# what recommends them over the univariate repeated measures F and its
# epsilon corrections.
manova_split_plot(d, within = c("t1", "t2", "t3"), between = "group")

# With six subjects left in group A, the types differ in the B rows
# alone: Type III (the default) averages the two groups' changes over
# time equally, and Type II weights each group by its size.
d_unequal <- d[-(1:4), ]
manova_split_plot(d_unequal, within = c("t1", "t2", "t3"),
                  between = "group")
```

```
manova_split_plot(d_unequal, within = c("t1", "t2", "t3"),
                  between = "group", ss_type = 2)
```

`mauchly_test` *Mauchly's Test of Sphericity for a One-Way Within-Subjects Design*

Description

Tests the null hypothesis that the covariance matrix of the orthonormal contrasts among the k repeated measurements is proportional to the identity (the *sphericity* assumption underlying univariate repeated measures F-tests).

Usage

```
mauchly_test(x, id = NULL, time = NULL, outcome = NULL)
```

Arguments

`x`	Either an $n \times k$ numeric matrix or `data.frame` (rows = subjects, columns = repeated measurements); *or* a long-format `data.frame` together with `id`, `time`, and `outcome` column names.
`id`	Column name in `x` identifying the subject when `x` is in long format (`NULL` otherwise).
`time`	Column name in `x` identifying the within-subjects factor level when `x` is in long format (`NULL` otherwise).
`outcome`	Column name in `x` identifying the dependent variable when `x` is in long format (`NULL` otherwise).

Details

Sphericity is the assumption that the variances of all pairwise differences among the k levels are equal, equivalently, that the covariance matrix Σ_C of any orthonormal set of $k-1$ contrasts among the levels is proportional to the identity. Mauchly's (1940) test statistic is

$$W = \frac{\det(\hat{\Sigma}_C)}{\left(\mathrm{tr}(\hat{\Sigma}_C)/(k-1)\right)^{k-1}},$$

and the chi square approximation

$$X^2 = -\,m\,\log W \quad \text{with } m = (n-1) - \frac{2(k-1)^2 + (k-1) + 2}{6\,(k-1)}$$

has approximately $(k-1)k/2 - 1$ degrees of freedom under H_0. The reported p-value uses Box's (1949) second-order correction, a weighted combination of the chi square tails on $(k-1)k/2 - 1$ and $(k-1)k/2 + 3$ degrees of freedom, which improves the first-order approximation in small samples; this matches `mauchly.test`.

When sphericity is rejected, the univariate F test is liberal; correct using the Greenhouse-Geisser, Huynh-Feldt, or lower-bound epsilon adjustments via `epsilon_corrections` or directly via `anova_within`.

The test is only defined for $k \geq 3$; with $k = 2$, sphericity is trivially true and the function returns `W = 1, p = 1`.

Value

A one-row `data.frame` with columns `W` (Mauchly's statistic), `statistic` (the chi square approximation), `df`, `p_value`, `n_subjects`, `n_levels`, and `method`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Mauchly, J. W. (1940). Significance test for sphericity of a normal n-variate distribution. *Annals of Mathematical Statistics, 11*(2), 204–209.

Box, G. E. P. (1949). A general distribution theory for a class of likelihood criteria. *Biometrika, 36*(3/4), 317–346.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 11 for sphericity in within-subjects designs.)

See Also

`epsilon_corrections`, `anova_within`

Other within-subjects analysis: `anova_within()`, `anova_within_two_way()`, `epsilon_corrections()`, `pairwise_within()`, `plot_trajectories_fitted()`

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# Wide-format example: simulated within-subjects data with 4 levels.
set.seed(113)
Y <- matrix(rnorm(20 * 4), nrow = 20)
mauchly_test(Y)

# Long-format example using built-in nlme::Orthodont (4 ages per subject).
mauchly_test(nlme::Orthodont, id = "Subject", time = "age",
             outcome = "distance")
```

`measurement_alignment` *Approximate Measurement Invariance by Factor Alignment*

Description

Estimates the group factor means and factor variances that make the measurement parameters as nearly invariant as possible across groups, following the alignment method of Asparouhov and Muthén (2014). The invariance ladder in `measurement_invariance` asks whether loadings and intercepts are exactly equal across groups. With many groups that hypothesis is essentially never true, the ladder stalls at configural invariance, and the comparison of factor means the researcher wanted never happens. Alignment takes the configural solution as given and searches for the group factor means and variances that concentrate the noninvariance in a few parameters instead of spreading it thinly over many, so that factor means stay comparable without claiming that exact invariance holds. Requires **lavaan**.

Usage

```
measurement_alignment(
  data,
  items,
  group,
  model = NULL,
  alignment = c("fixed", "free"),
  estimator = "ML",
  n_starts = 10,
  seed = NULL,
  epsilon = 0.01
)
```

Arguments

`data`	A `data.frame` holding the items and the grouping variable.
`items`	Character vector naming the indicator columns of `data`. Three or more items are required, and each must be numeric. Items that cannot be read from `data` (a name that is not a column, a name given twice, fewer than three) are refused with an error of class `dmar_items_invalid` (see `dmar_metadata`).
`group`	Single character string naming the grouping column of `data`. Two or more groups are required.
`model`	Optional **lavaan** model syntax for the measurement model, for the case where the default single factor over `items` is not what is wanted (a residual covariance, for example). The syntax must define exactly one latent variable and its indicators must be exactly `items`. The factor is standardized (mean 0, variance 1) in every group by this function, so do not set its scale in the syntax. When `NULL` (the default) the model `f =~ item1 + item2 + ...` is used.

`alignment` Identification rule for the metric of the factor, either `"fixed"` (the default) or `"free"`. Under `"fixed"` the first group's factor mean is held at 0 and its factor variance at 1. Under `"free"` the group factor means are constrained to average 0 and the group factor standard deviations to have a geometric mean of 1, so no group serves as the reference. The choice sets the metric the factor means and variances are reported on; see Details.

`estimator` Estimator passed to **lavaan** for the configural model. Defaults to `"ML"`; `"MLR"` supplies the robust corrections.

`n_starts` Number of starting values for the optimizer, a positive integer (default 10). The first start sets every factor mean to 0 and every factor variance to 1, the second uses a median based heuristic, and any remaining starts are random. The simplicity function has local minima, so several starts is the default rather than a refinement.

`seed` Optional integer seed for the random starts. Defaults to `NULL`, which draws the random starts from the current random number stream and so advances that stream. When an integer is supplied the seed is set locally and the caller's state is restored on exit.

`epsilon` Smoothing constant $\epsilon > 0$ inside the component loss function, default 0.01. Smaller values push the loss closer to $\sqrt{|x|}$ and make the surface harder to optimize; larger values smooth it at the cost of blurring the distinction between a few large differences and many small ones.

Details

Alignment starts from the *configural* solution: the multiple group single factor model with every loading, intercept, and residual variance free across groups and the factor standardized (mean 0 and variance 1) in each group. Write λ^0_{gi} and ν^0_{gi} for the resulting loading and intercept of item i in group g. That solution is not unique: for any group factor means α_g and factor variances ψ_g the reparameterized measurement parameters

$$\lambda_{gi} = \lambda^0_{gi}/\sqrt{\psi_g}$$

$$\nu_{gi} = \nu^0_{gi} - \alpha_g \lambda^0_{gi}/\sqrt{\psi_g}$$

reproduce the observed means and covariances exactly and so fit the data identically. Alignment picks the member of that family whose measurement parameters are closest to invariant, by minimizing the total simplicity function

$$F = \sum_i \sum_{g_1<g_2} w_{g_1 g_2} f(\lambda_{g_1 i} - \lambda_{g_2 i}) + \sum_i \sum_{g_1<g_2} w_{g_1 g_2} f(\nu_{g_1 i} - \nu_{g_2 i})$$

over α_g and ψ_g, with weights $w_{g_1 g_2} = \sqrt{N_{g_1} N_{g_2}}$ and the component loss function

$$f(x) = \sqrt{\sqrt{x^2 + \epsilon}} = (x^2 + \epsilon)^{1/4}.$$

The fourth root is the substance of the method, not a technical detail. A squared loss would spread a fixed amount of noninvariance evenly over all the parameters, because halving two differences beats zeroing one. The fourth root is concave in $|x|$, so the marginal penalty falls as a difference

grows: the criterion prefers a solution in which most parameters agree closely and a few disagree substantially, which is exactly the pattern of approximate invariance a researcher wants to find and report. The constant ϵ only rounds off the kink of $\sqrt{|x|}$ at zero so the criterion is differentiable. One consequence is worth knowing: each of the $2I$ terms in a group pair contributes at least $\epsilon^{1/4}$, so the smallest attainable value of F is $2I\epsilon^{1/4}\sum_{g_1<g_2} w_{g_1g_2}$, reached only when every aligned parameter is exactly invariant. The achieved value is comparable across runs on the same items, groups, and ϵ, not across data sets.

Two constraints identify the solution. The scale of the factor is genuinely undetermined by F: multiplying every $\sqrt{\psi_g}$ and every α_g by the same constant leaves the aligned intercepts alone and shrinks the aligned loadings toward each other, so without a constraint the criterion is minimized by letting the factor variances run away. The location constraint plays the same role in the limiting case of exact invariance, where shifting all the factor means by a constant leaves F unchanged. Under `alignment = "fixed"` the constraints are $\alpha_1 = 0$ and $\psi_1 = 1$. Under `alignment = "free"` they are $\sum_g \alpha_g = 0$ and $\prod_g \sqrt{\psi_g} = 1$, which treats the groups symmetrically and is the more natural choice when no group is a meaningful reference. The two rules give genuinely different solutions, not a relabeling of one another, because F is not invariant to shifting the factor means.

The rule also sets the metric the answer is reported on, which matters when the estimates are compared with anything else. The constraint $\psi_1 = 1$ makes the first group's factor the common metric, so a reported factor mean of 0.4 says that group's factor mean is 0.4 of the *first group's* factor standard deviations above the first group's, and a reported factor variance of 1.3 says its factor variance is 1.3 times the first group's. Under `"free"` the common metric is the one in which the group factor standard deviations have a geometric mean of 1, and the factor means are deviations from their own average on that metric. Simulating data with known group factor means and then comparing them with the estimates requires dividing the generating means by the generating factor standard deviation of the reference group first.

The minimization runs `optim` with the BFGS method and the analytic gradient, over the factor means α_g and the log factor standard deviations $\log\sqrt{\psi_g}$, reduced to the coordinates the identification rule leaves free. The optimizer is run from `n_starts` starting values and the best solution is kept, because a single start is not safe. Two things can go wrong, and they are different problems.

The first is ordinary multimodality. When several loadings and intercepts are noninvariant the criterion has more than one local minimum, typically within a percent or so of each other in value but at different factor means. The number of distinct minima the starts reached is returned in the `"n_optima"` attribute and the value achieved from each start in `"simplicity_starts"`. More than one is a signal to raise `n_starts` and to check whether the reported solution is stable.

The second is a degenerate branch, and it is the one that bites. Sending the factor variances of every group but the reference off to infinity drives those groups' aligned loadings to zero, which flattens the loading half of the criterion; the optimizer stops there and reports convergence with factor variances of 10^{36} or `Inf`. That is not an estimate, and on real data a meaningful share of random starts finds it. A start whose factor standard deviations or factor means leave a very wide sanity range (a factor of 10^4 either way) is therefore discarded and its entry in `"simplicity_starts"` is `Inf`. If every start is discarded the function stops rather than return the degenerate solution.

Item level invariance is summarized with the R^2 measure of Asparouhov and Muthén (2014), which asks how much of the group to group variation in an item's configural parameter is accounted for by the estimated factor means and variances alone. Let $\bar\lambda_i$ and $\bar\nu_i$ be the aligned parameters of item i averaged over groups. The factor means and variances by themselves imply the configural values $\sqrt{\psi_g}\bar\lambda_i$ and $\bar\nu_i + \alpha_g\bar\lambda_i$. Write T_i for the sum over groups of the squared configural parameter of

item i and E_i for the sum over groups of its squared departure from that implied value. Then

$$R_i^2 = 1 - E_i/T_i,$$

computed separately for the loadings and for the intercepts. A value of 1 for every item on the loadings is metric invariance, and on both loadings and intercepts is scalar invariance. The complementary view is the `"item_loss"` attribute, which splits the achieved simplicity function into the contribution of each item, so the items carrying the noninvariance can be named.

The criterion adds up raw parameter differences, so items on very different measurement scales do not contribute equally: an item whose raw variance is a hundred times another's dominates the sum. When the items are not already on a common metric, put them on one (standardizing them is the simple choice) before aligning.

The method is defined for two groups and is computed here for two, but it has little to offer there. With two groups the standard invariance ladder is tractable and interpretable, and the alignment criterion has only one group pair to work with. Alignment earns its keep when the number of groups makes exact invariance implausible and the ladder uninformative.

A configural solution whose likelihood has no maximum, flat or still rising as an error variance in some group falls below zero (three items whose correlations in one group have a negative product can do it), has no loadings or intercepts to align, only the point where **lavaan**'s optimizer stopped. The function checks a configural fit with a negative error variance as `fa_bifactor` describes and stops with an error of class `dmar_no_maximum` that names the group and the item and prints the **lavaan** code of the fit, where `bounds = "pos.var"` or a different model are the choices.

Value

A `data.frame` (class `dmar_tbl`) with one row per group and columns `group` (the group label), `n` (the number of cases the configural model used in that group), `factor_mean` (the estimated α_g), and `factor_variance` (the estimated ψ_g). The rows follow the group ordering **lavaan** uses.

Attributes carry the rest of the solution:

`aligned_loadings`, `aligned_intercepts` Matrices of the aligned λ_{gi} and ν_{gi}, groups in rows and items in columns.

`configural_loadings`, `configural_intercepts` Matrices of the configural λ_{gi}^0 and ν_{gi}^0, same layout.

`simplicity_function` The achieved minimum of F.

`simplicity_starts` The value of F achieved from each starting value, in start order. A start the optimizer could not complete, or one that ended in the degenerate branch described in Details, is recorded as `Inf` and discarded.

`converged` `TRUE` when the optimizer reported convergence at the retained solution.

`n_starts` The number of starting values used.

`n_optima` The number of distinct local minima the converged starts reached.

`alignment` The identification rule used, `"fixed"` or `"free"`.

`epsilon` The smoothing constant used.

`R2_loadings`, `R2_intercepts` Named numeric vectors, one entry per item, holding the per item R^2 invariance measures defined in Details.

`R2_total` The same two measures pooled over items, as a named numeric vector with elements `loadings` and `intercepts`. This is the overall effect size of approximate invariance reported by Asparouhov and Muthén (2014).

`item_loss` Matrix with one row per item and columns `loadings`, `intercepts`, and `total`, splitting the achieved simplicity function into per item contributions.

`fit` The configural **lavaan** fit object.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Asparouhov, T., & Muthén, B. (2014). Multiple-group factor analysis alignment. *Structural Equation Modeling, 21*(4), 495–508. doi:10.1080/10705511.2014.919210

Marsh, H. W., Guo, J., Parker, P. D., Nagengast, B., Asparouhov, T., Muthén, B., & Dicke, T. (2018). What to do when scalar invariance fails: The extended alignment method for multi-group factor analysis comparison of latent means across many groups. *Psychological Methods, 23*(3), 524–545. doi:10.1037/met0000113

Muthén, B., & Asparouhov, T. (2014). IRT studies of many groups: The alignment method. *Frontiers in Psychology, 5*, Article 978. doi:10.3389/fpsyg.2014.00978

Muthén, B., & Asparouhov, T. (2018). Recent methods for the study of measurement invariance with many groups: Alignment and random effects. *Sociological Methods & Research, 47*(4), 637–664. doi:10.1177/0049124117701488

Robitzsch, A. (2025). *sirt: Supplementary item response theory models*. R package version 4.2-133. https://CRAN.R-project.org/package=sirt

See Also

`measurement_invariance` for the exact invariance ladder alignment is meant to rescue; `cfa_1` for the single group measurement model; `htmt` for discriminant validity of the same items.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
# Five groups that differ in factor mean and factor variance, with two
# deliberately noninvariant measurement parameters (the loading of item
# 3 in group 2 and the intercept of item 5 in group 4). The first group's
# factor standard deviation is 1, which is the metric the default "fixed"
# rule reports on, so the estimates compare directly with the generating
# values below.
set.seed(113)
```

```
n_g <- c(400, 450, 380, 500, 420)
factor_mean <- c(0, 0.30, -0.50, 0.80, 0.20)
factor_sd   <- c(1, 1.20,  0.80, 1.10, 0.90)
Lambda <- matrix(0.8, nrow = 5, ncol = 6)
Nu     <- matrix(1.0, nrow = 5, ncol = 6)
Lambda[2, 3] <- 0.3
Nu[4, 5]     <- 1.8
d <- do.call(rbind, lapply(1:5, function(g) {
  eta <- rnorm(n_g[g], factor_mean[g], factor_sd[g])
  x <- sapply(1:6, function(i)
    Nu[g, i] + Lambda[g, i] * eta + rnorm(n_g[g], 0, 0.6))
  data.frame(x, cohort = paste0("cohort_", g))
}))
names(d)[1:6] <- paste0("x", 1:6)

out <- measurement_alignment(d, items = paste0("x", 1:6),
                             group = "cohort", seed = 113)
out                                  # recovered means and variances
attr(out, "R2_loadings")             # item 3 stands out
attr(out, "R2_intercepts")           # item 5 stands out
attr(out, "item_loss")

# Holzinger and Swineford's verbal tests across four groups formed by
# crossing school with sex. The raw tests are on very different scales,
# so they are standardized first (see Details).
data(holzinger_swineford)
hs <- holzinger_swineford
hs$school_sex <- interaction(hs$school, hs$sex, sep = ", ")
verbal <- c("t5_general_information", "t6_paragraph_comprehension",
            "t7_sentence", "t8_word_classification", "t9_word_meaning")
hs[verbal] <- scale(hs[verbal])
ma <- measurement_alignment(hs, items = verbal, group = "school_sex",
                            seed = 113)
ma

# The broom verbs: one row per group, and the alignment summary.
generics::tidy(ma)
generics::glance(ma)
```

`measurement_invariance` *Measurement Invariance Across Groups*

Description

Fits the standard ladder of multiple group invariance models for a measurement model of any number of factors and reports the comparison of its rungs in one table: configural invariance (same pattern, all parameters free by group), metric (equal loadings; required before comparing relations involving the factor), scalar (equal loadings and intercepts; required before comparing factor

means), and strict (equal residual variances as well; required before comparing observed-score variances). With ordered indicators a thresholds rung comes first, because thresholds rather than intercepts carry the location information (Wu & Estabrook, 2016); with dichotomous indicators the two are not separately identified, so that rung is folded into metric. Each rung is tested against the previous with a likelihood ratio test, scaled when the estimator in force carries a robust test, which is what declaring ordered items arranges, and the practical-fit changes (delta CFI, delta RMSEA) are reported alongside, since with large samples the chi square will flag trivial differences (Cheung & Rensvold, 2002, suggest delta CFI of about -.01 as a red flag). The fitted **lavaan** objects come back as an attribute, so score tests and partial invariance refits do not require refitting the ladder. Requires **lavaan**.

Usage

```
measurement_invariance(
  data,
  model = NULL,
  group,
  items = NULL,
  levels = NULL,
  ordered = NULL,
  estimator = "ML",
  missing = "fiml",
  group_partial = NULL,
  parameterization = c("delta", "theta"),
  ...
)
```

Arguments

data	A data.frame with the items and the grouping variable.
model	The measurement model, given either as **lavaan** model syntax (a single string, or a character vector of lines that is collapsed with newlines, such as c("visual =~ x1 + x2 + x3", "verbal =~ x4 + x5 + x6")) or as a named list mapping each factor name to a character vector of its item names, from which the syntax is built. Exactly one of model and items must be supplied. An indicator of either form that data lacks is refused with an error of class dmar_items_invalid (see dmar_metadata).
group	Single character string naming the grouping column (two or more groups).
items	Character vector (three or more) naming the indicator columns of a single factor. A convenience for the one-factor (congeneric) case of cfa_1, equivalent to model = "f =~ item_1 + item_2 + ...". Exactly one of model and items must be supplied. Items that cannot be read from data (a name that is not a column, a name given twice, fewer than three) are refused with an error of class dmar_items_invalid.
levels	Which rungs of the ladder to fit, in order: any leading subset of the ladder in force. With continuous indicators the ladder is c("configural", "metric", "scalar", "strict"); with ordered indicators it is c("configural", "thresholds",

"metric","scalar", "strict"); when every indicator is ordered and dichotomous it is c("configural", "metric", "scalar"), because neither the thresholds nor the strict rung is identified for a two-category item (see ordered below). NULL (the default) fits the whole ladder in force.

ordered — Ordered categorical (including binary) items: TRUE for every indicator in the model, or a character vector naming the ordered ones. NULL (default) treats all indicators as continuous. Declaring ordered items switches the ladder and, unless the estimator already carries the mean and variance adjusted test, switches the estimator (see estimator below). The number of observed categories per declared item is counted before the ladder is built, because a dichotomous item has a single threshold that is not separately identified from the intercept of its underlying response (Millsap & Yun-Tein, 2004; Wu & Estabrook, 2016), and its residual variance is not a free parameter under either parameterization. When every declared item is dichotomous the thresholds rung is folded into metric, and the strict rung is dropped as well unless a continuous indicator is left to carry a residual variance; when only some declared items are dichotomous, both rungs are kept and a message names those items, whose thresholds and residual variances the rungs leave untested.

estimator — Estimator passed to **lavaan**. Defaults to "ML"; use "MLR" for robust corrections, in which case the likelihood ratio tests use the scaled difference. When ordered is supplied, a maximum likelihood or generalized least squares (the "GLS" label; Browne, 1974) estimator is replaced by "WLSMV", and "DWLS" and "ULS" are replaced by "WLSMV" and "ULSMV". That second substitution changes only the test: in **lavaan** "WLSMV" is "DWLS" and "ULSMV" is "ULS" with the mean and variance adjusted statistic, which is what makes the rung-to-rung difference test asymptotically valid for ordered data. The one categorical estimator left alone is "WLS", the full weight matrix estimator of Browne (1984) and Muthén (1984), whose statistic is asymptotically chi square already and whose weight matrix a substitution would silently change.

missing — Missing data handling passed to **lavaan**: one of "fiml" (the default) or "ml" (full information maximum likelihood, which fits every case with an observed item), "listwise" (listwise deletion), "pairwise", "two_stage", or "robust_two_stage" (two-stage maximum likelihood, with normal theory or nonnormality robust standard errors and tests; lavaan's "two.stage" and "robust.two.stage" are accepted for these two). Full information maximum likelihood and two-stage estimation are not available with the categorical estimator, and two-stage estimation needs estimator = "ML"; asking for another combination is an error. The default analysis is full information maximum likelihood whenever a value of a model variable is missing and the complete data analysis otherwise, unlike fa_confirmatory, which fits complete data by FIML too: data in which no such value is missing, or none but in cases with every model variable missing (which both treatments set aside), are fitted as they stand (under FIML the robust indices of an MLR ladder would be the corrected ones of Zhang and Savalei, 2023, not the complete data ones), and incomplete data by FIML. FIML needs continuous items, estimator = "ML" or "MLR", standard errors other than se = "robust.sem", and a test other than the Satorra-Bentler kind passed through ...; when the default meets incomplete items and one of these, the call stops and names the option to choose (missing = "listwise", "pairwise" for ordered

items, `estimator = "MLR"`, or, with `estimator = "MLR"`, leaving out `test`) rather than deleting cases unasked, and an explicit `missing = "fiml"` with `se = "robust.sem"` or such a test stops too, whether or not a value is missing, since **lavaan** would fit the complete cases instead. Under FIML a group that never observes a variable of the model, or never observes two of them together, is refused before any fit (an error of class `dmar_group_coverage`). See *Details*.

`group_partial` Character vector of parameters to leave free across groups at every rung, passed to **lavaan**'s `group.partial`. Either user-supplied parameter labels or **lavaan** parameter specifications such as `"visual =~ x2"` or `"x2 ~1"`. This is the partial invariance case of Byrne, Shavelson, and Muthén (1989). `NULL` (default) constrains everything the rung calls for, as does an empty vector (`character(0)` or **lavaan**'s own default `""`), which frees nothing in **lavaan** either.

`parameterization`
Identification of ordered indicators, passed to **lavaan**: `"delta"` (the default, lavaan's) or `"theta"`. Residual variances are free parameters only under `"theta"`, so a strict rung requested with ordered items switches to `"theta"` with a message. Ignored when no item is ordered.

`...` Further arguments passed to `cfa` at every rung (for example `std.lv`, `orthogonal`, `cluster`). **lavaan**'s own `group.partial` is merged into `group_partial`, and an empty one (`NULL`, `character(0)`, or `""`) frees nothing, as in **lavaan**, so a call such as `group.partial = if (free) terms else NULL` runs either way; `group.equal`, which the ladder sets rung by rung, is an error here (choose the rungs with `levels`).

Details

The models are nested by construction, each adding equality constraints across groups to the previous. With continuous indicators the constraint sets are none (beyond the configuration), then `group.equal = "loadings"`, then `c("loadings", "intercepts")`, then `c("loadings", "intercepts", "residuals")`.

The ordered ladder differs, and the difference is substantive rather than cosmetic. For an ordered indicator the observed response is a coarsening of an underlying continuous response at a set of thresholds, and it is the thresholds, not an intercept, that locate the item on the latent scale. Constraining loadings while leaving thresholds free across groups therefore does not deliver what metric invariance is supposed to deliver, and the accepted sequence (Millsap & Yun-Tein, 2004; Wu & Estabrook, 2016) constrains thresholds first: `"thresholds"`, then `c("thresholds", "loadings")` for metric, then `c("thresholds", "loadings", "intercepts")` for scalar, then adding `"residuals"` for strict. The intercept rung is not vacuous even when every indicator is ordered: once thresholds and loadings are constrained, **lavaan** frees the underlying-response intercepts in the non-reference groups, and the scalar rung is what returns them to zero and lets the latent means be estimated instead. Residual variances, however, are free parameters only under the theta parameterization, so `parameterization` switches to `"theta"` when a strict rung is requested with ordered items.

Dichotomous items are the exception the ordered ladder has to make room for. A two-category item contributes one threshold, and that threshold, the intercept of the underlying response, and its residual variance are not separately identified: the data give one proportion per group per item, which pins down a single standardized location and nothing else (Millsap & Yun-Tein, 2004; Wu & Estabrook, 2016). Constraining thresholds alone across groups is then a reparameterization rather than

a restriction, since **lavaan** frees the underlying-response intercepts by exactly as many parameters as the constraint removes, and the residual variances stay fixed at one in every group whichever parameterization is in force. So the function counts the observed categories of each declared ordered indicator before building the ladder, and a message explains what the count implies. When all of them are dichotomous the thresholds rung is folded into metric, which constrains thresholds and loadings together; the strict rung goes too when every indicator in the model was declared ordered, leaving configural, metric, scalar. A continuous indicator alongside dichotomous ones keeps the strict rung, since its residual variance is a free parameter. When only some declared items are dichotomous the full ordered ladder is kept, since the polytomous items still carry testable thresholds and residual variances, and the message names the dichotomous items so their contribution to those two rungs is not overread. Every rung that is dropped is one that would have cost zero degrees of freedom.

When the estimator carries a robust test, the difference between two chi square statistics is not itself chi square distributed. `lavTestLRT` then returns the scaled difference test (the Satorra-Bentler or Satorra correction, chosen by **lavaan** to match the test in force), and that is what `delta_chi_square` and `p_value` report; the `"test"` attribute names the test used. In that case the model-level `chi_square`, `p_chi_square`, `cfi`, and `rmsea` columns are lavaan's scaled and robust versions, so `delta_chi_square` will not equal the difference of consecutive `chi_square` entries. That is a property of scaled difference testing, not an inconsistency.

One version of the test and one of the indices hold every rung, so that `delta_cfi` and `delta_rmsea` compare like with like, and the `"model_test"` and `"fit_indices"` attributes name the versions the columns hold. Where **lavaan** does not compute the robust indices at every rung, every rung falls back to one other version, and a line under the table says which. For a maximum likelihood fit of continuous items, whose robust indices estimate the population values of the ML indices with their small sample bias corrected for nonnormality, it is the standard indices, which estimate the same population values with the bias corrected as if the data were normal; the scaled indices, which put the scaled statistic into the standard formulas, estimate population values that change with the kurtosis of the data (Savalei, 2018). Beside the older mean and variance adjusted statistic (`test = "mean.var.adjusted"`, passed through `...`) **lavaan** computes no robust indices, and they are computed here, as `sem_fit_measures` computes them: they do not depend on which corrected statistic accompanies them (Savalei, 2018), so they equal those of the Satorra-Bentler statistic of the same fit, and a line under the table says that DMAR computed them. For ordered items, whose standard statistic is not the basis of an index, it is the scaled indices: **lavaan** leaves the robust indices `NA` at the rungs that constrain the thresholds under the default delta parameterization, as in the ordered example below. A ladder of continuous items fit by ULS (`estimator = "ULS"`, `"ULSM"`, or `"ULSMV"`) withholds the ML-derived robust RMSEA and incremental indices (Savalei, 2018). Invariance to a common change of units alone does not establish validity outside their derivation. Eligible scaled indices are reported with a qualification: they do not estimate the same population discrepancy as ML-based indices, and ML cutoffs do not automatically apply. The standard ULS statistic is in the units of squared covariances, so its CFI and RMSEA are withheld too. At a rung that restricts the means, as the scalar and strict rungs do through the intercepts, the ULS fit function adds the squared mean residuals, in the squared units of the items, to the squared covariance residuals, in the squared units of the covariances, so the statistic, the loadings, and every index change with the units: on the first example under `"ULSM"`, the scalar CFI is .9885 for the raw scores and .9840 with every score divided by 10, and the scalar delta CFI -.00005 and -.0046, a judgment against the cutoff of Cheung and Rensvold (2002) that turns on the units. The `cfi` and `rmsea` columns of such a rung are `NA`, and the `chi_square` column and the difference test are kept, since their reference distribution does not depend on the units, though their values do. A ladder fit by

`estimator = "ULS"` alone is tested by Browne's residual based statistic, which has no units, and its `cfi` and `rmsea` columns, formed from that statistic, are reported at the rungs whose ULS estimates do not change with the units, those that leave the means free. At a rung that restricts the means through parameters the covariances share, as the scalar and strict rungs do through the loadings, which enter the means through the latent means, the ULS estimates change with the units, and so do Browne's statistic evaluated at them and the indices formed from it: on the first example, the scalar CFI is .8874 for the raw scores and .8861 with every score divided by 10, and the scalar delta CFI -.0212 and -.0225. Those columns are `NA` there too, and the `chi_square` column is kept. **lavaan**'s difference test of a ULS ladder without a corrected statistic refers the difference of the ULS statistics, in the units of the squared covariances and not chi square distributed, to the chi square distribution, so `delta_chi_square` and `p_value` are `NA` at every rung, and `"test"` is `NA`; `"ULSM"` and `"ULSMV"` give a scaled difference test. `estimator = "MLR"`, which does not depend on the units, is the estimator for nonnormal continuous items. When **lavaan** reports no scaled statistic at every rung (beside a bootstrap test, `test = "bollen.stine"`), the `chi_square` column and the difference test are the standard ones, and the attributes say so. A saturated rung (zero degrees of freedom and a statistic of zero) fits exactly in every version and does not decide, since **lavaan** can leave its robust CFI `NA` there from a ratio of zeros, or not compute it. Its statistic is zero in every version, and with it the model term of the CFI, $T - c \cdot df$, whatever the scaling constant c (which is 0/0 there), so its `cfi` is 1 in every version; its RMSEA, $\sqrt{(T - c \cdot df)/(N \cdot df)}$, is 0/0, which no version defines, and it keeps **lavaan**'s 0, the convention **lavaan** applies in every version to a model that fits exactly, so that `delta_cfi` and `delta_rmsea` at the next rung compare with exact fit. A rung with zero degrees of freedom whose statistic is not zero, as when an inequality bound binds (**lavaan**'s count of the degrees of freedom leaves the bound out), does not fit exactly: its `cfi` is formed from its statistic and decides the version of the indices (a ladder whose robust CFI **lavaan** leaves `NA` at such a rung falls back to another version at every rung, the standard indices for a maximum likelihood ladder of continuous items), while its `rmsea`, which divides by the degrees of freedom, is undefined and `NA`, as is `delta_rmsea` at the next rung, with a line under the table that says so; it does not decide the version of the test, which no version defines on zero degrees of freedom.

Which estimators carry such a test is worth being precise about, because a naive chi square difference on ordered data is not asymptotically valid. `"MLM"`, `"MLMV"`, `"MLR"`, `"WLSM"`, `"WLSMV"`, `"ULSM"`, and `"ULSMV"` carry one. `"DWLS"` and `"ULS"` do not, which is why declaring ordered items promotes them to `"WLSMV"` and `"ULSMV"`: the discrepancy function and hence the parameter estimates are untouched, and only the statistic changes. The remaining case is `"WLS"`, the full weight matrix estimator, whose statistic is asymptotically chi square under the theory of Browne (1984) and Muthén (1984) and so needs no correction; the `"test"` attribute reports the standard difference test there. The sample size that theory asks for is large, which limits full weighted least squares to large samples, but that is a separate matter from whether the difference test is the right one.

Full information maximum likelihood. With `missing = "fiml"` (the default when an item value is missing) or `"ml"`, so that the ladder uses every case with an observed item, as `fa_confirmatory` fits the same model by default, the `chi_square` column is the FIML likelihood ratio statistic (its scaled version with `estimator = "MLR"`), and the `cfi` and `rmsea` columns are **lavaan**'s robust versions, the corrected indices of Zhang and Savalei (2023) that `fa_confirmatory` reports for the same model: the complete data fit function evaluated at the FIML estimates, with the saturated FIML estimates in place of the sample moments, so that they estimate the values the indices would have had with complete data (**lavaan** uses their version 6, which assumes normality, and their version 3, which does not, with `estimator = "MLR"`). The usual indices computed from the FIML statistic move with the amount and kind of missing data (Zhang & Savalei, 2023), so their change from rung to rung is not the change complete data would show, and a cutoff for a change in fit, such as the delta

CFI of about -.01 of Cheung and Rensvold (2002), is a statement about the latter. `delta_cfi` and `delta_rmsea` therefore compare the corrected indices, and one version holds every rung: when **lavaan** cannot compute them at some rung with degrees of freedom, every rung reports its scaled indices (for a fit with a scaled statistic) or its standard ones, and the line under the table says which. **lavaan** cannot combine FIML with the Satorra-Bentler or the mean and variance adjusted statistic (`test = "satorra.bentler"` or `"mean.var.adjusted"` passed through `...`), nor with `se = "robust.sem"`: it fits the complete cases instead, with a warning. Under the default of `missing` such a ladder is fitted as it stands when no item value is missing and stops otherwise; asked for by name, FIML with one of them stops before any fit whether or not a value is missing, as in `fa_confirmatory`, and so does a ladder whose fits **lavaan** made by another treatment than the FIML asked for (it fits `estimator = "GLS"`, `"WLS"`, and `"ULS"` in two stages).

FIML estimates the means, variances, and covariances of the model's variables within each group, so every group must observe every variable, and every pair of them together, at least once. A group that does not is refused by name before any fit, in an error of class `dmar_group_coverage` that carries the variables and pairs at issue (see `dmar_metadata`); **lavaan** would stop with an internal error. `holzinger_swineford` is such a data set, by school: `t25_paper_form_board_r` and `t26_flags` were given in Grant-White only. The message names what to do: leave the variable (or one of the pair) out of the model, or an auxiliary variable out of `aux`, for this comparison, so that the ladder compares the groups on what every group observes, or, when at least two groups would remain, leave the groups at issue out of `data`. Listwise deletion is no remedy, since a group that never observes a variable, or never observes two together, has no case observed on every variable, and the listwise ladder would stop on that group instead.

Auxiliary variables passed through `...` as **lavaan**'s `aux` enter a FIML fit as saturated correlates (Graham, 2003), and **lavaan** makes the mean and variance of each one ordinary parameters of each group, which the intercepts and residuals rungs would equate across the groups; the rung tests would then test whether the auxiliary is distributed alike in the groups, which says nothing about the measurement model. They are left free at every rung, beside any `group_partial` terms, so the ladder has the degrees of freedom of the ladder without the auxiliary. On six Holzinger and Swineford tests by school with 60 sentence completion scores deleted at random and general information (whose means are 44.8 and 36.6 in the two schools) as the auxiliary, the scalar rung's difference test is 23.88 on 4 df and its delta CFI -.031 (24.03 and -.031 without the auxiliary).

Two-stage estimation. With `missing = "two_stage"` every rung is fit in two stages (Yuan & Bentler, 2000; Savalei & Bentler, 2009): the means and covariances are first estimated in each group by maximum likelihood under the saturated normal model (the EM estimates, which use every partially observed case and any auxiliary variables passed through `...` as lavaan's `aux`), and the rung is then fit to them by the complete data maximum likelihood discrepancy function. A case observed on an auxiliary variable but on none of the model's variables is read by the first stage and kept, as in `fa_confirmatory`, which says why: **lavaan**'s two-stage estimator leaves such a case out of its sample size, so every rung is computed again with its sample size in each group set to the cases the first stage read there. The statistics, the indices, the difference tests, and the fits of the `"fits"` attribute then count every case. The second stage weights each group's discrepancy by its share of the sample size, so a rung that constrains parameters across groups has estimates that depend on the groups' counts, and when those grow in different ratios the rung is estimated again; the configural rung, which constrains nothing across groups, keeps **lavaan**'s estimates. On six Holzinger and Swineford tests by school with the whole block missing for 77 pupils at random given general information (the auxiliary), 130 and 94 cases with an observed test grow to 156 and 145 read; the common intercept of lozenges at the scalar rung is 18.00 (SE .697) rather than **lavaan**'s 18.21 (SE .782), and the scalar rung's difference test is 14.97 on 4 df, against 10.86 from **lavaan**'s own

ladder, 16.46 under FIML with the same auxiliary, and 17.44 after listwise deletion. A message says how many such cases there are, and a line under the table names the sample size. **lavaan**'s `summary()` of a fit in the `"fits"` attribute counts them as used, and `lavTestLRT` warns that a difference test between such a fit and one **lavaan** made of the same data on its own sample size is invalid. The standard errors are the two-stage sandwich, and the `chi_square` column of each rung is the residual based statistic of Savalei and Bentler (2009, Eq. 8), whose rejection rate their simulations with incomplete normal data found close to the nominal level, and the statistic `fa_confirmatory` reports after two-stage estimation. With `missing = "robust_two_stage"` the standard errors are robust to nonnormality (Yuan & Bentler, 2000) and `chi_square` is lavaan's scaled statistic (Savalei & Falk, 2014). Under both, `cfi` and `rmsea` are lavaan's robust indices, the corrected indices of Zhang and Savalei (2023): their TS V2 after robust two-stage estimation, and its normal theory analog after normal theory two-stage estimation (`fa_confirmatory` explains the difference), so `delta_cfi` and `delta_rmsea` compare those, and the rung-to-rung test is the scaled difference test of Satorra and Bentler (2001), which `lavTestLRT` applies to two-stage fits. The difference is formed from the two rungs' naive statistics and their scaling, so `delta_chi_square` is not the difference of the `chi_square` entries. Two-stage estimation needs continuous items and `estimator = "ML"`; robustness to nonnormality comes from `"robust_two_stage"`, not from `estimator = "MLR"`. The residual based statistic treats every observed variable as random, so a model given as syntax that regresses on an observed variable needs `fixed.x = FALSE` passed through `...`: under `"two_stage"` the statistic is refused for a fit that conditions on such a variable, as it is for sampling weights, clustered cases, inequality constraints, and a parameter on a bound, each with a message that says what to do instead.

A rung whose constraints cost no degrees of freedom tests nothing, so `delta_chi_square` and `p_value` are `NA` when `delta_df` is zero rather than reporting a difference in chi square that is numerical noise and can come out negative. The situation is reported in a message. It arises with dichotomous items (the ladder above avoids the two rungs where it is structural) and, for instance, with `group_partial` specifications that free every parameter a rung would constrain.

Failure at a rung does not end the conversation: partial invariance (freeing the offending parameter) is the usual next step, for which `group_partial` refits the whole ladder with named parameters free, and the `"fits"` attribute gives the fitted objects to `lavTestScore` for a score test of which constraint is doing the damage. This function deliberately reports the standard ladder rather than automating modification searches.

A rung whose likelihood has no maximum, flat or still rising as an error variance in some group falls below zero, has no estimates to compare, only the point where **lavaan**'s optimizer stopped; the function checks each rung with a negative error variance as `fa_bifactor` describes and stops with an error of class `dmar_no_maximum` that names the rung, the group, and the item and prints the **lavaan** code of that rung's fit, where `bounds = "pos.var"` (also accepted through `...` here) or a different model are the choices. Three items whose correlations in one group have a negative product give such a rung when `std.lv = TRUE` is passed through `...`; under the default scaling by a marker item, **lavaan**'s optimizer does not converge on such data. A negative error variance at a maximum of the likelihood is a finite estimate, and its rung is reported.

A rung that **lavaan**'s optimizer does not converge on has no estimates to report, and the function stops with an error of class `dmar_not_converged`, the class `fa_confirmatory` gives such a fit, whose message names the rung. The condition carries the rung as `level` and the **lavaan** code that reproduces its fit as `syntax` (the model) and `lavaan_call` (the call, with `my_data` in place of the data).

Value

A tidy wide `data.frame` (class `dmar_tbl`) with one row per fitted level and columns `level` (label), `chi_square`, `df`, `p_chi_square` (exact-fit test), `cfi`, `rmsea`, and, from the second row on, the step-comparison columns `delta_chi_square`, `delta_df`, `p_value` (the likelihood ratio test against the previous rung), `delta_cfi`, and `delta_rmsea`. `delta_chi_square` and `p_value` are `NA` wherever `delta_df` is zero, since a difference test on zero degrees of freedom tests nothing. Attributes carry the non-numeric information: `"fits"`, the named list of fitted **lavaan** objects, one per level; `"estimator"`, the estimator actually used; `"ordered"`, `TRUE` when any indicator was declared ordered; `"test"`, naming the chi square difference test used (`NA` when a single rung was fit and nothing was compared, and for a ULS ladder of continuous items without a corrected statistic, which has no difference test); `"fit_indices"`, the version of the `cfi` and `rmsea` columns at every rung, `"standard"`, `"robust"`, or `"scaled"` (`"robust"` is the corrected indices of Zhang and Savalei, 2023, under FIML and two-stage estimation, and `"scaled"` the fallback of a ladder of ordered items, or fit by FIML or in two stages, with a scaled statistic whose robust indices **lavaan** did not compute at every rung); `"fit_columns"`, the lines printed under the table that name that version for a ladder fit by FIML, one that fell back from the robust indices, or one whose robust indices DMAR computed, and that say why columns of a ULS ladder of continuous items are `NA`, or why the `rmsea` of a rung with zero degrees of freedom that does not fit exactly is (`NULL` otherwise; a two-stage ladder names the version in its own lines); `"model_test"`, naming the statistic in the `chi_square` column at every rung (`"standard"`, `"scaled"`, or under the two-stage options the two-stage residual based or scaled statistic); for a two-stage ladder, `"two_stage"`, a list with the method, the lines printed under the table, and, in `rows`, the cases the first stage read in each group and **lavaan**'s count of them when an auxiliary variable was named, how many were observed on an auxiliary variable alone, whether the rungs were computed again with all of them (`corrected`), and whether the estimates of a rung that constrains across groups were too (`reestimated`); and `"model"`, the **lavaan** syntax that was fitted.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Brosseau-Liard, P. E., & Savalei, V. (2014). Adjusting incremental fit indices for nonnormality. *Multivariate Behavioral Research, 49*(5), 460–470. doi:10.1080/00273171.2014.933697

Brosseau-Liard, P. E., Savalei, V., & Li, L. (2012). An investigation of the sample performance of two nonnormality corrections for RMSEA. *Multivariate Behavioral Research, 47*(6), 904–930. doi:10.1080/00273171.2012.715252

Browne, M. W. (1974). Generalized least squares estimators in the analysis of covariance structures. *South African Statistical Journal, 8*, 1–24.

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*(1), 62–83.

Byrne, B. M., Shavelson, R. J., & Muthén, B. (1989). Testing for the equivalence of factor covariance and mean structures: The issue of partial measurement invariance. *Psychological Bulletin, 105*(3), 456–466.

Cheung, G. W., & Rensvold, R. B. (2002). Evaluating goodness-of-fit indexes for testing measurement invariance. *Structural Equation Modeling, 9*(2), 233–255. doi:10.1207/S15328007SEM0902_5

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

Meredith, W. (1993). Measurement invariance, factor analysis and factorial invariance. *Psychometrika, 58*(4), 525–543. doi:10.1007/BF02294825

Millsap, R. E. (2011). *Statistical approaches to measurement invariance*. Routledge.

Millsap, R. E., & Yun-Tein, J. (2004). Assessing factorial invariance in ordered-categorical measures. *Multivariate Behavioral Research, 39*(3), 479–515. doi:10.1207/s15327906mbr3903_4

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika, 49*(1), 115–132.

Satorra, A., & Bentler, P. M. (2001). A scaled difference chi-square test statistic for moment structure analysis. *Psychometrika, 66*(4), 507–514. doi:10.1007/BF02296192

Savalei, V. (2018). On the computation of the RMSEA and CFI from the mean-and-variance corrected test statistic with nonnormal data in SEM. *Multivariate Behavioral Research, 53*(3), 419–429. doi:10.1080/00273171.2018.1455142

Savalei, V., & Bentler, P. M. (2009). A two-stage approach to missing data: Theory and application to auxiliary variables. *Structural Equation Modeling, 16*, 477–497. doi:10.1080/10705510903008238

Savalei, V., & Falk, C. F. (2014). Robust two-stage approach outperforms robust full information maximum likelihood with incomplete nonnormal data. *Structural Equation Modeling, 21*(2), 280–302. doi:10.1080/10705511.2014.882692

Wu, H., & Estabrook, R. (2016). Identification of confirmatory factor analysis models of different levels of invariance for ordered categorical outcomes. *Psychometrika, 81*(4), 1014–1045. doi:10.1007/s1133601695060

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*, 165–200.

Zhang, X., & Savalei, V. (2023). New computations for RMSEA and CFI following FIML and TS estimation with missing data. *Psychological Methods, 28*(2), 263–283. doi:10.1037/met0000445

See Also

`cfa_1` and `fa_confirmatory` for the single-group measurement models; `reliability_omega` for the reliability of the composite the model justifies; `compare_cov_structures` for covariance-structure comparisons outside the factor model; `lavTestScore` for the score test that localizes a failed rung.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
# Do the two schools measure the verbal and the reasoning construct
# in the same way? (Holzinger & Swineford, bundled.) The measurement
```

```
# model is a named list of factors; for a single factor, name its
# indicators with items = instead, or give lavaan syntax, as the last
# example does.
data(holzinger_swineford)
hs_factors <- list(
  verbal    = c("t6_paragraph_comprehension", "t7_sentence",
                "t9_word_meaning"),
  deduction = c("t20_deduction", "t22_problem_reasoning",
                "t23_series_completion"))

# The whole ladder, configural through metric, scalar, and strict.
# Configural invariance asks whether the same pattern of loadings
# holds at both schools; metric adds the constraint that the loadings
# are equal across schools, and it is the rung that has to hold before
# a relation involving one of these factors is compared across them;
# scalar equates the intercepts as well, which comparing factor means
# requires; strict equates the residual variances. Each row from the
# second on tests its rung against the one below it.
mi <- measurement_invariance(holzinger_swineford, hs_factors,
                             group = "school")
mi

# The fitted models travel with the table, so localizing a failed rung
# costs no refitting.
names(attr(mi, "fits"))

# The broom verbs: one row per rung of the ladder, and the model-level
# summary (estimator, test flavor, fit index flavor).
generics::tidy(mi)
generics::glance(mi)

# Partial invariance frees one loading across the schools at every
# rung (Byrne, Shavelson, & Muthén, 1989), so the metric rung costs
# one degree of freedom fewer than it did above. Naming a leading
# subset of the rungs in levels = stops the ladder there, which keeps
# this and the next two examples quick; a reported analysis climbs
# the whole ladder.
measurement_invariance(holzinger_swineford, hs_factors,
                       group = "school",
                       group_partial = "verbal =~ t7_sentence",
                       levels = c("configural", "metric"))

# With the indicators coded as ordered categories, ordered = TRUE puts
# the thresholds rung first, because thresholds rather than intercepts
# carry the location information there, and makes the rung-to-rung
# tests the scaled difference tests; the "test" attribute names the
# test that was used. The ladder stops at the thresholds rung here;
# metric, scalar, and strict follow it. lavaan computes no robust fit
# indices for the thresholds rung, so both rows report the scaled ones,
# as the line under the table says.
hs_ordered <- holzinger_swineford
for (item in unlist(hs_factors, use.names = FALSE)) {
  hs_ordered[[item]] <- as.integer(cut(
```

```
    holzinger_swineford[[item]],
    breaks = quantile(holzinger_swineford[[item]],
                      c(0, .25, .5, .75, 1)),
    include.lowest = TRUE))
}
mi_ordered <- measurement_invariance(hs_ordered, hs_factors,
                                     group = "school", ordered = TRUE,
                                     levels = c("configural",
                                                "thresholds"))
mi_ordered
attr(mi_ordered, "test")

# The measurement model can also be given as lavaan syntax, and
# missing = "fiml", the default, fits the ladder by full information
# maximum likelihood when continuous items have incomplete cases; here
# twenty scores on one item are set to missing first. Naming it states
# the choice in the call. With three indicators
# the single factor is just identified in each group, so the
# configural row is saturated (zero degrees of freedom, exact fit)
# and the metric row carries the test. Under FIML the cfi and rmsea
# columns are the corrected indices of Zhang and Savalei (2023), as the
# line under the table says.
hs_missing <- holzinger_swineford
set.seed(113)
hs_missing$t7_sentence[sample(nrow(hs_missing), 20)] <- NA
measurement_invariance(
  hs_missing,
  model = "verbal =~ t6_paragraph_comprehension + t7_sentence +
           t9_word_meaning",
  group = "school", missing = "fiml",
  levels = c("configural", "metric"))
```

`mediate` *Mediation Analysis With Bootstrap Confidence Intervals*

Description

Estimates the simple mediation model, a predictor X affecting an outcome Y directly and through a mediator M, and reports the indirect, direct, and total effects with confidence intervals in one tidy table. The indirect effect ab is the product of the $X \to M$ path and the $M \to Y$ path (holding X), and its sampling distribution is skewed, which is why the default interval is the percentile bootstrap rather than a normal approximation; the bias-corrected and accelerated (BCa) bootstrap, the Monte Carlo (parametric simulation) interval, and the Sobel normal-theory interval are available for comparison. This is the analysis counterpart of the planning functions `ss_aipe_indirect_effect` and `ss_power_indirect_effect`.

Usage

```
mediate(
```

```
  data,
  x,
  m,
  y,
  covariates = NULL,
  ci_method = c("boot_percentile", "boot_bca", "monte_carlo", "sobel"),
  B = 2000,
  conf_level = 0.95,
  seed = NULL
)
```

Arguments

`data`	A `data.frame` containing the variables.
`x`, `m`, `y`	Names (single character strings) of the predictor, the mediator, and the outcome columns in `data`. Each column must be numeric (a Date, date-time, or time difference column is read as its number, and a one-column matrix, such as `scale()` returns, as its column); a predictor with two categories is supplied coded 0 and 1. Any column name is accepted as it is.
`covariates`	Optional character vector of covariate column names, entered in both the mediator and the outcome models. Numeric, logical, factor, and character columns are accepted; a factor or character covariate is coded once, on the complete cases, with R's contrasts for the session. A covariate with a single value in the complete cases adds nothing to either model and is left out with a message. A column may fill one role only.
`ci_method`	Confidence interval method for the indirect effect: `"boot_percentile"` (default), `"boot_bca"`, `"monte_carlo"` (simulate a and b from their joint normal approximation; MacKinnon, Lockwood, & Williams, 2004), or `"sobel"` (the first-order normal-theory interval; reported for comparison, not recommended for inference). Direct and total effects always carry their ordinary t-based intervals.
`B`	Number of bootstrap or Monte Carlo replications, a single whole number. Defaults to 2000, because each bootstrap replication refits both regressions. The fewest accepted depends on the interval and the level: 400 for the percentile and Monte Carlo intervals at the default `conf_level` (200 at 0.90, 2000 at 0.99), and 2000 for the BCa interval (1000 at 0.90, 10000 at 0.99, which the default does not meet). A smaller `B` is refused before anything is computed (class `dmar_count_invalid`); see *Bootstrap replications* under *Details* and `dmar_bootstrap`. Not used, and not checked, for `ci_method = "sobel"`.
`conf_level`	Confidence level. Defaults to 0.95.
`seed`	Optional integer seed for the resampling, used locally (the caller's random number generator state is restored on exit). Default `NULL` draws the resamples from the current random number stream, which advances that stream.

Details

The model is the standard pair of regressions

$$M = i_M + aX + \mathbf{g}'\mathbf{C} + e_M, \qquad Y = i_Y + c'X + bM + \mathbf{h}'\mathbf{C} + e_Y,$$

with $\mathbf{C}$ the optional covariates. The indirect effect is ab, the direct effect c', and the total effect $c = c' + ab$ (an identity in linear models with the same cases, which the implementation exploits as an internal consistency check).

Bootstrap intervals resample cases (rows) with replacement `B` times, refitting both regressions in each resample (Efron & Tibshirani, 1993). `"boot_percentile"` takes the interval limits from the empirical quantiles of the bootstrapped ab estimates; it is not forced to be symmetric about the estimate, which is the point for a skewed sampling distribution. `"boot_bca"` (the bias-corrected and accelerated interval) additionally adjusts the two quantile positions for median bias, estimated from the bootstrap distribution, and for the rate at which the variance of the estimator changes with the parameter, the acceleration, estimated by the jackknife (which adds *N* extra pairs of fits); the adjustments make it second-order accurate where the percentile interval is first-order accurate (DiCiccio & Efron, 1996). Because the adjusted quantile positions sit farther into the tails of the bootstrap distribution, the BCa interval benefits more than the percentile interval does from a `B` well above the default.

Bootstrap replications. Each resample refits the two regressions by least squares on the rows drawn, with the covariates coded as they are in the data, and the interval is read from the quantiles of the `B` values of ab. A limit read from a quantile is located by the replications that lie beyond it, so the percentile and Monte Carlo intervals are run only when `B` puts at least 10 replications beyond each limit (400 at the 95 percent level), and the BCa interval, whose limits move farther into the tails, only with 50 beyond each nominal limit (2000 at the 95 percent level). A resample that leaves a covariate column redundant (a category with no case in the resample) still gives unique values of a and b and is kept; the `"n_covariate_redundant"` attribute counts these. A resample in which a or b is not estimable (the predictor constant, or the predictor or the mediator a linear combination of the other predictors there) has no indirect effect. Such a replication is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B`, `B_used`, `n_dropped`, and `B_required` as attributes and `B` and `B_used` as rows. One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the share of `B` beyond one limit (2.5 percent at the 95 percent level), because they could then decide by themselves where that limit falls. When more than half were dropped, no interval is reported: the two limits of the indirect effect are `NA`, every other row is as it would have been, and the warning has class `dmar_bootstrap_insufficient`. An interval from the replications that remain describes resamples in which the effect is defined. Dropping these replications is the rule DMAR applies to a replication that returns no value; it is a convention of the package, not a result that gives the interval its nominal coverage. Many dropped replications mean the data carry little information about a path (a predictor that varies in few cases), a design in which the case bootstrap is itself not reliable.

The BCa interval needs the indirect effect with each case left out; when one of those is not estimable, or the bias correction or the acceleration is not finite, the BCa interval is refused (class `dmar_bca_undefined`) and the percentile interval remains available. After the replications are run, the count of 10 is applied again at the two positions where the bias correction and the acceleration place the limits. When fewer than 10 replications lie beyond one of them, no BCa interval is reported (limits `NA`, warning of class `dmar_bootstrap_insufficient`, which carries the two positions as `positions` and a suggested `B` as `B_suggest`).

`options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration. Every such call gives a warning of class `dmar_bootstrap_exploratory`, and the attribute `"bootstrap_status"` is `"exploratory"`; the interval is not one to report.

The Sobel standard error is computed by `var_indirect_effect`. The proportion mediated is one of

the effect size measures for mediation models surveyed by Preacher and Kelley (2011); its instability when the total effect is small is why it is reported as NA near a zero total effect. Listwise deletion is applied to the analysis variables; for full information maximum likelihood under missingness, fit the model in **lavaan** (see mlmr for the package's FIML front end philosophy). The model has one mediator and no interaction, the cases are treated as independent, and incomplete cases are removed; for several mediators or missing data see mediation_mbco.

Mediation language implies causal structure: with observational data the estimates are conditional associations, and the causal reading requires the usual no-unmeasured-confounding assumptions for both the $X \rightarrow M$ and $M \rightarrow Y$ links (MacKinnon, 2008). The function computes; the design earns the interpretation.

Value

A data.frame (class dmar_tbl) with rows indirect_effect (with its ci_method interval), direct_effect, total_effect, the paths a and b, their standard errors (se_indirect per Sobel, se_a, se_b), the interval limits (indirect_effect_ci_lower / indirect_effect_ci_upper, direct_effect_ci_lower / direct_effect_ci_upper, total_effect_ci_lower / total_effect_ci_upper), proportion_mediated (ab/c; NA when the total effect is near zero, where the ratio is unstable), N, B, and B_used, the number of replications the interval of the indirect effect rests on (equal to B for the Monte Carlo interval and NA, as B is, for the Sobel interval). The interval method is recorded in the "ci_method" attribute and the confidence level in "conf_level". For the bootstrap and Monte Carlo intervals the attributes "B" (requested), "B_used" (the number the interval rests on), "n_dropped" (their difference), and "B_required" (the fewest replications a request may ask for; that count is held to B, not to B_used) hold the counts as integers, which is where dmar_metadata reads them, and "bootstrap_status" is one of "complete", "dropped", "exploratory", "insufficient_count", or "insufficient_share" (the last two mean that the limits of the indirect effect are NA). For the two bootstrap intervals "n_covariate_redundant" is the number of resamples kept in which a covariate column was redundant (see *Details*).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

DiCiccio, T. J., & Efron, B. (1996). Bootstrap confidence intervals. *Statistical Science, 11*(3), 189–228.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

MacKinnon, D. P. (2008). *Introduction to statistical mediation analysis*. Erlbaum.

MacKinnon, D. P., Lockwood, C. M., & Williams, J. (2004). Confidence limits for the indirect effect: Distribution of the product and resampling methods. *Multivariate Behavioral Research, 39*(1), 99–128. doi:10.1207/s15327906mbr3901_4

Preacher, K. J., & Hayes, A. F. (2008). Asymptotic and resampling strategies for assessing and comparing indirect effects in multiple mediator models. *Behavior Research Methods, 40*(3), 879–891. doi:10.3758/BRM.40.3.879

Preacher, K. J., & Kelley, K. (2011). Effect size measures for mediation models: Quantitative strategies for communicating indirect effects. *Psychological Methods, 16*(2), 93–115. doi:10.1037/a0022658

See Also

`ss_aipe_indirect_effect` and `ss_power_indirect_effect` for planning the study this function analyzes; `var_indirect_effect` for the Sobel variance.

Other mediation: `mediation_mbco()`, `ss_power_indirect_effect()`

Examples

```
# Simulated mediation: X raises M (a = .5), M raises Y (b = .4), and a
# little direct effect remains (c' = .2).
set.seed(113)
n <- 200
x <- rnorm(n)
m <- 0.5 * x + rnorm(n, 0, sqrt(1 - 0.25))
y <- 0.2 * x + 0.4 * m + rnorm(n, 0, 0.8)
d <- data.frame(x = x, m = m, y = y)

# The percentile bootstrap is the default and is what a reported
# analysis would use. Each resample refits both regressions.
# B = 400 is the fewest replications a 95 percent percentile interval
# is reported from and keeps the example quick; a reported interval
# deserves the default B = 2000 or more.
mediate(d, x = "x", m = "m", y = "y", B = 400, seed = 113)

# The Monte Carlo interval draws a and b from their joint normal
# approximation instead of refitting, so it stays quick even at a
# large B. Its limits sit close to the bootstrap limits here.
mediate(d, x = "x", m = "m", y = "y", ci_method = "monte_carlo",
        B = 10000, seed = 113)

# The Sobel interval is closed form. It assumes the product ab is
# normally distributed, which is why it is reported for comparison
# rather than used for inference; its B and B_used rows are NA
# because no replications are drawn.
mediate(d, x = "x", m = "m", y = "y", ci_method = "sobel")
```

`mediation_mbco` *Mediation Analysis via Model-Based Constrained Optimization*

Description

Tests hypotheses about mediation effects with the model-based constrained optimization (MBCO) procedure of Tofighi and Kelley (2020): a likelihood ratio test that compares the full mediation

model to a null model fit subject to the nonlinear constraint that the effect of interest (an indirect effect, a total effect, or any smooth function of path coefficients) equals zero. The model is specified in **lavaan** syntax but is fit with **OpenMx**, whose optimizers support the nonlinear equality constraints the procedure requires (see Details). The function enumerates the mediation effects implied by the model (the total effect, the direct effect, the total indirect effect, and every specific indirect pathway from `x` to `y`), reports each with a confidence interval, and tests each with the MBCO likelihood ratio statistic and its *p*-value. Models may include observed or latent variables, one or many mediators, in parallel or in sequence, and raw data or summary statistics may be supplied. With a grouping variable the model becomes a multiple-group SEM: every effect is estimated within each group and the between-group difference of each effect is tested, which is moderated mediation with a categorical moderator. With a numeric `moderator`, interactions stated in the model are probed: conditional effects at chosen moderator values, the index of moderated mediation, and a joint constancy test, all with MBCO likelihood ratio inference (see Details).

Usage

```
mediation_mbco(
  model,
  data = NULL,
  S = NULL,
  M = NULL,
  N = NULL,
  group = NULL,
  x = NULL,
  y = NULL,
  moderator = NULL,
  probe_values = NULL,
  hypotheses = NULL,
  ci_method = c("profile_likelihood", "monte_carlo", "wald"),
  conf_level = 0.95,
  B = 10000,
  optimizer = c("SLSQP", "CSOLNP", "NPSOL"),
  seed = NULL
)
```

Arguments

`model`	A character string with the model in **lavaan** syntax (regressions `~`, latent variable definitions `=~`, residual covariances `~~`, optionally labeled coefficients and defined quantities via `:=`), or a fitted **lavaan** object whose parameter table and data are then reused. Parameter labels must be valid R names (e.g., `b1`, not `1b`).
`data`	A `data.frame` of raw data containing the observed variables. Rows with missing values are retained and handled by full information maximum likelihood. Supply either `data` or `S` (with `N`), not both.
`S`	A covariance matrix of the observed variables (with dimnames naming the variables), used with `N` and optionally `M` when raw data are not available. Complete-data maximum likelihood depends on the data only through the sample means and covariance matrix, so an analysis from summary statistics reproduces the

raw-data analysis exactly (see Details). For a multiple-group analysis, a named list of such matrices, one per group.

`M` Optional named numeric vector of variable means accompanying `S` (for multiple groups, a named list of such vectors). When omitted, means of zero are used; intercepts are then uninteresting but every effect, test, and confidence interval is unaffected.

`N` Total sample size. Required with `S`; ignored for raw data. For a multiple-group analysis, a vector with one sample size per group in `S` (matched by name when named).

`group` Optional name (single character string) of the grouping variable in `data` for a multiple-group analysis. With summary input, supply `S`, `M`, and `N` as lists instead and leave `group` alone (it then merely names the constructed grouping column). See the Details section on multiple groups, including how lavaan's labeling rules decide which parameters differ by group.

`x, y` Names (single character strings) of the focal predictor and the outcome between which effects are traced. When both are omitted the function looks for a unique variable with no incoming regression path (the source) and a unique variable with no outgoing regression path (the outcome); if either is ambiguous an error asks for `x` and `y` explicitly. When the model defines quantities via `:=` or `hypotheses` are supplied, `x` and `y` may be omitted entirely and only those quantities are estimated and tested.

`moderator` Optional name (single character string) of a numeric moderator variable, turning on probing: every pathway effect that involves an interaction with the moderator is estimated at each probe value, the change in the effect per unit moderator is reported (for a pathway moderated in one place, the index of moderated mediation), and a constancy test asks whether the pathway effect depends on the moderator at all. State the interaction in the model syntax with a `:` term (e.g., `m ~ x + w + x:w`) or include a precomputed product column among the predictors; both are recognized. A categorical moderator goes through `group` instead.

`probe_values` Numeric vector of moderator values at which moderated effects are estimated; names, when given, label the rows. Defaults to the moderator's mean and one standard deviation either side (labeled `low`, `mean`, `high`), or to the two observed values of a two-valued moderator.

`hypotheses` Optional named character vector of additional quantities to estimate and test against zero, written as R expressions in the parameter labels and effect names (e.g., `c(difference = "indirect_via_imagery - indirect_via_repetition")` tests the equality of two specific indirect effects). A named *list* may mix such scalar expressions with character vectors of several expressions; a multi-expression element is tested jointly (all constrained to zero at once, with as many degrees of freedom as expressions), and its row reports the test alone, with no scalar estimate. Each expression of a joint test must restrict the parameters beyond what the others do. An expression given twice, a multiple of another (`"a"` and `"2*a"`), a combination of others (`"a"`, `"b"`, and `"a + b"`; `"a - b"` and `"b - a"`), or an expression that is zero wherever the others are (`"a*b"` with `"b"`; `"a*cp"` with `"a"`; `"a*b*cp"` with `"a*b"`) would be counted as a degree of freedom while restricting nothing, so such a set is refused before any model is fit, with a condition of class `dmar_argument_invalid` that names the redundant expressions (as

redundant). Equivalent sets of expressions that each restrict, such as `c("a", "b")` and `c("a + b", "a - b")`, give the same test. See *Constraints that add no restriction* under Details.

`ci_method` Confidence interval method for every reported effect: `"profile_likelihood"` (default; Neale & Miller, 1997, computed by **OpenMx**), `"monte_carlo"` (simulate the coefficients from their joint normal approximation and form percentile intervals of the effect; MacKinnon, Lockwood, & Williams, 2004), or `"wald"` (delta method symmetric interval; reported for comparison, not recommended for indirect effects, whose sampling distributions are skewed).

`conf_level` Confidence level. Defaults to 0.95.

`B` Number of Monte Carlo draws when `ci_method = "monte_carlo"`, a single whole number (not read, and not checked, for the other intervals). Defaults to 10000, because a draw costs one evaluation of each effect and no refit. The fewest accepted is 400 at the default `conf_level` (200 at 0.90, 2000 at 0.99); a smaller `B` is refused before the model is fit (class `dmar_count_invalid`). See *Monte Carlo draws* under *Details* and `dmar_bootstrap`.

`optimizer` Optimizer used by **OpenMx** for all fits: `"SLSQP"` (default), `"CSOLNP"`, or `"NPSOL"`. The CRAN build of **OpenMx** ships SLSQP and CSOLNP; NPSOL, the optimizer used in Tofighi and Kelley (2020), is available only in the build distributed from the OpenMx website.

`seed` Optional integer seed for the Monte Carlo interval, used locally (the caller's random number generator state is restored on exit). Default `NULL` draws the Monte Carlo replications from the current random number stream, which advances that stream.

Details

The MBCO procedure. A hypothesis about mediation is recast as a model comparison (Maxwell, Delaney, & Kelley, 2027). The full model estimates all path coefficients freely. The null model is the same model estimated subject to the constraint that the effect of interest is zero, for example H_0: $\beta_1\beta_2 = 0$ for the indirect effect of X on Y through one mediator. Writing $D = -2\log\text{likelihood}$ for the deviance of each fitted model, the test statistic is

$$\text{LRT}_{\text{MBCO}} = D_{\text{null}} - D_{\text{full}},$$

and the *p*-value refers it to the chi square distribution with degrees of freedom equal to the number of constraints (here 1). That distribution is the large sample distribution of the statistic when the null hypothesis is true (Wilks, 1938) and regular: the constraints are smooth and functionally independent, and the population parameter is identified, lies in the interior of the parameter space, and is a point near which the null set is a smooth surface with one dimension fewer per constraint than the full model. The null hypothesis of no indirect effect is regular at every point of its null set except those where both of its coefficients are zero (whatever the values of the other parameters), where the chi square reference does not hold (see *The reference distribution where the branches cross* below). The *p*-value is a continuous measure of the compatibility of the data with the null model, not merely a reject or fail-to-reject decision. In the simulations of Tofighi and Kelley (2020, p. 508), with one coefficient of the indirect effect zero, the Type I error rate of the test stayed close to the nominal .05 for every sample size (50 to 200) and size of the other coefficient, whereas the percentile bootstrap,

Monte Carlo, and profile likelihood intervals and the joint significance test rejected too rarely when the other coefficient was small (0.14) and the sample size was 100 or less.

Why the constraint is nonlinear. Null hypotheses about single parameters are linear constraints: fixing $\beta_3 = 0$ restricts the parameter space to a flat hyperplane, something every structural equation modeling program does by fixing the parameter and refitting. The null hypothesis of no mediation is different in kind. The constraint $\beta_1\beta_2 = 0$ involves a product of parameters, so it is a nonlinear function of the parameter vector, and its solution set is not a hyperplane but the union of two hyperplanes: the set where $\beta_1 = 0$ (with β_2 free) and the set where $\beta_2 = 0$ (with β_1 free). Mediation is absent in infinitely many ways, and no single fixed parameter expresses them all: fixing $\beta_1 = 0$ alone tests a more restrictive hypothesis than $H_0 : \beta_1\beta_2 = 0$. Maximizing the likelihood over such a constraint set requires an optimizer that handles general nonlinear equality constraints (sequential quadratic programming methods such as SLSQP and NPSOL, or CSOLNP); among R packages for structural equation modeling, **OpenMx** provides them, which is why the model is fit there even though it is specified in **lavaan** syntax. The same geometry explains why Wald-type (delta method) tests of a product behave badly near the null: at $\beta_1 = \beta_2 = 0$ the two hyperplanes intersect, the gradient of $\beta_1\beta_2$ vanishes, and the usual normal approximation for the product breaks down. The MBCO procedure does not use that approximation, since it compares maximized likelihoods directly; the chi square reference of its statistic is itself a large sample approximation, though, and it too fails at that point, as the next paragraph explains.

The reference distribution where the branches cross. Where exactly one of β_1 and β_2 is zero, the null set near the population values is the single hyperplane of that coefficient, the null hypothesis is regular there, and the chi square distribution on 1 degree of freedom is the large sample distribution of the statistic. At $\beta_1 = \beta_2 = 0$ the two hyperplanes cross and the null hypothesis is not regular. The best-fitting null model lies on one branch or the other, so the statistic is the smaller of the two branch statistics, each the likelihood ratio statistic for fixing one coefficient at zero. When the estimates of the two coefficients are independent, as in the single mediator model with observed variables, the branch statistics are asymptotically the squares Z_1^2 and Z_2^2 of two independent standard normal variables, and the statistic is asymptotically $\min(Z_1^2, Z_2^2)$ rather than a chi square variable on 1 degree of freedom. It exceeds the .95 quantile of that chi square distribution with probability $.05^2 = .0025$. At $\beta_1 = \beta_2 = 0$, then, the test at the .05 level rejects with probability near .0025 in large samples, not .05. When the two estimates are correlated, as they can be in models with latent variables and in some multiple-group or moderated designs, Z_1 and Z_2 are correlated standard normal variables, and the probability lies between .0025 and .05, nearer .05 the stronger the correlation (.0049, .0125, and .0296 for correlations of .3, .6, and .9). The test is conservative at the crossing in every case: its *p*-value is larger than one computed from the distribution that holds there, and its power against an indirect effect whose coefficients are both near zero is lower accordingly. For a chain of three or more coefficients the same holds wherever two or more of them are zero, and the test is more conservative still where more of them are zero (with independent estimates, $.05^3$ when three are zero). The function reports the chi square *p*-value throughout and does not adjust it, because which point of the null set holds is not known, and a critical value taken from the distribution at the crossing would make the test reject too often where only one coefficient is zero. The same qualification applies to every test of a product of coefficients the function reports (specific indirect effects, conditional effects at a probe value, and moderation coefficients that are products), and to a joint test whose constraints are satisfied together at a point where their gradients are linearly dependent, such as the crossing of two branches.

The rate at the crossing in the simulations of Tofighi and Kelley (2020). Their simulations report a different rate at $\beta_1 = \beta_2 = 0$. For the single mediator model with X binary (probability .5) and standard normal residuals, 5,000 replications at each of N = 50, 75, 100, and 200, the .05 level, and

their test fitted by the NPSOL optimizer of **OpenMx** 2.9.6 (p. 507 and its footnote 4), the empirical Type I error rate of $\mathrm{LRT}_{\mathrm{MBCO}}$ with both coefficients zero fell within the interval from .025 to .075 that they took as close enough to .05 for practical purposes, at every sample size (p. 508; about .03 to .04 in their Figure 3, p. 509), and the other five methods they studied rejected at rates below .01. The two numbers describe different things. The .0025 above is a limit: the rate that the statistic defined by the best-fitting null model approaches as *N* grows at that point. The published rates are estimates from a simulation of one design at four sample sizes. Neither is the rate at every sample size. For the statistic this function computes, the rate in that design is already close to the limit at these sample sizes: in 400,000 simulated data sets at each *N*, the smaller of the two branch statistics (the statistic `mediation_mbco()` returns for this model, to within 10^{-4} on 630 further data sets of the design at *N* = 50 to 200) exceeded the .95 chi square quantile at rates of .0032, .0029, .0030, and .0028 for *N* = 50, 75, 100, and 200 (Monte Carlo standard error .00009), so sample size in this range does not account for the difference. The statistic of the worse-fitting branch, the one a search reports when it stops on that branch (see *Local solutions and the search strategy* below), rejected at .100 to .111 in the same simulations, and the published rates lie between the two.

Local solutions and the search strategy. Because the null set is a union of surfaces, the constrained deviance surface generally has one local minimum per branch (one where the mediator does not respond to X, one where the outcome does not respond to the mediator, and so on along longer chains). A constrained optimizer started from the full-model estimates can converge to whichever branch is nearest rather than to the branch that fits best. The likelihood ratio test is defined by the globally best-fitting null model, so `mediation_mbco()` refits each null model from several starting configurations (the full-model estimates, and the estimates with each coefficient entering the constrained effect set to zero in turn), verifies that each candidate solution actually satisfies the constraint, and keeps the feasible solution with the smallest deviance. In the memory example below this matters: for the single mediator model the best-fitting null model sets the imagery-to-recall path to zero ($\mathrm{LRT}_{\mathrm{MBCO}} = 71.31$, the statistic the example reports), while the branch that sets the instruction-to-imagery path to zero fits worse ($\mathrm{LRT}_{\mathrm{MBCO}} = 179.02$). Tofighi and Kelley (2020) report 175.77 for this example, a value from that worse-fitting branch, on which their optimizer stopped; their statistic also differs from the 179.02 here because the example runs from the published (rounded) summary statistics of their Table 1 rather than the full-precision moments (which give 72.54 and 175.77 for the two branches). Either way the null model is overwhelmingly incompatible with the data, so the substantive conclusion is the same.

The null model of a product is fit branch by branch: a product is zero exactly where one of its factors is, so for each way of choosing one factor of each constraint, a factor that is a coefficient is fixed at zero and a factor that is a sum (the conditional coefficient $a_0 + a_1 w$ of a moderated path at a probe value w, for example) is kept as the constraint of that fit. A label common to every term of a sum is taken out as a factor ($a_0 b + a_1 b w = b(a_0 + a_1 w)$). The null set is the union of the branches, so when every branch fit is a valid solution the smallest of them is the constrained minimum, and the constrained searches from the starting configurations are run only when a branch fit is not, or when a constraint has no branches (a total effect, a difference between groups). Such a constraint is also searched from the full-model estimates moved onto its null set by Gauss-Newton steps.

Badly scaled variables. When the observed variables differ in scale by orders of magnitude (the ratio of the largest standard deviation to the smallest is at least 1000, or the largest is at least 1000, or the smallest at most 0.001, the rule **lavaan** uses for its `rescale_data` option), the free parameters do too, and the **OpenMx** optimizers can stop far from the maximum. The **OpenMx** models are then fit to the variables each divided by a power of 2 near its standard deviation ($2^{\mathrm{round}(\log_2 s)}$ for standard deviation s, over all groups), with every parameter in the matching metric (a latent variable takes the metric of the indicator whose loading is fixed), and every effect, constraint, and Monte

Carlo draw is written in the parameters of the original metric, so the estimates, standard errors, intervals, and likelihood ratio statistics are those of the variables as given, and the deviance, AIC, and BIC are returned to the original metric as well. A likelihood ratio statistic does not depend on the units of the variables, and the rescaling changes it only within the precision of the optimizer. A profile likelihood interval is profiled for the effect in the metric of the fit and its limits returned to the original metric. The constraints of a joint test are then handed to the optimizer as linear combinations whose gradients are orthonormal in the metric of the fit at the full-model estimates, which leaves their null set unchanged. Nothing is rescaled when a label shared by parameters in different metrics (an equality constraint across them) would need two multipliers. The divisors and the multipliers are returned in the `"rescale"` attribute. Scales that differ by less than that rule can still slow the optimizers; the branch fits and the start on the null set described above are the remedy there, and a variable can be put on a scale near that of the others before the call (the statistics do not depend on its units, and the estimates change by the factor applied).

Effects estimated and tested. With `x` and `y` resolved, the function enumerates every directed pathway from `x` to `y` along regression (`~`) paths. Each pathway through at least one intermediate variable contributes a specific indirect effect, the product of its path coefficients, reported as `indirect_via_` followed by the intermediate variable names. The direct effect is the `x` to `y` coefficient when that path is in the model. The total indirect effect is the sum of the specific indirect effects (reported when there are two or more), and the total effect is the direct effect plus the total indirect effect (reported when the direct path is in the model). Quantities defined in the model syntax via `:=` and any `hypotheses` are estimated and tested the same way, so contrasts of indirect effects, proportions mediated, or any other smooth function of parameters can be examined; each is tested against zero, so write an equality of two effects as their difference. Every reported row carries its estimate, a delta method standard error (via `mxSE`), the `ci_method` confidence interval, and the MBCO likelihood ratio test with its degrees of freedom and *p*-value.

The parallel two-mediator model. Tofighi and Kelley (2020) continue the memory example with imagery and repetition as parallel mediators and a residual covariance between them, asking whether the indirect effect through repetition is zero and whether the two specific indirect effects differ (their Research Questions 2 and 3). That analysis is the single-mediator call of the examples with the parallel model in place of the single one and the contrast supplied through `hypotheses`:

```
parallel <- "
  imagery    ~ b1*instruction
  repetition ~ b3*instruction
  recall     ~ b2*imagery + b4*repetition + b5*instruction
  imagery ~~ repetition
"
mediation_mbco(parallel, S = S_tk, M = M_tk, N = 369,
               x = "instruction", y = "recall",
               hypotheses = c(imagery_minus_repetition =
                 "indirect_via_imagery - indirect_via_repetition"),
               ci_method = "monte_carlo", seed = 113)
```

Six effects are reported, so six constrained null models are fit, about twice the cost of the single-mediator analysis. The indirect effect through repetition is near zero (the paper reports $\mathrm{LRT}_{\mathrm{MBCO}} = 0.083$, $p = .773$), while the contrast shows the imagery pathway is larger (the paper reports $\mathrm{LRT}_{\mathrm{MBCO}} = 25.828$, difference = 2.222, SE = 0.445).

Choosing the confidence interval. The default profile likelihood interval inverts the likelihood

ratio test for the effect itself (Neale & Miller, 1997), so its limits are free to sit asymmetrically about the estimate, as the skewed sampling distribution of a product of coefficients calls for, and the interval and the MBCO test tell the same story. Each bound is a constrained search of its own, which is what makes it the expensive choice. The Monte Carlo interval is the inexpensive alternative that also accommodates the skewness: `B` coefficient vectors are drawn from the joint normal approximation of the estimates, the effect is evaluated in each draw, and the limits are the empirical $(\alpha/2, 1-\alpha/2)$ quantiles of those `B` values (MacKinnon, Lockwood, & Williams, 2004). Ask for it when the profile searches are slow, when a profile bound fails to converge, or when comparing with a published analysis that reports one, as Tofighi and Kelley (2020) do for the memory data in the examples.

Monte Carlo draws. The limits of each effect are quantiles of its `B` drawn values, so the Monte Carlo interval is run only when `B` puts at least 10 draws beyond each limit (400 at the 95 percent level). A draw in which an effect is undefined (a division by a draw near zero, or a function outside its domain) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, and the line under the table states them, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). A call gives at most one such warning, whatever the number of effects. A warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the limits of that effect are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. The draws are counted effect by effect: an effect whose interval is withheld does not withhold the others, `B_used` is the fewest draws any effect's limits rest on, and `bootstrap_status` is the least favorable status among the effects. When several effects are affected the one warning lists them (draws dropped, draws used, and whether the interval is reported), gives in full the account of the effect with the fewest usable draws, has the least favorable class among them, and carries the list as the data frame `effects`. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the result is not one to report. See `dmar_bootstrap` for the rule and its reasons.

The Wald interval is the symmetric delta method interval and is reported only for comparison.

Multiple groups (moderated mediation across groups). With `group` (or list-form `S`, `M`, `N`), the model is fit as a multiple-group SEM: the same structure in every group, with group-specific parameters. Every enumerated effect is then estimated within each group (its term carries the group label as a suffix), and for each effect the difference from the reference group (the first group label) is estimated and tested with the same constrained-optimization machinery, since a difference of two products is itself a nonlinear function of the parameters. That difference test is moderated mediation with a categorical moderator: "does the indirect effect differ across groups?" is exactly the between-group contrast of the conditional indirect effects. lavaan's labeling rules decide what varies: a single label such as `b1` on a path applies to every group and therefore imposes cross-group equality (the corresponding difference is identically zero and its row is dropped); to let a path differ by group, leave it unlabeled or give per-group labels with the vector form `c("b1_f", "b1_m")*x`. With more than two groups, each non-reference group is compared with the reference group. The call below fits the simple model in both groups of a data frame `d` whose `condition` column takes two values, and reports every effect per group beside its between-group difference; leaving the paths unlabeled lets them differ by group.

```
mediation_mbco("m ~ x
                y ~ m + x",
               data = d, group = "condition", x = "x", y = "y",
               ci_method = "wald")
```

Each reported row costs its own constrained null model fit, so this two-group analysis fits nine null models where the single-group analysis fits three.

Moderated mediation, probed. With `moderator`, every regression coefficient along a pathway becomes a linear function of the moderator wherever the model contains the matching interaction (stated as `x:w` in the syntax, or as a product column among the predictors). A pathway effect, the product of its edge coefficients, is then a polynomial in the moderator, and the function derives that polynomial symbolically. Three kinds of rows follow for each moderated effect. First, the conditional effect at each probe value, each with its own confidence interval and MBCO test. Second, the polynomial's moderator coefficients: for a pathway moderated in one place the single such coefficient is exactly the index of moderated mediation (Hayes, 2015), here tested by likelihood ratio rather than bootstrap; a pathway moderated in several places gets one row per power of the moderator. Third, for a pathway moderated in several places (so the conditional effect is curved in the moderator), a joint constancy test of all moderator coefficients at once, with as many degrees of freedom as constraints, asking whether the pathway effect depends on the moderator at all; as a joint test its row reports no scalar estimate. Unmoderated effects in the same model keep their single rows. The null set of a conditional-effect constraint is again a union of branches (one edge's conditional coefficient or another's must vanish at the probed value), and the null fits are started on each branch, and at the model with all interactions removed for the moderation tests, so the reported statistics reflect the best-fitting null models. For example:

```
mediation_mbco("m ~ x + w + x:w
                y ~ m + x + w",
               data = d, x = "x", y = "y", moderator = "w")
```

`plot_mediation_mbco` draws the same conditional effects as curves over the moderator's whole range with a confidence band, the visual companion to the probe rows. Bespoke conditional quantities can still be written directly with `:=` definitions or `hypotheses` when the built-in probing does not cover them.

Moderated mediation and mediated moderation. Moderated mediation asks whether an indirect effect depends on a moderator W: the conditional indirect effect varies with w (Muller, Judd, & Yzerbyt, 2005; Preacher, Rucker, & Hayes, 2007). Mediated moderation asks whether an observed $X \times W$ interaction on Y is transmitted through the mediator. The two share their algebra: with first-stage moderation $M = a_1X + a_2W + a_3XW + e_M$ and $Y = bM + \ldots$, the quantity a_3b is at once the index of moderated mediation (Hayes, 2015) and the indirect effect of the product term through M; what differs is the question and the reporting emphasis. This function reports the conditional-indirect-effect framing: the probe rows and the moderation rows above. For a categorical moderator, use `group`.

Refusals, warnings, and judgment calls. The function stops where a computed number would be mislabeled: a nonrecursive (feedback) regression structure, categorical-endogenous syntax (thresholds), explicit `==` constraints, a model with no indirect pathway between `x` and `y`. It stops as well, with an error of class `dmar_no_maximum`, when the **lavaan** fit that the **OpenMx** fits start from has no maximum, an error variance running off below zero while the likelihood is flat or still rising (see `fa_bifactor`), printing the **lavaan** code of that fit, or none for a fit given as `model`. It warns where

the data make trouble detectable: an endogenous variable with only two distinct values (a binary mediator or outcome, where the product of coefficients is not the causal indirect effect), an interaction term among the predictors that no declared `moderator` accounts for (declaring the moderator resolves the warning by probing the moderation), an interaction included without its matching main effect (the principle of marginality), null models that converge imperfectly, profile bounds that fail. What it cannot check is left to the analyst, and stated rather than assumed: the no omitted confounder assumption, linearity, and the causal direction of the arrows come from the design, not from the fit.

Information criteria. The columns `delta_aic` and `delta_bic` report AIC and BIC for the null model minus the same criterion for the full model; positive values favor the full model (the effect improves fit by more than the parsimony penalty). A scalar equality constraint reduces the effective number of free parameters by one, so the differences equal $\mathrm{LRT}_{\mathrm{MBCO}} - 2$ for the AIC and $\mathrm{LRT}_{\mathrm{MBCO}} - \log N$ for the BIC; for a joint test the 2 and the $\log N$ are multiplied by its degrees of freedom. (The **OpenMx** summary counts the same number of estimated parameters in both models, so its printed AIC difference equals the likelihood ratio statistic; DMAR counts the constraint against the null model, consistent with the degrees of freedom of the test.)

Constraints that add no restriction. The degrees of freedom of a test are the number of its constraints, which is the number of restrictions only when each constraint restricts the parameters beyond what the others do. Before any **OpenMx** model is fit, the constraints of every test, written in the parameter labels, are checked twice. First, the function forms the Jacobian of the constraints (the matrix of their derivatives with respect to the free parameters) at the unrestricted estimates of the **lavaan** fit, which lie off the null set, and reads its rank after scaling every column by its largest entry and every row to unit length, so that the rank does not depend on the units of the parameters or the scale of the constraints. A singular value below 10^{-7} times the largest counts as zero (the same number as the default `tol` of `qr`). The rank is read again at a fixed point near the estimates, so that an estimate of exactly zero cannot make independent constraints look dependent, and the larger rank is used. A test whose constraints have a rank below their number (a constraint that is a function of the others) is refused (see `hypotheses`); so is a single constraint whose gradient is zero, which restricts nothing. The rank is not read on the null set: at the crossing of the branches of a product constraint its gradient vanishes, and a rank read there would call every set that contains it dependent.

Second, constraints whose Jacobian has full rank can still restrict the parameters in fewer directions than their number, when one of them is zero wherever the others are: `c("a*b", "b")` has the null set $b = 0$, `c("a", "a*cp")` the null set $a = 0$, and `c("a*b", "a*b*cp")` the null set $ab = 0$, one restriction each. For every constraint of a joint test the function finds points of the null set of the others, by Gauss-Newton steps from fixed starting points that cover both signs and a range of magnitudes of the parameters (in coordinates scaled by the estimates, and again in coordinates scaled by the derivatives of the constraints, so that neither the units of the parameters nor the starting points of one scaling decide the result), and refuses the test when the constraint is zero at all of them, to within 10^{-6} of its typical size, under both scalings. No random numbers are drawn. A set such as `c("a*b", "a*cp")`, whose null set is the union of $a = 0$ (one restriction) and $b = cp = 0$ (two), is accepted with 2 degrees of freedom: the chi square reference on 2 degrees of freedom is right where the second branch holds, and the test is conservative where only $a = 0$ holds, since the statistic there is asymptotically chi square on 1 degree of freedom, which exceeds the .95 quantile of the chi square distribution on 2 degrees of freedom with probability .0144. The checks do not detect a set none of whose constraints is zero wherever the others are but whose null set still imposes fewer restrictions than there are constraints, such as `c("a*b", "a*cp", "b*cp")` (any two of the three coefficients zero, two restrictions); its test is conservative. Neither check establishes that the

null hypothesis is regular (see *The reference distribution where the branches cross* above).

Change in explained variance. Following the reporting recommendation of Tofighi and Kelley (2020), the function computes R^2 for every endogenous variable under the full model (the `"R2"` attribute) and the drop in each R^2 under every null model (the `"delta_R2"` attribute, variables by tested effects), so the fit cost of removing an effect can be read as a change in effect size and not only as a test statistic.

Summary statistics input. With complete data the multivariate normal log likelihood depends on the data only through the sample means and the sample covariance matrix. When `S` (and optionally `M`) is supplied, the function therefore constructs an internal data set with exactly those moments (via `mvrnorm` with `empirical = TRUE`) and proceeds as with raw data; the estimates, likelihood ratio tests, and confidence intervals are identical to what the raw data would give, whatever internal data set realizes the moments. `S` is treated as the unbiased (divisor $N - 1$) covariance matrix, the form reported in articles. This is how a published mediation analysis can be reproduced, and its hypotheses re-tested with the MBCO procedure, from a table of descriptive statistics alone.

Assumptions. The causal reading of any mediation analysis rests on the no omitted confounder assumption for the predictor to mediator and mediator to outcome relations, in addition to the usual distributional assumptions (residuals multivariate normal, linear relations, no treatment by mediator interaction); randomizing X supports the first link but not the second (MacKinnon, 2008; Tofighi & Kelley, 2020). With a binary randomized X the variable enters the model as numeric 0/1, its exogenous variance freely estimated; the normality assumption applies to the residuals of the endogenous variables.

Value

A `data.frame` (classes `dmar_mediation_mbco`, `dmar_tbl`) with one row per effect and columns `pathway` (the traced pathway or defining expression), `term`, `estimate`, `se` (delta method), `ci_lower`, `ci_upper`, `lrt` (the MBCO likelihood ratio statistic), `df`, `p_value`, `delta_aic`, and `delta_bic`. A joint test row (a several-constraint moderation or hypothesis test) reports the test columns with `df` equal to the number of constraints, none of which is redundant (see `hypotheses`); its `estimate`, `se`, and interval are `NA` because no single number summarizes several constraints. Attributes: `"conf_level"`, `"ci_method"`, with the Monte Carlo interval `"B"`, `"B_used"`, `"n_dropped"`, `"B_required"`, and `"bootstrap_status"` (see *Monte Carlo draws* under Details), `"optimizer"`, `"x"`, `"y"`, `"N"`, `"deviance"`, `"aic"`, `"bic"`, `"n_par"` (full model fit information), `"groups"` (the group labels, reference first; multiple-group fits only), `"R2"` (named vector, endogenous variables under the full model, per group when grouped), `"delta_R2"` (matrix of full minus null R^2, variables by tested effects), `"moderation"` (for a moderated analysis: the moderator name, the probe values, and each moderated effect's moderator polynomial, which is what `plot_mediation_mbco` draws), `"rescale"` (only when badly scaled variables were rescaled for the fits, see *Badly scaled variables* under Details: `divisors`, the power of 2 each observed variable was divided by, and `multipliers`, the factor that turns each parameter of the rescaled fit into the parameter of the original metric), and `"mx_model"` (the fitted **OpenMx** full model, in the rescaled metric when there is a `"rescale"` attribute, an escape hatch for further OpenMx work). Use `tidy()` and `glance()` for broom-style views.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Tofighi, D., & Kelley, K. (2020). Improved inference in mediation analysis: Introducing the model-based constrained optimization procedure. *Psychological Methods, 25*(4), 496–515. doi:10.1037/met0000259

Tofighi, D., & Kelley, K. (2020). Indirect effects in sequential mediation models: Evaluating methods for hypothesis testing and confidence interval formation. *Multivariate Behavioral Research, 55*(2), 188–210. doi:10.1080/00273171.2019.1618545

Hayes, A. F. (2015). An index and test of linear moderated mediation. *Multivariate Behavioral Research, 50*(1), 1–22. doi:10.1080/00273171.2014.962683

MacKinnon, D. P. (2008). *Introduction to statistical mediation analysis*. Erlbaum.

MacKinnon, D. P., Lockwood, C. M., & Williams, J. (2004). Confidence limits for the indirect effect: Distribution of the product and resampling methods. *Multivariate Behavioral Research, 39*(1), 99–128. doi:10.1207/s15327906mbr3901_4

MacKinnon, D. P., Valente, M. J., & Wurpts, I. C. (2018). Benchmark validation of statistical models: Application to mediation analysis of imagery and memory. *Psychological Methods, 23*(4), 654–671. doi:10.1037/met0000174

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Muller, D., Judd, C. M., & Yzerbyt, V. Y. (2005). When moderation is mediated and mediation is moderated. *Journal of Personality and Social Psychology, 89*(6), 852–863. doi:10.1037/0022-3514.89.6.852

Neale, M. C., Hunter, M. D., Pritikin, J. N., Zahery, M., Brick, T. R., Kirkpatrick, R. M., Estabrook, R., Bates, T. C., Maes, H. H., & Boker, S. M. (2016). OpenMx 2.0: Extended structural equation and statistical modeling. *Psychometrika, 81*(2), 535–549. doi:10.1007/s1133601494358

Neale, M. C., & Miller, M. B. (1997). The use of likelihood-based confidence intervals in genetic models. *Behavior Genetics, 27*(2), 113–120. doi:10.1023/A:1025681223921

Preacher, K. J., Rucker, D. D., & Hayes, A. F. (2007). Addressing moderated mediation hypotheses: Theory, methods, and prescriptions. *Multivariate Behavioral Research, 42*(1), 185–227. doi:10.1080/00273170701341316

Wilks, S. S. (1938). The large-sample distribution of the likelihood ratio for testing composite hypotheses. *Annals of Mathematical Statistics, 9*(1), 60–62. doi:10.1214/aoms/1177732360

See Also

`plot_mediation_mbco` for the conditional-effect display of a moderated analysis; `mediate` for the regression-based simple mediation model with bootstrap intervals; `ss_aipe_indirect_effect` and `ss_power_indirect_effect` for planning; `var_indirect_effect` for the delta method variance.

Other mediation: `mediate()`, `ss_power_indirect_effect()`

Examples

```
# Replicate the memory experiment analyses of Tofighi and Kelley
# (2020) from the summary statistics in their Table 1 (data from
# MacKinnon, Valente, & Wurpts, 2018; N = 369). Instruction is 1 for
```

```
# imagery rehearsal instructions and 0 for repetition instructions.
# Requires the OpenMx and lavaan packages to be installed.
vars <- c("instruction", "imagery", "repetition", "recall")
sds  <- c(0.50, 2.96, 2.84, 3.40)
R_tk <- matrix(c(1.00,  .62, -.67,  .32,
                  .62, 1.00, -.56,  .51,
                 -.67, -.56, 1.00, -.28,
                  .32,  .51, -.28, 1.00), 4, 4,
               dimnames = list(vars, vars))
S_tk <- outer(sds, sds) * R_tk
M_tk <- c(instruction = 0.51, imagery = 5.66, repetition = 6.08,
          recall = 12.07)

# Single-mediator model: instruction -> imagery -> recall. Every
# reported effect costs its own constrained null model fit in OpenMx,
# which is where the run time goes; the Monte Carlo interval itself
# is inexpensive at any B. The indirect effect is about 2.1 words
# (the paper reports 2.121, SE = 0.276, 95% Monte Carlo CI
# [1.600, 2.682]), and its likelihood ratio statistic of 71.31 is the
# value discussed under Details.
single <- "
  imagery ~ b1*instruction
  recall  ~ b2*imagery + b3*instruction
"
mediation_mbco(single, S = S_tk, M = M_tk, N = 369,
               x = "instruction", y = "recall",
               ci_method = "monte_carlo", seed = 113)

# The parallel two-mediator model of the same paper, with the contrast
# of its two specific indirect effects, is described under Details;
# it is fit the same way from the same summary statistics, adding the
# 'hypotheses' argument. Raw data go in through 'data' rather than
# 'S', 'M', and 'N'; adding 'group' fits the model in every group and
# tests the between-group difference of each effect, which is
# moderated mediation with a categorical moderator (see Details). A
# continuous moderator goes in through 'moderator', and the help page
# for plot_mediation_mbco fits that analysis from a data frame and
# draws the conditional effects it estimates.
```

`meta_contrast` *Contrast Among Study Effect Sizes (Rosenthal-Rubin)*

Description

Tests a focused hypothesis about *differences* among independent study effect sizes by the method of Rosenthal and Rubin (1982): given effects y_i with sampling variances v_i and contrast weights λ_i summing to zero,

$$z = \frac{\sum \lambda_i y_i}{\sqrt{\sum \lambda_i^2 v_i}}$$

is referred to the standard normal. This is how a meta-analyst asks a pointed moderator question ("do the effects decline with weeks of prior teacher-student contact?") rather than the diffuse heterogeneity question ("do the effects differ at all?"). Raudenbush (1984) used exactly this test for the teacher expectancy literature, with weights inversely proportional to weeks of prior contact.

Usage

```
meta_contrast(yi, vi, weights, center = TRUE)
```

Arguments

`yi`	Numeric vector of study effect sizes (any metric whose sampling distribution is approximately normal; standardized mean differences and Fisher's Z correlations qualify).
`vi`	Sampling variances of `yi`, one per study.
`weights`	Contrast weights, one per study. If they do not already sum to zero they are mean-centered (with a message) when `center = TRUE`, the convenient route for weights built from a moderator such as `1 / (weeks + 2)`.
`center`	Logical: mean-center `weights` that do not sum to zero? Default `TRUE`.

Details

The two-sided p-value is reported; halve it for a directional hypothesis stated in advance (Raudenbush's $z = 2.75$ carried the one-tailed $p = .003$). Dividing the squared contrast statistic z^2 (the square of the `test_statistic` row) by the total heterogeneity statistic Q from `meta_es` gives the proportion of between-study heterogeneity the contrast accounts for, the meta-analytic analog of a contrast's share of the between-group sum of squares.

Value

A `data.frame` (class `dmar_meta_contrast`, a `dmar_tbl`) with the contrast `estimate` ($\sum \lambda_i y_i$), its `se`, the `test_statistic` (the estimate divided by `se`, the z above), the two-sided `p_value`, and `n_studies` (the number of studies). The reference distribution of `test_statistic` is the standard normal, recorded as in `meta_es`: the attribute `test_distribution` is `"normal"` and `test_df` is `NA`, and the line under the printed table says so.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Raudenbush, S. W. (1984). Magnitude of teacher expectancy effects on pupil IQ as a function of the credibility of expectancy induction: A synthesis of findings from 18 experiments. *Journal of Educational Psychology, 76*(1), 85–97.

Rosenthal, R., & Rubin, D. B. (1982). Comparing effect sizes of independent studies. *Psychological Bulletin, 92*(2), 500–504.

See Also

meta_es for the pooled effect and the total heterogeneity the contrast partitions; combine_p for combined significance tests; contrast_test for the single-study ANOVA analog.

Other meta-analysis: combine_p(), meta_es(), meta_r(), meta_smd(), plot_forest()

Examples

```
# Raudenbush (1984): do expectancy effects decline with weeks of prior
# teacher-student contact? Weights inversely proportional to weeks + 2,
# study-level data (Pellegrini & Hicks merged), d variances from the
# standard large-sample formula.
data(teacher_expectancy)
study <- teacher_expectancy[-c(4, 5), ]
d  <- append(study$d, 0.52, after = 3)
wk <- append(study$weeks, 0, after = 3)
ne <- append(study$n_experimental, 22, after = 3)
nc <- append(study$n_control, 22, after = 3)
v  <- (ne + nc) / (ne * nc) + d^2 / (2 * (ne + nc))
meta_contrast(d, v, weights = 1 / (wk + 2))
# A test statistic near the paper's z of 2.75: the better teachers knew
# their pupils, the smaller the expectancy effect (one-tailed p = .003
# in the paper).
```

---

meta_es *Random Effects Meta-Analysis of Generic Effect Sizes*

Description

Pools independent effect sizes given their sampling variances: the general engine behind meta_smd and meta_r, exposed for any effect metric whose estimates are approximately normal with known variances. The random effects model is the default and the fit reports the full uncertainty picture in one table: the pooled estimate with its confidence interval, the between-study standard deviation tau, the between-study variance tau-squared with its Q-profile confidence interval, I-squared with an interval mapped from the tau-squared limits, H-squared, Cochran's Q test, and, always, a prediction interval for the effect in a new study. Reporting the prediction interval by default is deliberate: when heterogeneity is real, the confidence interval for the average effect understates what the next study will show, and the package treats "where will the next study land" as part of the answer, not an option.

Usage

```
meta_es(
  yi,
  vi,
  method = c("reml", "pm", "dl", "fe"),
  hartung_knapp = TRUE,
  conf_level = 0.95
)
```

Arguments

`yi`	Numeric vector of effect sizes, one per independent study.
`vi`	Sampling variances of `yi`, one per study.
`method`	Between-study variance estimator: `"reml"` (restricted maximum likelihood, the default), `"pm"` (Paule-Mandel), `"dl"` (DerSimonian-Laird), or `"fe"` (a fixed effect / common effect analysis, which assumes tau-squared is zero and reports no prediction interval).
`hartung_knapp`	Logical: apply the Hartung-Knapp-Sidik-Jonkman small-sample adjustment (the pooled standard error rescaled from the weighted residuals, with a *t* reference on $k - 1$ degrees of freedom)? Default `TRUE`. The adjusted standard error is not bounded below by the conventional one, so the interval can be narrower than the normal interval; see *The Hartung-Knapp adjustment* and *When q is small* in Details. Identical effect sizes give a zero adjusted standard error and a warning; see *Identical effect sizes*. Ignored for `method = "fe"`.
`conf_level`	Confidence level for all intervals. Defaults to 0.95.

Details

The model is $y_i = \mu + u_i + e_i$ with $u_i \sim N(0, \tau^2)$ and $e_i \sim N(0, v_i)$, v_i treated as known. The τ^2 confidence interval inverts the generalized Q statistic (Viechtbauer, 2007); the I-squared interval maps the τ^2 interval through the typical within-study variance of Higgins and Thompson (2002). The prediction interval follows Higgins, Thompson, and Spiegelhalter (2009), using *t* with $k - 2$ degrees of freedom, and requires at least three studies.

The Hartung-Knapp adjustment. With weights $w_i = 1/(v_i + \hat{\tau}^2)$ and pooled estimate $\hat{\mu}$, the adjusted standard error is the conventional one, $(\sum w_i)^{-1/2}$, multiplied by $\sqrt{q}$, where

$$q = \frac{1}{k-1} \sum_{i=1}^{k} w_i (y_i - \hat{\mu})^2$$

is the weighted residual mean square on $k - 1$ degrees of freedom (Röver, Knapp, & Friede, 2015), and the interval is the pooled estimate plus and minus the $1 - \alpha/2$ quantile of the *t* distribution on $k - 1$ degrees of freedom times that standard error. The function applies this, the unmodified method, as `test = "knha"` in **metafor** does: q is not replaced by $\max(1, q)$. With few studies the coverage of the adjusted interval is closer to the nominal level than that of the normal interval. The prediction interval uses the standard error of the pooled estimate that the confidence interval uses, so under the adjustment it narrows and widens with q as well.

When q is small. Because q is not replaced by $\max(1, q)$, the adjusted standard error is smaller than the conventional one whenever q is below 1, which is common (when the effects are homogeneous, in most samples). That alone does not make the interval narrower than the normal interval, since the adjusted interval uses a *t* quantile on $k - 1$ degrees of freedom: it is narrower exactly when q is below the squared ratio of the normal quantile to the *t* quantile, which for a 95 percent interval is 0.02, 0.21, 0.50, and 0.75 at k = 2, 3, 5, and 10. A small q is part of the reference distribution of the method and not a fault in the data: when the weights are the true ones, the pooled estimate minus the mean effect, divided by the adjusted standard error, follows the *t* distribution exactly, small values of q included. Röver, Knapp, and Friede (2015) describe the modification $\max(1, q)$, which never gives a standard error below the conventional one; it is not applied here. In the simulations

run for this function the unmodified interval was at or near the nominal level when the effects were homogeneous or the studies equally precise, and below it (0.86 to 0.89 for a nominal 0.95, with the normal interval at 0.70 to 0.87) when one study was far more precise than the rest and the effects were heterogeneous. When the adjusted interval is much narrower than the conventional one, report both; `hartung_knapp = FALSE` gives the second. The attributes `hartung_knapp_q` and `se_conventional` hold q and the conventional standard error for that comparison. No warning is given for a q between 0 and 1, which is the method working as defined.

Identical effect sizes. When every effect size is the same, exactly or up to rounding error, q is 0 and so is the adjusted standard error. The confidence interval and the prediction interval then have no width, and the *t* statistic is infinite, or undefined when the common effect size is zero. That is the arithmetic of the method at its boundary and not evidence of an effect known exactly: each effect size still has its sampling variance, and identical values arise from coarse rounding, from one study entered more than once, or by chance with two or three studies. The function reports the numbers as computed, as `test = "knha"` in **metafor** does, and warns with a condition of class `dmar_meta_hk_zero_se` that states the conventional standard error and interval. The condition carries the number of studies as `n_studies`, the pooled estimate as `estimate`, the conventional standard error as `se_conventional`, and the limits of the normal interval that standard error gives as `lower_conventional` and `upper_conventional`. For four effect sizes of 0.5 with sampling variances of 0.04, the adjusted limits are 0.5 and 0.5; the conventional standard error is 0.1 and the normal interval 0.304 to 0.696. The check is made on the effect sizes themselves: effect sizes that differ in any digit beyond rounding error (0.5 against $0.5 + 10^{-9}$) are not identical, give a small positive q, and raise no warning.

I-squared is reported because readers expect it, but note its well-known limitation: it is a *proportion* of variability, not an amount, so the same tau matched with larger studies yields a larger I-squared. The quantity with direct scientific meaning is tau (the between-study standard deviation, in the metric of `yi`) together with the prediction interval.

Value

A `data.frame` (class `dmar_meta`, a `dmar_tbl`) with rows `estimate`, `se`, `test_statistic` (the estimate divided by `se`), `p_value`, `ci_lower` / `ci_upper`, `prediction_lower` / `prediction_upper`, `tau2` with `tau2_ci_lower` / `tau2_ci_upper`, `tau`, `I2` with limits, `H2`, `Q` / `Q_df` / `Q_p`, and `n_studies` (the number of studies, k). The estimator and adjustment are recorded in the `"method"` and `"hartung_knapp"` attributes. The reference distribution of `test_statistic` and `p_value` depends on the adjustment, and the attributes `test_distribution` and `test_df` record it: `"t"` on $k - 1$ degrees of freedom under the Hartung-Knapp adjustment, `"normal"` (with `test_df` `NA`) without it, as the line under the printed table says. The attributes `hartung_knapp_q` (the weighted residual mean square q) and `se_conventional` are recorded whenever the adjustment is applied. When the effect sizes are identical and the Hartung-Knapp adjustment is applied, `se` is zero, the limits equal the estimate, and the function warns (see *Identical effect sizes* in Details).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

DerSimonian, R., & Laird, N. (1986). Meta-analysis in clinical trials. *Controlled Clinical Trials, 7*(3), 177–188.

Hartung, J., & Knapp, G. (2001). On tests of the overall treatment effect in meta-analysis with normally distributed responses. *Statistics in Medicine, 20*(12), 1771–1782. doi:10.1002/sim.791

Higgins, J. P. T., & Thompson, S. G. (2002). Quantifying heterogeneity in a meta-analysis. *Statistics in Medicine, 21*(11), 1539–1558. doi:10.1002/sim.1186

Higgins, J. P. T., Thompson, S. G., & Spiegelhalter, D. J. (2009). A re-evaluation of random-effects meta-analysis. *Journal of the Royal Statistical Society: Series A, 172*(1), 137–159. doi:10.1111/j.1467985X.2008.00552.x

Raudenbush, S. W. (1984). Magnitude of teacher expectancy effects on pupil IQ as a function of the credibility of expectancy induction: A synthesis of findings from 18 experiments. *Journal of Educational Psychology, 76*(1), 85–97.

Röver, C., Knapp, G., & Friede, T. (2015). Hartung-Knapp-Sidik-Jonkman approach and its modification for random-effects meta-analysis with few studies. *BMC Medical Research Methodology, 15*, Article 99. doi:10.1186/s12874015009-11

Viechtbauer, W. (2007). Confidence intervals for the amount of heterogeneity in meta-analysis. *Statistics in Medicine, 26*(1), 37–52. doi:10.1002/sim.2514

See Also

`meta_smd` and `meta_r` for the metric- specific front ends; `meta_contrast` for focused moderator contrasts; `combine_p` for combined significance tests; `plot_forest` to see the studies and the pool together.

Other meta-analysis: `combine_p()`, `meta_contrast()`, `meta_r()`, `meta_smd()`, `plot_forest()`

Examples

```
# The teacher expectancy studies (Raudenbush, 1984), pooled in the d
# metric with variances from the standard large-sample formula.
data(teacher_expectancy)
d <- teacher_expectancy$d
n_e <- teacher_expectancy$n_experimental
n_c <- teacher_expectancy$n_control
v <- (n_e + n_c) / (n_e * n_c) + d^2 / (2 * (n_e + n_c))
meta_es(d, v)

# A fixed effect (common effect) analysis of the same studies.
meta_es(d, v, method = "fe")

# The adjusted standard error beside the conventional one. Five effect
# sizes that agree closely give a weighted residual mean square q far
# below 1, so the adjusted standard error is a small fraction of the
# conventional one; both intervals are worth reporting.
close <- meta_es(c(.30, .32, .29, .31, .33), c(.04, .05, .03, .06, .04))
attr(close, "hartung_knapp_q")
attr(close, "se_conventional")
meta_es(c(.30, .32, .29, .31, .33), c(.04, .05, .03, .06, .04),
        hartung_knapp = FALSE)
```

`meta_r` *Random Effects Meta-Analysis of Correlations*

Description

Pools correlations across independent studies on the Fisher's Z scale and reports the results back in the correlation metric. Optionally, each study's correlation is first corrected for attenuation due to measurement error in either or both variables (the Spearman correction of `correction_for_attenuation`, the basic artifact correction of Hunter and Schmidt's psychometric meta-analysis), using reliabilities you supply, for example from the `reliability` family. The pooled quantity is then the correlation between the constructs rather than the attenuated correlation between the observed scores, and the weight of each study comes from the sampling variance of its corrected correlation, which the correction enlarges.

Usage

```
meta_r(
  r,
  n,
  reliability_x = NULL,
  reliability_y = NULL,
  method = c("reml", "pm", "dl", "fe"),
  hartung_knapp = TRUE,
  conf_level = 0.95
)
```

Arguments

`r`	Numeric vector of observed correlations, one per study, each in (-1, 1).
`n`	Per-study sample sizes (integer, at least 4).
`reliability_x, reliability_y`	Optional per-study reliabilities in (0, 1] for the two measured variables; a single value is recycled across studies. When either is supplied, each correlation is disattenuated by $r_i / \sqrt{\rho_{xx,i}\,\rho_{yy,i}}$ before pooling (a reliability left `NULL` is treated as 1), and its sampling variance is that of the corrected correlation (see Details). The reliabilities are treated as known constants: the uncertainty of a reliability that was estimated is not carried into the results.
`method, hartung_knapp, conf_level`	Passed to `meta_es`. Under the Hartung-Knapp adjustment, identical correlations give a zero adjusted standard error and a warning of class `dmar_meta_hk_zero_se`; see `?meta_es`. In that warning the degenerate limits and the conventional normal interval are in the correlation metric, as the reported rows are, and the conventional standard error is on the Fisher's Z scale, where the model is fit, and is labeled so.

Details

Without reliabilities, pooling uses $z_i = \operatorname{atanh}(r_i)$ with sampling variance $1/(n_i - 3)$; with them, the effect sizes and their sampling variances are those given under Corrected correlations below. Either way, the pooled estimate, its confidence limits, and the prediction interval are transformed back through $\tanh$. The heterogeneity quantities (tau, tau-squared, I-squared, H-squared, Q) remain on the Fisher's *Z* scale, where the model lives; tau is therefore the between-study standard deviation of the Fisher's *Z* values.

Corrected correlations. With $a_i = \rho_{xx,i}\,\rho_{yy,i}$, the corrected correlation is $r_{c,i} = r_i/\sqrt{a_i}$ and the effect size pooled is $\operatorname{atanh}(r_{c,i})$. The correction divides the sampling error of r_i by $\sqrt{a_i}$ along with r_i itself, so a corrected correlation is less precise than the observed correlation it came from, and the Fisher's *Z* transformation, which makes the sampling variance of $\operatorname{atanh}(r_i)$ free of the population correlation, does not do the same for $\operatorname{atanh}(r_{c,i})$. The sampling variance used is the delta method approximation

$$v_i = \frac{(1 - r_i^2)^2}{a_i(1 - r_{c,i}^2)^2(n_i - 3)},$$

which is $1/(n_i - 3)$, the sampling variance of $\operatorname{atanh}(r_i)$, times the square of the derivative of $\operatorname{atanh}(r_{c,i})$ with respect to $\operatorname{atanh}(r_i)$. That derivative exceeds 1 whenever $a_i < 1$, and it grows without bound as $r_{c,i}$ approaches 1 in magnitude. It is evaluated at each study's observed correlation, so v_i depends on that study alone, as the model of `meta_es` takes the sampling variances to be known. With no correction ($a_i = 1$), v_i is $1/(n_i - 3)$ exactly. The v_i enter wherever the sampling variances do: the weights, the pooled estimate with its standard error and confidence interval, the between-study variance with its interval, I-squared, H-squared, Q, and the prediction interval. For five studies with $r = 0.40$, $n = 100$, and both reliabilities 0.80, each corrected correlation is 0.50 and each v_i is 1.96 times $1/(n_i - 3)$; the fixed effect interval for the correlation between the constructs is 0.401 to 0.588.

The reliabilities are treated as known constants. When they are estimates, from the same samples or from other sources, their sampling error is a further source of uncertainty in each corrected correlation that v_i does not include, and the intervals are then narrower than the total uncertainty warrants, by an amount that depends on how precisely the reliabilities are known. Because v_i grows with the magnitude of the study's own correlation, a study whose corrected correlation is larger in magnitude receives less weight, which moves the pooled estimate toward zero. In simulations run for this function (10 studies, a correlation of 0.5 between the constructs, reliabilities from 0.60 to 0.95, 30 to 400 observations per study, and 4,000 sets of studies per condition, so that a coverage rate near 0.95 has a Monte Carlo standard error of about 0.0035), the pooled estimate was, on average, about 0.003 to 0.015 below the construct correlation on the Fisher's *Z* scale, depending on the condition. When the effects were homogeneous, the fixed effect interval covered at the nominal 0.95 within Monte Carlo error (0.950 to 0.955; 0.80 to 0.83 with the variance of an uncorrected correlation). With heterogeneous effects (a between-study standard deviation of 0.15 on the Fisher's *Z* scale of the construct correlations), the interval for tau-squared covered at 0.95 to 0.97 (0.79 to 0.85 with the variance of an uncorrected correlation) and the Hartung-Knapp interval at 0.93 to 0.95. The delta method approximation is poorer when n_i is small or a corrected correlation is near 1 in magnitude: for a correlation of 0.7 between the constructs, reliabilities of 0.7, and 60 observations per study, a corrected correlation reached 1 in 7 percent of the sets of studies, and in the rest the pooled estimate was, on average, about 0.08 below the construct correlation on the Fisher's *Z* scale and the intervals covered at 0.84 to 0.87. The artifact distribution methods of Hunter and Schmidt (2004) are not implemented. A corrected correlation that reaches or exceeds 1 in magnitude (possible when an observed *r* outruns the supplied reliabilities) is an error at the pooling

stage, unlike the single-study correction_for_attenuation, which reports it with a warning: atanh is undefined there.

Value

A data.frame (class dmar_meta, a dmar_tbl) with the same rows and attributes as meta_es: the estimate, ci_lower / ci_upper, and prediction interval rows in the correlation metric; the se, test_statistic, and heterogeneity rows on the Fisher's Z scale where the model lives. The reference distribution of test_statistic is in the test_distribution and test_df attributes (Student's t on $k - 1$ degrees of freedom under the Hartung-Knapp adjustment, the standard normal without it). With reliabilities supplied, every row describes the corrected correlations, with the sampling variances given in Details. When the Hartung-Knapp adjustment is applied, the attributes hartung_knapp_q and se_conventional hold the weighted residual mean square and the conventional standard error, both on the Fisher's Z scale.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Hunter, J. E., & Schmidt, F. L. (2004). *Methods of meta-analysis: Correcting error and bias in research findings* (2nd ed.). Sage.

See Also

meta_es for the engine; correction_for_attenuation for the single-study correction and its connection to latent variable modeling; reliability for estimating the reliabilities; convert_r_Z / convert_Z_r for the transformation used.

Other meta-analysis: combine_p(), meta_contrast(), meta_es(), meta_smd(), plot_forest()

Examples

```
# Five validity studies of the same selection instrument.
r <- c(.28, .35, .22, .40, .31)
n <- c(120, 85, 200, 60, 150)
meta_r(r, n)

# The same studies corrected for criterion unreliability (reliability
# 0.80 in every study): the construct-level validity. Its interval is
# wider than the one above: the correction spreads the corrected
# correlations farther apart and enlarges their sampling variances.
meta_r(r, n, reliability_y = 0.80)
```

`meta_smd` *Random Effects Meta-Analysis of Standardized Mean Differences*

Description

Pools two-group standardized mean differences across independent studies. Each study contributes its standardized mean difference and per-group sample sizes; the function computes the within-study sampling variances, applies the Hedges small-sample bias correction by default (the same J factor as `expected_smd` and `smd`), and fits the random effects model of `meta_es`, returning the pooled effect with its confidence interval, tau and tau-squared with intervals, I-squared, Cochran's Q, and a prediction interval for the effect in a new study.

Usage

```
meta_smd(
  smd,
  n_1,
  n_2,
  unbiased = TRUE,
  method = c("reml", "pm", "dl", "fe"),
  hartung_knapp = TRUE,
  conf_level = 0.95
)
```

Arguments

`smd`	Numeric vector of standardized mean differences (Cohen's d), one per study, positive in the direction of the common hypothesis.
`n_1`, `n_2`	Per-group sample sizes for each study.
`unbiased`	Logical: convert each d to Hedges g (the small-sample unbiased estimator) before pooling? Default `TRUE`. Set `FALSE` to pool the raw d values, for example when reproducing a historical analysis such as Raudenbush (1984) that predates routine use of the correction.
`method`, `hartung_knapp`, `conf_level`	Passed to `meta_es`: the tau-squared estimator (`"reml"` default), the Hartung-Knapp small-sample adjustment (default `TRUE`), and the confidence level. Under the adjustment, identical effect sizes give a zero adjusted standard error and a warning of class `dmar_meta_hk_zero_se`; see `?meta_es`.

Details

The within-study variance is the standard large-sample form

$$v_i = \frac{n_{1i} + n_{2i}}{n_{1i} n_{2i}} + \frac{g_i^2}{2(n_{1i} + n_{2i})},$$

computed from the bias-corrected g_i when `unbiased = TRUE` (Hedges, 1981; Borenstein, Hedges, Higgins, & Rothstein, 2009). All reported quantities are in the standardized mean difference metric.

Value

A data.frame (class dmar_meta, a dmar_tbl) with the same rows and attributes as meta_es, the attributes hartung_knapp_q and se_conventional included when the Hartung-Knapp adjustment is applied. The test statistic is the row test_statistic, and the test_distribution and test_df attributes give its reference distribution.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Borenstein, M., Hedges, L. V., Higgins, J. P. T., & Rothstein, H. R. (2009). *Introduction to meta-analysis*. Wiley.

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Raudenbush, S. W. (1984). Magnitude of teacher expectancy effects on pupil IQ as a function of the credibility of expectancy induction: A synthesis of findings from 18 experiments. *Journal of Educational Psychology, 76*(1), 85–97.

See Also

meta_es for the engine and the reported rows; smd and ci_smd for the single-study quantities; plot_forest for the picture; teacher_expectancy for the example data.

Other meta-analysis: combine_p(), meta_contrast(), meta_es(), meta_r(), plot_forest()

Examples

```
# Pool the teacher expectancy studies (Raudenbush, 1984). Hedges g and
# the Hartung-Knapp adjustment are on by default; the prediction
# interval shows where a new expectancy study would be expected to land.
data(teacher_expectancy)
meta_smd(smd = teacher_expectancy$d,
         n_1 = teacher_expectancy$n_experimental,
         n_2 = teacher_expectancy$n_control)
```

---

missing_coverage *Covariance Coverage of Incomplete Data*

Description

Reports, for every pair of items, the number and proportion of cases on which both were observed (the covariance coverage), with each item's own response rate on the diagonal. A covariance or a regression slope is informed most directly by the cases that observed both of its variables, so the coverage of a pair is a first, model-free look at how much data stand behind the parameters that involve it. Full information maximum likelihood (FIML) uses every case, and the other variables

can carry information about a pair with low coverage, but a pair with no jointly observed case leaves its covariance with no direct information at all: its fraction of missing information is then one (Savalei & Rhemtulla, 2012, p. 488). `fraction_missing_information` gives that fraction for each parameter of a fitted model.

Usage

```
missing_coverage(data, items = NULL)
```

Arguments

`data`	A `data.frame`.
`items`	Optional character vector naming the columns of `data` to describe; `NULL` (the default) uses every column. Any column type is allowed, since only whether a value is `NA` matters.

Details

The denominator of every proportion is the number of cases with at least one observed item, the cases a FIML fit uses. That is the denominator of `lavaan::lavInspect(fit, "coverage")`, which the `matrix` attribute reproduces for a single group fit to the same items; cases with every item missing are counted in the `N_all_missing` attribute instead.

Value

A `data.frame` (class `dmar_tbl`) with one row per pair of items, the item itself included, in the order of `items` (the pairs (i, j) with $i \leq j$), and the columns `var_1`, `var_2`, `n_both` (the cases that observed both; for `var_1 == var_2`, the cases that observed the item), and `coverage` (`n_both` as a proportion of the cases with at least one observed item). The attributes are `matrix` (the symmetric coverage matrix), `N_used` (the denominator), `N_all_missing`, and `min_coverage`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Savalei, V., & Rhemtulla, M. (2012). On obtaining estimates of the fraction of missing information from full information maximum likelihood. *Structural Equation Modeling, 19*(3), 477–494. doi:10.1080/10705511.2012.687669

See Also

`missing_patterns` for the patterns behind the coverage; `fraction_missing_information` for what the missing values cost each parameter of a FIML fit.

Other missing data: `cov_fiml()`, `df_barnard_rubin()`, `df_missing()`, `fraction_missing_information()`, `missing_patterns()`

Examples

```
# The revised paper form board (t25) and the flags test (t26) were
# given only at Grant-White, so every pair involving either is covered
# by the 145 Grant-White pupils, 48 percent of the 301.
spatial <- c("t1_visual_perception", "t2_cubes", "t3_paper_form_board",
             "t4_lozenges", "t25_paper_form_board_r", "t26_flags")
cov_tbl <- missing_coverage(holzinger_swineford, items = spatial)
cov_tbl
round(attr(cov_tbl, "matrix"), 2)
```

`missing_patterns` *Missing Data Patterns*

Description

Tabulates the distinct patterns of observed and missing values across a set of items: which items each pattern leaves missing, and how many cases show it. The table is the first description of incomplete data, the one a reader needs before any model is fit. It shows whether the missing values are scattered across many patterns or concentrated in a few, whether they arrive together (two tests given to only one school share one pattern), and how many cases are complete, which is all listwise deletion would keep. Full information maximum likelihood (FIML) uses every pattern, and the fraction of missing information (`fraction_missing_information`) then says what the patterns cost each parameter.

Usage

```
missing_patterns(data, items = NULL)
```

Arguments

`data`	A `data.frame`.
`items`	Optional character vector naming the columns of `data` to describe; `NULL` (the default) uses every column. Any column type is allowed, since only whether a value is `NA` matters.

Details

A pattern is written as a string of 1s and 0s, one character per item in the order of `items`, with 1 for an observed value and 0 for a missing one (the convention of `mice::md.pattern()`). The rows are sorted by the number of cases, largest first, then by the number of items missing, fewest first, so the complete cases usually lead and the rare patterns trail.

The pattern in which every item is missing is kept as a row (`n_observed = 0`) when it occurs. **lavaan** sets such cases aside before fitting, without reporting them among its own patterns, so the number of cases a FIML fit uses is `N` minus `N_all_missing`.

Value

A data.frame (class dmar_tbl) with one row per distinct pattern and the same columns whatever the number of items: pattern (the string of 1s and 0s), missing_items (the names of the items the pattern leaves missing, separated by commas; "" for the complete pattern), n (the number of cases), proportion (of all rows of data), n_observed, and n_missing (the numbers of items observed and missing in the pattern). The attributes are matrix (the logical pattern matrix, TRUE for observed, one row per pattern in the order of the table, the layout of lavaan::lavInspect(fit, "patterns")), N (the rows of data), N_complete, N_all_missing, n_patterns, and items.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

missing_coverage for the proportion of cases observed on each pair of items; fraction_missing_information for what the missing values cost each parameter of a FIML fit.

Other missing data: cov_fiml(), df_barnard_rubin(), df_missing(), fraction_missing_information(), missing_coverage()

Examples

```
# The six spatial tests of the Holzinger and Swineford data: two
# patterns, the 145 Grant-White pupils with every test and the 156
# Pasteur pupils, who were not given the revised paper form board
# (t25) or the flags test (t26).
spatial <- c("t1_visual_perception", "t2_cubes", "t3_paper_form_board",
             "t4_lozenges", "t25_paper_form_board_r", "t26_flags")
missing_patterns(holzinger_swineford, items = spatial)

# Values missing completely at random on two more tests add patterns.
set.seed(113)
d <- holzinger_swineford[spatial]
d$t1_visual_perception[runif(nrow(d)) < .10] <- NA
d$t2_cubes[runif(nrow(d)) < .10] <- NA
pat <- missing_patterns(d)
pat
attr(pat, "N_complete")
```

---

mixed_anova *Mixed-Model ANOVA F-Ratios for One- and Two-Way Designs*

Description

Computes the classical *F*-ratios for one- and two-way ANOVA designs when one or more factors are random rather than fixed, using the expected-mean-square (EMS) rules that determine the correct denominator for each test (Searle, Casella, & McCulloch, 1992). The function is the closed-form alternative to fitting via `lmer` and is the standard treatment in classical psychometrics and design-of-experiments texts (Maxwell, Delaney, & Kelley, 2027, Ch. 10).

Usage

```
mixed_anova(
  data,
  outcome,
  factor_A,
  factor_B = NULL,
  A_type = c("fixed", "random"),
  B_type = c("random", "fixed")
)
```

Arguments

`data`	A `data.frame` containing the response, the factor(s), and (when both factors are crossed) the subject identifier.
`outcome`	Character name of the response column.
`factor_A`	Character name of factor *A*.
`factor_B`	Character name of factor *B*, or `NULL` for one-way designs. Default `NULL`.
`A_type`	One of `"fixed"` (default) or `"random"`: the EMS classification of factor *A*.
`B_type`	One of `"fixed"` or `"random"` (default): the EMS classification of factor *B*. Ignored when `factor_B` is `NULL`.

Details

One way design (only `factor_A`).

- Both *A* fixed and *A* random use the same observed *F*-ratio MS_A/MS_{within}; the test of "is there an effect of *A*?" is identical. The interpretation differs: the random-effects test asks whether the variance component σ^2_A is zero.

Two-way design, both fixed (Model I).

- $F_A = MS_A/MS_{AB}$ (when interaction is present in the model and treated as error) or $F_A = MS_A/MS_{\text{within}}$ (when interaction is pooled into error). The function uses MS_{within} as the denominator throughout for Model I.

Two-way design, both random (Model II).

- $F_A = MS_A/MS_{AB}$, $F_B = MS_B/MS_{AB}$, $F_{AB} = MS_{AB}/MS_{\text{within}}$.

Two-way mixed design (Model III, e.g., *A* fixed, *B* random).

- $F_A = MS_A/MS_{AB}$ (fixed factor against the interaction with the random factor)

- $F_B = MS_B / MS_{\text{within}}$ (random factor against the within-cell residual)
- $F_{AB} = MS_{AB} / MS_{\text{within}}$.

Balanced data assumed. The classical EMS rules require equal cell sizes. The function errors out on unbalanced data and recommends a mixed-effects fit via `lmer`.

Sums of squares. For the balanced designs this function targets, the Type I, Type II, and Type III sums of squares for each effect coincide, so the decomposition is unambiguous and no sums-of-squares type needs to be chosen (Maxwell, Delaney, & Kelley, 2027, Ch. 7). The returned object carries a numeric `sum_of_squares_type` attribute equal to 3, with the understanding that it equals Types I and II here; it records the convention without implying a choice that would matter for these designs.

Value

A `data.frame` (class `dmar_tbl`) with one row per testable effect. Columns: `effect`, `ss`, `df`, `ms`, `denominator` (the mean square the effect is tested against), `F_value`, `p_value`. Its numeric `sum_of_squares_type` attribute is 3 (see Details). The table prints through the `dmar_tbl` display layer: whole numbers such as the degrees of freedom without a decimal part, other values to 3 decimal places, and *p*-values to 4 decimal places (one below 0.0001 prints as "< 0.0001"); the stored values keep full precision. `tidy()` gives one row per effect, its `term` joining `effect` and `denominator` and its `estimate` the `ss`, and `glance()` gives one row, the number of rows (`n_terms`) with `conf_level` and `B_used` `NA`, since no interval or bootstrap is formed (see `dmar_tidiers`). `as.data.frame()` returns the table as a plain `data.frame` with the same columns, values, and attribute.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 7 on the sums of squares for balanced designs and Chapter 10 on random and mixed effects.)

Searle, S. R., Casella, G., & McCulloch, C. E. (1992). *Variance components*. Wiley.

See Also

`anova_within`, `anova_within_two_way`, `lmer`

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Other mixed models: `R2_mixed_effects()`, `R2_mixed_effects_decomposition()`, `icc_lmer()`, `manova_split_plot()`, `ss_aipe_mixed_effects()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_power_mixed_effects()`, `ss_power_split_plot_anova()`

Examples

```
# 1. Two-way mixed design: A fixed, B random.
set.seed(113)
grid <- expand.grid(A = factor(1:3), B = factor(1:5), rep = 1:4)
grid$y <- with(grid, 0.8 * as.integer(A) + rep(rnorm(5, 0, 1.5), each = 1)[as.integer(B)] +
                     rnorm(nrow(grid), 0, 1))
mixed_anova(grid, outcome = "y", factor_A = "A", factor_B = "B",
            A_type = "fixed", B_type = "random")
```

`mlmr` *Maximum Likelihood Multiple Regression*

Description

Fits a multiple regression model by maximum likelihood, with full information likelihood handling of missing values by default. The formula interface and S3 methods mirror `lm` so that calls such as `coef()`, `vcov()`, `confint()`, `summary()`, `fitted()`, `residuals()`, and `predict()` continue to work. Confidence intervals default to the likelihood ratio (profile) form; Wald and bootstrap variants are also available.

Usage

```
mlmr(
  formula,
  data,
  missing = c("fiml", "ml", "listwise", "pairwise", "available.cases"),
  ci_method = c("profile", "wald", "boot"),
  conf_level = 0.95,
  B = 1000L,
  boot_type = c("ordinary", "bollen.stine"),
  boot_seed = NULL,
  estimator = c("ML", "MLR", "MLM", "GLS"),
  se = NULL,
  fixed_x = FALSE,
  auxiliary = NULL,
  effect_sizes = TRUE,
  enforce_es_bounds = FALSE,
  ...,
  wald_df = Inf
)
```

Arguments

`formula`	A two-sided `formula` of the form `y ~ x1 + x2 + ...`. Factor predictors, interactions (`x1 * x2`), polynomial terms (`poly(x, 2)`), and transformations (`I(x^2)`) are supported through the usual `model.matrix` expansion. An intercept-only

formula, `y ~ 1`, fits the null model (the mean and the residual variance only); it is the natural restricted model in a model comparison and pairs with `anova()` for a likelihood ratio test against a fuller model.

`data` A `data.frame` containing the variables in `formula`. Rows with missing values on *any* of the modeled variables are retained (under `missing = "fiml"`) and contribute to the likelihood through whichever components are observed.

`missing` Character; how missing values are handled. The default `"fiml"` (equivalently `"ml"` in **lavaan**) uses the full information maximum likelihood over all rows that have at least one observed value on the modeled variables. Other choices are `"listwise"` (drop rows with any missing modeled variable), `"pairwise"` (sample moments computed pairwise; not recommended with this model), and `"available.cases"`.

`ci_method` Character; the method for confidence intervals on the regression coefficients. `"profile"` (default) inverts the likelihood ratio test on each parameter through a sequence of constrained refits. `"wald"` returns the symmetric estimate $\pm\, z_{1-\alpha/2}$ standard error interval. `"boot"` resamples the rows of `data` `B` times, refits the model on each resample, and reports the percentile interval of the resampled slopes.

`conf_level` Desired level of confidence (the complement of the Type I error rate). Defaults to `0.95`.

`B` Number of bootstrap resamples when `ci_method = "boot"`, a single whole number. Default `1000`, because every resample refits the model. The fewest accepted is 400 at the default `conf_level` (200 at 0.90, and 2000 at 0.99, which the default does not meet); a smaller `B` is refused before anything is computed (class `dmar_count_invalid`). See *Bootstrap replications* under *Details* and `dmar_bootstrap`.

`boot_type` Character; `"ordinary"` (default) for the nonparametric resampling of rows, or `"bollen.stine"` for the Bollen and Stine (1992) model-based bootstrap implemented in **lavaan**.

`boot_seed` Integer or `NULL`; optional seed for the bootstrap resampling RNG. The default `NULL` draws the resamples from, and advances, the current random number stream, so successive bootstrap calls draw fresh resamples; supply an integer to make the bootstrap reproducible. When supplied, the function seeds the RNG locally and restores the prior state on exit so the user's global RNG is not polluted.

`estimator` Character; the **lavaan** estimator. One of `"ML"` (default), `"MLR"`, `"MLM"`, or `"GLS"`. `ML` is standard maximum likelihood under conditional normality of *Y*. `MLR` is robust maximum likelihood with Yuan-Bentler scaled standard errors and a Yuan-Bentler scaled test statistic, recommended when the conditional distribution of *Y* departs from normality and FIML is in use (Yuan & Bentler, 2000). `MLM` is Satorra- Bentler scaled χ^2 statistics under complete data (Satorra & Bentler, 1994). `GLS` is normal theory generalized least squares. The ordinal-data estimators (`"DWLS"`, `"WLS"`, `"ULS"` and their robust variants) are intentionally not exposed; for ordinal outcomes, fit a different model class.

`se` Character or `NULL`; the standard error type passed to **lavaan**. When `NULL` (default), the standard error type is chosen automatically from `estimator`: `"ML"` and `"GLS"` use `"standard"`, `"MLR"` uses `"robust.huber.white"` (Huber-White

heteroskedasticity consistent), and "MLM" uses "robust.sem" (Satorra-Bentler). Override only when the default does not match the desired analysis. Other values include "robust", "first.order", and "none".

fixed_x — Logical; whether to treat predictors as fixed (not modeled jointly) or as random (jointly modeled). Defaults to FALSE, which is required for the full information likelihood to use rows with missing predictors. Set to TRUE only when the predictors are fully observed and the user wants the lm-style conditional model.

auxiliary — Character vector of variable names in data to include as auxiliary variables, or NULL (default) for none. Auxiliaries are entered as saturated correlates (Graham, 2003): correlated with the outcome residual, every predictor, and each other, but not as predictors, while the predictors keep their free covariances, so the model is saturated, the regression coefficients keep their meaning, and the full information maximum likelihood draws on the auxiliaries' observed values (the inclusive analysis strategy; Collins, Schafer, & Kam, 2001). A name in auxiliary must be numeric, must be present in data, must not appear in formula, and requires fixed_x = FALSE. Auxiliaries are refused with missing = "listwise", which would drop every row on which an auxiliary is missing and so change the sample the coefficients describe, and with missing = "pairwise" or "available.cases", which compute the moments from each pair of variables on its own, so that an auxiliary would leave every estimate unchanged.

effect_sizes — Logical; whether to compute regression effect sizes (standardized betas, semi-partial R^2, Cohen's f^2 per predictor, and the overall LR omnibus test). Defaults to TRUE. Disabling saves $p + 1$ additional lavaan refits.

enforce_es_bounds — Logical; whether to clamp semi-partial R^2 and Cohen's f^2 estimates to their theoretical lower bound of zero. Defaults to FALSE: the raw maximum likelihood estimates of R^2_{reduced} and R^2_{full} are reported as-is, and the difference can be slightly negative as a finite-sample artifact when the two are nearly equal. Setting to TRUE replaces any negative value with zero, which yields an estimate that respects the parameter space but is no longer the maximum likelihood estimate. When the clamp fires, the affected rows of the returned effect_sizes table carry the attribute "clamped" for diagnostics.

... — Additional arguments forwarded to lavaan.

wald_df — The reference distribution of the Wald interval and of the *p*-values: Inf (the default), the normal distribution; a single positive number, the degrees of freedom of a *t* distribution for every coefficient; or "missing_information", von Hippel's (2016) degrees of freedom $\max\{3, N(1-\gamma_j) - k\}$ for each coefficient, from its fraction of missing information γ_j. A finite value or the keyword requires ci_method = "wald". The keyword is an error where, with values missing, **lavaan** fit the complete cases in place of the FIML fit that missing = "fiml" asks for, as it does for se = "robust.sem" or a test statistic of the Satorra-Bentler kind passed through ...; with no value missing that fit is the FIML fit, and the keyword gives $N - k$. It follows ..., so it must be named in full. See *Details*.

Details

Why a separate function from `lm`. `lm` uses ordinary least squares and listwise deletes any row with a missing value on the outcome or on any predictor. Two situations motivate a maximum likelihood alternative.

First, when a predictor is missing on some rows, listwise deletion can be biased if the missingness mechanism depends on other observed variables (the missing at random or MAR pattern). Full information maximum likelihood (FIML) jointly models the distribution of $(Y, X_1, \ldots, X_p)$, the outcome and the p predictors, and yields consistent regression estimates under MAR, while listwise estimates can be biased away from the population values (Enders, 2010; Schafer & Graham, 2002).

Second, the joint likelihood estimates the predictor distribution as well, so quantities that depend on the predictor moments (the standardized coefficients, the model implied R^2, and the predictor variances and covariances) draw on every row with an observed predictor, not only the rows that are complete on the outcome. When the missing values are confined to the outcome, however, the unstandardized slopes and their standard errors match listwise deletion up to the maximum likelihood N versus $N - p - 1$ variance divisor. The rows with an observed predictor but a missing outcome inform the marginal distribution of *X*, not the conditional distribution of *Y* given *X* that identifies the slopes, so they leave the slope estimates and their conditional-model standard errors unchanged.

Full information estimation differs from listwise deletion in the slope estimates themselves when any predictors are missing on some rows, or when auxiliary variables that correlate with the outcome or with the missingness mechanism are supplied through `auxiliary` (see below). The bootstrap gives inference that does not depend on the multivariate normality assumption.

Auxiliary variables. A variable that is not part of the regression but is correlated with the outcome or with the missingness can be supplied through `auxiliary`. Auxiliaries are added to the model as saturated correlates (Graham, 2003): each one is correlated with the residual of the outcome, with every predictor, and with every other auxiliary, but is never entered as a predictor, and the predictors keep their free means, variances, and covariances. The model over the outcome, the predictors, and the auxiliaries is then saturated (its chi square is zero on zero degrees of freedom), so the auxiliaries impose no restriction and the focal regression coefficients keep their meaning: on complete data they equal those of `lm()` to working precision, and with missing values they are the regression implied by the full information maximum likelihood estimates of the means and covariances of every variable, auxiliaries included, which `cov_fiml` computes. The full information maximum likelihood uses the auxiliaries' observed values to make the MAR assumption hold conditional on more of the observed data and to recover information that listwise deletion discards. This is the inclusive analysis strategy of Collins, Schafer, and Kam (2001): a variable that predicts the missingness or the incomplete outcome belongs in the analysis even when it is of no substantive interest. Auxiliary variables must be numeric and require `fixed_x = FALSE` (the default).

Why likelihood ratio confidence intervals by default. Wald intervals (point estimate $\pm\, z_{1-\alpha/2}$ standard error) are symmetric by construction and assume the sampling distribution of the estimator is approximately normal over the relevant range. Likelihood ratio intervals invert the likelihood ratio test directly: an interval contains every value of the parameter that would not be rejected at level α. Likelihood ratio intervals are invariant under monotone reparameterizations, often have better coverage in small samples, and respect parameter boundaries (Pawitan, 2001). The cost is computational: each parameter requires a sequence of refits with that parameter constrained. The examples below ask for the Wald and bootstrap intervals so the help page stays quick; a reported analysis leaves `ci_method` at its default.

A *t* reference distribution for the Wald interval. The Wald interval and every *p*-value refer the ratio of a coefficient to its standard error to the normal distribution, which treats the standard error as known. That holds only in the limit: in small samples, and more so when values are missing, the normal intervals are too short. In von Hippel's (2016, pp. 432–433) simulations of bivariate normal data with half of one variable missing, the normal FIML intervals for the intercept and slope of a regression covered 5 to 6 points below the nominal 95% at $n = 25$, and 0 to 3 points below it at $n = 100$. He kept the maximum likelihood estimate and its standard error and replaced the normal quantile with a *t* quantile (his Eq. 29), on degrees of freedom that take the missing information into account. With `ci_method = "wald"`, `wald_df` does this. A number refers every coefficient's interval and *p*-value to the *t* distribution on that many degrees of freedom. The keyword `"missing_information"` applies von Hippel's rule for regression coefficients (his Eqs. 31 and 34) to each coefficient separately,

$$\nu_j = \max\{3, N(1 - \gamma_j) - k\},$$

the complete data degrees of freedom $N - k$ of a regression with k parameters (the slopes and the intercept) with the sample size replaced by the coefficient's effective sample size, where N is the number of cases the fit used and γ_j the coefficient's fraction of missing information. The fraction is estimated from the fit as `fraction_missing_information` estimates it, by the unstructured information of Chen and Savalei (2021), which cannot be negative. The model leaves the means and covariances of the outcome, the predictors, and any auxiliary variables unrestricted (it is saturated), and the three estimators of Chen and Savalei then coincide up to numerical error (Chen & Savalei, 2021, pp. 3, 9), so the fraction agrees with **lavaan**'s `parameterEstimates(fit, fmi = TRUE)` to the precision of **lavaan**'s own computation. Unlike **lavaan**'s, its computation does not depend on the units of the variables: an outcome measured in dollars, with a standard deviation of 100,000,000, gets the degrees of freedom it gets in millions of dollars (see `fraction_missing_information`). It is computed at the fit's estimates, which **lavaan** can leave short of the maximum of the likelihood for variables measured in small units (with every standard deviation near .003 the intercept's degrees of freedom moved by .15); the call then warns, and `rescale_data = TRUE`, passed through `...` to **lavaan**, is the remedy. A fraction that cannot be computed stops the call with the reason, rather than leave a coefficient with no degrees of freedom. A coefficient estimated mostly from fully observed variables keeps nearly all of the complete data degrees of freedom, and with no missing values every coefficient has $N - k$. The floor of 3 keeps a *t* distribution with finite variance when the estimated fraction is near one. `df_missing` computes the same degrees of freedom from any fractions of missing information, and its help page describes both of von Hippel's rules and the evidence for them; the table it returns for the fit is the `wald_df_table` attribute. Two cautions. The *t* interval is an approximation, since the small-sample distribution of a maximum likelihood estimate from incomplete data is not exactly *t* (von Hippel, 2016, p. 431). And the standard error is the maximum likelihood one, whose variance estimate divides by N rather than $N - k$, so with complete data the interval on $N - k$ degrees of freedom is shorter than the interval `lm()` reports by the factor $\sqrt{(N - k)/N}$: von Hippel's construction pairs the maximum likelihood standard error with the *t* quantile, and it is reproduced here as he published it, without a variance correction. The default, `Inf`, is the normal reference. The `test_statistic` column holds the ratio of the estimate to its standard error, the same number under either reference. `summary()` heads the printed coefficient table `t value` and `Pr(>|t|)` under a *t* reference, as `summary.lm()` does, and names the degrees of freedom directly beneath it.

The bootstrap interval. With `ci_method = "boot"` the rows of `data` are resampled with replacement `B` times (1000 by default) and the model is refit on each resample; with `boot_type = "bollen.stine"` the resamples are instead drawn from data transformed to satisfy the fitted model

(Bollen & Stine, 1992), a model-based bootstrap. Only the percentile interval is offered: each coefficient's limits are the empirical quantiles of its resampled estimates (Efron & Tibshirani, 1993); there is no bias-corrected and accelerated (BCa) or bootstrap standard error variant. Raising `B` tightens the Monte Carlo error of the reported limits. Bootstrap results vary from run to run; supply `boot_seed` for reproducibility.

Bootstrap replications. Each coefficient's limits are quantiles of its `B` resampled estimates, so the interval is run only when `B` puts at least 10 resamples beyond each limit (400 at the 95 percent level). A replication in which some coefficient is undefined (the refit fails or does not converge, or returns a coefficient that is not a finite number) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as elements of the fitted object, and `print()` and `summary()` state them in one line, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the limits of every coefficient are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. Such a resample is dropped for every coefficient, so all intervals rest on the same resamples. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the result is not one to report. See `dmar_bootstrap` for the rule and its reasons.

Model representation. Internally the model is fit through **lavaan** as a structural equation model in which Y is regressed on the predictors and (when `fixed_x = FALSE`) the predictor distribution is also estimated. With `fixed_x = FALSE` and complete data, point estimates of the slopes are identical to `lm` and the residual variance differs only by the usual N versus $N - p - 1$ divisor.

Caveats. The function assumes that, conditional on the modeled predictors, the dependent variable is normally distributed with constant variance. Missingness is assumed to be at most MAR; missing not at random patterns require selection or pattern mixture models outside the scope of this function. Factor predictors and interactions are expanded through `model.matrix` and entered as numeric covariates, so the same caveats about dummy variable encoding that apply to `lm` apply here as well.

Value

An object of class `"mlmr"`, a list with components modeled on the structure of an `lm` fit:

`call` The matched call.

`formula` The model formula.

`terms` The terms object.

`model` The model frame (with missing values preserved when `missing = "fiml"`).

`coefficients` Named numeric vector of regression coefficients, with `(Intercept)` first when an intercept is in the formula.

`vcov` The variance-covariance matrix of the regression coefficients, returned by `vcov()`.

`ci` A two-column matrix (`lower`, `upper`) of confidence limits in the order of `coefficients`.

`ci_method` Which method was used to compute `ci`.

`B`, `B_used`, `n_dropped`, `B_required`, `bootstrap_status` Present only with `ci_method = "boot"`: the resamples requested, the number the intervals rest on, their difference, the fewest resamples a request may ask for, and the verdict (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`; under the last two the limits are `NA`). See *Bootstrap replications* under Details.

`conf_level` The confidence level used.

`coef_table` A `data.frame` with columns `term`, `estimate`, `se`, `test_statistic`, `p_value`, `ci_lower`, `ci_upper`, carrying the confidence level in its `conf_level` attribute, so that `results_sentence` on a row reports it.

`sigma2` Residual variance of *Y*, on the maximum likelihood scale (divisor *N*, not $N - p - 1$).

`R2` Model implied squared multiple correlation, $1 - \hat{\sigma}_e^2 / \hat{\sigma}_Y^2$, where both variances come from the FIML estimated model implied covariance matrix.

`adj_R2` Adjusted R^2 using the number of complete cases (`N_complete`, the rows complete on the outcome and every predictor, which are the rows that identify the regression) and the number of slopes; the lavaan reported `N` can be larger under FIML because it counts rows that inform only the predictor distribution.

`logLik` The log likelihood at the maximum, with attributes `df` and `nobs` for compatibility with `stats::AIC` and `stats::BIC`.

`N` Sample size used by lavaan (rows with at least one observed value when `missing = "fiml"`; rows with no missing values when `missing = "listwise"`).

`N_complete` Number of rows that are complete on all modeled variables.

`fitted.values` Vector of fitted values, length `nrow(data)`, with `NA` for rows missing any predictor.

`residuals` Vector of residuals (`y - fitted`), with `NA` where `y` or any predictor was missing.

`lavaan_fit` The underlying **lavaan** fit object, returned for advanced users who want to apply **lavaan** accessors directly.

The object carries the attribute `wald_df`, the degrees of freedom of the *t* reference of the Wald interval and the *p*-values: `Inf` for the normal reference, the number supplied, or under `wald_df = "missing_information"` a named vector with one value per coefficient. In the last case the attribute `wald_df_table` is the `df_missing` table the values came from, with each coefficient's fraction of missing information, N, k, and whether the floor applied.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bollen, K. A., & Stine, R. A. (1992). Bootstrapping goodness of fit measures in structural equation models. *Sociological Methods & Research, 21*, 205–229. doi:10.1177/0049124192021002004

Chen, L., & Savalei, V. (2021). Three sample estimates of fraction of missing information from full information maximum likelihood. *Frontiers in Psychology, 12*, 667802. doi:10.3389/fpsyg.2021.667802

Collins, L. M., Schafer, J. L., & Kam, C.-M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods, 6*(4), 330–351. doi:10.1037/1082989X.6.4.330

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford Press.

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

Pawitan, Y. (2001). *In all likelihood: Statistical modelling and inference using likelihood*. Oxford, UK: Oxford University Press.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis. In A. von Eye & C. C. Clogg (Eds.), *Latent variables analysis: Applications for developmental research* (pp. 399–419). Sage.

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods, 7*, 147–177. doi:10.1037/1082989X.7.2.147

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*(1), 165–200. doi:10.1111/00811750.00078

See Also

lm, sem, lavaan, ci_reg_coef, ci_rc, ci_src.

Examples

```
# Complete data: the maximum likelihood estimates agree with lm() to
# working precision. The fit asks for the Wald interval; the choice
# among the intervals is taken up below.
fit_mlmr <- mlmr(t6_paragraph_comprehension ~ t5_general_information +
                   t9_word_meaning,
                 data = holzinger_swineford, ci_method = "wald")
fit_lm   <- lm(t6_paragraph_comprehension ~ t5_general_information +
                 t9_word_meaning,
               data = holzinger_swineford)
cbind(mlmr = coef(fit_mlmr), lm = coef(fit_lm))

fit_mlmr
confint(fit_mlmr)

# summary() adds the intervals, the per-predictor semi-partial R^2
# and Cohen's f^2, and the omnibus likelihood ratio test of all
# slopes equal to zero.
summary(fit_mlmr)

# The interval menu is profile, Wald, and bootstrap. The default,
# ci_method = "profile", inverts the likelihood ratio test one
```

```
# parameter at a time through a sequence of constrained refits; it
# is what a reported interval deserves, and leaving ci_method at its
# default asks for it. The fits on this page ask for the Wald or the
# bootstrap interval because those refits take longer than a help
# page should. The bootstrap resamples rows and takes percentile
# limits; it is what to ask for when the normality the likelihood
# assumes is doubtful. Every resample refits the model. B = 10 keeps
# the example quick and is below the 400 a reported interval needs;
# the option lets it run and the warning says so. A reported interval
# deserves the default B = 1000. boot_seed fixes the resamples, so
# the limits are reproducible rather than moving from run to run, and
# effect_sizes = FALSE skips the effect size refits, which
# summary() above already showed.
old <- options(dmar.bootstrap_exploratory = TRUE)
fit_boot <- mlmr(t6_paragraph_comprehension ~ t5_general_information +
                   t9_word_meaning, data = holzinger_swineford,
                 ci_method = "boot", B = 10, boot_seed = 113,
                 effect_sizes = FALSE)
options(old)
confint(fit_boot)

# Missing values on a predictor are where maximum likelihood and
# least squares part company. The full information likelihood keeps
# every row that carries information; listwise deletion keeps only
# the rows that are complete. The Holzinger and Swineford battery
# carries real missingness for this: the revised second-form test
# t26_flags was administered to only 145 of the 301 students, so a
# model using it loses more than half the sample under listwise
# deletion while the full information fit keeps all 301 rows. Notice
# the two sample sizes, and that the two fits do not agree on the
# intercept.
fit_fiml <- mlmr(t6_paragraph_comprehension ~ t7_sentence +
                   t26_flags, data = holzinger_swineford,
                 ci_method = "wald", effect_sizes = FALSE)
fit_lwd  <- mlmr(t6_paragraph_comprehension ~ t7_sentence +
                   t26_flags, data = holzinger_swineford,
                 missing = "listwise",
                 ci_method = "wald", effect_sizes = FALSE)
rbind(FIML = coef(fit_fiml), listwise = coef(fit_lwd))
c(N_fiml = nobs(fit_fiml), N_listwise = nobs(fit_lwd))

# An auxiliary variable is not a predictor. The complete speed test
# t13_straight_and_curved_capitals enters as a saturated correlate,
# correlated with the outcome residual and with the predictors, so
# the likelihood can draw on it for the rows where t26_flags is
# missing while the coefficients keep their meaning. Continuing from
# the model above, the coefficients move a little and the slope
# standard errors shrink a little: among the 145 students who took
# t26_flags its correlation with t13 is .34, so the auxiliary
# restores only part of the information the missing scores would
# have carried.
fit_aux <- mlmr(t6_paragraph_comprehension ~ t7_sentence +
                  t26_flags, data = holzinger_swineford,
```

```
                ci_method = "wald",
                auxiliary = "t13_straight_and_curved_capitals",
                effect_sizes = FALSE)
cbind(no_aux = coef(fit_fiml), aux = coef(fit_aux))
cbind(se_no_aux = sqrt(diag(vcov(fit_fiml))),
      se_aux = sqrt(diag(vcov(fit_aux))))

# With most of a variable missing, the normal reference understates the
# uncertainty in a small sample. Here the outcome t26_flags is observed
# for 145 of the 301 students, and wald_df = "missing_information"
# refers each coefficient to t on von Hippel's degrees of freedom,
# N(1 - FMI) - k, fewer than half of the N - k = 298 that complete
# data would give. The summary lists each coefficient's fraction of
# missing information and degrees of freedom under the intervals.
fit_t <- mlmr(t26_flags ~ t1_visual_perception + t4_lozenges,
              data = holzinger_swineford, ci_method = "wald",
              wald_df = "missing_information", effect_sizes = FALSE)
summary(fit_t)
```

`mlmr_mv` *Multivariate Maximum Likelihood Regression With Full Information Missing Data Handling*

Description

Fits a multivariate multiple regression model by maximum likelihood, with full information maximum likelihood handling of missing values by default. Multiple outcomes are regressed on a shared predictor set simultaneously, with the residual covariance among outcomes estimated as part of the model. The formula interface mirrors `lm`'s multivariate syntax, `cbind(y1, y2, y3) ~ x1 + x2`, and the returned object supports the same family of S3 methods as a univariate `mlmr` fit.

Usage

```
mlmr_mv(
  formula,
  data,
  missing = c("fiml", "ml", "listwise", "pairwise", "available.cases"),
  ci_method = c("profile", "wald", "boot"),
  conf_level = 0.95,
  B = 1000L,
  boot_type = c("ordinary", "bollen.stine"),
  boot_seed = NULL,
  estimator = c("ML", "MLR", "MLM", "GLS"),
  se = NULL,
  fixed_x = FALSE,
  auxiliary = NULL,
  effect_sizes = TRUE,
```

```
  enforce_es_bounds = FALSE,
  ...,
  wald_df = Inf
)
```

Arguments

formula	A two-sided formula whose left-hand side is cbind(y1, y2, ...) for two or more numeric outcomes and whose right-hand side names the shared predictor set. Factor predictors, interactions, polynomial terms, and transformations are supported as in mlmr.
data	A data.frame containing the variables in formula.
missing	Character; passed to **lavaan**. Defaults to "fiml" (equivalently "ml"). See mlmr for the full list of accepted values.
ci_method	Character; confidence interval method for the regression coefficients. "profile" (default), "wald", or "boot". The same trade-offs apply as in mlmr; with J outcomes and K predictors, profile likelihood requires $O(JK)$ constrained refits. "boot" resamples the rows of data B times (or draws Bollen-Stine model-based resamples), refits on each, drops resamples whose refit does not converge, and reports each coefficient's percentile interval, the empirical quantiles of its resampled estimates; as in mlmr, the percentile interval is the only bootstrap interval offered. The examples ask for the Wald and bootstrap intervals so the help page stays quick; a reported analysis leaves ci_method at its default.
conf_level	Desired level of confidence. Defaults to 0.95.
B	Number of bootstrap resamples when ci_method = "boot", a single whole number. Default 1000, because every resample refits the model for every outcome. The fewest accepted is 400 at the default conf_level (200 at 0.90, and 2000 at 0.99, which the default does not meet); a smaller B is refused before anything is computed (class dmar_count_invalid). See *Bootstrap replications* under *Details* and dmar_bootstrap.
boot_type	Character; "ordinary" (default) or "bollen.stine".
boot_seed	Integer or NULL. Defaults to NULL, under which the resamples are drawn from, and advance, the current random number stream; supply an integer for reproducible bootstraps. When set, the seed applies for the duration of the call and the caller's generator state is restored on exit.
estimator	Character; one of "ML" (default), "MLR", "MLM", "GLS".
se	Character or NULL; defaults to NULL, meaning "choose automatically from estimator" (same logic as mlmr).
fixed_x	Logical; defaults to FALSE (jointly model the predictor distribution, required for FIML to use rows with missing predictors).
auxiliary	Character vector of variable names in data to include as auxiliary variables (saturated correlates; Graham, 2003), or NULL (default) for none. Each auxiliary is correlated with every outcome's residual, every predictor, and each other auxiliary, but is never entered as a predictor, while the predictors keep their free covariances, so the model is saturated, the per-outcome regression coefficients

keep their meaning, and the full information maximum likelihood draws on the auxiliaries' observed values (the inclusive analysis strategy; Collins, Schafer, & Kam, 2001). A name in `auxiliary` must be numeric, present in `data`, absent from `formula`, and requires `fixed_x = FALSE`. Auxiliaries are refused with `missing = "listwise"`, which would drop every row on which an auxiliary is missing and so change the sample the coefficients describe, and with `missing = "pairwise"` or `"available.cases"`, which compute the moments from each pair of variables on its own, so that an auxiliary would leave every estimate unchanged.

`effect_sizes` Logical; whether to compute per-outcome standardized betas, semi-partial R^2, and Cohen's f^2. Defaults to `TRUE`.

`enforce_es_bounds`
Logical; if `TRUE`, negative per-outcome sr^2 or f^2 estimates (finite-sample artifacts) are clamped to zero. Defaults to `FALSE`.

`...` Additional arguments forwarded to `lavaan`.

`wald_df` The reference distribution of the Wald interval and of the p-values: `Inf` (the default), the normal distribution; a single positive number, the degrees of freedom of a t distribution for every coefficient; or `"missing_information"`, von Hippel's (2016) degrees of freedom for each coefficient from its fraction of missing information (see `mlmr`). A finite value or the keyword requires `ci_method = "wald"`; the keyword is an error where, with values missing, **lavaan** fit the complete cases in place of the FIML fit asked for, as in `mlmr`. It follows `...`, so it must be named in full.

Details

Why a separate function from `mlmr`. A univariate `mlmr` fit handles the case of one outcome regressed on one or more predictors. `mlmr_mv` extends to the case of two or more outcomes regressed on the same predictor set, modeling the residual covariance among outcomes explicitly. Under FIML a row that is missing on one outcome still contributes information about the other outcomes (through the modeled residual covariance) and about the joint distribution of the predictors, where listwise deletion drops the row. A user who fits separate univariate regressions for each outcome under listwise deletion can discard a great deal of information when the outcomes are correlated and missingness patterns differ.

Same predictor set across outcomes. The formula `cbind(y1, y2) ~ x1 + x2` regresses both `y1` and `y2` on `x1` and `x2`. Per-outcome predictor sets (sometimes called seemingly unrelated regression with heterogeneous predictors) are not supported; users who need that can fit separate `mlmr` models or call `sem` directly with a custom model string.

Auxiliary variables. As in `mlmr`, variables that are not part of the regression but are correlated with an outcome or with the missingness can be supplied through `auxiliary` and are entered as saturated correlates (Graham, 2003): each is correlated with every outcome's residual, every predictor, and each other auxiliary, but never as a predictor, and the predictors keep their free means, variances, and covariances. The model over the outcomes, the predictors, and the auxiliaries is then saturated, so the per-outcome coefficients keep their meaning (on complete data they equal those of `lm()`) while the likelihood draws on the auxiliaries' observed values (the inclusive analysis strategy of Collins, Schafer, & Kam, 2001).

The bootstrap interval. `ci_method = "boot"` resamples the rows of `data` with replacement `B` times (1000 by default), refits the model on each resample, and reports each coefficient's percentile interval. It is the interval to ask for when the multivariate normality the likelihood assumes is doubtful, since its coverage does not rest on that assumption. The price is `B` refits of a model that already carries J outcomes, so a bootstrap interval is a deliberate request rather than a default. Bootstrap results vary from run to run; supply `boot_seed` for reproducibility. The mechanics, including the Bollen-Stine variant, are given in the `ci_method` argument description and in `mlmr`.

Bootstrap replications. Each coefficient's limits are quantiles of its `B` resampled estimates, so the interval is run only when `B` puts at least 10 resamples beyond each limit (400 at the 95 percent level). A replication in which some coefficient is undefined (the refit fails or does not converge, or returns a coefficient that is not a finite number) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as elements of the fitted object, and `print()` and `summary()` state them in one line, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the limits of every coefficient are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. Such a resample is dropped for every coefficient, so all intervals rest on the same resamples. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the result is not one to report. See `dmar_bootstrap` for the rule and its reasons.

A *t* reference distribution for the Wald interval. With `ci_method = "wald"`, `wald_df` refers every coefficient's interval and *p*-value to a *t* distribution rather than the normal one, keeping the estimate and its standard error (von Hippel, 2016, Eq. 29), exactly as in `mlmr`, whose help page gives the reasoning. A number applies to every coefficient of every outcome. Under `"missing_information"` each coefficient gets von Hippel's $\max\{3, N(1-\gamma) - k\}$ from its own fraction of missing information γ, with k the number of regression parameters of its outcome (the slopes and the intercept), so the coefficients of a fully observed outcome keep nearly $N - k$ degrees of freedom while those of an outcome observed for few cases get fewer.

Caveats. Same as `mlmr`: the function assumes that, conditional on the predictors, the joint distribution of the outcomes is multivariate normal with constant covariance, and that missingness is at most MAR. Factor predictors and interactions are expanded through `model.matrix` once and reused for every outcome.

Value

An object of class `"mlmr_mv"`, a list with components similar to a univariate `mlmr` fit but extended for multiple outcomes:

`call`, `formula`, `terms`, `model`, `xlevels` As in `mlmr`.

`coefficients` A matrix with predictors (and an intercept row, when present) as rows and outcomes as columns, matching `coef.mlm`.

`B`, `B_used`, `n_dropped`, `B_required`, `bootstrap_status` Present only with `ci_method = "boot"`: the resamples requested, the number the intervals rest on, their difference, the fewest resamples a request may ask for, and the verdict (`"complete"`, `"dropped"`, `"exploratory"`,

"insufficient_count", or "insufficient_share"; under the last two the limits are NA). See *Bootstrap replications* under Details.

coef_table A long data.frame with one row per (outcome, term) combination; columns include outcome, term, estimate, se, test_statistic, p_value, ci_lower, ci_upper, std_estimate, carrying the confidence level in its conf_level attribute, so that results_sentence on a row reports it.

vcov The variance-covariance matrix of the regression coefficients across all outcomes, returned by vcov(). Rows and columns follow the outcome-major order of coef_table (the column-major flattening of coefficients) and are named "outcome:term", for example "mpg:wt", the naming vcov uses for an "mlm" fit. The cross-outcome blocks carry the sampling covariance between coefficients of different outcomes, so joint Wald tests across outcomes compose with coef().

residual_cov The estimated residual covariance matrix among outcomes (J by J).

R2 Named vector of model implied R^2 per outcome.

adj_R2 Named vector of adjusted R^2 per outcome.

effect_sizes When effect_sizes = TRUE, a long data.frame with one row per (outcome, predictor) combination giving sr^2 and Cohen's f^2.

fitted.values, residuals Matrices with rows = observations and columns = outcomes; NA in rows where any predictor is missing.

logLik, N, N_complete As in mlmr.

lavaan_fit The underlying **lavaan** fit.

The object carries the attributes wald_df and, under wald_df = "missing_information", wald_df_table, as an mlmr fit does, with the coefficients named "outcome:term" in the order of coef_table.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Collins, L. M., Schafer, J. L., & Kam, C.-M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods, 6*(4), 330–351. doi:10.1037/1082989X.6.4.330

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

See Also

mlmr for the univariate sibling; lm (and the "mlm" object class) for the OLS multivariate analog; sem for the underlying engine.

Examples

```
# Two outcomes on a shared predictor set. The residual covariance
# between the outcomes is estimated as part of the model, which is
# what separates this from two separate regressions. The fit asks
# for the Wald interval, a choice taken up below, and keeps the
# default effect_sizes = TRUE: the per-outcome effect sizes come
# back on the fit rather than in summary(), one row per outcome and
# predictor, giving the semi-partial R^2 and Cohen's f^2, and they
# are what fills the standardized coefficients in coef_table.
fit <- mlmr_mv(cbind(t6_paragraph_comprehension, t9_word_meaning) ~
                 t5_general_information + t7_sentence,
               data = holzinger_swineford,
               ci_method = "wald")
coef(fit)              # matrix: rows = predictors, cols = outcomes
summary(fit)
fit$R2                 # per-outcome R^2
fit$residual_cov       # residual covariance among outcomes
print(fit$effect_sizes, row.names = FALSE)

# The interval menu is profile, Wald, and bootstrap. The default,
# ci_method = "profile", inverts the likelihood ratio test one
# coefficient at a time, and with two outcomes there are twice as
# many coefficients to profile; it is what a reported interval
# deserves, and leaving ci_method at its default asks for it. The
# fits on this page ask for the Wald or the bootstrap interval
# because those refits take longer than a help page should. The
# bootstrap resamples rows and takes percentile limits; it is what
# to ask for when the multivariate normality the likelihood assumes
# is doubtful. Every resample refits the two-outcome model. B = 10
# keeps the example quick and is below the 400 a reported interval
# needs; the option lets it run and the warning says so. A reported
# interval deserves the default B = 1000. boot_seed fixes the
# resamples, so the limits are reproducible rather than moving from
# run to run. The effect sizes cost one constrained refit per outcome
# and predictor, so this fit and the ones after it leave them off.
old <- options(dmar.bootstrap_exploratory = TRUE)
fit_boot <- mlmr_mv(cbind(t6_paragraph_comprehension, t9_word_meaning) ~
                      t5_general_information + t7_sentence,
                    data = holzinger_swineford,
                    ci_method = "boot", B = 10, boot_seed = 113,
                    effect_sizes = FALSE)
options(old)
confint(fit_boot)

# FIML versus listwise when one outcome has missing values. The
# revised second-form test t26_flags was administered to only 145
# of the 301 students, so it carries real missingness. A row with
# t26_flags missing still informs the likelihood about the other
# outcome, about the predictors, and, through the residual
# covariance, about t26_flags itself, so no row is discarded. Notice
# the two sample sizes, and that the coefficients of the complete
# outcome differ between the fits: listwise deletion drops 156 of
```

```
# its observed rows along with the missing t26_flags values.
fit_fiml <- mlmr_mv(cbind(t6_paragraph_comprehension,
                          t26_flags) ~
                    t7_sentence + t9_word_meaning,
                  data = holzinger_swineford,
                  ci_method = "wald", effect_sizes = FALSE)
fit_lwd  <- mlmr_mv(cbind(t6_paragraph_comprehension,
                          t26_flags) ~
                    t7_sentence + t9_word_meaning,
                  data = holzinger_swineford,
                  missing = "listwise", ci_method = "wald",
                  effect_sizes = FALSE)
c(N_fiml = nobs(fit_fiml), N_listwise = nobs(fit_lwd))
cbind(FIML = coef(fit_fiml)[, "t6_paragraph_comprehension"],
      listwise = coef(fit_lwd)[, "t6_paragraph_comprehension"])

# Auxiliary variable (saturated correlates): the complete speed test
# t13_straight_and_curved_capitals informs the likelihood without
# entering either regression. Continuing from the model above, the
# coefficients of the complete outcome are unchanged to working
# precision, while those of t26_flags move, since the auxiliary
# carries information about the rows where t26_flags is missing.
fit_aux <- mlmr_mv(cbind(t6_paragraph_comprehension,
                         t26_flags) ~
                   t7_sentence + t9_word_meaning,
                 data = holzinger_swineford,
                 ci_method = "wald",
                 auxiliary = "t13_straight_and_curved_capitals",
                 effect_sizes = FALSE)
coef(fit_aux)
```

`moments_nc_chisq` *Moments of the Noncentral Chi Square Distribution*

Description

Returns the mean, variance, standard deviation, skewness, and excess kurtosis of a noncentral chi square distribution with `df` degrees of freedom and noncentrality parameter `ncp`. The noncentral chi square is the distribution of a sum of squared independent normals with nonzero means ($\sum(Z_i + \mu_i)^2$, with $\lambda = \sum \mu_i^2$); it is the building block of the noncentral *F* (whose numerator is a noncentral chi square) and the reference distribution for likelihood ratio and Wald statistics under the alternative. Unlike the noncentral *t* and *F*, every moment exists, so none of the returned values is ever `NA`.

Usage

```
moments_nc_chisq(df, ncp = 0)
```

Arguments

df	Degrees of freedom, a single positive number (need not be a whole number).
ncp	Noncentrality parameter λ, a single non-negative number. Defaults to 0, the central chi square.

Details

The cumulants of the noncentral chi square are $\kappa_n = 2^{n-1}(n-1)!\,(\nu + n\lambda)$ for $n \geq 1$, from which the moments follow directly: the mean is $\kappa_1 = \nu + \lambda$, the variance is $\kappa_2 = 2(\nu + 2\lambda)$, the skewness is $\kappa_3/\kappa_2^{3/2} = \sqrt{8}\,(\nu + 3\lambda)/(\nu + 2\lambda)^{3/2}$, and the excess kurtosis is $\kappa_4/\kappa_2^2 = 12(\nu + 4\lambda)/(\nu + 2\lambda)^2$. At $\lambda = 0$ these reduce to the central chi square values: mean ν, variance 2ν, skewness $\sqrt{8/\nu}$, and excess kurtosis $12/\nu$.

Value

A `data.frame` (class `dmar_tbl`) in `term` / `value` layout with the `mean`, `variance`, `sd`, `skewness`, and `excess_kurtosis`, followed by the `df` and `ncp` that produced them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Johnson, N. L., Kotz, S., & Balakrishnan, N. (1995). *Continuous univariate distributions* (Vol. 2, 2nd ed., Chapter 29). Wiley.

See Also

`moments_nc_F` (whose numerator is a noncentral chi square) and `moments_nc_t` for the other noncentral moments; `ci_nc_chisq` for the noncentral chi square confidence limits; `dchisq` for the density.

Other noncentral distribution moments: `moments_nc_F()`, `moments_nc_t()`

Examples

```
# A noncentral chi square with 5 df and noncentrality 3.
moments_nc_chisq(df = 5, ncp = 3)

# ncp = 0 is the central chi square: mean df, variance 2 * df.
moments_nc_chisq(df = 5)

# Every moment exists for any positive df, so nothing is ever NA.
anyNA(moments_nc_chisq(df = 1, ncp = 10)$value)
```

`moments_nc_F` *Moments of the Noncentral* F *Distribution*

Description

Returns the mean, variance, standard deviation, skewness, and excess kurtosis of a noncentral *F* distribution with `df_1` numerator and `df_2` denominator degrees of freedom and noncentrality parameter `ncp`, in one table. A moment that does not exist for the given `df_2` is returned as `NA`.

Usage

```
moments_nc_F(df_1, df_2, ncp = 0)
```

Arguments

`df_1`	Numerator degrees of freedom, a single positive number.
`df_2`	Denominator degrees of freedom, a single positive number.
`ncp`	Noncentrality parameter λ, a single non-negative number. Defaults to 0, the central *F*.

Details

The noncentral F is the reference distribution of the F statistic when an effect is present, so its moments describe the sampling behavior of R^2, eta squared, and the omnibus F test under the alternative. A central F (`ncp = 0`) is the special case.

The distribution. The noncentral F is $F = (X_1/\nu_1)/(X_2/\nu_2)$ with $X_1 \sim \chi^2_{\nu_1}(\lambda)$ a noncentral chi square and $X_2 \sim \chi^2_{\nu_2}$ independent of it. Its raw moments are

$$\mathrm{E}[F^k] = \left(\tfrac{\nu_2}{\nu_1}\right)^k \mathrm{E}[X_1^k] \prod_{i=1}^{k} \frac{1}{\nu_2 - 2i}, \qquad \nu_2 > 2k,$$

where the noncentral chi square moments $\mathrm{E}[X_1^k]$ follow from its cumulants $\kappa_j = 2^{j-1}(j-1)!\,(\nu_1 + j\lambda)$ (Johnson, Kotz, & Balakrishnan, 1995).

What is returned. The mean is $\nu_2(\nu_1+\lambda)/[\nu_1(\nu_2-2)]$ and the variance is $2(\nu_2/\nu_1)^2[(\nu_1+\lambda)^2 + (\nu_1+2\lambda)(\nu_2-2)]/[(\nu_2-2)^2(\nu_2-4)]$. The skewness is the third central moment over the variance to the power 3/2, and the excess kurtosis is the fourth central moment over the squared variance, minus 3 (so that a normal distribution has 0). The mean exists for $\nu_2 > 2$, the variance for $\nu_2 > 4$, the skewness for $\nu_2 > 6$, and the excess kurtosis for $\nu_2 > 8$; a moment whose condition is not met is returned as `NA`, and so is the standard deviation when the variance is.

How the moments are computed. A central moment formed as a difference of raw moments loses digits when the degrees of freedom are large, because the raw moments agree in their leading digits: for $\nu_1 = \nu_2 = 10^6$ and $\lambda = 0$ that form gives an excess kurtosis of -5.29×10^{-5}, where the value is 6.60006×10^{-5}. The function therefore works with the two independent factors of F. Write $F = (\nu_2/\nu_1)XY$ with $X = X_1$ and $Y = 1/X_2$, and let $U = X/\mathrm{E}[X] - 1$ and $V = Y/\mathrm{E}[Y] - 1$, so that $F/\mathrm{E}[F] - 1 = U + V + UV$ with U and V independent and of mean 0. The central moments

of $F/\mathrm{E}[F]$ are then sums of products of the central moments of U and V, in which every term is nonnegative and nothing cancels. With $n = \nu_2$,

$$u_2 = \kappa_2/\kappa_1^2, \quad u_3 = \kappa_3/\kappa_1^3, \quad c_u = \kappa_4/\kappa_1^4, \quad u_4 = c_u + 3u_2^2,$$

$$v_2 = \frac{2}{n-4}, \quad v_3 = \frac{16}{(n-4)(n-6)}, \quad c_v = \frac{48(5n-22)}{(n-4)^2(n-6)(n-8)}, \quad v_4 = c_v + 3v_2^2,$$

the second and third central moments and the fourth cumulant of U and of V (those of V from $\mathrm{E}[Y^k] = \prod_{i=1}^{k} 1/(n-2i)$). Then

$$m_2 = u_2 + v_2 + u_2 v_2, \qquad \mathrm{Var}(F) = \mathrm{E}[F]^2\, m_2,$$

$$m_3 = u_3 + v_3 + u_3 v_3 + 3u_3 v_2 + 3u_2 v_3 + 6u_2 v_2, \qquad \text{skewness} = m_3/m_2^{3/2},$$

and the excess kurtosis is e_4/m_2^2, where

$$e_4 = c_u + c_v + c_u c_v + 3c_u v_2^2 + 3u_2^2 c_v + 6u_2^2 v_2^2 + 4u_4 v_3 + 4u_3 v_4 + 6c_u v_2 + 12u_2^2 v_2 + 6u_2 c_v + 12u_2 v_2^2 + 12u_3 v_2 + 12u_2 v_3 + 12u_3 v_3$$

is the fourth central moment of $U + V + UV$ minus $3m_2^2$, with the terms that cancel removed algebraically. These expressions are the same quantities as the differences of raw moments above, derived for DMAR; they are not taken from the reference. Checked against exact rational arithmetic in 14 cases with degrees of freedom up to 10^{12} and noncentrality up to 10^6, the four moments are right to a relative error of 5×10^{-16}. At ordinary degrees of freedom (200 or fewer) the values agree with those of the difference of raw moments to a relative difference of 10^{-11}.

Value

A `data.frame` (class `dmar_tbl`) in `term` / `value` layout with the `mean`, `variance`, `sd`, `skewness`, and `excess_kurtosis` (any of which may be `NA` when `df_2` is too small), followed by the `df_1`, `df_2`, and `ncp` that produced them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Johnson, N. L., Kotz, S., & Balakrishnan, N. (1995). *Continuous univariate distributions* (Vol. 2, 2nd ed., Chapter 30). Wiley.

See Also

`moments_nc_t` for the noncentral *t*; `ci_nc_F` for the noncentral *F* confidence limits used in effect size intervals; `df` for the density.

Other noncentral distribution moments: `moments_nc_chisq()`, `moments_nc_t()`

Examples

```
# A noncentral F with 3 and 40 df and noncentrality 8.
moments_nc_F(df_1 = 3, df_2 = 40, ncp = 8)

# ncp = 0 is the central F: mean df_2 / (df_2 - 2).
moments_nc_F(df_1 = 3, df_2 = 40)

# The variance is undefined for four or fewer denominator df.
moments_nc_F(df_1 = 2, df_2 = 4, ncp = 5)
```

---

moments_nc_t *Moments of the Noncentral* t *Distribution*

Description

Returns the mean, variance, standard deviation, skewness, and excess kurtosis of a noncentral *t* distribution with `df` degrees of freedom and noncentrality parameter `ncp`. These are the closed-form moments surveyed by Owen (1968); they are the engine behind the bias and variance of the standardized mean difference (Cohen's *d*), since *d* is a scaled noncentral *t* variate. A central *t* (`ncp = 0`) is the special case with mean 0 and the familiar $df/(df-2)$ variance.

Usage

```
moments_nc_t(df, ncp = 0)
```

Arguments

`df`	Degrees of freedom, a single positive number (need not be a whole number). May be `Inf`.
`ncp`	Noncentrality parameter δ, a single number (may be negative, which mirrors the distribution about 0). Must be finite. Defaults to 0, the central *t*.

Details

Writing the noncentral *t* as $T = (Z+\delta)/\sqrt{W/\nu}$ with $Z \sim N(0,1)$ and $W \sim \chi^2_\nu$ independent, the raw moments are

$$\mathrm{E}[T^k] = \mathrm{E}[(Z+\delta)^k]\left(\tfrac{\nu}{2}\right)^{k/2}\frac{\Gamma\big((\nu-k)/2\big)}{\Gamma(\nu/2)}, \qquad \nu > k.$$

The mean is formed from the logarithm of the chi mean factor that `sd_unbiased` uses. The variance, skewness, and excess kurtosis are formed from the central moments of $\sqrt{\nu/W}$, so that no raw moments are subtracted; from 100 degrees of freedom on, the two smallest of those are their expansions in $1/\nu$. Against values computed to several hundred digits, the largest relative error found is about 10^{-12} for degrees of freedom from just above each boundary to 10^{300} and any finite `ncp`. A mean, variance, or standard deviation above the largest number R holds is returned as `Inf`. With `df = Inf` the result is the normal distribution with mean `ncp` and variance 1.

The mean exists for $\nu > 1$, the variance for $\nu > 2$, the skewness for $\nu > 3$, and the excess kurtosis for $\nu > 4$; a moment whose degrees of freedom condition is not met is returned as NA. The mean is $\delta\sqrt{\nu/2}\,\Gamma((\nu-1)/2)/\Gamma(\nu/2)$, the δ-scaled reciprocal of the Hedges (1981) bias-correction factor that expected_smd and smd use; that is why the standardized mean difference is upward biased.

Value

A data.frame (class dmar_tbl) in term / value layout with the mean, variance, sd, skewness, and excess_kurtosis (any of which may be NA when the degrees of freedom are too small), followed by the df and ncp that produced them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Owen, D. B. (1968). A survey of properties and applications of the noncentral t-distribution. *Technometrics, 10*(3), 445–478. doi:10.1080/00401706.1968.10490594

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

See Also

moments_nc_F for the noncentral *F*; expected_smd and var_smd for the same moments specialized to Cohen's *d*; dt for the density.

Other noncentral distribution moments: moments_nc_F(), moments_nc_chisq()

Examples

```
# A noncentral t with 20 df and noncentrality 2.5.
moments_nc_t(df = 20, ncp = 2.5)

# ncp = 0 is the central t: mean 0, variance df / (df - 2), no skew.
moments_nc_t(df = 10)

# The mean is the noncentrality times the Hedges bias factor's reciprocal,
# which is why Cohen's d (a scaled noncentral t) is upward biased.
m <- moments_nc_t(df = 18, ncp = 1.2)
m$value[m$term == "mean"]
```

mr_cv *Minimum Risk Point Estimation of the Population Coefficient of Variation*

Description

Sequential estimation of the coefficient of variation with minimum risk. The function implements the procedure of Chattopadhyay and Kelley (2016), which considers study cost and the accuracy of the estimated coefficient of variation together. For the variance of the sample coefficient of variation it uses an improved analytic solution, Equation 32 of Kelley, Darku, and Chattopadhyay (2018), in place of Equation 22 of the 2016 article, which carries an error in its kurtosis term (see the part of Details on Equation 22). It returns the pilot sample size, or it evaluates the risk and the stopping rule on the observations collected so far and says whether sampling can stop.

Usage

```
mr_cv(
  data,
  A,
  structural_cost,
  epsilon,
  sampling_cost,
  pilot = FALSE,
  m0 = 4,
  gamma = 0.49,
  verbose = FALSE
)
```

Arguments

data	A numeric vector of the observations collected so far (at least 4), from a quantity with a positive mean. It is needed when pilot = FALSE.
A	The structural cost divided by the square of epsilon (see Details), a single positive number. Give either A or both structural_cost and epsilon.
structural_cost	The structural cost: what one is willing to pay for the study apart from the cost of sampling (see Details), a single positive number.
epsilon	The maximum desired difference between the estimated coefficient of variation and the population value, a single positive number.
sampling_cost	The cost of collecting one more observation, a single positive number. For example, if each survey costs 10 dollars to distribute and score, sampling_cost would be 10 dollars per additional observation.
pilot	TRUE to plan the pilot sample size, or FALSE (the default) to evaluate the risk function and the stopping rule at the current sample size.
m0	The smallest pilot sample size allowed: a whole number of at least 4 (the default), since the variance estimate needs four observations.

`gamma` The exponent γ of the correction term $n^{-2\gamma}$ in the stopping rule, which keeps sampling from stopping too early: a single number strictly between 0 and 1/2. Chattopadhyay and Kelley (2016) take $0 < \gamma < 1/2$ and use .49 for practical purposes, the default.

`verbose` If `TRUE`, rows with the quantities behind the stopping rule are added to the table; defaults to `FALSE`.

Details

The value of `epsilon` is context specific; the smaller the value the closer the estimated value will tend to be to the population value.

When the aim of a study is to estimate a parameter accurately, such as the coefficient of variation, the structural cost and the maximum probable error of the estimate, ϵ, are combined to form A, the structural cost divided by ϵ^2: the price paid per squared unit of maximum probable error. The structural cost is the investment in the study apart from the cost of sampling; in a grant application it is the funds requested less those for sampling observations (Chattopadhyay & Kelley, 2016). A larger structural cost or a smaller ϵ gives a larger A, and a larger A leads to a more expensive study, other things being equal. A is fixed before any data are collected, whether it is given directly or through its two components.

The sampling cost is not fixed in advance, because the sample size needed to estimate the coefficient of variation accurately is unknown. The risk function of Chattopadhyay and Kelley (2016) adds the mean square error of the estimator, weighted by A, to the sampling cost cn, where c is `sampling_cost` and n the sample size. Their sequential procedure collects a pilot sample, then adds one observation at a time, and stops at the first n for which $n^2 \geq (A/c)(V_n^2 + n^{-2\gamma})$ (their Equation 23), where V_n^2 estimates ξ^2, the variance of the limiting normal distribution of $\sqrt{n}(k_n - \kappa)$, with k_n the sample coefficient of variation and κ the population value. Both sides of the rule are on the scale of n^2: the criterion the function reports is the right-hand side, and it is compared with the square of the sample size, not with the sample size. The row `N` of the table holds this n; N^2, here and in the line printed under the table, is its square. With `pilot = FALSE` the function evaluates the risk and the stopping rule at the current sample size; with `pilot = TRUE` it returns the pilot sample size $\max(m_0, \lceil (A/c)^{1/(2+2\gamma)} \rceil)$. A sample size is a whole number, so the formula is rounded up, and a formula value that is itself a whole number is that number: with $A = 100{,}000$, $c = 1$, and $\gamma = .25$ it is $100{,}000^{0.4} = 100$, which R computes as 100.00000000000003, and the pilot sample size is 100, not 101. `mr_smd` is the corresponding procedure for the standardized mean difference (Chattopadhyay & Kelley, 2017). For a confidence interval for the coefficient of variation of normally distributed scores, and for the sample size that makes it narrow, see `ci_cv` and `ss_aipe_cv` (Kelley, 2007).

The Variance Estimate: For scores with mean μ, variance σ^2, and third and fourth central moments μ_3 and μ_4,

$$\xi^2 = \frac{\mu_4}{4\mu^2\sigma^2} - \frac{\sigma^2}{4\mu^2} - \frac{\mu_3}{\mu^3} + \frac{\sigma^4}{\mu^4}$$

(Kelley, Darku, & Chattopadhyay, 2018, Equation 30). With the skewness γ_1 and the excess kurtosis γ_2 of the scores (neither is the exponent γ of the stopping rule), this is

$$\xi^2 = \frac{\kappa^2}{2} + \kappa^4 - \kappa^3\gamma_1 + \frac{\kappa^2\gamma_2}{4}.$$

V_n^2 is the expression in moments with the sample mean, the sample variance, and the unbiased estimates of the third and fourth central moments in place of the population values (their Equation 32), each moment computed from the scores centered at their mean. It is consistent for ξ^2 whenever the fourth moment is finite, so the procedure does not assume a specific distribution for the scores. The scores must have a positive mean: the coefficient of variation divides by the mean, and the procedure is for a quantity that cannot be negative (Chattopadhyay & Kelley, 2016, p. 627).

A Zero or Negative Estimate: V_n^2 can be zero or negative in a small sample. A positive estimate enters the stopping rule and the risk as $\max(V_n^2, n^{-3})$, the estimator of Equation 31 of Kelley, Darku, and Chattopadhyay (2018). The bound is small beside the correction term of the rule: n^{-3} is $n^{2\gamma-3}$ times $n^{-2\gamma}$, which is $n^{-2.02}$ times it at the default γ and less than n^{-2} times it for any γ. It replaces a positive estimate below n^{-3}. With a small coefficient of variation that is the usual case in a small sample: for normal scores ξ^2 is close to $\kappa^2/2$, which is below n^{-3} while n is below $(2/\kappa^2)^{1/3}$, about 27 observations for $\kappa = .01$ and 9 for $\kappa = .05$. From the pilot sample size on, the bound adds less than 1 to the criterion, so it can change the decision only at the pilot sample size, and only when N^2 there is within 1 of the criterion; the stop then usually comes one observation later.

When the estimate is zero or negative (it is taken as 0 for scores that do not vary, for scores that differ only by rounding error, with a range of at most 10^{-10} times the largest absolute score, such as equal gains computed as posttest minus pretest, and when it is within 10^{-10} of the size of its four terms), the function reports the stopping rule as not met, with the risk and the criterion missing, and says so under the table; the next observation is collected and the rule is evaluated again. This is a deliberate departure from Equation 23 read literally, under which a negative V_n^2 lowers the criterion and can end sampling at the pilot sample, and from Equation 31, which would put n^{-3} in place of such an estimate: the rule can then be met by a sample that says nothing usable about the variance.

Large-Sample Properties: The properties of the procedure are asymptotic (Chattopadhyay & Kelley, 2016). As A/c grows, the final sample size approaches the large-sample value $\sqrt{A/c}\,\xi$, at which the approximate risk $A\xi^2/n + cn$ is smallest. Short of that limit the procedure can stop early, on average, when the scores are skewed. In 20,000 simulated runs with $A = 5{,}000{,}000$ and $c = 10$, the mean final sample size was 480 for gamma scores with $\kappa = .8$, which is 0.94 of the large-sample value of 512, and 440 for lognormal scores with $\kappa = .6$, which is 0.81 of the large-sample value of 545.

Equation 22 of Chattopadhyay and Kelley (2016): The variance estimate of this function improves on the one the 2016 article uses, and its results therefore differ from the article's. Chattopadhyay and Kelley (2016) estimate ξ^2 by their Equation 22, which the article prints in moments,

$$\frac{s_n^4}{4\bar{X}_n^4} + \frac{\hat{\mu}_{4n}}{4\bar{X}_n^4} + \frac{s_n^2}{2\bar{X}_n^2} - \frac{\hat{\mu}_{3n}}{\bar{X}_n^3},$$

and of whose four terms only the last is a term of Equation 32 of Kelley, Darku, and Chattopadhyay (2018). Written with the skewness and the excess kurtosis, Equation 22 estimates $\kappa^2/2 + \kappa^4 - \kappa^3\gamma_1 + \kappa^4\gamma_2/4$, the first-order term of Bao (2009, Equation 3) as printed. That term is in error. Bao posted an erratum to the article on his web page at Purdue University (Bao, n.d.; the file is dated February 20, 2023), which says that the corrections were missed in proofreading and makes three: in the first line of the mean squared error of his Equation 3 (p. 293) the term

$\alpha^4\gamma_2/(4n)$ should be $\alpha^2\gamma_2/(4n)$, where α is the coefficient of variation and γ_2 the excess kurtosis; in the last line of his Remark 2 (p. 296) $\alpha^4\gamma_2$ should be $\alpha^2\gamma_2$; and in the second line of the expression for $E(\lambda_{22})$ (p. 295) the term $(3/n^2 + 1/n)\alpha^2\gamma_2$ should be $(3/n^2 + 1/(n\alpha^2))\alpha^2\gamma_2$. It credits Dennis Boos of North Carolina State University with finding the errors. The kurtosis term of the first-order variance is therefore $\kappa^2\gamma_2/4$, as Equation 30 of the 2018 article has, and Equation 32, the estimate used here, is the sample version of that corrected variance. The two population values are equal only when the excess kurtosis is 0 or $\kappa = 1$; with positive excess kurtosis the value of Equation 22 is too small for κ below 1 and too large above it. For gamma scores with $\kappa = .8$, ξ^2 is 0.525 and the value of Equation 22 is 0.304. In a sample the two estimates differ by $(k_n^2/4)(1 - k_n^2)(\hat{\mu}_{4n}/s_n^4 - 3)$, so the results of this function differ from those of the article for nearly any data, normal data included. For the article's example, the first of the Examples, the stopping rule is met at 35 observations with Equation 22, as the article reports, and is not met there with the variance estimate used here: the criterion is 1261.332 and N^2 is 1225.

Value

A `dmar_tbl` with columns `term` and `value`. With `pilot = FALSE` the rows are

`risk`	The estimated risk at the current sample size, $AV_n^2/n + cn$, with $\max(V_n^2, n^{-3})$ for V_n^2.
`N`	The current sample size (the number of observations in `data`).
`cv`	The current coefficient of variation.

With `verbose = TRUE` the rows `v2_n`, `mu_3_n` and `mu_4_n` (the unbiased estimates of the third and fourth central moments it is built from), and `criterion` follow. The row `v2_n` always holds the estimate V_n^2 itself, also when n^{-3} was used in its place and when the estimate is zero or negative. The attribute `is_satisfied` is `TRUE` when the stopping rule is satisfied, so that sampling can stop; the attribute `criterion` holds $(A/c)(\max(V_n^2, n^{-3}) + n^{-2\gamma})$, the value that N^2 is compared with; and the attribute `variance_unusable` is `TRUE` when the estimate is zero, negative, or not finite. In that case `is_satisfied` is `FALSE`, and the row `risk`, the row `criterion`, and the attribute `criterion` are `NA`. The decision is printed under the table: whether the stopping rule is met, the two numbers compared, and whether to stop or to collect another observation. With `pilot = TRUE` the single row is `pilot_N`, the pilot sample size.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bao, Y. (n.d.). Erratum: "Finite sample moments of the coefficient of variation" [Econometric Theory 25(1) (2009) 291–297]. Department of Economics, Purdue University. Retrieved November 15, 2026, from https://web.ics.purdue.edu/~ybao/Research/erratum_BaoET.pdf

Bao, Y. (2009). Finite-sample moments of the coefficient of variation. *Econometric Theory, 25*(1), 291–297. doi:10.1017/S0266466608090555

Chattopadhyay, B., & Kelley, K. (2016). Estimation of the coefficient of variation with minimum risk: A sequential method for minimizing sampling error and study cost. *Multivariate Behavioral Research, 51*(5), 627–648. doi:10.1080/00273171.2016.1203279

Chattopadhyay, B., & Kelley, K. (2017). Estimating the standardized mean difference with minimum risk: Maximizing accuracy and minimizing cost with sequential estimation. *Psychological Methods, 22*(1), 94–113. doi:10.1037/met0000089

Kelley, K. (2007). Sample size planning for the coefficient of variation from the accuracy in parameter estimation approach. *Behavior Research Methods, 39*(4), 755–766. doi:10.3758/BF03192966

Kelley, K., Darku, F. B., & Chattopadhyay, B. (2018). Accuracy in parameter estimation for a general class of effect sizes: A sequential approach. *Psychological Methods, 23*(2), 226–243. doi:10.1037/met0000127

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ci_cv, cv, ss_aipe_cv, mr_smd

Examples

```
# The example of Chattopadhyay and Kelley (2016): a structural cost of
# 1000 and a maximum probable error of .05, so that A is 400000, with a
# sampling cost of 75 per observation. The pilot sample size, from A or
# from its two components:
mr_cv(pilot = TRUE, A = 400000, sampling_cost = 75)
mr_cv(pilot = TRUE, structural_cost = 1000, epsilon = .05, sampling_cost = 75)

# The 18 pilot observations of the article. The line under the table
# gives the decision.
books <- c(36, 53, 19, 11, 10, 24, 14, 65, 18, 48, 25, 35, 13, 18, 3, 41, 5, 3)
mr_cv(data = books, A = 400000, sampling_cost = 75)

# The rule is not met, so one more observation is collected and the rule
# is checked again, and so on. The article stops at 35 observations; with
# the variance estimate used here the rule is not yet met there (see the
# part of Details on Equation 22). The extra rows show what the decision
# rests on.
books <- c(books, 44, 26, 13, 39, 2, 3, 26, 22, 8, 15, 12, 22, 5, 21, 23, 40, 18)
mr_cv(data = books, A = 400000, sampling_cost = 75, verbose = TRUE)

# A complete run. The simulated scores stand in for observations that a
# study would collect one at a time. The loop starts at the pilot sample
# size, adds an observation whenever the rule is not met, and cannot go
# past the scores on hand.
set.seed(113)
scores <- rgamma(1000, shape = 4, scale = 5)
n <- mr_cv(pilot = TRUE, A = 100000, sampling_cost = 1)$value
repeat {
  check <- mr_cv(data = scores[1:n], A = 100000, sampling_cost = 1)
  if (attr(check, "is_satisfied") || n == length(scores)) break
  n <- n + 1
}
check
```

mr_smd *Minimum Risk Point Estimation of the Population Standardized Mean Difference*

Description

A function for the sequential estimation of the standardized mean difference with minimum risk. The function implements the procedure of Chattopadhyay and Kelley (2017), which considers study cost and the accuracy of the estimated standardized mean difference together. `mr_smd` assumes normally distributed data, equal sample sizes in the two groups, and an equal cost of sampling per observation in each group.

Usage

```
mr_smd(
  A,
  structural_cost,
  epsilon,
  d,
  n_per_group,
  sampling_cost,
  pilot = FALSE,
  m0 = 4,
  gamma = 0.49
)
```

Arguments

`A`	The structural cost divided by the square of `epsilon`: the price one is willing to pay for a maximum allowable difference of `epsilon` between the estimate of the standardized mean difference and its parameter, a single positive number. Give either `A` or both `structural_cost` and `epsilon`.
`structural_cost`	The structural cost of what one is willing to pay in a study, a single positive number.
`epsilon`	The maximum desired difference between the estimated standardized mean difference and the population value, a single positive number.
`d`	The current estimate of the standardized mean difference, a single finite number. It is needed when `pilot = FALSE`.
`n_per_group`	Current sample size *per group* (thus total sample size is $2n$), a whole number of at least 2; requires equal sample size *per group*. It is needed when `pilot = FALSE`.
`sampling_cost`	The cost of collecting one more observation, a single positive number. For example, if each survey costs 10 dollars to distribute and score, `sampling_cost` would be 10 dollars per additional observation.

pilot	TRUE to plan the pilot sample size per group, or FALSE (the default) to evaluate the risk function and the stopping rule at the current sample size.
m0	The smallest pilot sample size per group allowed: a whole number of at least 4 (the default).
gamma	The exponent γ of the correction term $n^{-\gamma}$ in the stopping rule, which keeps sampling from stopping too early: a single number strictly between 0 and 1/2. Chattopadhyay and Kelley (2017) take $0 < \gamma < 1/2$ and suggest .49 for practical purposes, the default.

Details

Chattopadhyay and Kelley (2017) estimate the population standardized mean difference sequentially, minimizing a risk function that adds the mean square error of the estimator, weighted by A, to the total sampling cost. With n observations per group and $V_n^2 = 2 + d^2/4$, sampling stops at the first n for which $n \geq \sqrt{A/(2c)}\,(V_n + n^{-\gamma})$, where c is sampling_cost. With pilot = FALSE the function evaluates that risk and the stopping rule at the current sample size; with pilot = TRUE it returns the pilot sample size per group, $\max(m_0, \lceil (A/(2c))^{1/(2+2\gamma)} \rceil)$. A formula value that is itself a whole number is that number, although R can compute it a little above (with $A = 1{,}000{,}000$, $c = 5$, and $\gamma = .25$ it is 100, not 101; see mr_cv). See Kelley and Rausch (2006) for additional information on the standardized mean difference.

Value

A dmar_tbl with columns term and value. With pilot = FALSE the rows are

risk	The estimated risk at the current sample size, $AV_n^2/n + 2cn$.
n_1	Sample size for group 1 (the input n_per_group).
n_2	Sample size for group 2 (the input n_per_group).
d	The input estimate of the standardized mean difference.

The attribute is_satisfied is TRUE when the stopping rule is satisfied, so that sampling can stop, and the attribute criterion holds $\sqrt{A/(2c)}\,(V_n + n^{-\gamma})$, the value that the sample size per group is compared with. The decision is printed under the table: whether the stopping rule is met, the two numbers compared, and whether to stop or to collect another observation in each group. With pilot = TRUE the single row is pilot_n_per_group (see Note).

Note

When pilot=TRUE the function returns one row, pilot_n_per_group: the size of the pilot sample *per group* that should be used (thus, the total sample size is twice the pilot sample size).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., & Kelley, K. (2016). Estimation of the coefficient of variation with minimum risk: A sequential method for minimizing sampling error and study cost. *Multivariate Behavioral Research, 51*(5), 627–648. doi:10.1080/00273171.2016.1203279

Chattopadhyay, B., & Kelley, K. (2017). Estimating the standardized mean difference with minimum risk: Maximizing accuracy and minimizing cost with sequential estimation. *Psychological Methods, 22*(1), 94–113. doi:10.1037/met0000089

Kelley, K., Darku, F. B., & Chattopadhyay, B. (2018). Accuracy in parameter estimation for a general class of effect sizes: A sequential approach. *Psychological Methods, 23*(2), 226–243. doi:10.1037/met0000127

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ci_smd, mr_cv

Examples

```
# To obtain pilot sample size in a situation in which A=10000. Note that 'A' is
# 'structural_cost' divided by the square of 'epsilon'.

# From Chattopadhyay and Kelley (2017)
mr_smd(pilot = TRUE, A = 10000, sampling_cost = 2.4, gamma = .49)

High.SLS <- c(11, 7, 22, 13, 6, 9, 11, 16, 12, 17, 14, 8, 16)
Low.SLS <- c(3, 6, 10, 8, 14, 5, 12, 10, 6, 8, 13, 5, 9)

mr_smd(d = 1.021484, n_per_group = 13, A = 10000, sampling_cost = 2.40, gamma = .49)

# Or, using the smd() function:
mr_smd(d = smd(group_1 = High.SLS, group_2 = Low.SLS)[1,2], n_per_group = 13, A = 10000,
       sampling_cost = 2.40, gamma = .49)

# Here, for this situation, the stopping rule is satisfied:
mr_smd(d = 1.00, n_per_group = 75, A = 10000, sampling_cost = 2.40, gamma = .49)
```

---

nnt_from_smd *Number Needed to Treat From a Standardized Mean Difference*

Description

Converts a standardized mean difference to the success rate difference (SRD) of Kraemer and Kupfer (2006) and to its reciprocal, which they define as the number needed to treat (NNT) for a continuous outcome. The SRD is the probability that a randomly chosen treated person has a better outcome than a randomly chosen control person, minus the probability of the reverse. When sample sizes or confidence limits for the standardized mean difference are given, the function also returns the confidence interval for the SRD and the corresponding confidence set for the NNT, which is two intervals when the interval for the standardized mean difference includes zero.

Usage

```
nnt_from_smd(
  smd,
  n_1 = NULL,
  n_2 = NULL,
  conf_level = 0.95,
  smd_ci_lower = NULL,
  smd_ci_upper = NULL
)
```

Arguments

`smd`	The standardized mean difference (Cohen's *d*): the treatment mean minus the control mean, with higher scores the better outcome, divided by the pooled within-group standard deviation. A single finite number. A negative value gives a negative SRD and a negative NNT, read as the number needed to harm.
`n_1`, `n_2`	Sample sizes of the treatment and control groups. Supply both or neither. With both, the noncentral *t* confidence interval for the population standardized mean difference is computed (`ci_smd`) and carried to the SRD and the NNT.
`conf_level`	Confidence level of the noncentral *t* interval when `n_1` and `n_2` are given. Default `0.95`. With supplied limits, `conf_level` enters no computation: it is the caller's statement of the level of those limits, and it is recorded and printed only when it is given in the call.
`smd_ci_lower`, `smd_ci_upper`	Optional confidence limits for the population standardized mean difference, computed elsewhere. Supply both or neither, with `smd_ci_lower` not above `smd_ci_upper`. When supplied they are used as given, the noncentral *t* computation is skipped, and `n_1` and `n_2` are not used. An estimate outside its limits is accepted with a warning.

Details

What is estimated. Take one person at random from the treated population and one at random from the control population, with outcomes Y_T and Y_C, higher being better. The success rate difference is

$$\mathrm{SRD} = \Pr(Y_T > Y_C) - \Pr(Y_C > Y_T),$$

the probability that the treated person has the better outcome minus the probability that the control person has, and Kraemer and Kupfer (2006, p. 994) define the number needed to treat for a scaled outcome as its reciprocal,

$$\mathrm{NNT} = 1/\mathrm{SRD}.$$

No response threshold enters: the comparison is between two people, not between each person and a cut. An SRD of 0.25 says that in 100 such pairs the treated person is ahead in about 25 more pairs than the control person is, and the NNT of 4 says that about 4 people must be treated for one more of those comparisons to favor treatment. A negative standardized mean difference gives a negative SRD and a negative NNT, read as the number needed to harm. A standardized mean difference of zero gives an SRD of zero and an infinite NNT (no advantage of either group).

Assumptions, by the part each serves. The point conversion assumes two independent normal populations and a standardized mean difference whose standardizer is the root mean square of the two population standard deviations, which the pooled standard deviation estimates under a common variance (or with equal sample sizes). The interval from `n_1` and `n_2` assumes normality and a common variance, the conditions of the noncentral *t* interval for the standardized mean difference (`ci_smd`; Kelley, 2007). Supplied limits carry whatever assumptions produced them. When the populations are not normal, as with a skewed outcome, a ceiling or a floor, or an ordinal scale, the same standardized mean difference corresponds to a different $\Pr(Y_T > Y_C)$, and the SRD and the NNT computed here are then answers for a normal model that does not describe the data. How far off they are depends on the two distributions and is not quantified here. `cliff_delta` estimates the SRD itself from the two samples with no distributional model (Cliff's delta is the sample version of the same difference of probabilities), and `vargha_delaney_A` estimates $\Pr(Y_T > Y_C)$, counting half of each tie.

The conversion. Under the normal model the difference $Y_T - Y_C$ is normal with mean d and variance 2 in units of the standardizer, so

$$\Pr(Y_T > Y_C) = \Phi(d/\sqrt{2}),$$

$$\mathrm{SRD} = 2\Phi(d/\sqrt{2}) - 1, \qquad \mathrm{NNT} = \frac{1}{2\Phi(d/\sqrt{2}) - 1}.$$

The function computes the SRD as $\mathrm{sign}(d)\,\Pr(\chi^2_1 \le d^2/2)$, which is the same number written without the subtraction, so that a standardized mean difference near zero loses no digits to it. The values below agree with Table 1 of Kraemer and Kupfer (2006, p. 992) to the decimals printed there, with one exception: at $d = 1.0$ the table prints an SRD of .521, and the value is 0.5205, which is 0.520 to three decimals.

d	SRD	NNT
0.1	0.056	17.739
0.2	0.112	8.892
0.3	0.168	5.953
0.4	0.223	4.490
0.5	0.276	3.619
0.6	0.329	3.043
0.7	0.379	2.636
0.8	0.428	2.334
0.9	0.475	2.103
1.0	0.520	1.921

The confidence set, and why it can be two intervals. The function starts from a confidence interval for the population standardized mean difference, either the noncentral *t* interval computed from `n_1` and `n_2` or the limits supplied. Because the SRD is an increasing function of the standardized mean difference, the SRD at the two limits is a confidence interval for the SRD with the same coverage, and it is always an ordinary interval. The NNT is the reciprocal of the SRD, and the reciprocal is not continuous at zero, so what the interval for the SRD becomes depends on where zero falls:

- Zero outside the interval. The reciprocal reverses the order of the limits: the interval for the NNT runs from $1/\mathrm{SRD}_{\mathrm{upper}}$ to $1/\mathrm{SRD}_{\mathrm{lower}}$.
- Lower limit exactly zero. The interval for the NNT runs from $1/\mathrm{SRD}_{\mathrm{upper}}$ to infinity.
- Upper limit exactly zero and lower limit below zero. The interval runs from minus infinity to $1/\mathrm{SRD}_{\mathrm{lower}}$.
- Zero strictly inside the interval. The SRD values from the lower limit up to zero are sent to the NNT values from $1/\mathrm{SRD}_{\mathrm{lower}}$ (negative) down to minus infinity, and the SRD values from zero up to the upper limit are sent to the NNT values from plus infinity down to $1/\mathrm{SRD}_{\mathrm{upper}}$. The confidence set is therefore two intervals, $\mathrm{NNT} \leq 1/\mathrm{SRD}_{\mathrm{lower}}$ (harm) and $\mathrm{NNT} \geq 1/\mathrm{SRD}_{\mathrm{upper}}$ (benefit), which meet at infinity, the NNT of no effect (Altman, 1998).
- No interval requested. Only `smd`, `srd`, and `nnt` are returned.

One lower limit and one upper limit cannot hold a set of two intervals. Reading the two reciprocals as the ends of a single interval would describe the values between them, which are exactly the NNT values the data exclude, and would leave out the NNT of no effect, which the data include. For that case `nnt_ci_lower` and `nnt_ci_upper` are therefore `NA`, the set is in the `nnt_confidence_set` attribute, and a line under the printed table states it. In every case the confidence set for the NNT is the image of the interval for the standardized mean difference under a one-to-one transformation, so it holds the population NNT exactly when that interval holds the population standardized mean difference, and it has the same coverage. Kraemer and Kupfer (2006) call confidence intervals for the NNT "not advisable" (pp. 992 to 993) and prefer the SRD for interval estimation (p. 994), which is why the limits of the SRD are reported here beside those of the NNT.

Edge cases.

- `smd = 0`: the SRD is 0 and `nnt` is `Inf`. The sign of that infinity has no meaning.
- A negative `smd`: the SRD and the NNT are negative, and with both limits negative the interval for the NNT is an ordered negative interval (for $d = -0.5$ with 50 per group, -17.636 to -2.109).
- A limit exactly zero: one end of the interval for the NNT is infinite, as above; the set is one interval.
- A very small $|d|$: the NNT is very large and is computed without loss of digits (for $d = 10^{-8}$, about 1.77×10^{8}). An SRD that underflows to zero gives an infinite NNT with the sign of d.
- Very small samples: the noncentral *t* interval is wide, includes zero for most values of d, and gives two intervals (for $d = 0.5$ with `n_1 = 1` and `n_2 = 2`, limits for d of -2.07 and 2.88).
- An estimate outside the supplied limits is accepted with a warning, and the rows are computed from the three numbers as given.
- Supplied limits take precedence over `n_1` and `n_2`, which are then neither used nor checked.
- `conf_level` must be a single number in (0, 1) even when it enters no computation; `NULL` is an error.

Which NNT this is. The NNT returned here depends on the standardized mean difference alone. The NNT of evidence based medicine is the reciprocal of a difference between two response proportions at a stated threshold. Under the same normal model, with control response proportion p_0, that NNT is

$$1/\left[\Phi\left(\Phi^{-1}(p_0) + d\right) - p_0\right],$$

which changes with p_0 and, for a positive d, is larger than the Kraemer and Kupfer NNT at every threshold. Its smallest value is $1/[2\Phi(d/2) - 1]$, at $p_0 = \Phi(-d/2)$. At $d = 0.5$ the Kraemer and Kupfer NNT is 3.62; the threshold NNT is at least 5.07 (at $p_0 = 0.40$), 5.22 at $p_0 = 0.5$, and 6.01 at $p_0 = 0.2$. The two are different quantities and neither is an approximation to the other, so a report should say which one it gives. Furukawa and Leucht (2011) compared the two conversions in individual patient data from antipsychotic trials and found the control event rate conversion closer to the observed NNTs: it fell inside the observed 95 percent interval in 35 of 36 comparisons, against 20 of 36 for the conversion used here. For the NNT at a threshold computed from data, see `responder_analysis`.

A worked example. `nnt_from_smd(smd = 0.5, n_1 = 50, n_2 = 50)` returns an SRD of 0.276 and an NNT of 3.62: in 100 pairs of one treated and one control person the treated person is ahead in about 28 more pairs than the control person is, and about 3.6 people must be treated for one more such comparison to favor treatment. The 95 percent limits for the standardized mean difference are 0.101 and 0.897, which exclude zero. The SRD at those limits is 0.057 and 0.474, and the interval for the NNT is their reciprocals in reverse order, 2.11 to 17.64. `nnt_from_smd(smd = 0.2, n_1 = 50, n_2 = 50)` returns an SRD of 0.112 and an NNT of 8.89, with limits for the standardized mean difference of -0.193 and 0.592, which include zero. The interval for the SRD, -0.109 to 0.325, is an ordinary interval and is the one to report. `nnt_ci_lower` and `nnt_ci_upper` are `NA`, and the line under the table gives the confidence set for the NNT as "NNT <= -9.189 (harm) or NNT >= 3.079 (benefit)": the data are compatible with a benefit as large as one more favorable comparison for every 3.1 people treated, with no effect (an infinite NNT), and with harm as large as one more unfavorable comparison for every 9.2 people treated. A report of this result gives the estimate, 8.89, with both intervals of the set, or gives the SRD with its interval.

Value

A `data.frame` (class `dmar_nnt`, a `dmar_tbl`) with columns `term` and `value`. Without an interval the rows are `smd`, `srd` (the success rate difference), and `nnt`. With an interval the rows are, in this order, `smd`, `srd`, `nnt`, `smd_ci_lower`, `smd_ci_upper`, `nnt_ci_lower`, `nnt_ci_upper`, `srd_ci_lower`, and `srd_ci_upper`.

What each row is: `smd` is the standardized mean difference as given. `srd` is the success rate difference, $2\Phi(d/\sqrt{2}) - 1$. `nnt` is `1 / srd`. `smd_ci_lower` and `smd_ci_upper` are the confidence limits for the population standardized mean difference, computed from `n_1` and `n_2` or as supplied. `nnt_ci_lower` and `nnt_ci_upper` are the limits of the NNT when its confidence set is one interval. `srd_ci_lower` and `srd_ci_upper` are the limits of the SRD. The result is a `dmar_tbl`, so `tidy()`, `glance()`, and `rbind()` work on it as on any other. The confidence level is in the `conf_level` attribute: with `n_1` and `n_2` it is the level of the computed interval, and with supplied limits it is recorded only when `conf_level` is given in the call. With an interval the `ci_source` attribute names `smd` as the estimate whose interval the others are computed from, so that `results_sentence` reports the SRD (`term = "nnt"` or `term = "smd"` for another) and `tidy()` gives the confidence level on the rows of the SRD and the NNT and not on those of `smd`.

`srd_ci_lower` and `srd_ci_upper` are the SRD at the two limits of the standardized mean difference

and always form an ordinary interval. `nnt_ci_lower` and `nnt_ci_upper` depend on where zero falls in the interval for the standardized mean difference:

- Zero outside the interval: `nnt_ci_lower` is `1 / srd_ci_upper` and `nnt_ci_upper` is `1 / srd_ci_lower`.
- Lower limit exactly zero: `nnt_ci_lower` is `1 / srd_ci_upper` and `nnt_ci_upper` is `Inf`.
- Upper limit exactly zero and lower limit below zero: `nnt_ci_lower` is `-Inf` and `nnt_ci_upper` is `1 / srd_ci_lower`.
- Zero strictly inside the interval: the confidence set for the NNT is two intervals, which one lower and one upper limit cannot hold, so `nnt_ci_lower` and `nnt_ci_upper` are `NA`. The set is given in the `nnt_confidence_set` attribute and in a line printed under the table.

In every case, each finite end of the confidence set for the NNT is `1 / srd_ci_lower` or `1 / srd_ci_upper`.

A result with an interval carries two attributes. `nnt_set` is `"interval"` or `"two_intervals"`. `nnt_confidence_set` is a `data.frame` with columns `lower` and `upper` and one row per interval of the set, in increasing order: one row for `"interval"`, and for `"two_intervals"` the rows (`-Inf`, `1 / srd_ci_lower`) and (`1 / srd_ci_upper`, `Inf`). The two attributes belong to the table itself: `tidy()` and `glance()` return the rows without them, so for a set of two intervals they show `nnt_ci_lower` and `nnt_ci_upper` as `NA`; its finite ends are `1 / srd_ci_lower` and `1 / srd_ci_upper`, which both carry. A table made by `rbind()` or by taking rows keeps the attributes of the first table, as every `dmar_tbl` does, so the line printed beneath it describes that table.

The NNT of no effect is infinite: `nnt` is `Inf` when `smd` is zero, and the sign of that infinity has no meaning.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Altman, D. G. (1998). Confidence intervals for the number needed to treat. *BMJ, 317*(7168), 1309–1312. doi:10.1136/bmj.317.7168.1309

Furukawa, T. A., & Leucht, S. (2011). How to obtain NNT from Cohen's *d*: Comparison of two methods. *PLoS ONE, 6*(4), e19070. doi:10.1371/journal.pone.0019070

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. (The noncentral *t* interval for the standardized mean difference that is carried to the SRD and the NNT here.) doi:10.18637/jss.v020.i08

Kraemer, H. C., & Kupfer, D. J. (2006). Size of treatment effects and their importance to clinical research and practice. *Biological Psychiatry, 59*(11), 990–996. doi:10.1016/j.biopsych.2005.09.014

See Also

`smd` and `ci_smd` for the standardized mean difference and its interval; `cles` for $\Pr(Y_T > Y_C)$ under the same normal model; `cliff_delta` and `vargha_delaney_A` for the same comparisons estimated without the normal model; `responder_analysis` for the NNT at a threshold.

Other effect size estimates: cles(), cliff_delta(), correction_for_attenuation(), eta_squared(), eta_squared_generalized(), expected_partial_r(), expected_r(), expected_smd(), omega_squared(), probability_of_superiority_paired(), proportion_of_superiority(), responder_analysis(), smd_trimmed()

Examples

```
# 1. Point estimate only.
nnt_from_smd(smd = 0.5)

# 2. With the noncentral t interval from the sample sizes.
nnt_from_smd(smd = 0.5, n_1 = 50, n_2 = 50, conf_level = 0.95)

# 3. With limits for the standardized mean difference computed elsewhere.
nnt_from_smd(smd = 0.5, smd_ci_lower = 0.20, smd_ci_upper = 0.80)

# 4. An interval for the standardized mean difference that includes zero:
#    two intervals for the NNT, printed under the table.
nnt_from_smd(smd = 0.3, smd_ci_lower = -0.10, smd_ci_upper = 0.70)

# 5. The same case from sample sizes, with the set as an attribute: one
#    row per interval of the set.
res <- nnt_from_smd(smd = 0.2, n_1 = 50, n_2 = 50)
res
attr(res, "nnt_set")
attr(res, "nnt_confidence_set")

# 6. A negative standardized mean difference: the number needed to harm,
#    with an ordered negative interval.
nnt_from_smd(smd = -0.5, n_1 = 50, n_2 = 50)
```

---

now *A Plain-Language Date / Time Stamp and a Simple Stopwatch*

Description

now() is a small utility for two related tasks: (1) printing the current date and time in a form that reads naturally in a console log or report ("September 21, 2026 (3:42 PM)"), and (2) measuring how long a piece of work takes by capturing the time before and after the work and subtracting the two stamps.

Usage

```
now(time = TRUE, tidy = FALSE)
```

Arguments

time	Logical; if TRUE (default) the hour, minute, and AM/PM are appended to the printed date. FALSE returns the date only.
tidy	Logical; if TRUE, the timestamp is returned as a data.frame with columns term and value (numeric components only: day, year, hour, minute) and the non-numeric components (month name, AM/PM) attached as attributes. Defaults to FALSE, which returns the printable dmar_now stamp. Use tidy = TRUE only when the components are needed individually for programmatic processing; the dmar_now default supports subtraction and prettily prints in any context.

Details

The function returns an object of class "dmar_now" that carries the underlying Sys.time value and prints in the human-readable form. Two dmar_now objects can be subtracted with the ordinary - operator; the result is a difftime object with units chosen automatically from the magnitude of the elapsed interval. The pattern is the R analog of a stopwatch: capture before, capture after, take the difference.

For analyses that should record *when* a long-running computation was performed (a bootstrap confidence interval, a Monte Carlo simulation, an ss_aipe_*_sensitivity run), inserting a now() call at the start and end of the work creates a self-documenting log of when the computation ran and how long it took. The print method's natural-language form is designed to be copy-pasted into a methods section or an analysis note without further formatting.

The name coincides with lubridate::now(), and the masking is deliberate: both functions return the current time as a POSIXct object, so a script written for either remains correct with the other attached; only the printed form differs.

Value

When tidy = FALSE (default), an object of class c("dmar_now", "POSIXct", "POSIXt") whose print method yields the natural-language form ("September 21, 2026 (3:42 PM)" or, with time = FALSE, "September 21, 2026"). The underlying numeric value is the Sys.time() stamp at the moment of the call, so the - operator gives the elapsed time between two captures as a difftime object.

When tidy = TRUE, a data.frame with columns term and value, where value is numeric (day, year, hour, minute) and the month name and AM/PM marker are attached as the "month" and "am_pm" attributes.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

Sys.time for the underlying timestamp; difftime for the elapsed-time class returned by the - operator.

Examples

```
# Print the current date and time.
now()

# Time how long a piece of work takes. The pattern is the same
# whether the work is a bootstrap, a simulation, or a numeric
# search: capture a stamp before, capture one after, subtract.
# The "-" operator returns the elapsed time as a difftime, with
# units chosen automatically. Here the work is a descriptive
# summary of three Holzinger and Swineford cognitive tests, small
# enough that the elapsed time is a fraction of a second.
start <- now()
d <- descriptives(holzinger_swineford[, c("t1_visual_perception",
                                          "t2_cubes", "t4_lozenges")])
end <- now()
end - start

# A deliberate wait shows the same pattern on a longer interval. The
# pause is a twentieth of a second, long enough to register in the
# difference and short enough not to slow the help page.
start <- now()
Sys.sleep(0.05)
end <- now()
end - start

# Date only.
now(time = FALSE)

# Tidy data.frame form, when the components are needed
# individually for programmatic processing, for example when the
# stamp is embedded in a report's metadata block.
now(tidy = TRUE)
```

---

obrien_test *O'Brien's Test for Homogeneity of Variance*

Description

Tests the null hypothesis that two or more groups have equal population variances using O'Brien's (1981) procedure: each observation is transformed into a quantity whose expected value equals the group's variance, and a one-way analysis of variance is then run on those transformed values. Bartlett's test assumes normally distributed data and is very sensitive to departures from normality; because O'Brien's test applies the ordinary ANOVA F test to the transformed scores, it is robust to violations of normality (Maxwell, Delaney, & Kelley, 2027, Section 3.10.3).

Usage

```
obrien_test(x, group = NULL, data = NULL, na_action = stats::na.omit)
```

Arguments

x	Either a numeric vector of observations (in which case group must also be supplied), or a one-sided formula of the form y ~ group, in which case data is consulted for the variables.
group	A grouping vector or factor of the same length as x; used only when x is a numeric vector.
data	An optional data.frame containing the variables named in the formula.
na_action	Function specifying how missing values are handled (default na.omit).

Details

Following O'Brien (1981) and the version given in Abdi (2007), each observation Y_{ij} (the jth observation in group i, with size n_i and sample variance s_i^2) is transformed to

$$r_{ij} = \frac{(n_i - 1.5)\, n_i\, (Y_{ij} - \bar{Y}_i)^2 - 0.5\, s_i^2\, (n_i - 1)}{(n_i - 1)(n_i - 2)}.$$

The mean of the r_{ij} within group i equals s_i^2, so a one-way ANOVA on the r_{ij} tests $H_0 : \sigma_1^2 = \cdots = \sigma_k^2$. Each group must have at least three observations for the transformation to be defined.

When the group sizes are very unequal (the largest four or more times the smallest), O'Brien recommends a Welch-type ANOVA of the transformed scores (Maxwell, Delaney, & Kelley, 2027, Section 3.10.3); this function runs the ordinary one-way ANOVA.

Value

A one-row data.frame with columns statistic (the *F*-value from the ANOVA on the transformed scores), df_1, df_2, p_value, n_groups, N (the total sample size), and method.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Abdi, H. (2007). O'Brien's test for homogeneity of variance. In N. J. Salkind (Ed.), *Encyclopedia of measurement and statistics*. Sage.

Hunter, I. M. L. (1964). *Memory*. London: Penguin.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on testing the homogeneity of variance assumption.)

O'Brien, R. G. (1981). A simple test for variance effects in experimental designs. *Psychological Bulletin, 89*(3), 570–574.

See Also

bartlett.test, var.test

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), pairwise_within(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# Hunter's (1964) "one-is-a-bun" peg-word memory experiment, as discussed
# by Abdi (2007). Sixty-four participants were assigned to a control group
# (no mnemonic instruction) or an experimental group (peg-word mnemonic).
# The score is the number of word pairs (out of 10) recalled. Abdi (2007,
# Table 6) reports F = 1.29 (df = 1, 62) for the O'Brien test of equal
# variances, p = .260 as computed here; the experimental group's apparent
# ceiling effect does not produce statistically detectable variance
# heterogeneity.
hunter_1964 <- data.frame(
  group = factor(
    c(rep("Control", 32), rep("Experimental", 32)),
    levels = c("Control", "Experimental")
  ),
  recall = c(
    # Control group (n = 32):
    5, 5, 5, 5, 5,
    6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
    7, 7, 7, 7, 7, 7, 7, 7, 7,
    8, 8, 8,
    9, 9,
    10, 10,
    # Experimental group (n = 32):
    6,
    7, 7,
    8, 8, 8, 8,
    9, 9, 9, 9, 9, 9, 9, 9, 9,
    10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10
  )
)
obrien_test(recall ~ group, data = hunter_1964)

# Comparison against Bartlett's test on the same data.
bartlett.test(recall ~ group, data = hunter_1964)

# Vector / grouping-variable interface, on DMAR's depression_bdi data.
# The wait list variance is about twice the SSRI variance, but with ten
# observations per group the test does not reject equal variances.
obrien_test(depression_bdi$bdi_post, depression_bdi$condition)
```

omega_squared *Omega Squared (Effect Size for ANOVA)*

Description

Computes the sample omega squared (ω^2), Hays' (1994) bias-corrected estimator of the proportion of the population variance of the dependent variable that a fixed effect accounts for, as two estimands: the *total* ω^2, a proportion of the total variance, and the *partial* ω^2, a proportion of the variance due to the effect and error alone. With `estimand = "all"`, the default, the function reports every estimand its inputs determine; `"total"` or `"partial"` reports one. In a one-way design the two are the same number.

Usage

```
omega_squared(
  object = NULL,
  estimand = c("all", "total", "partial"),
  SS_effect = NULL,
  SS_error = NULL,
  SS_total = NULL,
  MS_error = NULL,
  F_value = NULL,
  df_effect = NULL,
  df_error = NULL,
  N = NULL
)
```

Arguments

`object`	Optional. A fitted model of class `aov`, `lm`, or `aovlist` (a multi-stratum fit such as `aov(y ~ A + Error(subject/A), data = d)`). The function returns one row per effect, with the stratum named for an `aovlist`. Give either `object` or summary statistics, not both.
`estimand`	Which estimand to report: `"all"` (the default, every estimand the inputs determine, `NA` for the others), `"total"`, or `"partial"`. An estimand asked for by name that the inputs do not determine is refused (see Details).
`SS_effect`	The sum of squares of the effect, or a vector of the sums of squares of several effects of one analysis (named for the effects, which label the rows). Given with `df_effect` and the error mean square.
`SS_error`	The error (residual) sum of squares, given with `df_error` as an alternative to `MS_error`.
`SS_total`	The corrected total sum of squares, $\sum(y-\bar{y})^2$ (see the Details of `eta_squared`). With the sums of squares, it determines the total ω^2.
`MS_error`	The error (residual) mean square. Give either `MS_error` or `SS_error` with `df_error`.

`F_value` The observed F statistic of the effect, or a vector of the F statistics of several effects tested against the same error term. With `df_effect`, `df_error`, and `N`, it determines the partial ω^2 only.

`df_effect` The numerator degrees of freedom of the effect (one value for each effect, or one for all). Required with `F_value` and with the sums of squares.

`df_error` The error (residual) degrees of freedom. Required with `F_value` and with `SS_error`.

`N` The total sample size, the number of observations. With the sums of squares or `F_value`, it determines the partial ω^2. For an F from a design with an `Error()` stratum, give the size of the effect's stratum instead (see Details): the number of subjects for an effect tested between subjects, and for an effect tested within subjects the stratum's total degrees of freedom, which is `df_effect + df_error` when the effect is the only one tested in its stratum. `N` must be at least `df_effect + df_error`.

Details

The function takes a fitted model (`aov`, `lm`, or an `aovlist` from `aov()` with an `Error()` term), the sums of squares of a published ANOVA table, or an F statistic with its degrees of freedom and N. The confidence interval is computed by `ci_omega_squared`.

Total and partial. For a fixed effect with sum of squares SS_{effect} on df_{effect} degrees of freedom, error mean square MS_{error}, corrected total sum of squares SS_{total}, and N observations,

$$\hat{\omega}^2_{\text{total}} = \frac{SS_{\text{effect}} - df_{\text{effect}} MS_{\text{error}}}{SS_{\text{total}} + MS_{\text{error}}},$$

$$\hat{\omega}^2_{\text{partial}} = \frac{SS_{\text{effect}} - df_{\text{effect}} MS_{\text{error}}}{SS_{\text{effect}} + (N - df_{\text{effect}}) MS_{\text{error}}} = \frac{df_{\text{effect}}(F-1)}{df_{\text{effect}}(F-1) + N}$$

(Olejnik & Algina, 2000, p. 266 and Table 10, p. 268; Maxwell, Delaney, & Kelley, 2027, Eqs. 3.97, 3.98, and 7.44 to 7.46). The total value estimates the effect's share of the total population variance, all other effects and the error included; the partial value estimates its share of the variance due to the effect and the error alone, the variance that remains when the other effects are held constant. Which answers the question depends on whether the design's other factors would vary where the effect is studied (see the Details of `eta_squared`). In a one-way design $N - df_{\text{effect}} = df_{\text{error}} + 1$, so the two denominators agree and the estimands are the same number. Some sources print the total's denominator as $SS_{\text{total}} - MS_{\text{error}}$ (Kim, 2016, Table 4, which gives .326 for an effect whose total $\hat{\omega}^2$ is .323); the denominator of Hays' estimator adds the error mean square, as here.

Bias correction and negative estimates. The sample η^2 is the proportion of *sample* variance accounted for and overestimates its population value. $\hat{\omega}^2$ subtracts $df_{\text{effect}} MS_{\text{error}}$ from the effect's sum of squares and adds to the denominator, which gives an estimator of the population proportion with much smaller bias (Hays, 1994; Olejnik & Algina, 2000; Kelley, 2007). Its numerator is negative whenever $F < 1$, and a negative estimate of either estimand is reported as 0, since $\omega^2 \geq 0$ by definition (Olejnik & Algina, 2000, p. 262; Maxwell, Delaney, & Kelley, 2027, Section 3.9.2).

What the inputs determine. A fitted model with one error stratum determines both estimands of every effect. Sums of squares determine the total when `SS_total` is given and the partial when `N` is given, with `df_effect` and the error mean square in either case. An F statistic with its degrees of freedom and N determines the partial value only: the total also depends on the sums of squares of the design's other effects, and the function cannot tell from F whether there are any, so the F interface

reports the total as NA even for a one-way design, where it equals the partial value. With estimand = "all" an estimand the inputs do not determine is NA, with the reason in the estimand_notes attribute and beneath the printed table; asked for by name, it is refused with an error of class dmar_estimand_undetermined (see dmar_metadata). Sums of squares and F are two ways of giving the same effect, and a call that gives both, or a fitted model with summary statistics, or both MS_error and SS_error, is refused.

Designs. The total ω^2 is defined for fixed effects in designs whose factors are all between subjects, with one error term (Olejnik & Algina, 2000, p. 266). For an aovlist fit, which has an error term for each stratum, the total is NA; designs with random factors and designs with factors within subjects have ω^2 formulas of their own (Olejnik & Algina, 2000, Tables 11 and 12; Olejnik & Algina, 2003), which this function does not compute. An interaction or a covariate of an lm fit is an effect like any other, with its own sum of squares and degrees of freedom. The sums of squares of a fitted model are the sequential (Type I) ones of anova(); see the Details of eta_squared on the corrected total, on Types II and III, and on the coding of factors.

Within-subjects designs. An effect of an aovlist fit is tested in its own error stratum, and its partial estimate is computed with the size of that stratum in place of N, the size with which ci_eta_squared and ci_omega_squared convert the limits of its interval: the stratum's total degrees of freedom for an effect tested within subjects, and the number of subjects for an effect in the between-subjects stratum of a mixed design (see the Details of ci_eta_squared). Since $df_{\text{effect}}(F-1)$ estimates the noncentrality parameter λ of the effect's test, $\hat{\omega}^2_{\text{partial}}$ then estimates $\lambda/(\lambda+M)$, with M the stratum's size, which is the value the sample partial η^2 converges to as subjects are added and the quantity whose interval ci_eta_squared(estimand = "partial") reports for the row. Computed with the number of observations N, the estimate of a within-subjects effect estimates $\lambda/(\lambda+N)$, a smaller quantity, which the sample partial η^2 does not estimate. That estimate is the partial ω^2 that Olejnik and Algina (2003, p. 443) compute for a repeated measures factor, whose N is the total number of observations, and the F interface given the number of rows as N reproduces it. For the three IQ occasions of pygmalion, $F(2, 618)$ with 930 rows gives .150, below the partial η^2 of .212 and its interval [.157, .265]; with the stratum's 2 + 618 = 620, the estimate is .209.

Value

A data.frame (class dmar_anova_es, printed through dmar_tbl) with one row per effect and the columns effect, omega_squared_total and omega_squared_partial (only the one asked for when estimand is "total" or "partial"), stratum (for an aovlist fit), F_value, df_effect, df_error, and N. An effect given through the summary interfaces without names is "overall" (or effect_1, effect_2, and so on, for several), and a quantity the inputs do not give is NA. N is the number of observations (for the summary interfaces, N as given); the partial value of an effect of an aovlist fit is computed with the size of its own stratum (see Details), so that its row's F, degrees of freedom, and that size, given to the F interface, reproduce it. The estimand attribute records the estimand asked for, and the estimand_notes attribute names each estimand column that is NA with the reason, which the line beneath the printed table states. tidy() gives one row per effect and estimand (see dmar_tidiers).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bargmann, R. E. (1970). Interpretation and use of a generalized discriminant function. In R. C. Bose, I. M. Chakravarti, P. C. Mahalanobis, C. R. Rao, & K. J. C. Smith (Eds.), *Essays in probability and statistics* (pp. 35–60). University of North Carolina Press.

Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40*(3), 659–670. doi:10.1177/001316448004000309

Hays, W. L. (1994). *Statistics* (5th ed.). Fort Worth, TX: Harcourt Brace College Publishers.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*, 137–152. doi:10.1037/a0028086

Keppel, G. (1991). *Design and analysis: A researcher's handbook* (3rd ed.). Prentice Hall.

Kim, H.-Y. (2016). Statistical notes for clinical researchers: Sample size calculation 3. Comparison of several means using one-way ANOVA. *Restorative Dentistry & Endodontics, 41*(3), 231–234. doi:10.5395/rde.2016.41.3.231

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on ω^2 and Chapter 7 on total and partial ω^2 in factorial designs.)

Olejnik, S., & Algina, J. (2000). Measures of effect size for comparative studies: Applications, interpretations, and limitations. *Contemporary Educational Psychology, 25*(3), 241–286. doi:10.1006/ceps.2000.1040

Olejnik, S., & Algina, J. (2003). Generalized eta and omega squared statistics: Measures of effect size for some common research designs. *Psychological Methods, 8*(4), 434–447. doi:10.1037/1082989X.8.4.434

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Venables, W. (1975). Calculation of confidence intervals for noncentrality parameters. *Journal of the Royal Statistical Society, Series B, 37*(3), 406–412. doi:10.1111/j.25176161.1975.tb01554.x

See Also

`ci_omega_squared` for the confidence interval, `eta_squared`, `var_omega_squared`, `ci_pvaf`

Other effect size estimates: `cles()`, `cliff_delta()`, `correction_for_attenuation()`, `eta_squared()`, `eta_squared_generalized()`, `expected_partial_r()`, `expected_r()`, `expected_smd()`, `nnt_from_smd()`, `probability_of_superiority_paired()`, `proportion_of_superiority()`, `responder_analysis()`, `smd_trimmed()`

Examples

```
# 1. The sums of squares of a published two-way table: Kim (2016,
#    Table 3), light by material, 2 by 4 with 10 per cell (N = 80).
omega_squared(SS_effect = c(LIGHT = 34.716, MATERIAL = 1996.998,
                            "LIGHT:MATERIAL" = 1570.389),
              df_effect = c(1, 3, 3), SS_error = 2261.612,
```

```
              df_error = 72, SS_total = 5863.715, N = 80)

# 2. An F statistic determines the partial value only. Bargmann's
#    (1970) example, used in turn by Venables (1975), Fleishman (1980),
#    and Steiger (2004): a one-way ANOVA with five groups of 11 and an
#    observed F of 11.221 on 4 and 50 df. In a one-way design the
#    partial value is the total.
omega_squared(F_value = 11.221, df_effect = 4, df_error = 50, N = 55,
              estimand = "partial")

# 3. One-way ANOVA from a fitted model (depression_bdi: three
#    treatment arms, 10 per arm, N = 30); the two estimands agree.
fit_one <- aov(bdi_post ~ condition, data = depression_bdi)
omega_squared(fit_one)

# 4. Two-factor ANOVA for the manipulated expectancy treatment and the
#    measured grade classification (pygmalion data, unequal cell
#    sizes, N = 310). The treatment by grade interaction is weak here
#    (F = 1.19), so the additive model is used.
fit_additive <- aov(iq_8 ~ treatment + factor(grade), data = pygmalion)
omega_squared(fit_additive)

# 5. The same partial values with their confidence intervals.
ci_omega_squared(fit_additive, estimand = "partial")
```

`orthogonal_polynomial` *Orthogonal-Polynomial (Trend) Contrast Coefficients*

Description

Builds the set of orthogonal-polynomial contrasts (linear, quadratic, cubic, ...) for a factor whose a levels are quantitative, so that between-group variation can be decomposed into independent trend components. The columns are mutually orthogonal and each sums to zero (so each is a contrast on the group means). By default the coefficients are returned in *orthonormal* form, meaning each column also has unit length, $\sum_i c_i^2 = 1$; the alternative `type = "integer"` rescales every column to the small whole numbers used in the published orthogonal-polynomial tables, which are easier to read by eye and match hand computation. The per-trend sum of squared coefficients $\sum_i c_i^2$ is carried on the returned object and shown when it is printed.

Usage

```
orthogonal_polynomial(
  levels,
  scores = NULL,
  type = c("orthonormal", "integer"),
  degree = NULL
)
```

Arguments

`levels`	One of: a single integer giving the number of levels a (labels default to `"L1"`, `"L2"`, ..., and the levels are treated as equally spaced); a character or factor vector of level labels (equally spaced unless `scores` is given); or a numeric vector of the quantitative level values themselves, which are used both as the labels and as the spacing.
`scores`	Optional numeric vector of length a giving the quantitative position of each level. Supply this when the labels are non-numeric but the spacing is unequal (e.g., doses 0, 1, 2, and 4). Defaults to equally spaced positions `1:a`.
`type`	Either `"orthonormal"` (default) for unit-length columns ($\sum_i c_i^2 = 1$, identical to `contr.poly`) or `"integer"` for the minimal whole-number coefficients of the classic trend-coefficient table. `"integer"` requires equally spaced `scores`.
`degree`	Highest-order trend to return, an integer between 1 and $a - 1$. Defaults to $a - 1$ (the full set the design can support).

Details

What a trend contrast is. When the levels of a factor are quantitative and ordered (minutes of study, dose, day), the omnibus between-group variation can be partitioned into a linear trend (does the mean rise or fall steadily?), a quadratic trend (is there curvature?), a cubic trend (an S-shape?), and so on, up to order $a - 1$. Each trend is a single 1-*df* contrast on the group means, $\hat{\psi} = \sum_i c_i \bar{Y}_i$, and because the contrasts are mutually orthogonal their sums of squares add up to the omnibus between-group sum of squares exactly.

Orthonormal versus integer scaling. The two `type` values return the *same* trends (same directions, same hypotheses, identical *F*, *t*, and *p* for a given trend); they differ only in how each column is scaled.

- `"orthonormal"` normalizes each column to unit length, $\sum_i c_i^2 = 1$. The trend portion of the design is then an orthonormal basis, the sum of squares for a trend is simply $\hat{\psi}^2$, and in a fitted `lm` the `.linear` / `.quadratic` coefficients are the trend estimates on a common scale. This is the form `lm` and `aov` use internally and is defined for any spacing, equal or unequal.
- `"integer"` rescales each column to the smallest whole numbers with the same ratios, reproducing the published orthogonal-polynomial coefficient table (e.g., for $a = 4$ the linear contrast is $-3, -1, 1, 3$). These are easy to read and to compute with by hand, but the columns no longer share a common length: $\sum_i c_i^2$ varies by trend (for $a = 4$: 20, 4, and 20), so the sum of squares for a trend is $n\,\hat{\psi}^2 / \sum_i c_i^2$. Integer coefficients exist only for equally spaced levels.

The reported $\sum_i c_i^2$ (shown on printing and stored in `attr(*, "sum_sq")`) is exactly the divisor in that sum-of-squares formula; for the orthonormal form it is 1 for every trend.

Unequal spacing. With unequally spaced `scores` the orthonormal polynomials are still uniquely defined and are returned by the `"orthonormal"` type. Whole-number coefficients generally do not exist in that case, so `type = "integer"` is an error.

Relation to base R. For equally spaced levels the orthonormal output is identical to `contr.poly(a)`; `orthogonal_polynomial` adds the integer table form, an explicit `scores` argument for unequal spacing, meaningful trend names, and the $\sum_i c_i^2$ report.

Value

A numeric $a \times$ degree matrix with row names = the level labels and column names = the trend names (`"linear"`, `"quadratic"`, `"cubic"`, `"quartic"`, ...). The matrix can be assigned directly to `contrasts(factor)` or passed to `contrast_test`, `is_orthogonal_set`, or `ci_c`. The per-column sum of squared coefficients is stored as `attr(*, "sum_sq")` (a named numeric vector), and the level spacing and `type` are stored as `attr(*, "scores")` and `attr(*, "type")`. The object carries class `"orthogonal_polynomial"` so that printing shows the coefficients alongside $\sum_i c_i^2$. When printed, the object is shown in the textbook Table A.10 orientation (trends in rows, levels in columns) with the $\sum_i c_i^2$ values as a final column; the stored matrix is the transpose of that display (levels in rows) so it can be assigned directly to `contrasts()`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Fisher, R. A., & Yates, F. (1953). *Statistical tables for biological, agricultural and medical research* (4th ed.). Oliver and Boyd. (Origin of the tabulated integer coefficients.)

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 6 on trend analysis; Appendix Table A.10 reports these coefficients.)

See Also

`contr.poly`, `effects_coding`, `helmert_coding`, `is_orthogonal_set`, `contrast_test`, `ci_c`

Other design utilities: `design_consequences()`, `design_effect()`, `effects_coding()`, `helmert_coding()`, `is_orthogonal_set()`

Examples

```
# Trend (orthogonal-polynomial) analysis decomposes the omnibus effect of a
# quantitative factor (dose, time, trial block, stimulus intensity) into
# independent linear, quadratic, cubic, ... components, as in the trend
# analysis of Maxwell, Delaney, and Kelley (2027, Chapter 6).

# 1. The a = 4 coefficient table in the integer form of the textbook
#    appendix (Table A.10). Printing puts the trends in rows and appends a
#    final sum-of-squared-coefficients (sum c^2) column: linear -3 -1 1 3,
#    and per-trend sum c^2 of 20, 4, 20.
orthogonal_polynomial(4, type = "integer")

# 2. The default orthonormal form (identical to stats::contr.poly(4)); every
#    trend has sum c^2 = 1, the scale lm() and aov() use internally.
orthogonal_polynomial(4)

# 3. A worked trend analysis in the style of MDK Chapter 6. An outcome is
#    measured at a = 5 equally spaced levels of a quantitative factor (say
#    stimulus intensity 1..5), n = 8 per level, from a population with a
#    strong linear and a mild quadratic trend. Assigning the trend contrasts
```

```
#    to the factor makes lm() report each trend test directly.
set.seed(113)
intensity <- factor(rep(1:5, each = 8))
contrasts(intensity) <- orthogonal_polynomial(levels(intensity))
y <- c(10, 9, 7, 6, 6)[as.integer(intensity)] +
       rnorm(length(intensity), sd = 1.2)
round(coef(summary(lm(y ~ intensity))), 3)

# 4. The same trends through DMAR's contrast_test(), which reports each
#    trend's estimate, standard error, t, p, and confidence interval from a
#    fitted one-way model. The integer columns are the contrast weights.
op <- orthogonal_polynomial(5, type = "integer")
contrast_test(aov(y ~ intensity),
              contrasts = list(linear    = op[, "linear"],
                               quadratic = op[, "quadratic"],
                               cubic     = op[, "cubic"]))

# 5. Unequally spaced doses (0, 1, 2, 4 mg): integer coefficients no longer
#    exist, but the orthonormal trends remain uniquely defined.
orthogonal_polynomial(c("0 mg", "1 mg", "2 mg", "4 mg"),
                      scores = c(0, 1, 2, 4))

# 6. Confirm a returned set is mutually orthogonal.
is_orthogonal_set(orthogonal_polynomial(5, type = "integer"))
```

`pairwise_within` *Paired Pairwise Comparisons With Multiple-Comparison Adjustment*

Description

Computes all pairwise paired-*t* comparisons among the levels of a within-subjects factor and returns the mean difference, paired SD, paired *t*-statistic, degrees of freedom, raw and adjusted *p*-values, and a confidence interval on the mean difference, one row per pair. *p*-values are adjusted across comparisons by the user-specified method (Bonferroni, Holm, Hochberg, Hommel, BH, BY, or none).

Usage

```
pairwise_within(
  data,
  subject = NULL,
  condition = NULL,
  outcome = NULL,
  adjust = c("holm", "bonferroni", "hochberg", "hommel", "BH", "BY", "none"),
  conf_level = 0.95,
  bonferroni_ci = FALSE
)
```

Arguments

data	Either an $n \times k$ wide numeric matrix / data.frame (one row per subject, one column per condition), or a long-format data.frame together with subject, condition, and outcome column names.
subject	Long-format only: character name of the subject-id column.
condition	Long-format only: character name of the within- subjects factor column.
outcome	Long-format only: character name of the response column.
adjust	Multiple-comparison adjustment method. One of "holm" (default), "bonferroni", "hochberg", "hommel", "BH", "BY", or "none". Passed to stats::p.adjust().
conf_level	Family-wise confidence level for the per-pair CIs. Default 0.95. CIs are computed at the per-pair nominal level ($1 - \alpha/m$ under Bonferroni; conf_level otherwise).
bonferroni_ci	Logical. If TRUE, the CIs use a per-pair confidence level of $1-(1-\text{conf_level})/m$ to give simultaneous conf_level coverage across the m comparisons (Bonferroni-corrected CIs). Default FALSE.

Details

Why a paired pairwise. stats::pairwise.t.test() returns a square matrix of p-values, which doesn't compose with the rest of the **DMAR** pipeline. This function returns one row per comparison, matching the data.frame(term, value) style used elsewhere.

CI scale. CIs are on the mean-difference scale (unstandardized). When bonferroni_ci = TRUE, the per-pair confidence level is $1-(1-\text{conf_level})/m$, giving Bonferroni-style simultaneous coverage. The CI is built from the paired-t distribution with $n - 1$ degrees of freedom.

Value

A data.frame (class dmar_tbl) with one row per pair. Columns: contrast (the labeled difference, e.g. "B - A"), mean_difference, sd_difference, t_value, df, p_value (raw), p_adjusted, ci_lower, ci_upper, n_pairs. The table records conf_level in its conf_level attribute, which the console prints beneath the table and results_sentence reports. With bonferroni_ci = TRUE that is the simultaneous level of the m intervals, the convention of contrast_test with adjust = "bonferroni" and of the post hoc intervals (ci_tukey_kramer and the rest), and the line beneath the table names the Bonferroni adjustment and the level of each interval, $1 - (1 - \text{conf_level})/m$.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 11.)

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65–70.

See Also

pairwise.t.test, anova_within

Other within-subjects analysis: anova_within(), anova_within_two_way(), epsilon_corrections(), mauchly_test(), plot_trajectories_fitted()

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), randomization_test(), randomization_test_paired(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# 1. Wide-format input: 4 timepoints x 10 subjects.
set.seed(113)
n <- 10; k <- 4
Y <- matrix(rnorm(n * k, 0, 1), n, k) +
     matrix(rep(seq(0, 0.9, length.out = k), n), n, k, byrow = TRUE) +
     rnorm(n, 0, 1.5)
colnames(Y) <- paste0("T", 1:k)
pairwise_within(Y)

# 2. Long-format input:
long <- data.frame(
  subject = factor(rep(1:n, times = k)),
  time    = factor(rep(paste0("T", 1:k), each = n)),
  y       = as.vector(Y)
)
pairwise_within(long, subject = "subject", condition = "time", outcome = "y")
```

---

parallel_analysis *Parallel Analysis of Common-Factor Eigenvalues*

Description

Compares factor-analytic eigenvalues with independently generated null references. Each correlation matrix is reduced by replacing its diagonal with the communalities of a fixed one-factor ML solution. The procedure concerns common factors, not principal components. The reference percentile is a null threshold; it is not a sampling confidence interval for an observed eigenvalue or a confidence statement about a factor count.

Usage

```
parallel_analysis(
  data = NULL,
  S = NULL,
```

```
  N = NULL,
  items = NULL,
  missing = "fiml",
  ordered = NULL,
  reference = "normal",
  B = 1000L,
  percentile = 0.95,
  conf_level = 0.95,
  seed = NULL
)
```

Arguments

data	Numeric raw observations, with items selecting columns. Supply exactly one of data or S.
S	Named positive definite covariance matrix, with N. Available only for a normal-reference analysis of continuous items.
N	Reference sample size for covariance input. For raw incomplete data it may be supplied as an explicit benchmark. With complete data the default is the number of cases; with FIML it is the complete-case count, and the output explicitly identifies this benchmark.
items	The items to analyze, by name: columns of data, or rows and columns of S, which is then reduced to them, in the order given; only the reduced matrix need be finite and positive definite. NULL, the default, analyzes every column of data or of S. Names that cannot be read as at least three distinct items are refused with an error of class dmar_items_invalid (see dmar_metadata).
missing	"fiml" (default), "ml", "listwise", or "pairwise". Continuous FIML obtains the unrestricted correlation matrix from cov_fiml. Ordered items follow the rule of fa_exploratory and factor_count: with no value missing, leaving missing out analyzes the items as they stand (listwise deletion with nothing to delete, recorded as "listwise"); with a value missing, the choice of "pairwise" or "listwise" is the user's, and leaving missing out is refused; giving "fiml" or "ml" for ordered items is refused whether or not a value is missing. Each refusal has class dmar_argument_invalid, as in those functions, and comes before anything is computed.
ordered	NULL, TRUE, or names of ordered items. The current ordered interface requires all analysis items to be ordered.
reference	"normal" (default) generates independent normal responses. With ordered items these are categorized using the observed univariate category probabilities, then polychoric correlations are estimated. "resample" samples each item's observed responses independently with replacement, preserving its empirical marginal distribution in expectation; ordered correlations remain polychoric.
B	Number of independent reference datasets, a single whole number, default 1000 (each one is generated and fit by a one-factor model). The fewest accepted depends on percentile: the reference quantile is reported only when at least 10 reference datasets lie beyond it and never from fewer than 100, which takes 200

	at the default 95th percentile, 1000 at the 99th, and 100 at the 90th or at the median. A smaller `B` is refused before anything is computed (class `dmar_count_invalid`); see *Details* and `dmar_bootstrap`.
`percentile`	Null reference percentile, default 0.95.
`conf_level`	Confidence level for Monte Carlo uncertainty in the reference percentile, default 0.95. This is not observed-data sampling uncertainty.
`seed`	Optional seed, default `NULL`. A supplied seed restores the caller's random stream on exit.

Details

The fixed communality reduction is computed by `stats::factanal()` with its lower uniqueness bound of 0.005. Observed and null matrices use exactly the same reduction. Defined boundary solutions are retained and counted; failed fits are counted separately.

Reference datasets. The threshold of each rank is the `percentile` quantile of the eigenvalues of the `B` reference datasets, so it is located by the reference datasets that lie beyond it. The function runs only when `B` puts at least 10 of them beyond the quantile, and never with fewer than 100 (200 at the 95th percentile). A reference dataset whose fit fails is dropped and counted, never replaced. The count is held to the number requested. Thresholds and a factor count are reported only when no more than half of the attempted reference datasets failed; otherwise they are `NA`, the observed eigenvalues are returned, and the warning has class `dmar_bootstrap_insufficient`. When some failed and the thresholds were formed the warning has class `dmar_bootstrap_replications_dropped`. The result records `B`, `B_used`, `n_dropped`, `B_required`, and `bootstrap_status` as attributes beside the fuller `counts` list. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the line under the table says so. These rules limit the error of drawing a finite number of reference datasets; they are not a coverage guarantee.

The suggested factor count is the number of consecutive ranks, starting at one, whose observed eigenvalues exceed their reference thresholds. It is evidence to combine with theory, model fit, and interpretability. Selection itself has no confidence interval here. Monte Carlo limits for the reference quantile use binomial order-statistic bounds; the quantile estimate uses R's type 7 interpolation. When reference fits fail, these quantities describe the null distribution conditional on returning a defined fit. Discarding failures can select a different distribution; the half-return rule does not repair that selection or establish the intended unconditional null quantile.

An incomplete-data FIML correlation matrix uses all observed values, whereas the complete-case reference size is only an explicitly labeled benchmark. It is not an effective sample size and is not guaranteed to be conservative. These references do not reproduce a study's missingness mechanism. They do not establish general MAR calibration. If fewer than three complete cases remain, choose a justified reference `N` explicitly rather than silently substituting the total sample size.

Value

A `dmar_tbl` with ranks, observed `estimate`, `reference_quantile`, `reference_mean`, `reference_quantile_ci_lower`, `reference_quantile_ci_upper` (the Monte Carlo confidence limits for the reference quantile), and `exceeds_reference`. Attributes retain `suggested_factors`, `reference_draws`, `counts`, `observed_boundary`, `method`, and an explicit `analysis_note`, with the counts every result of the package that ran replications carries: `B`, `B_used`, `n_dropped`, `B_required`, and `bootstrap_status` (see `dmar_bootstrap`). Observed sampling confidence limits are unavailable and are not inferred from the null distribution.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika, 30*, 179–185. doi:10.1007/BF02289447

Examples

```
set.seed(113)
items <- c("t1_visual_perception", "t2_cubes", "t4_lozenges",
           "t6_paragraph_comprehension", "t7_sentence", "t9_word_meaning")
# B = 200 is the fewest reference data sets the 95th percentile is
# reported from and keeps the example quick; the default is B = 1000.
parallel_analysis(holzinger_swineford, items = items, B = 200, seed = 113)
```

---

plot_ci *Forest-Plot-Style Confidence Interval Display*

Description

Plots one or more effect size estimates with their confidence intervals.

Usage

```
plot_ci(
  ci = NULL,
  estimate = NULL,
  lower = NULL,
  upper = NULL,
  names = NULL,
  n = NULL,
  conf_level = 0.95,
  show_n = TRUE,
  reference_line = NULL,
  xlab = "Effect Size",
  title = NULL,
  palette = "okabe_ito"
)
```

Arguments

ci	A `data.frame` from a DMAR `ci_*` function. When supplied, the function auto-detects the format and extracts the point estimate(s), lower limit(s), and upper limit(s). Explicit `estimate`, `lower`, and `upper` arguments override values parsed from `ci`.

estimate	Numeric vector of point estimates.
lower	Numeric vector of lower confidence limits.
upper	Numeric vector of upper confidence limits.
names	Optional character vector of labels for each effect.
n	Optional numeric vector (or scalar) of sample sizes. Recycled to match the number of effects.
conf_level	Confidence level of the limits, used only for the title (default 0.95). When ci records the level of its interval, as every DMAR table does, that level is used, and a conf_level that differs from it is refused, since the plot draws the limits it is given and cannot change their level.
show_n	Logical. If TRUE (the default), the sample size is annotated above each estimate, with the estimate and its interval printed below.
reference_line	Optional numeric value at which to draw a vertical reference line (e.g., 0 for mean differences, 1 for ratios).
xlab	Label for the horizontal (effect size) axis. Defaults to "Effect Size".
title	Optional plot title.
palette	Character string naming the color palette. The point estimates and interval bars are drawn in the palette's primary color. Defaults to "okabe_ito", base R's colorblind-safe Okabe-Ito palette; "tableau" is also available.

Details

The function accepts either (a) a data.frame produced by an DMAR ci_* function (e.g., ci_smd, ci_R2, ci_omega_squared), or (b) explicit numeric vectors for the estimate(s), lower bound(s), and upper bound(s).

The function recognizes three DMAR output formats:

Long term/value with estimate row Output from ci_smd, which includes a row for the point estimate (e.g., term = "smd") in addition to "ci_lower" and "ci_upper".

Long term/value without estimate Output from ci_R or ci_R2, which contains only "ci_lower" and "ci_upper". Supply the point estimate via the estimate argument.

Wide per-effect format Output from ci_omega_squared or ci_eta_squared, which has one row per effect with columns for each estimate and its limits (omega_squared_partial, omega_squared_partial_ci_lowe omega_squared_partial_ci_upper, and the same for the total) and N, or from ci_eta_squared_generalized, with ci_lower and ci_upper. Each estimate that has limits is drawn (the total of an effect of a factorial design has none), and when a table has limits for two, each row is labeled with the effect and the estimate.

The default title names the level the limits were formed at. An interval formed from unequal tail areas (alpha_lower and alpha_upper of a ci_* function) is drawn with a subtitle that gives the two areas, or, when one of them is 0, says which limit is the one-sided confidence bound, as results_sentence does.

Value

A ggplot2 object.

Note

Requires **ggplot2** (listed in Suggests).

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

ci_smd, ci_R, ci_R2, ci_omega_squared, plot_smd, plot_R2

Other plotting: plot_R2(), plot_equivalence(), plot_fa_confirmatory(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_randomization_test(), plot_regions_of_significance(plot_smd(), plot_trajectories(), plot_trajectories_fitted(), power_equivalence_md_plot()

Examples

```
# From explicit values.
plot_ci(estimate = 0.45, lower = 0.15, upper = 0.75,
        names = "Cohen's d", n = 60, reference_line = 0)

# From ci_smd() output.
ci_result <- ci_smd(smd = 0.5, n_1 = 50, n_2 = 50)
plot_ci(ci_result, n = 100, reference_line = 0)

# Multiple effects from ci_omega_squared(): the expectancy treatment
# and the grade classification in the pygmalion data.
pyg <- pygmalion
pyg$grade <- factor(pyg$grade)
fit <- aov(iq_8 ~ treatment + grade, data = pyg)
omega_result <- ci_omega_squared(fit, estimand = "partial")
plot_ci(omega_result, reference_line = 0,
        xlab = expression(omega^2))
```

---

plot_equivalence *Plot Contrasts Against an Equivalence Region*

Description

Draws a forest-style plot of one or more contrast estimates with their 100(1 - 2α)% confidence intervals against the equivalence region $(-\delta_L, \delta_U)$ and the noninferiority bound $-\delta_L$, colored by the six-way verdict of equivalence_c: an interval entirely inside the region is equivalent; entirely above δ_U, superior; entirely below $-\delta_L$, inferior; a lower limit above $-\delta_L$ with an upper limit at or past δ_U, noninferior only; an upper limit below δ_U with a lower limit at or below $-\delta_L$, not relevantly superior only; and an interval containing both bounds, inconclusive. The geometry *is* the decision rule, so the verdict for each contrast can be read from where its interval falls.

Usage

```
plot_equivalence(
  x = NULL,
  estimate = NULL,
  lower = NULL,
  upper = NULL,
  names = NULL,
  effect_lower = NULL,
  effect_upper = NULL,
  xlab = "Contrast",
  title = NULL,
  palette = "okabe_ito"
)
```

Arguments

`x`	Either a single result from `equivalence_c` or a list of them (a named list supplies the row labels). Alternatively, supply `estimate`, `lower`, and `upper` directly.
`estimate`, `lower`, `upper`	Numeric vectors of contrast estimates and their confidence limits, used when `x` is not supplied.
`names`	Optional character vector of row labels.
`effect_lower`, `effect_upper`	Equivalence bounds on the scale of the estimates, as positive magnitudes (the region drawn is $(-\delta_L, +\delta_U)$, with `effect_lower` as δ_L and `effect_upper` as δ_U). Taken from `x` when it carries `equivalence_c` results; required otherwise. If only `effect_upper` is supplied, the bounds are symmetric.
`xlab`	The horizontal axis label. Default `"Contrast"`.
`title`	Optional plot title.
`palette`	Character string naming the color palette for the verdict colors. Defaults to `"okabe_ito"`, base R's colorblind-safe Okabe-Ito palette; `"tableau"` is also available.

Details

The shaded band is the equivalence region and the dashed vertical lines are its bounds; the solid line at zero marks exact equality, which is the null value of ordinary significance testing and is deliberately *not* a decision boundary here. Verdicts are recomputed from the supplied limits and bounds by the rule `equivalence_c` applies, so the plot cannot disagree with it. The legend sets each one-sided verdict beside the stronger one that implies it, with inconclusive between them (equivalent; superior, noninferior only; inconclusive; not relevantly superior only, inferior).

Value

A `ggplot` object. Requires **ggplot2** to be installed.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., Bandyopadhyay, T., Kelley, K., & Padalunkal, J. J. (2025). A sequential approach for noninferiority or equivalence of a linear contrast under cost constraints. *Psychological Methods, 30*(2), 425–439. doi:10.1037/met0000570

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

See Also

equivalence_c, plot_ci

Other equivalence testing: equivalence_c(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), power_density_equivalence_md(), power_equivalence_c(), power_equivalence_md(), power_equivalence_md_plot(), ss_power_equivalence_c()

Other plotting: plot_R2(), plot_ci(), plot_fa_confirmatory(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_randomization_test(), plot_regions_of_significance(), plot_smd(), plot_trajectories(), plot_trajectories_fitted(), power_equivalence_md_plot()

Examples

```
# Six constructed intervals, one per verdict, against bounds of 5
# (A equivalent, B superior, C noninferior only, D inconclusive,
#  E not relevantly superior only, F inferior):
plot_equivalence(estimate = c(-1.0, 7.0, 3.5, 0.5, -1.5, -8.0),
                 lower    = c(-3.2, 5.5, -1.4, -6.0, -6.6, -10.5),
                 upper    = c( 1.2, 8.5,  8.4,  7.0,  3.6,  -5.5),
                 names    = c("A", "B", "C", "D", "E", "F"),
                 effect_upper = 5)

# From equivalence_c() results; a named list supplies the labels.
res <- list(
  "Focal vs. reference" = equivalence_c(contrast = -5.28, se = 2.49,
                                        df_error = 399, contrast_upper = 5),
  "Within pipeline"     = equivalence_c(contrast = -0.53, se = 2.66,
                                        df_error = 399, contrast_upper = 5)
)
plot_equivalence(res)
```

plot_fa_confirmatory *Plot the Estimates of a Multiple-Factor CFA*

Description

Displays the item-level estimates of a fa_confirmatory fit, one panel per factor, with each estimate's confidence interval. The display is built to make the equality questions behind the classical measurement structures visible: a dashed vertical line marks, per factor, either the common (equated) estimate when the plotted parameter was constrained equal, or the mean of the free estimates as an informal anchor for the question "could these plausibly be one value?". Confidence intervals that all cover the anchor are what equal loadings (or equal error variances, or equal intercepts) would look like; an interval far from it shows which item resists the constraint, and the likelihood ratio test of the two nested fa_confirmatory() fits is the formal companion (see the examples in fa_confirmatory). plot_cfa_k() is the same function under another name.

Usage

```
plot_fa_confirmatory(
  x,
  what = c("loadings", "errors", "intercepts"),
  show_equal_reference = TRUE,
  xlab = NULL,
  title = NULL,
  palette = "okabe_ito"
)

plot_cfa_k(
  x,
  what = c("loadings", "errors", "intercepts"),
  show_equal_reference = TRUE,
  xlab = NULL,
  title = NULL,
  palette = "okabe_ito"
)
```

Arguments

x	A dmar_fa_confirmatory object from fa_confirmatory with the default output = "verbose".
what	Which parameter to display: "loadings" (default, the lambda terms), "errors" (the psi terms), or "intercepts" (the nu terms; requires a fit with the mean structure).
show_equal_reference	Logical. If TRUE (default), draw the dashed per-factor reference line described above. When the parameter was constrained equal the line is the common estimate and is always drawn.

xlab	Label for the horizontal axis. Defaults to a description of the plotted parameter.
title	Optional plot title.
palette	Character string naming the color palette. Defaults to "okabe_ito", base R's colorblind-safe Okabe-Ito palette; "tableau" is also available.

Value

A ggplot2 object.

Note

Requires **ggplot2** (listed in Suggests).

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

fa_confirmatory for the fit; plot_ci for the general forest-style confidence interval display.

Other plotting: plot_R2(), plot_ci(), plot_equivalence(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_randomization_test(), plot_regions_of_significance(), plot_smd(), plot_trajectories(), plot_trajectories_fitted(), power_equivalence_md_plot()

Examples

```
data(holzinger_swineford)
hs_factors <- list(
  verbal = c("t6_paragraph_comprehension",
             "t7_sentence", "t9_word_meaning"),
  deduction = c("t20_deduction", "t22_problem_reasoning",
             "t23_series_completion"))
res <- fa_confirmatory(holzinger_swineford, hs_factors)

# Are equal loadings plausible? Compare each interval with the anchor.
plot_fa_confirmatory(res)

# The same question for the error variances, the additional constraint
# that separates essentially parallel from essentially tau-equivalent.
plot_fa_confirmatory(res, what = "errors")

# After imposing the constraint, every item in a factor sits at the
# common estimate and the dashed line is that estimate rather than
# the mean of the free ones.
res_equal <- fa_confirmatory(holzinger_swineford, hs_factors,
                   equal_loading = TRUE)
plot_fa_confirmatory(res_equal)
```

plot_forest *Forest Plot of Study Effect Sizes With the Pooled Estimate*

Description

Draws the meta-analyst's central picture: every study's effect size with its confidence interval, the random effects pooled estimate beneath them, and, by default, the prediction interval showing where the effect of a *new* study is expected to land. Point sizes are proportional to precision (inverse variance), so the eye weighs the studies the way the model does. Requires **ggplot2**.

Usage

```
plot_forest(
  yi,
  vi,
  labels = NULL,
  method = c("reml", "pm", "dl", "fe"),
  hartung_knapp = TRUE,
  conf_level = 0.95,
  show_prediction = TRUE,
  xlab = "Effect size",
  title = NULL,
  palette = "okabe_ito",
  colors = NULL
)
```

Arguments

yi	Numeric vector of study effect sizes.
vi	Sampling variances of yi.
labels	Optional study labels, one per study; defaults to "Study 1", "Study 2", and so on, in the supplied order.
method, hartung_knapp	Passed to meta_es for the pooled row ("reml" and TRUE by default). Under the Hartung-Knapp adjustment, identical effect sizes give a pooled interval of no width and the warning of class dmar_meta_hk_zero_se that meta_es gives, which states the conventional interval; see ?meta_es.
conf_level	Confidence level for the per-study and pooled intervals. Defaults to 0.95.
show_prediction	Logical: draw the prediction interval band on the pooled row? Default TRUE (ignored for method = "fe", which has none).
xlab	Label for the effect size axis. Defaults to "Effect size".
title	Optional plot title.
palette	Palette name. Defaults to "okabe_ito", base R's colorblind-safe Okabe-Ito palette; "tableau" is also available.

`colors` Optional length-2 vector overriding the palette: the study color and the pooled-estimate color.

Value

A **ggplot** object; print it, or add further layers.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Raudenbush, S. W. (1984). Magnitude of teacher expectancy effects on pupil IQ as a function of the credibility of expectancy induction: A synthesis of findings from 18 experiments. *Journal of Educational Psychology, 76*(1), 85–97.

See Also

`meta_es`, `meta_smd`, and `meta_r` for the numbers behind the picture; `teacher_expectancy` for the example data.

Other meta-analysis: `combine_p()`, `meta_contrast()`, `meta_es()`, `meta_r()`, `meta_smd()`

Other plotting: `plot_R2()`, `plot_ci()`, `plot_equivalence()`, `plot_fa_confirmatory()`, `plot_irt_information()`, `plot_mediation_mbco()`, `plot_randomization_test()`, `plot_regions_of_significance()`, `plot_smd()`, `plot_trajectories()`, `plot_trajectories_fitted()`, `power_equivalence_md_plot()`

Examples

```
# Twelve simulated studies whose true effects vary from study to study
# (between-study standard deviation 0.35), so the prediction interval
# for the effect of a new study is visibly wider than the confidence
# interval for the mean effect.
set.seed(113)
k <- 12
n <- sample(20:100, k)                    # per-group sample sizes
theta <- rnorm(k, mean = 0.4, sd = 0.35)  # true study effects
d <- rnorm(k, mean = theta, sd = sqrt(2 / n))
v <- 2 / n + d^2 / (4 * n)
plot_forest(d, v, xlab = "Standardized mean difference (d)")

# The teacher expectancy literature (Raudenbush, 1984): most studies
# cluster near zero, the estimated between-study variance is zero, and
# the prediction interval nearly coincides with the confidence interval.
data(teacher_expectancy)
d <- teacher_expectancy$d
n_e <- teacher_expectancy$n_experimental
n_c <- teacher_expectancy$n_control
v <- (n_e + n_c) / (n_e * n_c) + d^2 / (2 * (n_e + n_c))
plot_forest(d, v, labels = teacher_expectancy$author,
            xlab = "Standardized mean difference (d)")
```

`plot_irt_information` *Plot an Item Response Theory Information Curve*

Description

Draws the information function computed by `irt_information`: either the test information curve, with the standard error of the latent trait estimate on a secondary axis, or one curve per item. The test view answers "where on the latent continuum does this scale measure precisely?", and because the standard error is $1/\sqrt{I(\theta)}$ the same picture shows the precision directly. The item view decomposes that curve, since information is additive across items, and so shows which items cover which part of the continuum.

Usage

```
plot_irt_information(
  x,
  what = c("test", "item"),
  show_se = TRUE,
  show_peak = TRUE,
  palette = "okabe_ito",
  title = NULL,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

`x`	The result of `irt_information`.
`what`	Which curves to draw: `"test"` (default) for the test information function, or `"item"` for one curve per item.
`show_se`	Logical. When `TRUE` (the default) and `what = "test"`, the standard error of the latent trait estimate is drawn as a dashed curve against a secondary axis. The layer is omitted when the standard error is not finite and varying over the grid (for example when test information is zero somewhere).
`show_peak`	Logical. When `TRUE` (the default) and `what = "test"`, a vertical dotted line marks the value of `trait_value` at which test information peaks on the supplied grid.
`palette`	Character string naming the color palette. Defaults to `"okabe_ito"`, base R's colorblind-safe Okabe-Ito palette; `"tableau"` is also available.
`title`	Optional plot title.
`xlab`	Label for the horizontal axis. Defaults to a description of the latent trait metric.
`ylab`	Label for the vertical axis. Defaults to a description of the information plotted.

Details

The secondary axis is a linear rescaling of the primary axis, so the dashed standard error curve shares the panel with the information curve without either being distorted relative to its own axis. The standard error is largest where information is smallest, which is why the two curves run in opposite directions.

Value

A `ggplot2` object.

Note

Requires **ggplot2** (listed in `Suggests`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Embretson, S. E., & Reise, S. P. (2000). *Item response theory for psychologists*. Lawrence Erlbaum.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph Supplement, 34*(4, Pt. 2), 1–97.

See Also

`irt_information`

Other plotting: `plot_R2()`, `plot_ci()`, `plot_equivalence()`, `plot_fa_confirmatory()`, `plot_forest()`, `plot_mediation_mbco()`, `plot_randomization_test()`, `plot_regions_of_significance()`, `plot_smd()`, `plot_trajectories()`, `plot_trajectories_fitted()`, `power_equivalence_md_plot()`

Examples

```
info <- irt_information(
  discrimination = c(mood_1 = 1.4, mood_2 = 0.9, mood_3 = 1.1),
  threshold = c(-1.5, -0.5, 0.5, 1.5, 0.0, 0.8),
  item = c(rep("mood_1", 4), "mood_2", "mood_3")
)

# Test information with the standard error on the secondary axis.
plot_irt_information(info)

# One curve per item.
plot_irt_information(info, what = "item")
```

plot_mediation_mbco *Plot Conditional Effects From a Moderated Mediation Analysis*

Description

Draws the conditional effects from a mediation_mbco analysis that declared a moderator: for each moderated pathway effect, the curve tracing how the effect changes over the moderator's range, with a pointwise confidence band, the probed values marked, a dashed reference line at zero, and a rug showing where the moderator was actually observed. The picture answers, at a glance, the questions the table answers row by row: how large is the effect at any given moderator value, where (if anywhere) does its interval exclude zero, and over what part of the moderator's range the data can support either statement.

Usage

```
plot_mediation_mbco(
  x,
  effects = NULL,
  conf_level = NULL,
  B = 10000,
  from = NULL,
  to = NULL,
  n_grid = 200,
  show_probe_values = TRUE,
  show_rug = TRUE,
  palette = c("okabe_ito", "tableau"),
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  seed = NULL
)
```

Arguments

x	A dmar_mediation_mbco object returned by mediation_mbco with a moderator. An object fit without a moderator has no conditional effects to draw, and the function says so.
effects	Character vector naming which moderated effects to draw, using the base effect names from the result table (e.g., "indirect_via_m", "total_effect"). Defaults to all moderated effects. Unmoderated effects are flat lines and are not drawn.
conf_level	Confidence level for the band. Defaults to the level used when the object was fit.
B	Number of Monte Carlo draws behind the band, a single whole number. Defaults to 10000, because a draw costs one evaluation of each effect and no refit. The limits of the band are quantiles of the draws, so the fewest accepted is

400 at a 95 percent level (200 at 0.90, 2000 at 0.99), 10 draws beyond each limit; a smaller B is refused (class dmar_count_invalid). A draw whose conditional effect is not a finite number at a value of the moderator is dropped there and counted; one warning (class dmar_bootstrap_replications_dropped) says so, and where more than half were dropped the band is not drawn (class dmar_bootstrap_insufficient). options(dmar.bootstrap_exploratory = TRUE) lets a smaller B run, with a warning of class dmar_bootstrap_exploratory. See dmar_bootstrap.

from, to
: Range of moderator values to draw. Defaults to the observed range of the moderator. Values outside the observed range are extrapolation; the rug makes that visible.

n_grid
: Number of grid points along the moderator at which the curve and band are evaluated. Defaults to 200.

show_probe_values
: Logical. If TRUE (default), mark the probed moderator values (the _at_ rows of the result table) as points on each curve.

show_rug
: Logical. If TRUE (default), draw a rug of the observed moderator values along the horizontal axis.

palette
: Character string naming the color palette. Defaults to "okabe_ito", base R's colorblind-safe Okabe-Ito palette; "tableau" is also available.

xlab, ylab, title
: Optional axis labels and title. The defaults name the moderator on the horizontal axis and describe the vertical axis as the conditional effect of x on y.

seed
: Optional integer seed for the Monte Carlo band, used locally (the caller's random number generator state is restored on exit). Default NULL draws the band from the current random number stream, which advances that stream.

Details

What is drawn, and where it comes from. A pathway effect in a model with interactions is a polynomial in the moderator: a straight line when the pathway is moderated in one place (its slope is the index of moderated mediation), a curve when it is moderated in more than one. mediation_mbco derives each polynomial symbolically and stores it with the result, so this function evaluates the same quantity the table probes, just everywhere in the moderator's range instead of at two or three values. The marked points are exactly the table's _at_ rows.

The band is pointwise. At each grid value of the moderator, the band is a conf_level Monte Carlo confidence interval for the conditional effect at that one value: the path coefficients are drawn from their joint normal approximation (MacKinnon, Lockwood, & Williams, 2004), each draw's polynomial is evaluated along the grid, and the band connects the pointwise quantiles. Read vertically at a single moderator value of interest, it is an ordinary confidence interval. Read horizontally, the moderator values where the band crosses zero estimate the Johnson-Neyman boundaries (Johnson & Neyman, 1936; Preacher, Rucker, & Hayes, 2007), the values separating "interval excludes zero" from "interval includes zero". That horizontal reading scans many intervals at once, so the pointwise band understates the uncertainty of the boundary locations themselves; treat the crossing points as estimates, not as sharp cutoffs, and lean on the table's moderation and constancy tests for the formal question of whether the effect depends on the moderator at all.

The rug guards against extrapolation. The curve can be evaluated at any moderator value, but the data only inform it where the moderator was observed. The rug shows that support directly; a confident-looking band in a region with no rug beneath it is arithmetic, not evidence.

The band and the table may differ slightly. The band is always Monte Carlo, whichever `ci_method` the table used. At a probed value, a Monte Carlo band and a profile likelihood or Wald interval agree closely in large samples but are not the same construction; small discrepancies between the band and an `_at_` row's interval are expected, not a defect.

The plot is an ordinary **ggplot2** object, so any further customization (themes, additional layers, institutional color scales) can be added to the returned value with `+`.

Value

A `ggplot2` object. Its data contains one row per effect and grid value with columns `effect_label`, `w_value`, `estimate`, `band_lower`, and `band_upper`, so the numbers behind the picture are recoverable from the object itself. The band limits are `NA` at a value of the moderator where no interval is formed (see `B`).

Note

Requires **ggplot2** (listed in `Suggests`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Johnson, P. O., & Neyman, J. (1936). Tests of certain linear hypotheses and their application to some educational problems. *Statistical Research Memoirs, 1*, 57–93.

MacKinnon, D. P., Lockwood, C. M., & Williams, J. (2004). Confidence limits for the indirect effect: Distribution of the product and resampling methods. *Multivariate Behavioral Research, 39*(1), 99–128. doi:10.1207/s15327906mbr3901_4

Preacher, K. J., Rucker, D. D., & Hayes, A. F. (2007). Addressing moderated mediation hypotheses: Theory, methods, and prescriptions. *Multivariate Behavioral Research, 42*(1), 185–227. doi:10.1080/00273170701341316

Tofighi, D., & Kelley, K. (2020). Improved inference in mediation analysis: Introducing the model-based constrained optimization procedure. *Psychological Methods, 25*(4), 496–515. doi:10.1037/met0000259

See Also

`mediation_mbco` for the analysis this function displays; `regions_of_significance` for the analogous display for mixed-effects model interactions.

Other plotting: `plot_R2()`, `plot_ci()`, `plot_equivalence()`, `plot_fa_confirmatory()`, `plot_forest()`, `plot_irt_information()`, `plot_randomization_test()`, `plot_regions_of_significance()`, `plot_smd()`, `plot_trajectories()`, `plot_trajectories_fitted()`, `power_equivalence_md_plot()`

Examples

```
# First-stage moderated mediation: the effect of x on m depends on
# w, so the indirect effect of x on y through m is a line in w. The
# simulated x reaches y only through m, and the model fit below
# carries no direct path, so the indirect effect is the whole effect
# of x and the picture has one curve.
set.seed(113)
n <- 300
x <- rnorm(n)
w <- rnorm(n)
m <- 0.5 * x + 0.3 * w + 0.4 * x * w + rnorm(n)
y <- 0.5 * m + 0.1 * w + rnorm(n)
d_mod <- data.frame(x = x, w = w, m = m, y = y)

# Fit with the moderator declared. Every reported row costs its own
# constrained null model fit in OpenMx, so the Wald interval and two
# probe values keep the fit quick; the default probe values are the
# moderator's mean and one standard deviation either side, and the
# curve and its band cover the whole range of w either way.
model_mod <- "
  m ~ x + w + x:w
  y ~ m + w
"
res <- mediation_mbco(model_mod, data = d_mod, x = "x", y = "y",
                      moderator = "w", ci_method = "wald",
                      probe_values = c(low = -1, high = 1))

# The band draws B coefficient vectors from their joint normal
# approximation. B = 2000 keeps the example quick; a reported figure
# deserves the default B = 10000. The probed values are marked on
# the curve, and the rug shows where w was observed.
plot_mediation_mbco(res, B = 2000, seed = 113)

# The same curve over a chosen range of w, with a 90% band. The
# 'effects' argument names the pathway to draw; in a model that also
# carries a direct path the total effect is moderated too, and it is
# drawn as a second curve unless 'effects' selects one of them.
plot_mediation_mbco(res, effects = "indirect_via_m", from = -2,
                    to = 2, conf_level = 0.90, B = 2000, seed = 113)
```

`plot_R2` *Visualize the Proportion of Variance Explained (R^2)*

Description

Creates a horizontal bar chart showing the observed R^2 as a proportion of total variance, with an optional confidence interval displayed beneath the bar and sample size / predictor-count annotations.

Usage

```
plot_R2(
  R2,
  N = NULL,
  p = NULL,
  conf_level = 0.95,
  show_ci = TRUE,
  show_n = TRUE,
  random_predictors = TRUE,
  title = NULL,
  palette = "okabe_ito",
  colors = NULL
)
```

Arguments

R2	The observed squared multiple correlation coefficient ($0 \leq R^2 \leq 1$).
N	Total sample size.
p	Number of predictors.
conf_level	Confidence level for the confidence interval (default 0.95).
show_ci	Logical. If TRUE (the default), a confidence interval is shown beneath the proportion bar. Requires both N and p.
show_n	Logical. If TRUE (the default), N and p are annotated on the plot.
random_predictors	Logical. Whether the predictors are random (TRUE, the default) or fixed. Passed to ci_R2.
title	Optional plot title.
palette	Character string naming the color palette used for the "Explained" portion of the bar when colors is NULL. Defaults to "okabe_ito", base R's colorblind-safe Okabe-Ito palette; "tableau" is also available.
colors	Optional character vector of length 2: the first color fills the "Explained" portion of the bar, the second the "Unexplained" portion. When NULL (the default), the "Explained" portion uses the first color of palette and the "Unexplained" portion a neutral light gray.

Value

A ggplot2 object.

Note

Requires **ggplot2** (listed in Suggests).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*, 524–555. doi:10.1080/00273170802490632

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

See Also

ci_R2, ci_R, plot_ci, plot_smd

Other plotting: plot_ci(), plot_equivalence(), plot_fa_confirmatory(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_randomization_test(), plot_regions_of_significance(plot_smd(), plot_trajectories(), plot_trajectories_fitted(), power_equivalence_md_plot()

Examples

```
# The default display: the observed R2 fills its share of the bar, the
# confidence interval sits beneath it, and N and p are annotated.
plot_R2(R2 = 0.25, N = 100, p = 5)

# With fixed predictors and a 90% confidence interval. The interval
# beneath the bar is now the fixed predictor interval from ci_R2, and
# its label reads 90% rather than 95%.
plot_R2(R2 = 0.35, N = 200, p = 3, conf_level = 0.90,
        random_predictors = FALSE)

# Without the interval or the sample size annotation, the bar stands
# alone, which is the display to use when N and p are not known.
plot_R2(R2 = 0.10, show_ci = FALSE, show_n = FALSE)
```

---

plot_randomization_test

Plot the Randomization Distribution Behind a Randomization Test

---

Description

Displays the reference distribution that randomization_test built by reassigning the observed scores to the two groups, with the observed statistic marked and every reassignment at least as extreme as the observed one shaded. The shaded reassignments are the ones the test counted, so under exact enumeration the shaded proportion is the *p*-value, and the figure shows where that number came from instead of only reporting it.

Usage

```
plot_randomization_test(object, bins = 40L, palette = "okabe_ito")
```

Arguments

object	A result of randomization_test. A reassignment whose statistic is infinite has no place on the axis; the histogram shows the finite values and the subtitle states how many reassignments are not drawn. Infinite values occur in the reference distribution only, under statistic = "t", for a reassignment that leaves both groups constant at different values (with a two-valued outcome, the one that puts every high score in one group). The observed statistic is always finite, because randomization_test refuses statistic = "t" when the observed groups are themselves constant at different values, so the line that marks it is always drawn.
bins	Number of histogram bins used to display the reference distribution. Defaults to 40.
palette	Character; the color palette. Defaults to "okabe_ito", base R's colorblind-safe Okabe-Ito palette; "tableau" is also available.

Details

Reading the figure is the point of it. The spread of the distribution is what the reassignments alone can produce when the grouping is irrelevant, which is the null hypothesis of the test. If the observed statistic sits inside that spread, reassignment alone explains it. If it sits out in a tail, few reassignments reproduce it, and that scarcity is the evidence. No normal or *t* distribution appears anywhere in the construction. This is the display Chapter 1 of Maxwell, Delaney, and Kelley (2027) uses to introduce the logic of the randomization test.

Value

A ggplot object, which can be printed or further modified with the usual **ggplot2** verbs.

Author(s)

Ken Kelley

References

Fisher, R. A. (1935). *The design of experiments*. Oliver & Boyd.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 1 on the logic of the randomization test.)

See Also

randomization_test for the test itself.

Other plotting: plot_R2(), plot_ci(), plot_equivalence(), plot_fa_confirmatory(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_regions_of_significance(), plot_smd(), plot_trajectories(), plot_trajectories_fitted(), power_equivalence_md_plot()

Examples

```
treatment <- c(80, 84, 79, 88, 83)
control   <- c(72, 75, 68, 81, 74)
rt <- randomization_test(group_1 = treatment, group_2 = control)
plot_randomization_test(rt)
```

`plot_regions_of_significance` *Plot Regions of Significance for a Covariate by Group Interaction*

Description

Draws the estimated group difference $\hat{D}(x)$ across the observed range of the covariate, with the confidence band that the region of significance is read from, a reference line at zero, and vertical lines at the boundaries of the region. Wherever the band clears zero the groups differ significantly, so the boundaries are exactly the covariate values at which the band touches the zero line: the plot *is* the decision rule, so the region of significance can be read from it directly.

Usage

```
plot_regions_of_significance(
  x,
  data = NULL,
  conf_level = 0.95,
  method = c("simultaneous", "pointwise"),
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  palette = "okabe_ito",
  facet = NULL,
  n_points = 200L
)
```

Arguments

`x`	A result of `regions_of_significance`. Alternatively a fitted `lm` or `aov`, or a formula with `data` supplied, in which case `regions_of_significance` is called first with `conf_level` and `method`.
`data`, `conf_level`, `method`	Passed to `regions_of_significance`, and used only when `x` is a model or a formula rather than an already computed result.
`xlab`, `ylab`	Axis labels. The defaults name the covariate and the group difference.
`title`	Optional plot title.
`palette`	Character string naming the color palette. Defaults to `"okabe_ito"`, base R's colorblind-safe Okabe-Ito palette; `"tableau"` is also available.

facet	Logical. Draw one panel per pair of groups. Defaults to TRUE when there is more than one pair, which keeps the panels from overplotting each other; set it to FALSE to lay the pairs over one another in a single panel.
n_points	Number of covariate values at which the difference and its band are evaluated. Default 200.

Details

The band is $\hat{D}(x) \pm t_{crit}\sqrt{\mathrm{Var}[\hat{D}(x)]}$ with the same critical value used to find the boundaries, so the picture and the table can never disagree. With the default simultaneous critical value (Potthoff, 1964) the band is a simultaneous band: it holds over the whole covariate range at once, which is what licenses scanning it for the covariate values where the groups differ.

The band is drawn over the covariate values actually observed in the two groups. A boundary that falls outside that range is therefore not drawn, deliberately: it is an extrapolation of two fitted lines into a region with no data, and drawing it would invite reading it as a place where something was observed.

Value

A ggplot object. Requires **ggplot2** to be installed.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Johnson, P. O., & Neyman, J. (1936). Tests of certain linear hypotheses and their application to some educational problems. *Statistical Research Memoirs, 1*, 57–93.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 and its extension on heterogeneity of regression.)

Potthoff, R. F. (1964). On the Johnson-Neyman technique and some extensions thereof. *Psychometrika, 29*(3), 241–256. doi:10.1007/BF02289721

See Also

regions_of_significance, plot_ci

Other plotting: plot_R2(), plot_ci(), plot_equivalence(), plot_fa_confirmatory(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_randomization_test(), plot_smd(), plot_trajectories(), plot_trajectories_fitted(), power_equivalence_md_plot()

Examples

```
# The Pygmalion teacher-expectancy data: post-test IQ (averaged over
# the two follow-ups) on pretest IQ, by condition. The expectancy
# effect is significant only in a band of pretest IQ values.
data(pygmalion)
```

```
pygmalion$iq_post <- (pygmalion$iq_4 + pygmalion$iq_8) / 2
fit <- lm(iq_post ~ iq_pre * treatment, data = pygmalion)
plot_regions_of_significance(fit)

# Three groups: one panel per pair.
set.seed(113)
n <- 150
g <- factor(rep(c("control", "low", "high"), each = n / 3))
x <- rnorm(n, 50, 10)
y <- 2 + 0.5 * x + (g == "high") * (0.4 * x - 15) + rnorm(n, 0, 5)
plot_regions_of_significance(y ~ x * g, data = data.frame(y, x, g))
```

`plot_smd` *Visualize a Standardized Mean Difference With Overlapping Distributions*

Description

Plots two normal distributions separated by the standardized mean difference (*d*). The plot includes a confidence interval for the population effect size and sample size annotations, both shown by default.

Usage

```
plot_smd(
  smd = NULL,
  n_1 = NULL,
  n_2 = NULL,
  group_1 = NULL,
  group_2 = NULL,
  conf_level = 0.95,
  show_ci = TRUE,
  show_n = TRUE,
  title = NULL,
  group_labels = c("Group 1", "Group 2"),
  palette = "okabe_ito",
  colors = NULL
)
```

Arguments

`smd`	The standardized mean difference (Cohen's *d*).
`n_1`	Sample size for Group 1.
`n_2`	Sample size for Group 2.
`group_1`	Raw data for Group 1. When provided, `smd`, `n_1`, and `n_2` are computed from the data.

group_2	Raw data for Group 2.
conf_level	Confidence level for the confidence interval (default 0.95).
show_ci	Logical. If TRUE (the default), a confidence interval for the population standardized mean difference is displayed beneath the distributions. Requires both n_1 and n_2.
show_n	Logical. If TRUE (the default), per-group sample sizes are annotated on the plot.
title	Optional character string for the plot title. Defaults to "Standardized Mean Difference".
group_labels	Character vector of length 2 giving labels for the two groups. Defaults to c("Group 1", "Group 2").
palette	Character string naming the color palette used when colors is NULL. Defaults to "okabe_ito", base R's colorblind-safe Okabe-Ito palette; "tableau" is also available.
colors	Optional character vector of length 2 giving fill colors for the two groups. When NULL (the default), the first two colors of palette are used.

Details

Two unit-variance normal distributions are drawn, centered at 0 (Group 2 / reference) and d (Group 1 / focal). The semi-transparent fills make the overlap visible, giving a direct visual impression of how much the distributions differ.

When show_ci = TRUE and both n_1 and n_2 are available, the function calls ci_smd to compute the noncentral t based confidence interval and displays it as a horizontal bar beneath the curves. A filled dot marks the point estimate and vertical caps mark the confidence bounds.

Value

A ggplot2 object that can be further customized with standard **ggplot2** layers, scales, and themes.

Note

Requires **ggplot2** (listed in Suggests). Install it with install.packages("ggplot2") if needed.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

See Also

smd, ci_smd, plot_ci, plot_R2

Other plotting: plot_R2(), plot_ci(), plot_equivalence(), plot_fa_confirmatory(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_randomization_test(), plot_regions_of_significance(plot_trajectories(), plot_trajectories_fitted(), power_equivalence_md_plot()

Other confidence intervals for effect sizes: ci_R2(), ci_c(), ci_c_ancova(), ci_c_ancova_bp(), ci_correlation, ci_cv(), ci_eta_squared(), ci_eta_squared_generalized(), ci_mahalanobis(), ci_omega_squared(), ci_pvaf(), ci_ratio_of_means(), ci_ratio_of_means_paired(), ci_rc(), ci_reg_coef(), ci_rmsea(), ci_sc(), ci_sc_ancova(), ci_sm(), ci_smd(), ci_smd_c(), ci_snr(), ci_src(), ci_srsnr(), contrast_adjusted()

Examples

```
# From a known standardized mean difference and the two sample sizes:
# the two curves, the interval beneath them, and the sample sizes in
# the corner are the default display.
plot_smd(smd = 0.50, n_1 = 50, n_2 = 50)

# From raw data, where the standardized mean difference and both
# sample sizes are taken from the data. The d annotation here reports
# the sample value rather than a supplied one, and the group labels
# and the title are the reader's own.
set.seed(113)
g1 <- rnorm(40, mean = 0.6, sd = 1)
g2 <- rnorm(40, mean = 0.0, sd = 1)
plot_smd(group_1 = g1, group_2 = g2,
         group_labels = c("Treatment", "Control"),
         title = "Treatment Effect on Reading Scores")

# Without the confidence interval or the sample size annotations, the
# figure is the two curves alone.
plot_smd(smd = 0.80, show_ci = FALSE, show_n = FALSE)
```

---

plot_trajectories *Visualize Observed Individual Trajectories in a Longitudinal Data Set*

Description

Plots one trajectory per subject from a long-format data frame, optionally colored by a grouping variable, and optionally faceted into one panel per subject. Returns a **ggplot2** object that can be further customized.

Usage

```
plot_trajectories(
  data,
  id,
```

```
  time,
  outcome,
  group = NULL,
  ids = NULL,
  n_random = NULL,
  pct_random = NULL,
  facet = FALSE,
  nrow = NULL,
  ncol = NULL,
  show_points = TRUE,
  point_size = 1.5,
  linewidth = 0.5,
  alpha = 0.7,
  palette = "okabe_ito",
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = NULL
)
```

Arguments

data	A long-format data.frame (one row per subject-occasion).
id	Character. Column name in data identifying the subject.
time	Character. Column name for the time / occasion variable (the *x* axis).
outcome	Character. Column name for the outcome / score variable (the *y* axis).
group	Optional character. Column name for a grouping variable used to color the trajectories (and panels, if faceted).
ids	Optional vector of subject IDs to plot.
n_random	Optional integer; randomly sample this many subjects.
pct_random	Optional numeric; sample this percentage of subjects. Values ≤ 1 are interpreted as proportions; values > 1 as percentages. At most one of ids, n_random, or pct_random may be supplied.
facet	Logical. If TRUE, draw one panel per subject via facet_wrap. If FALSE (default), overlay all trajectories in one panel.
nrow, ncol	Optional integers passed to facet_wrap() when facet = TRUE.
show_points	Logical. If TRUE (default), draw the observed points as well as the connecting lines.
point_size	Size of the observed points (default 1.5).
linewidth	Line width for the connecting segments (default 0.5).
alpha	Transparency for points and lines (default 0.7).
palette	Character string naming the color palette used to color the trajectories when group is a discrete (factor, character, or logical) variable. Defaults to "okabe_ito", base R's colorblind-safe Okabe-Ito palette; "tableau" is also available. Ignored when group is NULL or numeric.

`title`, `xlab`, `ylab`
: Optional plot labels. Sensible defaults are taken from `outcome` and `time` when these are `NULL`.

`seed`
: Optional integer random seed used when `n_random` or `pct_random` is supplied. Defaults to `NULL`, under which the subjects are sampled from, and advance, the current random number stream; supply an integer for reproducible subject sampling, which is set for the call only, the caller's random number state being restored when it returns.

Details

The function reworks the original `vit()` (visualize individual trajectories) function to return a single **ggplot2** object instead of producing graphical side effects. Saving is handled by the user via `ggsave`; multi-page output via faceting and `facet_wrap`'s `nrow`/`ncol`.

Value

A `ggplot` object.

Note

Requires **ggplot2** (a `Suggests` dependency).

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

`plot_trajectories_fitted` for plotting observed trajectories together with a fitted multilevel model's predictions.

Other plotting: `plot_R2()`, `plot_ci()`, `plot_equivalence()`, `plot_fa_confirmatory()`, `plot_forest()`, `plot_irt_information()`, `plot_mediation_mbco()`, `plot_randomization_test()`, `plot_regions_of_significance()`, `plot_smd()`, `plot_trajectories_fitted()`, `power_equivalence_md_plot()`

Examples

```
# The Orthodont data from nlme: 27 children, 4 measurements each.
d <- nlme::Orthodont

# Overlay all trajectories, colored by sex.
plot_trajectories(d, id = "Subject", time = "age",
                  outcome = "distance", group = "Sex")

# One panel per child, for six children drawn at random, few enough
# that the figure is quick to draw; drop n_random to get a panel for
# every child. The seed makes the draw reproducible, and the session's
# generator state is left as it was.
plot_trajectories(d, id = "Subject", time = "age",
                  outcome = "distance",
                  n_random = 6, facet = TRUE, ncol = 3,
```

```
                    seed = 113)
```

`plot_trajectories_fitted` *Plot Observed and Fitted Individual Trajectories From a Multilevel Model*

Description

Given a fitted `lme`/`nlme` (**nlme**) or `lmer` (**lme4**) model, plots each subject's observed values and fitted curve on a smooth time grid, faceted one panel per subject. Per-subject R^2 (squared correlation between observed and fitted values) and root-mean-square error are computed and attached to the returned **ggplot2** object as the `quality_of_fit` attribute.

Usage

```
plot_trajectories_fitted(
  model,
  id = NULL,
  time = NULL,
  outcome = NULL,
  ids = NULL,
  n_random = NULL,
  pct_random = NULL,
  n_grid = 100,
  show_points = TRUE,
  point_size = 1.5,
  linewidth = 0.6,
  alpha = 0.8,
  palette = "okabe_ito",
  nrow = NULL,
  ncol = NULL,
  show_quality = TRUE,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = NULL
)
```

Arguments

`model`	A fitted model object of class `lme`, `nlme`, or `lmerMod`.
`id`, `time`, `outcome`	Optional character names of the ID, time, and outcome columns. When `NULL` (the default), each is auto-detected from the model object: the outcome is the

response variable in the formula, the ID is the first random-effect grouping factor, and time is the first fixed-effect predictor. Override these when the auto-detection is wrong.

`ids`, `n_random`, `pct_random`
: Subject-subsetting options identical to those of `plot_trajectories`; at most one may be supplied.

`n_grid`
: Integer. Number of points used to draw each subject's smooth fitted curve (default `100`).

`show_points`
: Logical. Whether to draw the observed values (default `TRUE`).

`point_size`, `linewidth`, `alpha`, `nrow`, `ncol`
: Visual / layout controls.

`palette`
: Character string naming the color palette; the fitted curve is drawn in the palette's primary color. Defaults to `"okabe_ito"`, base R's colorblind-safe Okabe-Ito palette; `"tableau"` is also available.

`show_quality`
: Logical. If `TRUE` (the default), each panel strip text includes the subject's R^2 and RMSE.

`title`, `xlab`, `ylab`
: Optional plot labels.

`seed`
: Optional integer random seed used when `n_random` or `pct_random` is supplied. Defaults to `NULL`, under which the subjects are sampled from, and advance, the current random number stream; supply an integer for reproducible subject sampling, which is set for the call only, the caller's random number state being restored when it returns.

Details

Reworks the original `vit_fitted()` function by:

- returning a **ggplot2** object instead of writing to graphics devices,
- attaching per-subject quality-of-fit as an attribute rather than assigning it into the global environment, as the original did,
- drawing a smooth fitted curve from a per-subject time grid via `predict(..., re.form = NULL)` for **lme4** fits and `predict(..., level = 1)` for **nlme** fits,
- identifying **lme4** fits by their class `lmerMod` (not `lmer`).

Value

A `ggplot` object. The per-subject quality-of-fit `data.frame` (columns: id column, `r_squared`, `rmse`) is attached as `attr(<plot>, "quality_of_fit")`.

Note

Requires **ggplot2** plus, depending on the model class, **nlme** or **lme4** (`Suggests` dependencies).

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

plot_trajectories

Other plotting: plot_R2(), plot_ci(), plot_equivalence(), plot_fa_confirmatory(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_randomization_test(), plot_regions_of_significance(plot_smd(), plot_trajectories(), power_equivalence_md_plot()

Other within-subjects analysis: anova_within(), anova_within_two_way(), epsilon_corrections(), mauchly_test(), pairwise_within()

Examples

```
# nlme: linear growth in tooth distance over age for the 27 Orthodont
# children. Four of the children are paneled here so the figure is
# quick to draw; drop n_random to get a panel for every child.
fm_nlme <- nlme::lme(distance ~ age, random = ~ age | Subject,
                     data = nlme::Orthodont)
p <- plot_trajectories_fitted(fm_nlme, n_random = 4, seed = 113)
p
attr(p, "quality_of_fit")  # per-subject R^2 and RMSE

# An lme4 fit is handled the same way: the outcome, the subject
# identifier, and the time variable are read from the model. Six of
# the eighteen sleepstudy subjects are paneled here.
fm_lme4 <- lme4::lmer(Reaction ~ Days + (Days | Subject),
                      data = lme4::sleepstudy)
plot_trajectories_fitted(fm_lme4, n_random = 6, seed = 113)
```

---

power_density_equivalence_md

Density Underlying the TOST Power Calculation

Description

Evaluates the integrand whose integral over $(0, \text{upper})$ yields the power of the Schuirmann (1987) two one-sided tests procedure; see power_equivalence_md. Useful for plotting the integrand and for diagnostic work.

Usage

```
power_density_equivalence_md(
  sd_values,
  alpha_level,
  lower_bound,
  upper_bound,
  true_effect,
  true_sd,
  n_per_treatment,
  df_error
)
```

Arguments

sd_values	Numeric vector of σ values at which to evaluate the integrand.
alpha_level	Type I error rate for each of the two one-sided tests.
lower_bound	Lower limit of the equivalence interval on the analysis scale: a difference, or the logarithm of a ratio.
upper_bound	Upper limit of the equivalence interval on the analysis scale: a difference, or the logarithm of a ratio.
true_effect	The true (population) effect on the analysis scale: the difference in treatment means, or the logarithm of their ratio.
true_sd	$\sqrt{\text{error variance}}$.
n_per_treatment	Number of subjects who receive each treatment (the size of each group in a parallel design; every subject in a crossover design), as in power_equivalence_md.
df_error	Error degrees of freedom for true_sd.

Details

The analysis scale. Unlike power_equivalence_md, which takes the bounds and the true effect on the original scale and logs ratios when logscale = TRUE, this function has no logscale argument and transforms nothing: lower_bound, upper_bound, and true_effect are on the scale of the analysis, differences of the means, or, for means compared on the log scale, logarithms of the ratios (log(0.8) for a ratio of 0.8).

Value

A data.frame with one row per supplied sd_values, and columns sd_values and power_density.

Note

See the legacy MBESS package (Kelley, 2007a, 2007b) for additional details and discussion.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Phillips, K. F. (1990). Power of the two one-sided tests procedure in bioequivalence. *Journal of Pharmacokinetics and Biopharmaceutics, 18*(2), 137–144. doi:10.1007/BF01063556

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

See Also

power_equivalence_md, power_equivalence_md_plot

Other equivalence testing: equivalence_c(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), plot_equivalence(), power_equivalence_c(), power_equivalence_md(), power_equivalence_md_plot(), ss_power_equivalence_c()

Examples

```
# Density at a single value of true_sd:
power_density_equivalence_md(sd_values = 0.10, alpha_level = .05,
                             lower_bound = -.2, upper_bound = .2,
                             true_effect = .05, true_sd = .20,
                             n_per_treatment = 24, df_error = 22)

# Vectorized over a grid:
grid <- power_density_equivalence_md(
  sd_values = seq(0.01, 0.40, length.out = 50),
  alpha_level = .05, lower_bound = -.2, upper_bound = .2,
  true_effect = .05, true_sd = .20, n_per_treatment = 24, df_error = 22
)
head(grid)

# On the log scale the bounds and the true effect are logarithms of the
# ratios: the ratios .8, 1.25, and 1.05 of power_equivalence_md(logscale =
# TRUE) are given here as their logarithms.
power_density_equivalence_md(sd_values = 0.10, alpha_level = .05,
                             lower_bound = log(.8), upper_bound = log(1.25),
                             true_effect = log(1.05), true_sd = .20,
                             n_per_treatment = 18, df_error = 16)
```

---

power_equivalence_c *Power of the TOST or Noninferiority Test for a Linear Contrast*

Description

Computes the exact power of the Schuirmann (1987) two one-sided tests procedure, or of the one-sided noninferiority test, for a linear contrast of group means $\psi = \sum_j c_j \mu_j$ with one pooled error term, or for a contrast measured within each unit of a paired design. For equivalence, the power is the probability that the $(1 - 2\alpha)$ confidence interval for ψ lies entirely inside $(-\delta_L, \delta_U)$, computed by numerical integration over the chi distribution of the estimated error standard deviation; for non-inferiority, the power is a noncentral *t* probability in closed form. This is the contrast generalization of power_equivalence_md.

Usage

```
power_equivalence_c(
  c_weights = NULL,
```

```
  n_per_group = NULL,
  N = NULL,
  true_sd,
  contrast_lower = NULL,
  contrast_upper = NULL,
  true_contrast = 0,
  alpha_level = 0.05,
  side = c("equivalence", "noninferiority"),
  df_error = NULL,
  design = c("independent", "paired")
)
```

Arguments

`c_weights`	The contrast weights, required for `design = "independent"`. The weights must sum to zero with the positive weights summing to 1 and the negative weights to -1, so that the bounds are on the raw scale of the response. Leave `NULL` for `design = "paired"`, where the weights are already in the per-unit contrast scores.
`n_per_group`	For `design = "independent"`, the sample sizes per group (if length 1, equal group sizes are assumed); together with `c_weights`, `n_per_group` determines the standard error factor $\sqrt{\sum_j c_j^2/n_j}$. Leave `NULL` for `design = "paired"`, which has no groups of different units and takes `N`; a paired call that gives `n_per_group` is refused.
`N`	For `design = "paired"`, the number of units, each measured under every condition the contrast involves. Every unit contributes one contrast score, so this is the total sample size. Leave `NULL` for `design = "independent"`, where the sizes are given per group in `n_per_group`; an independent call that gives `N` is refused.
`true_sd`	For `design = "independent"`, the error standard deviation (the square root of the mean square error). For `design = "paired"`, the standard deviation of the per-unit contrast scores (for two conditions, of the difference scores). It is σ in the formulas below.
`contrast_lower, contrast_upper`	Bounds on the raw scale of the response, each positive; the equivalence region is $(-\delta_L, +\delta_U)$, where δ_L is `contrast_lower` and δ_U is `contrast_upper`. For `side = "equivalence"`, `contrast_upper` is required and `contrast_lower` defaults to it (symmetric bounds). For `side = "noninferiority"`, the test is against $-\delta_L$ alone, so `contrast_lower` is the argument to give; a call that gives only the upper bound uses it as the margin.
`true_contrast`	The population value of the contrast at which the power is evaluated (ψ in the formulas). Default `0`, the most favorable point for an equivalence declaration.
`alpha_level`	One-sided significance level for each test. Default `0.05`.
`side`	`"equivalence"` (default) for the TOST power, or `"noninferiority"` for the one-sided test against $-\delta_L$.
`df_error`	The error degrees of freedom, ν in the formulas below. Defaults to $\sum_j n_j - J$ for `design = "independent"`, the sum of `n_per_group` less the number of groups,

and to $N - 1$ for `design = "paired"`; supply it directly when the error term comes from a model with more groups or additional predictors than the contrast involves (for example, a five-group model supplying the pooled error for a two-group contrast).

`design` `"independent"` (default) for groups of different units sharing one pooled error term, or `"paired"` for a contrast measured within each unit (two conditions, or a pre and a post measurement, on the same units), analyzed as the mean of the N per-unit contrast scores. The design decides which size argument is read: `n_per_group` for independent groups, `N` for a paired design. See Details.

Details

Equivalence power. Conditional on the estimated error standard deviation S, the $(1 - 2\alpha)$ CI fits inside the bounds on a computable event, and the unconditional power integrates that event over the scaled chi distribution of S on `df_error` degrees of freedom. With `c_weights = c(1, -1)` and equal `n_per_group`, the result reproduces `power_equivalence_md` exactly.

Noninferiority power. The one-sided test rejects when $t = (\hat{\psi} + \delta_L)/\mathrm{SE}(\hat{\psi})$ exceeds $t_{1-\alpha,\nu}$, so the power is $\Pr(T'_\nu(\lambda) > t_{1-\alpha,\nu})$ with noncentrality $\lambda = (\psi + \delta_L)/(\sigma\sqrt{\sum_j c_j^2/n_j})$.

Paired designs. With `design = "paired"`, each unit contributes one contrast score, $d_i = \sum_j c_j y_{ij}$, and the analysis is of their mean: the standard error is $\sigma/\sqrt{N}$ on $N - 1$ degrees of freedom, where N is the number of units (the argument `N`), so $\sqrt{\sum_j c_j^2/n_j}$ above becomes $1/\sqrt{N}$ and both powers are computed as for the independent design. The weights that form the scores should follow the same rule as `c_weights` (positive weights summing to 1, negative to -1), so that the bounds are on the raw scale; for two conditions the score is the difference. Its standard deviation `true_sd` carries the correlation between the measures: for two measures with a common standard deviation σ_y correlated ρ, it is $\sigma_y\sqrt{2(1-\rho)}$, smaller than $\sigma_y\sqrt{2}$, the standard deviation of a difference between independent observations, whenever $\rho > 0$; that is the precision a paired design buys.

The feasibility condition. If the expected half-width of the CI is not smaller than the bounds allow, the equivalence power is zero or near zero regardless of `true_contrast`: an imprecise design cannot declare equivalence even when the arms are truly identical. Planning should target a half-width of about half the bound; see `ss_power_equivalence_c` and `ss_aipe_c`.

Value

A one-row `data.frame` with columns `term` (`"power"`) and `value` (the computed power, in $[0, 1]$).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., Bandyopadhyay, T., Kelley, K., & Padalunkal, J. J. (2025). A sequential approach for noninferiority or equivalence of a linear contrast under cost constraints. *Psychological Methods, 30*(2), 425–439. doi:10.1037/met0000570

Phillips, K. F. (1990). Power of the two one-sided tests procedure in bioequivalence. *Journal of Pharmacokinetics and Biopharmaceutics, 18*(2), 137–144. doi:10.1007/BF01063556

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

See Also

power_equivalence_md, ss_power_equivalence_c, equivalence_c, ss_aipe_c

Other equivalence testing: equivalence_c(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), plot_equivalence(), power_density_equivalence_md(), power_equivalence_md(), power_equivalence_md_plot(), ss_power_equivalence_c()

Examples

```
# 1. Two groups of 61 and 113 sharing a five-group pooled error term
#    (so df_error = 404 - 5 = 399), bounds of 5 raw-scale points:
#    the design's probability of declaring equivalence when the
#    groups are truly identical.
power_equivalence_c(c_weights = c(1, -1), n_per_group = c(61, 113),
                    true_sd = 15.67, contrast_upper = 5,
                    true_contrast = 0, df_error = 399)

# 2. The same design's noninferiority power at the same point.
power_equivalence_c(c_weights = c(1, -1), n_per_group = c(61, 113),
                    true_sd = 15.67, contrast_upper = 5,
                    true_contrast = 0, df_error = 399,
                    side = "noninferiority")

# 3. Agreement with power_equivalence_md() in the two-group case
#    (Phillips, 1990, Table 1: expected 0.8029678).
power_equivalence_c(c_weights = c(1, -1), n_per_group = 24, true_sd = 0.20,
                    contrast_lower = 0.2, contrast_upper = 0.2,
                    true_contrast = 0.05, df_error = 22)

# 4. A paired design: 50 units measured under both conditions, with
#    difference scores whose SD is 1, and a noninferiority margin of
#    0.5. No upper bound is needed for a noninferiority question.
power_equivalence_c(N = 50, true_sd = 1, contrast_lower = 0.5,
                    side = "noninferiority", design = "paired")
```

---

power_equivalence_md *Power of the Two One-Sided Tests Procedure (TOST) for Equivalence*

Description

Computes the power of the Schuirmann (1987) two one-sided tests procedure , the probability that a $(1-2\alpha)$ confidence interval for the mean difference (or ratio, on the log scale) lies entirely within the equivalence interval $[\theta_1, \theta_2]$ (lower_bound and upper_bound), by numerical integration over the chi distribution of the sample standard deviation.

Usage

```
power_equivalence_md(
  alpha_level,
  logscale,
  lower_bound,
  upper_bound,
  true_effect,
  true_sd,
  n_per_treatment,
  df_error
)
```

Arguments

alpha_level	Type I error rate for each of the two one-sided tests (typically 0.05). The full equivalence test uses a $(1 - 2\alpha)$ confidence interval.
logscale	Logical. If TRUE, treatment means are compared on the logarithmic scale; lower_bound, upper_bound, and true_effect are then given as ratios of test to reference on the original scale (untransformed) and are log-transformed internally. If FALSE, they are differences of the means.
lower_bound	Lower limit of the equivalence interval on the original scale: a ratio when logscale = TRUE (logged internally), a difference otherwise.
upper_bound	Upper limit of the equivalence interval on the original scale: a ratio when logscale = TRUE (logged internally), a difference otherwise.
true_effect	The true (population) effect: the ratio of the treatment means on the original scale when logscale = TRUE (logged internally), the difference in treatment means otherwise.
true_sd	$\sqrt{\text{error variance}}$; root-MSE from an ANOVA. On the log scale, this is the coefficient of variation.
n_per_treatment	Number of subjects who receive each treatment: the size of each group in a parallel design, and the number of subjects in a crossover design, where every subject receives both treatments. It is the same quantity in both designs, the n of the standard error $\sigma\sqrt{2/n}$ of the estimated difference, so in a parallel design it is the size of one group and never the total. Must be greater than 1.
df_error	Error degrees of freedom associated with true_sd.

Details

The computation conditions on the error standard deviation the study will actually observe. Given that value, whether the confidence interval fits inside the equivalence interval is an ordinary normal probability, and the power is that probability averaged over the chi distribution the error standard deviation follows on df_error degrees of freedom. The averaging is carried out on a unit-free scale, with the equivalence limits expressed in standard errors and the error standard deviation as a multiple of true_sd, so the power depends on the design rather than on the units of the response: with logscale = FALSE, multiplying lower_bound, upper_bound, true_effect, and true_sd by a common factor leaves the answer unchanged.

The scale of the bounds and the true effect. This function and power_equivalence_md_plot take lower_bound, upper_bound, and true_effect on the original scale: ratios of test to reference with logscale = TRUE, which are logged before the computation, and differences with logscale = FALSE, which are used as given. power_density_equivalence_md takes them on the analysis scale and transforms nothing, so power_equivalence_md(logscale = TRUE, lower_bound = .8, ...) corresponds to power_density_equivalence_md(lower_bound = log(.8),...).

For Phillips's (1990) original example (regular-scale two-period crossover with $\theta_1 = -0.2$, $\theta_2 = 0.2$, CV $= 0.20$, $\delta = 0.05$, $n = 24$, $\nu = 22$, which are lower_bound, upper_bound, true_sd, true_effect, n_per_treatment, and df_error), this function reproduces the published value of 0.8029678 (Phillips, 1990, Table 1, 5th row, 5th column).

Value

A one-row data.frame with columns term ("power") and value (the computed power, in $[0, 1]$).

Note

See the legacy MBESS package (Kelley, 2007a, 2007b) for additional details and discussion.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Diletti, E., Hauschke, D., & Steinijans, V. W. (1991). Sample size determination of bioequivalence assessment by means of confidence intervals. *International Journal of Clinical Pharmacology, Therapy and Toxicology, 29*(1), 1–8.

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Phillips, K. F. (1990). Power of the two one-sided tests procedure in bioequivalence. *Journal of Pharmacokinetics and Biopharmaceutics, 18*(2), 137–144. doi:10.1007/BF01063556

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

See Also

power_equivalence_md_plot, power_density_equivalence_md

Other equivalence testing: equivalence_c(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), plot_equivalence(), power_density_equivalence_md(), power_equivalence_c(), power_equivalence_md_plot(), ss_power_equivalence_c()

Examples

```
# Table 1 of Phillips, 1990, fifth row and fifth column, where the
# published power is 0.8029678.
power_equivalence_md(alpha_level = .05, logscale = FALSE,
                     lower_bound = -.2, upper_bound = .2,
                     true_effect = .05, true_sd = .20,
                     n_per_treatment = 24, df_error = 22)

# Table 1 of Diletti et al., 1991, on the log scale, so the limits and
# the true difference are ratios of test to reference. The published
# power is 0.7922796.
power_equivalence_md(alpha_level = .05, logscale = TRUE,
                     lower_bound = .8, upper_bound = 1.25,
                     true_effect = 1.05, true_sd = .20,
                     n_per_treatment = 18, df_error = 16)
```

---

power_equivalence_md_plot

Plot TOST Equivalence-Test Power Curves Over a Range of True Differences

---

Description

For each sample size in `n_per_treatment`, draws power as a function of the true mean difference (or ratio, on the log scale), evaluated at 201 equally spaced points across the equivalence interval. Returns a **ggplot2** object; the underlying numerical grid is attached as `attr(<plot>, "power_grid")`.

Usage

```
power_equivalence_md_plot(
  alpha_level,
  logscale,
  lower_bound,
  upper_bound,
  true_sd,
  n_per_treatment,
  df_error,
  title = NULL,
  subtitle = NULL
)
```

Arguments

`alpha_level`	Type I error rate for each of the two one-sided tests.
`logscale`	Logical. If `TRUE`, the means are compared on the logarithmic scale.

lower_bound	Lower limit of the equivalence interval on the original scale, as in power_equivalence_md: a ratio when logscale = TRUE, a difference otherwise.
upper_bound	Upper limit of the equivalence interval on the original scale: a ratio when logscale = TRUE, a difference otherwise.
true_sd	$\sqrt{\text{error variance}}$.
n_per_treatment	Vector of sample sizes, each the number of subjects who receive each treatment (the size of each group in a parallel design; every subject in a crossover design), as in power_equivalence_md. One curve is drawn per element.
df_error	Vector of error degrees of freedom for true_sd, the same length as n_per_treatment.
title	Optional plot title (default "Power of TOST").
subtitle	Optional subtitle (typically a reference like "Phillips, Figure 3").

Value

A ggplot object. The 201-row power grid (column 1, true_effect: the true effect on the original scale, a ratio when logscale = TRUE and a difference otherwise, from lower_bound to upper_bound; remaining columns: power for each element of n_per_treatment, labeled n=<size>) is attached as attr(<plot>, "power_grid").

Note

See the legacy MBESS package (Kelley, 2007a, 2007b) for additional details and discussion.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Diletti, E., Hauschke, D., & Steinijans, V. W. (1991). Sample size determination of bioequivalence assessment by means of confidence intervals. *International Journal of Clinical Pharmacology, Therapy and Toxicology, 29*(1), 1–8.

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Phillips, K. F. (1990). Power of the two one-sided tests procedure in bioequivalence. *Journal of Pharmacokinetics and Biopharmaceutics, 18*(2), 137–144. doi:10.1007/BF01063556

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

See Also

power_equivalence_md, power_density_equivalence_md

Other equivalence testing: equivalence_c(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), plot_equivalence(), power_density_equivalence_md(), power_equivalence_c(), power_equivalence_md(), ss_power_equivalence_c()

Other plotting: plot_R2(), plot_ci(), plot_equivalence(), plot_fa_confirmatory(), plot_forest(), plot_irt_information(), plot_mediation_mbco(), plot_randomization_test(), plot_regions_of_significance(plot_smd(), plot_trajectories(), plot_trajectories_fitted()

Examples

```
# One curve per sample size, showing power against the true mean
# difference. The seven sample sizes are the ones behind Figure 3 of
# Phillips, 1990, so the figure reproduces that one; every curve
# evaluates the power integral at 201 true differences.
n      <- c(9, 12, 18, 24, 30, 40, 60)
df_err <- c(7, 10, 16, 22, 28, 38, 58)
fig <- power_equivalence_md_plot(
  alpha_level = .05, logscale = FALSE,
  lower_bound = -.2, upper_bound = .2, true_sd = .20,
  n_per_treatment = n, df_error = df_err,
  subtitle = "Phillips Figure 3"
)
fig

# The numbers behind the curves travel with the figure, so a particular
# power value can be read off rather than eyeballed. The first column,
# true_effect, is the true difference and the remaining columns give power,
# one column per sample size. Power is highest where the true difference
# is zero.
power_grid <- attr(fig, "power_grid")
power_grid[which.min(abs(power_grid[, "true_effect"])), ]

# Figure 1c of Diletti et al., 1991, is the same idea on the log scale,
# where the equivalence limits are the 0.80 to 1.25 ratio bounds used
# in bioequivalence work, given as ratios.
n_d      <- c(8, 12, 18, 24, 30, 40, 60)
df_err_d <- c(6, 10, 16, 22, 28, 38, 58)
power_equivalence_md_plot(
  alpha_level = .05, logscale = TRUE,
  lower_bound = .8, upper_bound = 1.25, true_sd = .20,
  n_per_treatment = n_d, df_error = df_err_d,
  subtitle = "Diletti, Figure 1c"
)
```

---

power_fisher_exact *Power of Fisher's Exact Test (Noncentral Hypergeometric)*

Description

Computes the power of Fisher's exact test (Fisher, 1934) for the 2×2 table under Fisher's noncentral hypergeometric distribution, where the alternative is parameterized by the true odds ratio ψ. The power is the probability that the (conditional) exact test rejects $H_0 : \psi = 1$ when in fact $\psi = \psi_1 \neq 1$.

Usage

```
power_fisher_exact(
  n_1,
  n_2,
  p_1,
  p_2,
  alpha_level = 0.05,
  alternative = c("two_sided", "less", "greater")
)
```

Arguments

`n_1`, `n_2`	Group sample sizes for the two columns of the 2×2 table (e.g., treatment vs control).
`p_1`, `p_2`	Success probabilities in the two groups under the alternative. The odds ratio under the alternative is $\psi = (p_1/(1-p_1))/(p_2/(1-p_2))$.
`alpha_level`	Two-sided significance level. Default `0.05`.
`alternative`	One of `"two_sided"` (default; the base-R spelling `"two.sided"` is accepted as an alias), `"less"`, or `"greater"`.

Details

Setup. Fisher's exact test conditions on the marginal totals of the 2×2 table:

	Success	Failure	Total
Group 1	X_1	$n_1 - X_1$	n_1
Group 2	$S - X_1$	$(n_1 + n_2) - n_1 - S + X_1$	n_2
Total	S	$n_1 + n_2 - S$	$n_1 + n_2$

Under $H_0 : \psi = 1$, X_1 given the marginals follows the *central* hypergeometric. Under the alternative ψ_1, X_1 follows *Fisher's noncentral* hypergeometric with odds ratio parameter ψ_1 (Fisher, 1935; Fog, 2008), the conditional distribution of one binomial count given the total of two independent binomials. (Wallenius' noncentral hypergeometric, which arises from sequential biased urn sampling, is a different distribution and is not the relevant one here.)

Power calculation. For each possible value of the column-1 total $S = 0, 1, \ldots, n_1 + n_2$:

1. Determine the rejection region under $H_0 : \psi = 1$ using the central hypergeometric.
2. Compute $\Pr(X_1 \in \text{reject} \mid \psi = \psi_1, S)$ under the noncentral hypergeometric.
3. Weight by $\Pr(S \mid \psi_1)$, the marginal probability of total column-1 successes under the alternative.

The power is the resulting weighted sum.

Value

A `data.frame` with rows for the power, the alternative-odds-ratio ψ_1, the alternative success probabilities p_1 and p_2, the expected column-1 total $E[S]$ across both groups, and the row / column totals.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Fisher, R. A. (1934). *Statistical methods for research workers* (5th ed.). Oliver & Boyd.

Fisher, R. A. (1935). The logic of inductive inference. *Journal of the Royal Statistical Society, 98*(1), 39–82.

Fog, A. (2008). Sampling methods for Wallenius' and Fisher's noncentral hypergeometric distributions. *Communications in Statistics – Simulation and Computation, 37*(2), 241–257. doi:10.1080/03610910701790236

Good, P. I. (2000). *Permutation tests: A practical guide to resampling methods for testing hypotheses* (2nd ed.). Springer.

O'Brien, R. G. (1998). A tour of UnifyPow: A SAS module/macro for sample size analysis. *Proceedings of the 23rd SAS Users Group International Conference*, 1346–1355.

See Also

`fisher.test`

Other sample size for power: `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_composite_sc`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`, `ss_power_reg_coef_sensitivity()`, `ss_power_rm_anova()`, `ss_power_sc()`, `ss_power_sem()`, `ss_power_smd()`, `ss_power_split_plot_anova()`

Examples

```
# 1. Power for n_1 = n_2 = 30 when the two population proportions are
#    0.6 and 0.3. The value comes from enumerating the conditional
#    reference set the test itself uses, not from a normal
#    approximation, so it is the power of the test as conducted.
power_fisher_exact(n_1 = 30, n_2 = 30, p_1 = 0.6, p_2 = 0.3)

# 2. A difference of 0.10 between proportions is much harder to detect:
#    100 per group is not close to enough. Sample size requirements grow
#    quickly as the difference between the two proportions shrinks.
power_fisher_exact(n_1 = 100, n_2 = 100, p_1 = 0.45, p_2 = 0.35)
```

prime_time_achievement

Indiana Prime Time Third Grade Achievement Evaluation Data

Description

The complete student level data file from the 2000 to 2001 Indiana Department of Education program evaluation of Project Prime Time, reported in Lapsley, Daytner, Kelley, and Maxwell (2002, ERIC ED466679). The evaluation examined the academic performance of N = 10,927 third grade students in n = 573 classrooms (here 586 by the `paste(corp, school, class)` rule), n = 163 schools, n = 61 school corporations, and 9 Indiana educational service regions as a function of class size, pupil to teacher ratio, and the presence of a Prime Time instructional assistant. The data have been used as a multilevel example through chapters 3, 4, 6, 9, and 10 of Finch, Bolin, and Kelley (2019, *Multilevel Modeling Using R*, 2nd ed., CRC Press) and are made available here as a benchmark data set for the design, measurement, and analysis of nested data.

Usage

```
prime_time_achievement
```

Format

A data frame with 10,927 observations on 113 variables. Variables fall into seven blocks: three derived unique cluster identifiers, student level demographics and ability and achievement scores, classroom level variables, school level variables, and school corporation (district) level variables. Original Indiana DOE variable spellings are preserved, including the typos `calender` (calendar), `hispanc1` and `hispanc2` (Hispanic), and `rmediate` (remediate), for code compatibility with Finch, Bolin, and Kelley (2019). The SPSS variable label from the source file is available as `attr(prime_time_achievement$VAR, "label")` for every variable carried over from the SPSS file; the one derived recode without a source label is `classize` (see its entry below).

`id` Student identifier in the source file (not guaranteed unique; see `class_id` etc. for stable cluster keys).

`region` Indiana educational service region, coded 1 to 9. Sampling stratifier (25% of corporations per region).

`corp` School corporation (district) numeric identifier. **Note:** corp 2400 appears in both region 2 (12 schools) and region 3 (1 school) in the source file, so `paste(region, corp)` is the cleaner cluster key; see `corp_id`.

`school` Numeric school identifier (unique within corporation).

`class` Classroom number within school, 1 to 8. **Not** unique across schools.

`corp_id` Derived. `paste(region, corp, sep = "_")`. 61 distinct values, matching the count of school corporations reported in Lapsley et al. (2002).

`school_id` Derived. `paste(corp, school, sep = "_")`. 163 distinct values.

`class_id` Derived. `paste(corp, school, class, sep = "_")`. 586 distinct values. (The published report counted 573 classrooms; the small discrepancy reflects a different counting convention used in the manuscript.)

`gender` 1 = Female, 2 = Male. 45 `NA`.

`age` Student age in months.

`race` Indiana DOE 6-category ethno-racial code: 1 = American Indian / Alaskan, 2 = African American, 3 = Asian American, 4 = Hispanic American, 5 = Caucasian American, 6 = Multiracial.

`geread` Gates-MacGinitie reading.

`gevocab` Gates-MacGinitie vocabulary.

`gereadcm` Gates-MacGinitie reading composite.

`gelang` Gates-MacGinitie language.

`gelangmc` Gates-MacGinitie language mechanics.

`gelangcm` Gates-MacGinitie language composite.

`gemath` Gates-MacGinitie mathematics.

`gemathcp` Gates-MacGinitie mathematics computation.

`gemathcm` Gates-MacGinitie mathematics composite.

`getotal` Gates-MacGinitie total.

`ncread` NCE reading (ISTEP+).

`ncvocab` NCE vocabulary (ISTEP+).

`ncreadcm` NCE reading composite (ISTEP+).

`nclang` NCE language (ISTEP+).

`nclangmc` NCE language mechanics (ISTEP+).

`nclangcm` NCE language composite (ISTEP+).

`ncmath` NCE mathematics (ISTEP+).

`ncmathcp` NCE mathematics computation (ISTEP+).

`ncmathcm` NCE mathematics composite (ISTEP+).

`nctotal` **NCE total composite (ISTEP+)**, the criterion variable in the Lapsley et al. (2002) HLM analyses.

`aaread` AANCE reading.

`aavocab` AANCE vocabulary.

`aareadcm` AANCE reading composite.

`aalang` AANCE language.

`aalangmc` AANCE language mechanics.

`aalangcm` AANCE language composite.

`aamath` AANCE mathematics.

`aamathcp` AANCE mathematics computation.

`aamathcm` AANCE mathematics composite.

`aatotal` AANCE total.

`npanverb` NPA nonverbal reasoning.

`npamem` NPA working memory.

`npaverb` NPA verbal reasoning.

`npatotal` NPA total.

`csi` Cognitive Skills Index (student level).

`multi` Multi-age classroom indicator (1 = Yes, 2 = No).

`typmulti` Type of multi-age classroom (1 = 1st-2nd-3rd grades, 2 = 2nd-3rd, 3 = 3rd-4th, 4 = 2nd-3rd- 4th; `NA` when `multi == 2`).

`clenroll` Official class enrollment.

`classize` Project STAR style class size category: 1 = small (roughly 12-17), 2 = regular (roughly 18-22), 3 = regular-larger (roughly 23-26), 4 = large (27 or more). Boundaries follow the STAR classification (Pate-Bain and Achilles, 1986).

`ptratio` Classroom pupil to teacher ratio (IDOE formula: enrollment / [1.00 per full time teacher + 0.33 per full time aide + 0.165 per part time aide]).

`ptia` Prime Time Instructional Aide status: 1 = aide present, 2 = no aide, 3 = other assistant listed. The focal treatment indicator.

`ptstatus` Status of Prime Time aide: 1 = full time in classroom, 2 = part time in classroom; `NA` when `ptia != 1`.

`locale` NCES locale code (1 = large central city, 2 = mid-size central city, 3 = urban fringe of large city, 4 = urban fringe of mid-size city, 5 = large town, 6 = small town, 7 = rural).

`chapter1` School receives Title I (legacy "Chapter 1") money? 1 = Yes, 2 = No.

`ses` School SES, the IDOE percentage of students *not* eligible for subsidized lunch (0 to 100; higher = more affluent). The school level SES variable used in the Lapsley et al. (2002) HLM analyses.

`context` IDOE contextual rank for the school.

`calender` School calendar type (1 = traditional, 2 = year round). Typo preserved from source.

`senroll` Building (school) enrollment.

`sattend` Building attendance rate (percent).

`white1` School percent White.

`black1` School percent Black.

`hispanc1` School percent Hispanic (typo preserved).

`asian1` School percent Asian.

`aindian1` School percent American Indian.

`multi1` School percent multi-racial.

`total1` School total percent non-white.

`noteach` Number of teachers in the building (full time equivalent).

`avgage1` School average teacher age.

`avgexp1` School average teacher experience (years).

`avgsal1` School average teacher salary (dollars).

`spert` School students per teacher.

`thrdclss` Number of third grade classrooms in the building.

`thrdstud` Number of third graders who took ISTEP+ in the building.

`passla1` Building percent passing language arts.

`passmth1` Building percent passing math.

`passbth1` Building percent passing both.

`tmnnce1` Building total battery mean NCE.

`rmdnce1` Building reading *median* NCE. **Note:** the 56 rows from one building (`corp` 5740, `school` 6187) carry the source value 7603, an evident data-entry error in the Indiana DOE file (an NCE is on the 1 to 99 scale, and the same building's other median-NCE columns are in range). The value is preserved as shipped rather than silently corrected, since the true value cannot be recovered; drop or set it to `NA` before analyzing this column.

`lamdnce1` Building language arts *median* NCE.

`mmdnce1` Building mathematics *median* NCE.

`tmdnce1` Building total battery *median* NCE.

`avgcsi1` Building average Cognitive Skills Index.

`geog` Geographic category of the corporation (1 = urban, 2 = suburban, 3 = town, 4 = rural). Sampling stratifier within region.

`totepp` Corporation total expense per pupil (1997–1999, dollars).

`cenroll` Corporation enrollment (all grades).

`cattend` Corporation attendance rate (percent).

`freelnch` Corporation percent eligible for free lunch.

`lep` Corporation percent with limited English proficiency.

`speced` Corporation percent in special education.

`minority` Corporation percent minority.

`white2` Corporation total White public enrollment (raw count).

`black2` Corporation total Black public enrollment.

`hispanc2` Corporation total Hispanic public enrollment (typo preserved).

`asian2` Corporation total Asian public enrollment.

`aindian2` Corporation total American Indian public enrollment.

`multi2` Corporation total multi-racial public enrollment.

`total2` Corporation total non-white public enrollment.

`thrdadm` Corporation third grade ADM (average daily membership).

`thrdtech` Corporation third grade teachers.

`avgage2` Corporation average teacher age.

`avgexp2` Corporation average teacher experience (years).

`avgsal2` Corporation average teacher salary (dollars).

`thrdaide` Corporation third grade aides.

`passla2` Corporation percent passing language arts.

`passmth2` Corporation percent passing math.

`passbth2` Corporation percent passing both.

`tmnnce2` Corporation total battery mean NCE.

`rmdnce2` Corporation reading *median* NCE.

`lamdnce2` Corporation language arts *median* NCE.

`mmdnce2` Corporation mathematics *median* NCE.

`tmdnce2` Corporation total battery *median* NCE.

`rmediate` Corporation remediation funding per pupil (dollars). Typo preserved.

Details

Study background. Indiana's Prime Time program, phased in beginning 1984 to 1985 (Indiana statute; House Bill 1166 of 2001 codified the modern funding formula), used a state funding formula to reduce class size and pupil to teacher ratio in Kindergarten through third grade. Funds were distributed to school corporations to maintain a corporation average pupil to teacher ratio of 18:1 in K and grade 1 and 20:1 in grades 2 and 3; corporations could meet the target by hiring additional teachers or, more commonly, paraprofessional instructional assistants. Lapsley, Daytner, Kelley, and Maxwell (2002) describe Prime Time as one of the first statewide efforts to address large enrollments in the primary grades and write that, along with Tennessee's Project STAR, it was widely considered a national model, citing Pate-Bain and Achilles (1986); Project STAR was also covered by Education Week (*Research: Sizing Up Small Classes*, February 2001).

In 1999 the Indiana Department of Education funded a three year program evaluation of Prime Time. The third year of the evaluation, conducted by Daniel K. Lapsley and Katrina M. Daytner (Ball State University and Western Illinois University) with technical assistance from Ken Kelley and Scott E. Maxwell (University of Notre Dame), examined the academic performance of randomly selected Indiana third graders on the state mandated ISTEP+ standardized achievement test as a function of class size, pupil to teacher ratio, and the presence of a Prime Time instructional aide, using hierarchical linear modeling. The preliminary report and the AERA 2002 paper that summarizes the analyses are archived as ERIC document ED466679 (Lapsley, Daytner, Kelley, and Maxwell, 2002). An earlier background report from the same evaluation team, prior to the Notre Dame group joining the project, is archived as ERIC document ED455220.

Sampling. School corporations were drawn by stratified cluster sampling with two rules: 25% of corporations from each of the nine Indiana educational service regions, and at least one urban corporation per region, with the remainder proportionally allocated across geographic categories (urban, suburban, town, rural). The achieved sample was 61 corporations (78% of the target), 163 schools, 573 classrooms as counted in the manuscript (586 by the `paste(corp, school, class)` rule used here), and 10,927 students (49.6% female; 85% Caucasian, 9.2% African American, 3.2% Hispanic). 4,016 students were in classrooms with a Prime Time instructional assistant (`ptia == 1`), 6,765 in classrooms without (`ptia == 2`); the file here shows 4,021 and 6,789 plus 117 `ptia == 3` (other assistant listed), with the small differences reflecting cleaning rules applied between the manuscript count and the final SPSS file.

Instruments. Third graders sit for the ISTEP+ (Indiana Statewide Testing for Educational Progress) in September of the school term. The ISTEP+ is published by CTB/McGraw-Hill and includes language arts, reading, and mathematics assessments. Normal Curve Equivalent (NCE) composite scores for these domains and for the total are the `ncread` / `nclang` / `ncmath` / `nctotal` columns and were the criterion variables in the published HLM analyses. NCE scores have a population mean of 50 and a standard deviation of approximately 21.06, with percentiles 1, 50, and 99 mapping

to NCE scores of 1, 50, and 99. The `ge*` family is the parallel Gates-MacGinitie battery; the `aa*` family is the African American comparison NCE (AANCE); the `npa*` family is the cognitive abilities battery used as student level covariates in Finch, Bolin, and Kelley (2019).

Nested data structure. The natural hierarchy is student *within* classroom *within* school *within* corporation *within* region. The derived identifiers `corp_id`, `school_id`, and `class_id` are precomputed and safe to use as grouping variables; the bare `corp` and `class` columns are *not* unique by themselves. Class sizes range from 3 to 28 students (median 19); schools have 1 to 8 third grade classrooms (median 3) and 11 to 166 students (median 65); corporations have 15 to 756 students (median 117). Variance decomposition for the published outcome `nctotal` based on the three level random intercept null model `lmer(nctotal ~ 1 + (1 | corp_id/school_id))` gives: between-corporation variance 16.29, between-school within corporation variance 22.72, and within school residual variance 240.43, so that $ICC_{\text{corp}} \approx 0.058$, $ICC_{\text{school|corp}} \approx 0.081$, and the combined cluster $ICC_{\text{cluster}} \approx 0.140$. These nontrivial intraclass correlations are the methodological reason multilevel modeling is preferred to ordinary least squares regression for these data.

Level 1, 2, 3 model framework. For an outcome Y_{ijk} on student i in classroom j in school k, with student level predictor $X^{(1)}_{ijk}$, classroom level predictor $X^{(2)}_{jk}$, and school level predictor $X^{(3)}_{k}$, the published Lapsley et al. (2002) family of HLM models has the equations

$$Y_{ijk} = \pi_{0jk} + \pi_{1jk} X^{(1)}_{ijk} + e_{ijk} \quad \text{(Level 1)},$$

$$\pi_{0jk} = \beta_{00k} + \beta_{01k} X^{(2)}_{jk} + r_{0jk}, \quad \pi_{1jk} = \beta_{10k} + r_{1jk} \quad \text{(Level 2)},$$

$$\beta_{00k} = \gamma_{000} + \gamma_{001} X^{(3)}_{k} + u_{00k}, \quad \beta_{01k} = \gamma_{010}, \quad \beta_{10k} = \gamma_{100} \quad \text{(Level 3)},$$

with $e_{ijk} \sim N(0, \sigma^2)$, $r_{jk} \sim N(0, \mathbf{T}_\pi)$, and $u_{00k} \sim N(0, \tau_{00})$. Substituting upward, the reduced form is

$$Y_{ijk} = \gamma_{000} + \gamma_{100} X^{(1)}_{ijk} + \gamma_{010} X^{(2)}_{jk} + \gamma_{001} X^{(3)}_{k} + u_{00k} + r_{0jk} + r_{1jk} X^{(1)}_{ijk} + e_{ijk},$$

which in **lme4** translates to `lmer(Y ~ X1 + X2 + X3 + (1 + X1 | corp_id/school_id))`. The examples fit three models of this family with **lme4**: the three-level null model, a main effects model, and a model with a cross-level interaction.

Suggested benchmark uses. The data set supports demonstrations and benchmarks including:

- Two-, three-, and four-level random intercept and random slope models with **lme4**, **nlme**, or **glmmTMB**.
- Cross-level interaction modeling (e.g., race × class size, ptia × SES).
- ICC, design effect, and cluster level sample size calculations.
- Comparisons of unweighted vs. design weighted estimators for stratified cluster samples.
- Bayesian multilevel modeling and prior sensitivity (**brms**, **MCMCglmm**, **rstanarm**).
- Missing data demonstrations (the `ge*`, `nc*`, and `aa*` columns have non-trivial missingness; see `vapply(prime_time_achievement, function(x) sum(is.na(x)), integer(1))`).
- Multilevel reliability, intraclass correlation, and measurement invariance demonstrations across schools, corporations, and the categorical predictors.

Privacy and identifiability. The student level rows contain no names, addresses, or other personally identifiable information. Demographic variables are age in months, gender, and a six category race

code; all other fields are test scores or aggregated school / corporation statistics. The numeric `corp`, `school`, and `class` identifiers are the same administrative numbers used in the original Indiana Department of Education public files for the 2000 to 2001 school year; they could in principle be cross referenced to that public information to identify specific schools or corporations. No individual student can be identified from any combination of variables in this file.

Missing data convention. The Indiana DOE source used `999` as the student level missing data code and `888` as the "not applicable" code for `typmulti` and `ptstatus`. The build script converts both to `NA` (the SPSS missingness ranges already do most of the recoding on import). The retained SPSS variable label is available via `attr(prime_time_achievement$X, "label")` on every variable carried over from the SPSS file (all columns except the derived recode `classize`).

Author(s)

Ken Kelley

Source

Indiana Department of Education program evaluation of Project Prime Time, 2000 to 2001 academic year. Sample of 10,927 third grade students in 586 classrooms in 163 schools in 61 corporations in 9 educational service regions. The records are public data that the author, a member of the evaluation team, is authorized to distribute.

Lapsley, D. K., Daytner, K. M., Kelley, K., and Maxwell, S. E. (2002). *Teacher aides, class size and academic achievement: A preliminary evaluation of Indiana's Prime Time*. Paper presented at the Annual Meeting of the American Educational Research Association, New Orleans, LA, April 1-5, 2002. ERIC document ED466679.

References

Primary citation. Lapsley, D. K., Daytner, K. M., Kelley, K., and Maxwell, S. E. (2002). *Teacher aides, class size and academic achievement: A preliminary evaluation of Indiana's Prime Time*. ERIC document ED466679. https://eric.ed.gov/?id=ED466679.

Use as a multilevel modeling running example. Finch, W. H., Bolin, J. E., and Kelley, K. (2019). *Multilevel modeling using R* (2nd ed.). CRC Press. The 2nd edition (Finch, Bolin, and Kelley, 2019) is the edition that uses these data; later editions are not authored by Kelley and should not be cited for that use.

Background report from the same evaluation team. Lapsley, D. K., and Daytner, K. M. (2001). Indiana's class size reduction initiative: Teacher perspectives on training, implementation, and pedagogy. ERIC document ED455220. https://files.eric.ed.gov/fulltext/ED455220.pdf.

Indiana statutory context. Indiana General Assembly, House Bill 1166 (2001). https://archive.iga.in.gov/2001/bills

Project STAR background and Education Week coverage. Pate-Bain, H., and Achilles, C. M. (1986). Interesting developments on class size. *Phi Delta Kappan, 67*, 662–665. See also Education Week, *Research: Sizing up small classes* (February 7, 2001), https://www.edweek.org/leadership/research-sizing-up-smal

Project STAR teacher aide null result that motivated the Prime Time evaluation. Finn, J. D., Gerber, S. B., Farber, S. L., and Achilles, C. M. (2000). Teacher aides: An alternative to small classes? In M. C. Wang and J. D. Finn (Eds.), *How small classes help teachers do their best* (pp. 131–174). Temple University Center for Research in Human Development and Education.

Examples

```
data(prime_time_achievement)
dim(prime_time_achievement)

# Variable labels from the SPSS source are preserved on every column:
attr(prime_time_achievement$nctotal, "label")
attr(prime_time_achievement$ptia,    "label")

# Cluster counts, reconciled with Lapsley et al., 2002:
length(unique(prime_time_achievement$corp_id))    # 61
length(unique(prime_time_achievement$school_id))  # 163
length(unique(prime_time_achievement$class_id))   # 586

# Reconciling with the manuscript:
table(prime_time_achievement$gender, useNA = "ifany")
table(prime_time_achievement$race,   useNA = "ifany")
table(prime_time_achievement$ptia)
table(prime_time_achievement$classize)

# ----- Selecting subsets of interest -----

# Caucasian and African American only, the matched race
# supplementary analyses in Lapsley et al., 2002:
pt_wb <- subset(prime_time_achievement, race %in% c(2L, 5L))

# Drop the few "other assistant listed" cases for a clean
# aide / no-aide contrast:
pt_clean <- subset(prime_time_achievement, ptia %in% c(1L, 2L))

# Only rural corporations, coded 4 on geog, which is what the source
# SPSS file's FILTER_$ variable encoded:
pt_rural <- subset(prime_time_achievement, geog == 4L)

# Complete cases on the nctotal-on-race-and-class-size analysis:
analysis_vars <- c("nctotal", "race", "classize", "ses",
                   "corp_id", "school_id", "class_id")
pt_complete <- prime_time_achievement[
  complete.cases(prime_time_achievement[, analysis_vars]),
  analysis_vars
]

# ----- Multilevel fits -----

# Three-level null random intercept model on the full student level
# file. The variance components in the random effects block of the
# summary are the corporation, school within corporation, and residual
# variances behind the intraclass correlations reported in the Details
# section.
m_null <- lme4::lmer(nctotal ~ 1 + (1 | corp_id/school_id),
                     data = prime_time_achievement)
summary(m_null)
```

```
# Main effects of race, a student level variable, of ptia and
# classize, classroom level variables, and of ses, a school level
# variable. Compare to Lapsley et al., 2002, which fit closely related
# HLM specifications.
m_main <- lme4::lmer(
  nctotal ~ factor(race) + factor(ptia) + classize + ses +
    (1 | corp_id/school_id),
  data = prime_time_achievement)
summary(m_main)

# Cross-level interaction of ptia with ses. The published finding was
# that the aide benefit was concentrated in higher SES schools, which
# is what the interaction coefficient carries.
m_inter <- lme4::lmer(
  nctotal ~ factor(race) + factor(ptia) * ses + classize +
    (1 | corp_id/school_id),
  data = prime_time_achievement)
summary(m_inter)
```

`print_anova` *Print a Model Comparison or ANOVA Table With DMAR* p*-value Formatting*

Description

Pretty-print an ANOVA-like object (the output of `stats::anova`, `car::Anova`, `lmerTest::anova`, etc.) with *p*-values formatted at a fixed number of decimal places (default 4) and with a "< 10^(-digits_p)" floor for values too small to express. The default behavior of `print.anova` routes *p*-values through `stats::format.pval`, which applies its own digit rule (`max(1L, getOption("digits") - 2L)`) and switches to scientific notation for tiny values. `print_anova()` sidesteps that by converting the *p*-value columns to character strings up front and printing as a data frame.

Usage

```
print_anova(x, digits_p = 4L)
```

Arguments

`x`	An ANOVA-like data frame with one or more `Pr(...)` columns. Accepts `anova` objects from `stats::anova`, `car::Anova`, `car::Manova`, and `lmerTest::anova`.
`digits_p`	Integer number of decimal places for the *p*-value column(s). Default `4L`.

Details

The returned object is the input `x` invisibly, unchanged: the underlying numeric *p*-values retain full precision and can still be indexed (for example as `x[["Pr(>F)"]]`).

Any column whose name starts with `Pr(` is formatted as a *p*-value column. Other columns print at whatever `getOption("digits")` dictates (so set `options(digits = 4)` for a uniformly compact display).

Value

The input `x`, invisibly and unchanged.

Author(s)

Ken Kelley

See Also

`format_p`, `print_summary`.

Examples

```
fit <- lm(weight ~ Time + Diet, data = ChickWeight)
print_anova(anova(fit))

print_anova(car::Anova(fit, type = "III"))

# Underlying numeric p-values are untouched:
a <- anova(fit)
print_anova(a)
a[["Pr(>F)"]]   # full-precision doubles
```

`print_summary` *Print a Model Summary With DMAR* p*-value Formatting*

Description

Pretty-print a model summary (the output of `summary.lm`, `summary.glm`, or `summary` on an **lme4** or **lmerTest** fit) with *p*-values formatted at a fixed number of decimal places (default 4) and with a "< 10^(-digits_p)" floor for values too small to express. The default `print.summary.lm` / `print.summary.merMod` routes *p*-values through `stats::format.pval`, which applies its own digit rule and switches to scientific notation for tiny values. `print_summary()` sidesteps that by converting the *p*-value columns to character strings up front and printing as a data frame.

Usage

```
print_summary(fit, digits_p = 4L)
```

Arguments

`fit`	A fitted model object with a `summary` method that returns coefficients via `coef(summary(fit))`, including a `Pr(...)` column. Tested with `lm`, `glm`, `lme4::lmer`, and `lmerTest::lmer`.
`digits_p`	Integer number of decimal places for the *p*-value column(s). Default `4L`.

Details

For a linear model, the function prints the coefficient table, the residual standard error and degrees of freedom, the multiple and adjusted R^2, and the omnibus F test and its p-value. For a mixed-effects model fit through **lme4 / lmerTest**, the function prints the random-effect variances (from `lme4::VarCorr`) and the fixed-effect coefficient table.

The returned object is the model summary, invisibly and unchanged: the underlying numeric p-values retain full precision and can still be indexed (for example as `coef(summary(fit))[, "Pr(>|t|)"]`).

Value

The model summary, invisibly and unchanged.

Author(s)

Ken Kelley

See Also

`format_p`, `print_anova`.

Examples

```
fit_lm <- lm(weight ~ Time + Diet, data = ChickWeight)
print_summary(fit_lm)

fit_lmer <- lme4::lmer(weight ~ Time + (1 | Chick), data = ChickWeight)
print_summary(fit_lmer)

# Underlying numeric p-values are untouched:
sm <- summary(fit_lm)
sm$coefficients[, "Pr(>|t|)"]   # full-precision doubles
```

---

probability_of_superiority_paired

Probability of Superiority for a Paired-Samples Design

Description

Takes paired observations and returns the probability of superiority for a paired design (Grissom & Kim, 2005, 2012), the probability that the second member of a pair scores higher than the first (a tied pair counting one half), with a confidence interval and the numbers of pairs won, lost, and tied. Only the order within each pair is used. It is the paired counterpart of `vargha_delaney_A` and `cliff_delta`, which are for two independent groups.

Usage

```
probability_of_superiority_paired(x, y, conf_level = 0.95)
```

Arguments

x, y	Paired numeric vectors of equal length. x and y are interpreted as repeated measurements on the same units (e.g., pre/post, condition 1 / condition 2, sibling-1 / sibling-2). Logical vectors and ordered factors with the same levels in the same order are also accepted, since only the order within each pair is used. Pairs with a missing member are removed (see the `n_missing` attribute). An unordered factor and a character vector are refused; see *What is refused, and why*.
conf_level	Confidence level. Default `0.95`.

Details

Estimand and interval. For a pair (X, Y) drawn from the population of pairs,

$$P_S \;=\; \Pr(Y > X) + 0.5 \cdot \Pr(Y = X).$$

The sampling unit is the pair, and the n pairs are taken to be independent and identically distributed; nothing is assumed about the distribution of the scores or about the dependence within a pair. The interval is a large-sample confidence interval for P_S, formed on a transformed scale that keeps it inside $[0, 1]$. A value of 0.5 says that neither member tends to score higher.

Sample estimate. With the score of pair i

$$s_i = \mathrm{I}(y_i > x_i) + 0.5 \cdot \mathrm{I}(y_i = x_i),$$

the estimate is the mean $\hat{P}_S = \bar{s}$: the proportion of pairs won by y, plus half the proportion tied. Being the mean of n independent scores with expectation P_S, it is unbiased. Each pair is compared directly, not through the difference $y_i - x_i$, so a pair of equal infinite values is a tie.

Why a function for pairs. P_S compares the two members of one pair. Vargha and Delaney's *A* and Cliff's δ compare an observation of one group with an observation of an unrelated unit in the other, which is a different quantity when the members of a pair are correlated, and the variance formulas of `vargha_delaney_A` and `cliff_delta` assume independent groups. Paired data therefore need the estimate and the variance given here.

Variance and confidence interval. The variance of the mean of n independent scores is the variance of a score divided by n, and the function estimates it by

$$\widehat{\mathrm{Var}}(\hat{P}_S) \;=\; \frac{1}{n^2}\sum_{i=1}^{n}(s_i - \bar{s})^2,$$

which uses the divisor n in the variance of the scores (it is $(n-1)/n$ times the estimate with the divisor $n-1$). This is the row `var_ps`. The interval is formed for $d = 2P_S - 1$, which lies in $[-1, 1]$, on the scale of $\operatorname{arctanh}(d)$, as the interval of `cliff_delta` is:

$$\tanh\left\{\operatorname{arctanh}(\hat{d}) \pm z_{1-\alpha/2}\,\frac{2\sqrt{\widehat{\mathrm{Var}}(\hat{P}_S)}}{1-\hat{d}^2}\right\},$$

and the two limits d_L and d_U are returned as $(d_L + 1)/2$ and $(d_U + 1)/2$.

Scale and missing values. Only the order within each pair is used, so any increasing rescoring applied to both members gives the same result, and an ordinal response may be supplied as two ordered factors with the same levels. A pair with a missing member (`NA` or `NaN`) is removed, and the number removed is in the `n_missing` attribute. `x` and `y` must have the same length. Fewer than three complete pairs is an error, since no variance can be estimated.

What is refused, and why. The statistic uses the order of the observations and nothing else, so the function must be given objects whose order is the order meant. It refuses, with a condition of class `dmar_scores_not_ordered` (see `dmar_metadata`), an unordered factor (its codes follow the order of `levels()`, alphabetical unless set, so the levels low, mid, high are stored as 2, 3, 1); a character vector (strings sort alphabetically, `"10"` before `"9"`); an ordered factor beside anything that is not an ordered factor with the same levels in the same order (each factor numbers its own levels from 1); time differences in different units, or one beside plain numbers (each such object is stored in its own units); a `Date` beside a `POSIXct`; and a list, data frame, or matrix of several columns. Nothing is computed in any of these cases. The message says which case applies and what to supply: an ordered factor with its levels from lowest to highest, `factor(x, levels = c("low", "mid", "high"), ordered = TRUE)`, or numeric scores.

When the estimated variance is zero. Two kinds of data give a variance estimate of exactly 0, and they are the only two. *Complete separation*: every pair favors the same member, so the estimate is at the end of its range (0 or 1). *No variation*: every pair is tied, so the estimate is 0.5. In both, the two limits are reported equal to the estimate, `var_ps` is 0, the table's `interval_degenerate` attribute names the case, and a warning of class `dmar_interval_degenerate` is given whose `reason` element is `"complete_separation"` or `"no_variation"`; the condition also carries `estimate` and `n` (the number of pairs), and its `limit_feng_cliff` is `NA`, since that limit is for two independent groups. The reported limits are not a confidence interval with the stated coverage. The zero describes the sample: the variance estimate measures how much the scores differ from one pair to the next, and when every pair favors the same member they do not differ. Complete separation is common when the effect is large and the pairs are few. Without ties, n pairs all favor the same member with probability $P_S^n + (1 - P_S)^n$: 0.47 for five pairs and 0.22 for ten when $P_S = 0.86$, and 0.82 and 0.66 when $P_S = 0.96$. In those samples the reported limits, both 0 or both 1, do not contain a population value inside (0, 1).

Particular cases. Infinite values are ordered like any others, and a pair of equal infinite values is a tie. Wins together with ties, and no losses, give a positive variance and an ordinary interval (three wins and a tie give 0.875 with limits 0.502 and 0.980). The mark travels with the result: `results_sentence` writes the estimate and says that there is no confidence interval, the printed table says so beneath itself, and `tidy()` and `glance()` carry it in the column `interval_degenerate`. The randomization test for paired observations (`randomization_test_paired`) remains valid in both cases.

Reading the output. In the example below 20 of the 30 pairs have a higher `post` than `pre` score and none is tied, so $\hat{P}_S = 20/30 = 0.667$, with 95 percent limits 0.484 and 0.810.

Value

A `data.frame` (classes `dmar_probability_of_superiority_paired` and `dmar_tbl`) with columns `term` and `value` and these rows:

`probability_of_superiority` The estimate $\hat{P}_S$, the proportion of pairs with $y_i > x_i$ plus half the proportion of tied pairs.

`ci_lower`, `ci_upper` The confidence limits at `conf_level`. Both equal the estimate when the estimated variance is 0, and are then not a confidence interval.

`var_ps` The estimated variance of $\hat{P}_S$.

`wins_y_over_x` The number of complete pairs with $y_i > x_i$.

`losses_y_under_x` The number with $y_i < x_i$.

`ties` The number with $y_i = x_i$.

The table carries the attributes `conf_level`; `n_missing`, the number of pairs removed because a member was missing; and `interval_degenerate`, present only when the estimated variance is 0, with the value `"complete_separation"` or `"no_variation"`. The `tidy()` and `glance()` verbs (see `dmar_tidiers`) add a logical column `interval_degenerate`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Grissom, R. J., & Kim, J. J. (2005). *Effect sizes for research: A broad practical approach*. Lawrence Erlbaum.

Grissom, R. J., & Kim, J. J. (2012). *Effect sizes for research: Univariate and multivariate applications* (2nd ed.). Routledge.

Vargha, A., & Delaney, H. D. (2000). A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics, 25*(2), 101–132. doi:10.3102/10769986025002101

See Also

`cliff_delta`, `vargha_delaney_A`, `cles`, `randomization_test_paired`, `proportion_of_superiority`

Other effect size estimates: `cles()`, `cliff_delta()`, `correction_for_attenuation()`, `eta_squared()`, `eta_squared_generalized()`, `expected_partial_r()`, `expected_r()`, `expected_smd()`, `nnt_from_smd()`, `omega_squared()`, `proportion_of_superiority()`, `responder_analysis()`, `smd_trimmed()`

Examples

```
# 1. Paired pre/post data:
set.seed(113)
pre  <- rnorm(30, mean = 100, sd = 15)
post <- pre + rnorm(30, mean =  5, sd = 10)
probability_of_superiority_paired(x = pre, y = post)

# 2. Every pair favors the same member: the estimated variance is 0,
#    the two limits equal the estimate, and a warning says that they
#    are not a confidence interval.
all_up <- probability_of_superiority_paired(x = c(1, 2, 3, 4),
                                            y = c(4, 5, 6, 7))
all_up
results_sentence(all_up)
```

`procrustes_phi` *Tucker's Congruence Coefficient ϕ (Factor Similarity)*

Description

Computes Tucker's (1951) congruence coefficient ϕ, a measure of similarity between two factor-loading patterns (typically the standardized loadings of the same factor estimated on two different samples or with different methods), together with a permutation-based *p*-value testing the null hypothesis of unrelated loading patterns. ϕ is the standard tool for factor-replication studies (Lorenzo-Seva & ten Berge, 2006).

Usage

```
procrustes_phi(loadings_1, loadings_2, n_resamples = 10000L, seed = NULL)
```

Arguments

`loadings_1`, `loadings_2`	Numeric vectors of factor loadings on the same indicator set, of equal length. Either standardized or raw loadings work; the coefficient is scale-invariant.
`n_resamples`	A single whole number of at least 0: the number of permutations for the significance test. Default `10000`. Set to 0 to skip the test.
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Definition. For two vectors of loadings $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2$ on a shared set of p indicators, Tucker's congruence coefficient is

$$\phi(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \;=\; \frac{\sum_{i=1}^{p} \lambda_{1i}\lambda_{2i}}{\sqrt{\sum_{i=1}^{p} \lambda_{1i}^2 \cdot \sum_{i=1}^{p} \lambda_{2i}^2}}.$$

ϕ is the cosine of the angle between the two loading vectors and ranges over $[-1, 1]$; values near ± 1 indicate high (anti-)congruence, values near 0 indicate orthogonality.

Permutation test. Under the null hypothesis that the two loading patterns are unrelated, in the sense that the pairing of the items' loadings across the two vectors is arbitrary, permuting one vector and recomputing ϕ gives a permutation distribution against which the observed ϕ can be evaluated. The test takes the two sets of loadings as given: it does not reflect the sampling variability of loadings estimated from data, and the function reports no standard error or interval for ϕ. The two-sided *p*-value is $(r + 1)/(m + 1)$, where *r* counts the permuted $|\phi|$ values at least as large as the observed $|\phi|$ and *m* is `n_resamples`. A permutation whose coefficient equals the observed one counts as at least as extreme, as when it exchanges two items with the same loading; the comparison allows for the rounding of the arithmetic. Adding one to each part counts the observed arrangement, which is itself a legitimate permutation; without it a *p*-value of exactly zero could be reported, a value a sampled permutation test cannot support (Phipson & Smyth, 2010). The smallest reportable *p*-value is therefore $1/(m + 1)$.

Interpretation. Benchmark values for ϕ have been proposed in the literature (Lorenzo-Seva & ten Berge, 2006), but context always matters; this package reports the coefficient and the permutation p-value and leaves interpretation to the context of the application.

Value

A `data.frame` (class `dmar_tbl`) in `term` / `value` layout with the row `tucker_phi`, the point estimate of ϕ. When `n_resamples` > `0` the table also carries `p_value_perm`, the two-sided permutation p-value, and `n_resamples`, the number of permutations requested.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lorenzo-Seva, U., & ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology, 2*(2), 57–64. doi:10.1027/16142241.2.2.57

Phipson, B., & Smyth, G. K. (2010). Permutation p-values should never be zero: Calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology, 9*(1), Article 39. doi:10.2202/15446115.1585

Tucker, L. R. (1951). *A method for synthesis of factor analysis studies* (Personnel Research Section Report No. 984). Department of the Army.

See Also

`factor_congruence` for the coefficients between the factors of two whole solutions, matched one to one and reflected, `cor`, `reliability_H`

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
set.seed(113)
# 1. Two highly similar loading patterns:
l1 <- c(0.72, 0.65, 0.81, 0.55, 0.69)
l2 <- c(0.70, 0.62, 0.83, 0.58, 0.66)
procrustes_phi(l1, l2)

# 2. Loadings on different factors should show low congruence:
l3 <- c(0.10, 0.05, 0.20, 0.85, 0.78)
procrustes_phi(l1, l3, n_resamples = 5000)
```

proportion_of_superiority *Proportion of Superiority (Sometimes Called Cohen's U_3)*

Description

Computes the proportion of the treatment-group population that exceeds the *control-group mean* under bivariate normality with equal variances. This quantity is sometimes called Cohen's U_3 (Cohen, 1988). Under those assumptions it equals $\Phi(\delta)$, where δ is the population standardized mean difference. When sample sizes are supplied, the CI on the proportion of superiority is constructed by transforming the noncentral *t* CI on Cohen's *d* via Φ, which is monotone and therefore preserves coverage exactly.

Usage

```
proportion_of_superiority(
  smd,
  n_1 = NULL,
  n_2 = NULL,
  conf_level = 0.95,
  smd_ci_lower = NULL,
  smd_ci_upper = NULL
)
```

Arguments

`smd`	Sample standardized mean difference (Cohen's *d*). Numeric scalar.
`n_1`, `n_2`	Group sample sizes (required if a CI is wanted).
`conf_level`	Confidence level for the CI. Default `0.95`.
`smd_ci_lower`, `smd_ci_upper`	Optional pre-computed CI limits on *d*; when supplied directly, the function skips the noncentral *t* step and just transforms these limits.

Details

The proportion of superiority is one of three "U" indices Cohen (1988) defined; the other two (U_1, the proportion of non-overlap, and U_2, the proportion of either population that exceeds the same percentile in the other) can be derived from it directly: with $U_3 = \Phi(\delta)$ for the proportion of superiority, Cohen's $U_2 = \Phi(\delta/2)$ and $U_1 = (2 \cdot U_2 - 1)/U_2$ (Cohen, 1988, Table 2.2.1).

Why this rather than `cles`. The proportion of superiority answers the question "what fraction of the treatment population exceeds the *control-group mean*," whereas `cles` answers "what fraction of randomly drawn pairs favor the treatment over the control." Both are unitless probability-scale summaries of a Cohen's-*d* difference, but the proportion of superiority is marginal while CLES is paired. Specifically, $\Phi(d)$ versus $\Phi(d/\sqrt{2})$; for $d = 0.5$, the proportion of superiority is 0.69 and CLES is 0.64.

CI construction. Because $\Phi(\cdot)$ is monotone, the CI on the proportion of superiority is just $[\Phi(d_L),\ \Phi(d_U)]$ where $[d_L,\ d_U]$ is the noncentral *t* CI on *d* from `ci_smd`.

Value

A `data.frame` with rows for *d* (`smd`), the proportion of superiority (`proportion_of_superiority`), and (when a CI is constructable) the confidence limits on *d* (`smd_ci_lower`, `smd_ci_upper`) and on the proportion of superiority (`ci_lower`, `ci_upper`). With an interval the table carries the attributes `conf_level`; `ci_estimate`, which names `proportion_of_superiority` as the row that `ci_lower` and `ci_upper` belong to, so that `results_sentence` reports the proportion of superiority with them; and `ci_source`, which names `smd` as the estimate whose interval that interval is computed from, so that `tidy()` gives the confidence level on the rows of the proportion of superiority and its limits alone.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum. (See Section 2.2 for the U_1, U_2, and U_3 indices.)

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. Academic Press.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. (The noncentral *t* interval on the standardized mean difference that is transformed here.) doi:10.18637/jss.v020.i08

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

`cles`, `nnt_from_smd`, `smd`, `ci_smd`, `probability_of_superiority_paired`

Other effect size estimates: `cles()`, `cliff_delta()`, `correction_for_attenuation()`, `eta_squared()`, `eta_squared_generalized()`, `expected_partial_r()`, `expected_r()`, `expected_smd()`, `nnt_from_smd()`, `omega_squared()`, `probability_of_superiority_paired()`, `responder_analysis()`, `smd_trimmed()`

Examples

```
# 1. Proportion of superiority at three reference d values:
proportion_of_superiority(smd = 0.2)
proportion_of_superiority(smd = 0.5)
proportion_of_superiority(smd = 0.8)

# 2. With a noncentral t CI from sample sizes:
proportion_of_superiority(smd = 0.5, n_1 = 50, n_2 = 50, conf_level = 0.95)
```

pygmalion *Pygmalion in the Classroom Teacher-Expectancy Data*

Description

The teacher-expectancy data from Rosenthal and Jacobson's (1968) *Pygmalion in the Classroom*, the study that introduced the "Pygmalion effect": the hypothesis that a teacher's expectations can become a self-fulfilling prophecy for a pupil's intellectual growth. Intelligence-test scores were obtained for $N = 310$ elementary school children in grades 1 through 6, of whom $n = 64$ were randomly designated to their teachers as likely "intellectual bloomers" while the remaining $n = 246$ served as controls. The data set is a classic benchmark for the analysis of covariance (ANCOVA) and, in particular, for ANCOVA with *heterogeneity of regression*: it is the running example for that topic in Maxwell, Delaney, and Kelley, *Designing Experiments and Analyzing Data: A Model Comparison Perspective* (Routledge), where it appears as a Chapter 9 example (and as a Chapter 3 exercise).

Usage

```
pygmalion
```

Format

A data frame with 310 observations on 6 variables.

`grade` Grade in school at the start of the study, an integer from 1 to 6.

`treatment` Factor with levels `Control` (reference, $n = 246$) and `Bloomer` ($n = 64$). The `Bloomer` children were a randomly selected ~20% of each classroom whose teachers were told, on the basis of a fictitious test purportedly predicting intellectual blooming, that they were likely to show unusual gains during the year; the `Control` children were not singled out. In the AMCP source this variable is coded `1` = Bloomer, `0` = Control.

`iq_pre` Pretest total IQ, measured before the expectancy manipulation. The covariate in the analysis of covariance.

`iq_4` Total IQ at an intermediate follow-up assessment.

`iq_8` Total IQ at the end-of-study follow-up assessment. This is the dependent variable in the book's Chapter 9 analysis of covariance.

`iq_gain` Total IQ change from pretest to the end-of-study assessment, equal to `iq_8 - iq_pre`.

Details

The study. Robert Rosenthal (Harvard University) and Lenore Jacobson (principal of an elementary school in South San Francisco referred to as "Oak School") set out to test experimentally whether teacher expectations influence pupil achievement. At the start of the school year all children were given a standardized test of general ability, described to teachers as the "Harvard Test of Inflected Acquisition," a test said to identify children poised for an intellectual growth spurt. In reality the instrument was Flanagan's Tests of General Ability (TOGA) and the children identified as likely

"bloomers" were chosen *at random*, about one in five per classroom. The only experimental manipulation was the expectation planted in the teachers' minds. Children were re-tested over the following year(s), and the question was whether the randomly labeled bloomers would out-gain their controls in measured IQ. Rosenthal and Jacobson reported that they did, most strongly in the earliest grades, and interpreted the difference as evidence that teacher expectations operate as a self-fulfilling prophecy. The study drew critiques; Thorndike (1968), for example, questioned the reliability of the TOGA at the extremes of the score range for the youngest children, which is itself part of why the data are instructive for teaching careful analysis.

Why it is a benchmark for heterogeneity of regression. A standard ANCOVA adjusts the group comparison for the pretest covariate under the assumption that the regression of the outcome on the covariate has the *same* slope in every group (homogeneity of regression). In these data that assumption is questionable: the within-group regression of `iq_8` on `iq_pre` is steeper for the bloomers than for the controls, so the estimated treatment effect depends on the covariate value at which it is evaluated. The data can therefore be used to teach (a) testing the homogeneity-of-regression assumption, (b) interpreting a treatment-by-covariate interaction, and (c) estimating the treatment effect, and its sampling variance, *at chosen covariate values* rather than only at the grand mean.

Reproducible quantities. Fitting the separate-slopes model `lm(iq_8 ~ iq_pre * treatment)` gives a within-group slope of 0.77799 for the controls and 0.96894 for the bloomers (reported as 0.96895 in `MBESS::var.ete`, a fifth-decimal rounding difference). The pooled within-group residual variance is $\hat{\sigma}^2 = 175.3251$ on 306 degrees of freedom, and the sample variance of the covariate is 348.91. These are exactly the inputs used in the worked example for the variance of the estimated treatment effect at selected covariate values under heterogeneity of regression (Li, McLouth, and Delaney; see `MBESS::var.ete`).

Relationship to the AMCP package. The same numeric data ship with the book's data companion, the **AMCP** package, as `chapter_9_exercise_15` and `chapter_9_extension_exercise_3` (with `IQGain`) and `chapter_3_exercise_22` (without it). The version here renames the columns to DMAR's descriptive snake_case style and labels the experimental condition as a factor; no measured value has been altered. See `data-raw/pygmalion.R` for the construction script and its verification checks.

Author(s)

Ken Kelley

Source

Rosenthal, R., & Jacobson, L. (1968). *Pygmalion in the classroom: Teacher expectation and pupils' intellectual development*. Holt, Rinehart and Winston.

Distributed with the **AMCP** data companion to Maxwell, Delaney, and Kelley (see References) as `chapter_9_exercise_15`.

References

Rosenthal, R., & Jacobson, L. (1968). *Pygmalion in the classroom: Teacher expectation and pupils' intellectual development*. Holt, Rinehart and Winston.

Rosenthal, R., & Jacobson, L. (1968). Pygmalion in the classroom. *The Urban Review, 3*(1), 16–20. doi:10.1007/BF02322211

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (Heterogeneity-of-regression ANCOVA example, Chapter 9.)

Thorndike, R. L. (1968). Review of *Pygmalion in the Classroom*. *American Educational Research Journal, 5*(4), 708–711.

See Also

ancova for an ANCOVA that returns adjusted means, effect size confidence intervals, and a homogeneity-of-regression test.

Examples

```
data(pygmalion)
str(pygmalion)

# Design: pupils per condition within each grade.
table(pygmalion$treatment, pygmalion$grade)

# ---- Heterogeneity-of-regression ANCOVA (book Chapter 9) ----
# Separate IQ8-on-IQpre slopes for the two conditions.
fit_het <- lm(iq_8 ~ iq_pre * treatment, data = pygmalion)
coef(fit_het)
# Control slope = 0.778; the interaction (0.191) gives the
# steeper Bloomer slope of 0.969.

# The treatment-by-covariate interaction is the
# heterogeneity-of-regression test (1 df): compare the additive
# ANCOVA model to the separate-slopes model.
fit_add <- lm(iq_8 ~ iq_pre + treatment, data = pygmalion)
anova(fit_add, fit_het)

# Pooled within-group residual variance (175.3251) and the
# covariate variance (348.91), as used by MBESS::var.ete.
sum(residuals(fit_het)^2) / fit_het$df.residual
var(pygmalion$iq_pre)

# ---- DMAR's ANCOVA, with the homogeneity-of-regression check ----
ancova(pygmalion, outcome = "iq_8", treatment = "treatment",
       covariates = "iq_pre")
```

R2_mixed_effects *Marginal and Conditional R^2 for a Mixed-Effects Model*

Description

Computes the Nakagawa and Schielzeth (2013) marginal and conditional coefficients of determination from a fitted mixed-effects model, returned in tidy long form. The marginal R^2 is the proportion of total variance explained by the fixed effects alone; the conditional R^2 is the proportion explained

by the fixed and random effects together. The two quantities are the mixed-effects companions to the intraclass correlation returned by `icc_lmer`: where the ICC isolates the share of variance attributable to a single grouping factor, the marginal and conditional R^2 summarize how much of the outcome variance the fixed and random parts of the model account for.

Usage

```
R2_mixed_effects(
  model,
  conf_level = 0.95,
  ci_method = c("none", "boot"),
  B = 1000,
  seed = NULL
)
```

Arguments

`model`	A fitted mixed-effects model. A `lmer` fit (class `lmerMod`) is the primary target; an `lme` fit is also accepted.
`conf_level`	Confidence level. Default `0.95`. Used only when a bootstrap interval is requested (see `ci_method`).
`ci_method`	Interval method for the two R^2 values. The default `"none"` returns point estimates only. `"boot"` adds a parametric bootstrap percentile interval (`lmerMod` models only, via `bootMer`); see *Details*.
`B`	Number of bootstrap replications when `ci_method = "boot"`, a single whole number. Default `1000`, because every replication simulates a response and refits the model. The fewest accepted is 400 at the default `conf_level` (200 at 0.90, and 2000 at 0.99, which the default does not meet); a smaller `B` is refused before anything is computed (class `dmar_count_invalid`). See *Bootstrap replications* under *Details* and `dmar_bootstrap`.
`seed`	Optional integer seed for the bootstrap. Default `NULL`, under which the bootstrap draws from, and advances, the caller's current random number stream. When supplied, the seed is set inside the function and the caller's RNG state is restored on exit.

Details

Variance decomposition. Writing σ_f^2 for the variance of the fixed-effect linear predictor, σ_r^2 for the variance attributable to the random effects, and σ_ε^2 for the residual variance,

$$R^2_{\text{marginal}} = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_r^2 + \sigma_\varepsilon^2}, \qquad R^2_{\text{conditional}} = \frac{\sigma_f^2 + \sigma_r^2}{\sigma_f^2 + \sigma_r^2 + \sigma_\varepsilon^2}.$$

The fixed-effect variance is $\sigma_f^2 = \text{var}(\mathbf{X}\boldsymbol{\beta})$, the variance of the fitted fixed-effect linear predictor across the observations. The residual variance is $\sigma_\varepsilon^2 = \text{sigma}(\text{model})^2$.

Random-effect variance. For a random-intercept model the random-effect variance is the sum of the variance components read off `VarCorr`. For a model with random slopes the variance contributed by a random-effects term depends on the values of the associated covariates, so the sum

of the diagonal variance components is not correct on its own. This function uses the Johnson (2014) extension: for each random-effects term with design matrix $\mathbf{Z}$ and estimated covariance matrix $\boldsymbol{\Sigma}$, its contribution is the mean over the observations of the quadratic form $\mathbf{z}_i^\top \boldsymbol{\Sigma}\, \mathbf{z}_i$, that is, $\frac{1}{n}\,\mathrm{tr}(\mathbf{Z}\boldsymbol{\Sigma}\mathbf{Z}^\top)$, and σ_r^2 is the sum of these contributions across all random-effects terms. For a random-intercept term this reduces to the intercept variance component, so the two paths agree.

Scope. The decomposition here is the one appropriate for a Gaussian (identity-link) linear mixed model, which is what `lmer` and `lme` fit. Generalized linear mixed models introduce a distribution-specific variance term and are not handled by this function.

The bootstrap interval. The default `ci_method = "none"` reports the two point estimates alone, so the bootstrap is what to ask for when the marginal and conditional R^2 are to be reported with an interval and the refits it costs are affordable. With `ci_method = "boot"` the interval comes from a parametric bootstrap (`bootMer`): each of the `B` replicates (1000 by default) simulates a new response vector from the fitted model, refits the model, and recomputes the two R^2 values. The unit of resampling is therefore a whole simulated data set drawn from the estimated model, not a resampled set of cases. Only the percentile interval is offered: the limits are the empirical quantiles of the `B` bootstrap values (Efron & Tibshirani, 1993); there is no BCa or bootstrap standard error variant. Raising `B` tightens the Monte Carlo error of the reported limits. Bootstrap results vary from run to run; supply `seed` for reproducibility.

Bootstrap replications. The limits of each interval are quantiles of the `B` replicated values, so the interval is run only when `B` puts at least 10 replicates beyond each limit (400 at the 95 percent level). A replication in which either R^2 is undefined (a refit that fails, or one whose variance components cannot be computed) is dropped and counted, never replaced, and the interval is computed from the rest. The result records `B` (requested), `B_used` (the number the interval rests on), `n_dropped`, and `B_required` (the fewest replications a request may ask for; that count is held to `B`, not to `B_used`) as attributes, with `bootstrap_status` (`"complete"`, `"dropped"`, `"exploratory"`, `"insufficient_count"`, or `"insufficient_share"`). One warning of class `dmar_bootstrap_replications_dropped` states the counts when any replication was dropped, with a caution when the dropped replications are at least the 2.5 percent of `B` that lies beyond one limit at the 95 percent level, because they could then decide by themselves where that limit falls. When more than half were dropped, the four limit rows are `NA`, the estimate is returned, and the warning has class `dmar_bootstrap_insufficient`. Such a replicate is dropped for both quantities, so the two intervals rest on the same replicates. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory` and the result is not one to report. See `dmar_bootstrap` for the rule and its reasons.

Value

A `data.frame` with rows `"R2_marginal"` and `"R2_conditional"` in the `value` column. When `ci_method = "boot"`, lower- and upper-limit rows for each quantity are appended (`R2_marginal_ci_lower`, `R2_marginal_ci_upper`, `R2_conditional_ci_lower`, `R2_conditional_ci_upper`), the confidence level is carried on the object, and the attributes `B`, `B_used`, `n_dropped`, `B_required`, and `bootstrap_status` record the replications (see *Bootstrap replications* under Details).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Johnson, P. C. D. (2014). Extension of Nakagawa & Schielzeth's R^2_{GLMM} to random slopes models. *Methods in Ecology and Evolution, 5*(9), 944–946. doi:10.1111/2041210X.12225

Nakagawa, S., & Schielzeth, H. (2013). A general and simple method for obtaining R^2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution, 4*(2), 133–142. doi:10.1111/j.2041210x.2012.00261.x

See Also

icc_lmer, ss_power_mixed_effects, lmer, VarCorr

Other agreement and measurement: content_validity_index(), gwet_ac(), icc_lmer(), krippendorff_alpha(), limits_of_agreement(), lin_ccc(), variance_components_mls()

Other mixed models: R2_mixed_effects_decomposition(), icc_lmer(), manova_split_plot(), mixed_anova(), ss_aipe_mixed_effects(), ss_aipe_mixed_effects_sensitivity(), ss_power_mixed_effects(), ss_power_split_plot_anova()

Examples

```
fit <- lme4::lmer(Reaction ~ Days + (Days | Subject),
                  data = lme4::sleepstudy)

# Marginal R2 is the proportion of variance the fixed effects account for;
# conditional R2 adds what the random effects account for, so the gap
# between the two is what the subject-level terms buy.
R2_mixed_effects(fit)

# A parametric bootstrap percentile interval for both quantities. Each
# replication refits the model. B = 20 keeps the example quick and is
# below the 400 a reported interval needs; the option lets it run and
# the warning says so. A reported interval deserves the default
# B = 1000. The seed makes the limits reproducible and leaves the
# caller's generator state as it was.
old <- options(dmar.bootstrap_exploratory = TRUE)
R2_mixed_effects(fit, ci_method = "boot", B = 20, seed = 113)
options(old)
```

`R2_mixed_effects_decomposition`

R-Squared Measures for Mixed-Effects Models

Description

Computes the Rights and Sterba (2019) integrative framework of R-squared measures for a fitted two-level linear mixed-effects (multilevel) model. The model implied outcome variance is fully decomposed into five sources: variance due to individual-level (i.e., Level 1) predictors via fixed slopes (f_1), cluster-level (i.e., Level 2) predictors via fixed slopes (f_2), predictors via random slope (co)variation (v), cluster-specific outcome means via random intercept variation (m), and individual-level (i.e., Level 1) residuals (σ^2). Proportions of the total, within-cluster, and between-cluster outcome variance attributable to combinations of these sources give the family of R-squared measures.

Usage

```
R2_mixed_effects_decomposition(model)
```

Arguments

`model`	A fitted two-level model of class `merMod` (from **lme4**, e.g.\ `lmer`) or `lme` (from **nlme**). The model must use numeric predictors (only the cluster variable may be a factor) and must not contain `I()` terms; create any transformed predictors as their own columns first.

Details

The companion `R2_mixed_effects` returns the Nakagawa and Schielzeth marginal and conditional R-squared; this function contains those two as special cases (`total_f` and `total_fvm`) within the fuller source decomposition.

The measure superscripts index the variance sources in the numerator and the subscripts index the outcome variance in the denominator: `total_*` use the total outcome variance, `within_*` the within-cluster variance ($f_1 + v + \sigma^2$), and `between_*` the between-cluster variance ($f_2 + m$). `total_fvm` is the omnibus measure (all explained sources over the total variance) and, for a random-intercept model, coincides with the Nakagawa and Schielzeth conditional R-squared computed by `R2_mixed_effects`; `total_f` coincides with their marginal R-squared. See Rights and Sterba (2019, Table 1) for the definitions.

The measures were derived under the assumption that the fitted model uses cluster-mean-centering of the individual-level predictors (with the cluster means entered as cluster-level predictors). When that centering is not detected, only the total-variance measures are returned, matching the reference implementation.

Fitting the model requires **lme4** (for a `merMod` fit) or **nlme** (for an `lme` fit) to be installed.

Value

A `data.frame` (`dmar_tbl`) with columns `term` and `value`. When the individual-level predictors are cluster-mean-centered, the full set of 12 measures is returned, named `total_f1`, `total_f2`, `total_v`, `total_m`, `total_f`, `total_fv`, `total_fvm`, `within_f1`, `within_v`, `within_fv`, `between_f2`, and `between_m`; otherwise the five total-variance measures `total_f`, `total_v`, `total_m`, `total_fv`, and `total_fvm` are returned (the within/between split requires cluster-mean-centering). The returned object carries the source-by-target variance decomposition in `attr(x, "decomposition")`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Rights, J. D., & Sterba, S. K. (2019). Quantifying explained variance in multilevel models: An integrative framework for defining R-squared measures. *Psychological Methods, 24*(3), 309–338. doi:10.1037/met0000184

Nakagawa, S., & Schielzeth, H. (2013). A general and simple method for obtaining R^2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution, 4*(2), 133–142. doi:10.1111/j.2041210x.2012.00261.x

See Also

R2_mixed_effects, icc_lmer

Other mixed models: R2_mixed_effects(), icc_lmer(), manova_split_plot(), mixed_anova(), ss_aipe_mixed_effects(), ss_aipe_mixed_effects_sensitivity(), ss_power_mixed_effects(), ss_power_split_plot_anova()

Examples

```
fit <- lme4::lmer(Reaction ~ Days + (Days | Subject),
                  data = lme4::sleepstudy)
R2_mixed_effects_decomposition(fit)
```

---

randomization_test *Randomization (Permutation) Test for Two Independent Groups*

Description

Compares two independent groups by referring the observed statistic to the distribution of that same statistic over reassignments of the observed scores to the two groups. That reference distribution, not a normal or a *t* distribution, supplies the *p*-value, so the test needs no assumption about the shape of the population. Alongside the test, the function reports the effect sizes that answer the question the test only screens: the mean difference and its randomization-based interval, the standardized mean difference with a noncentral *t* interval, the common language effect size, and Cliff's delta.

Usage

```
randomization_test(
  x = NULL,
  group = NULL,
  data = NULL,
  group_1 = NULL,
  group_2 = NULL,
  statistic = c("mean", "t"),
```

```
  alternative = c("two_sided", "less", "greater"),
  exact = NULL,
  n_resamples = 10000L,
  seed = NULL,
  conf_level = 0.95,
  shift_ci = TRUE
)
```

Arguments

x	A formula of the form y ~ group, or a numeric response vector to be paired with group. Leave NULL when the two samples are supplied through group_1 and group_2.
group	A grouping variable the same length as x, with exactly two levels after unused levels are dropped. Ignored when x is a formula.
data	An optional data.frame in which to find the variables named in the formula.
group_1, group_2	The two samples supplied directly as numeric vectors, an alternative to the formula and response-plus-grouping interfaces. Lengths need not be equal.
statistic	One of "mean" (default) or "t". "mean" uses the difference in group means. "t" uses the studentized (Welch) statistic, which divides that difference by its separate-variances standard error and is the better choice when the groups differ in size and may differ in variance (see Details). The statistic is refused when both observed groups are constant at different values (exactly, or to within the tolerance of the comparison), where its standard error is 0 (see *When the studentized statistic is undefined*); a reassignment with that property in the reference set of other data is ordered as Inf or -Inf, and when every score is equal the statistic is 0.
alternative	One of "two_sided" (default; the base-R spelling "two.sided" is accepted as an alias), "less", or "greater". The direction refers to the first group minus the second, the same orientation t.test uses.
exact	Logical. If NULL (default), the test enumerates every reassignment when choose(N, n_1) is at most 50,000 and samples reassignments otherwise. If TRUE, enumeration is forced (refused above 1,000,000 reassignments). If FALSE, Monte Carlo is forced.
n_resamples	A single whole number of at least 1: the number of randomly drawn reassignments when enumeration is not used. Default 10000L.
seed	Optional integer seed for the Monte Carlo branch. Default NULL, under which the reassignments are drawn from, and advance, the current random number stream; supply an integer for reproducibility. When a seed is supplied the RNG state in place before the call is restored on exit.
conf_level	Confidence level for every interval reported, the inverted randomization interval included. Default 0.95.
shift_ci	Logical. Compute the randomization-based interval for the shift by inverting the test? Default TRUE. Setting it to FALSE reports the two endpoints as NA and skips the inversion, which is the expensive part of the call.

Details

What the randomization distribution is. Suppose *N* participants were randomly assigned, n_1 to one condition and n_2 to the other. Under the null hypothesis that the condition a participant received made no difference to that participant's score, each score would have been the same number no matter which group the participant landed in. The assignment actually used was one draw from the $\binom{N}{n_1}$ assignments the randomization could equally well have produced, so every one of those assignments was equally likely, and each of them yields a value of the test statistic. Those values are the randomization distribution. The *p*-value is the proportion of them at least as extreme as the value the experiment actually produced.

Why there is no normality assumption. Nothing in that argument mentions a population, a normal curve, or a sampling model. The probability comes from the coin flips the experimenter performed, which are known exactly because the experimenter performed them. This is the inferential logic Fisher (1935) used to introduce experimental design, and it is where Chapter 1 of Maxwell, Delaney, and Kelley (2027) starts, for the same reason: the validity of the test rests on the randomization rather than on assumptions a data analyst cannot check.

What the test does and does not license. A small *p*-value says the observed separation between the groups would rarely arise from reassignment alone, which is evidence that the assignment mattered. It does not say how much it mattered, and with a large *N* an uninteresting difference will produce a small *p*-value. It also does not, by itself, license generalization beyond the participants at hand: randomization licenses a causal claim about these units, while generalization to a population is a separate argument that rests on how the units were recruited. That is why this function reports effect sizes with intervals rather than a *p*-value alone.

Why the studentized statistic. With $n_1 = n_2$ the separate-variances standard error equals the pooled one, and the pooled *t* is an increasing function of the mean difference over the reassignments, so the two statistics give the same *p*-value in every balanced design. They differ only when the groups are unequal in size. Reassigning scores between groups of unequal size mixes the two variances in proportions that the observed assignment does not have, so the reference distribution for the raw mean difference is built under a null that is false in a second way, and the test's actual Type I error rate drifts away from the nominal level. The studentized statistic rescales each reassignment by its own separate-variances standard error, which removes most of that drift and remains asymptotically valid under heteroscedasticity (Janssen, 1997; Neuhaus, 1993). Use `statistic = "t"` when the group sizes differ and unequal variances are plausible.

When the studentized statistic is undefined. The studentized statistic is a ratio, the difference in means over its standard error, and it has a value only when that standard error is positive. If each group is constant and the two constants differ (exactly, or to within the tolerance of the comparison), the standard error of the observed data is 0 and `statistic = "t"` is refused with a condition of class `dmar_studentized_statistic_undefined`. The observed data are then at the boundary of the reference set: among the reassignments only the observed one and, with equal group sizes, the one that exchanges the groups have a zero standard error. The refusal concerns the statistic and not the evidence. `statistic = "mean"` is defined for every arrangement and gives the smallest *p*-value the design allows in the direction of the data, $1/\binom{N}{n_1}$ one-sided, and two-sided $2/\binom{N}{n_1}$ with equal group sizes and $1/\binom{N}{n_1}$ otherwise; the message states the value, and the condition carries it as `p_value_mean`. The refusal is made before any reassignment is enumerated or drawn, so a refused Monte Carlo call leaves the random number stream where it was. With `statistic = "mean"` the same data return the table, with the warning of class `dmar_smd_undefined` described under Value.

When the observed statistic is defined, some arrangements in the reference set can still have a zero

standard error: with a two-valued outcome, the reassignment that puts every high score in one group. Those arrangements are kept and ordered as the limit of the statistic, `Inf` or `-Inf` with the sign of their mean difference, so they count as the most extreme arrangements, which is where the mean difference also places them. Refusing them would refuse ordinary binary and coarsely measured data. For `group_1 = c(1, 1, 1, 0)` against `group_2 = c(0, 0, 0, 1)` the reference distribution of the studentized statistic holds one `-Inf` and one `Inf` among its 70 values, and the *p*-value, 34/70, is the one the difference in means gives. When every score is equal, every arrangement is the same data set; the statistic is reported as 0 and the *p*-value is 1.

Ties. A reassignment whose statistic equals the observed one counts as at least as extreme. Two statistics that are equal in exact arithmetic can differ in their last binary digits as computed, so the comparison allows for the rounding of the arithmetic, with a bound proportional to the size of the scores. Where the bound cannot separate two statistics they are counted as tied, so the *p*-value can be above the exact one and is not below it. For the difference in means that happens only when two mean differences agree to within about 1e-14 of the largest absolute score, as when scores agree to about fourteen significant digits or one score is about 1e14 times the rest. For the studentized statistic it can also happen when one score is about ten million times the spread of the rest, or when every score lies within about a millionth of one of two values.

Exact or Monte Carlo. When `choose(N, n_1)` is at most 50,000 every reassignment is enumerated and the *p*-value is exact: it is a count divided by a known total, with no approximation anywhere. Above that threshold `n_resamples` reassignments are drawn at random and the *p*-value is $(r + 1)/(m + 1)$, where *r* counts the sampled reassignments at least as extreme as the observed one and *m* is `n_resamples`. Adding one to each part counts the observed assignment, which is itself a legitimate reassignment; without it a *p*-value of exactly zero could be reported for a hypothesis the data cannot rule out, and the test would be anticonservative (Phipson & Smyth, 2010). The reported `p_value_se` is $\sqrt{\hat{p}(1-\hat{p})/m}$, the standard error of the resampling itself. It describes how much the *p*-value would move if the reassignments were drawn again, not how much it would move in a new experiment. Raising `n_resamples` shrinks it at the usual $1/\sqrt{m}$ rate.

The randomization interval, and how it differs from the normal theory one. Suppose the treatment adds a constant δ to every score it touches. Subtracting δ from each first-group score should then leave scores that are exchangeable across groups, so the randomization test applied to the subtracted data is a test of $H_0 : \text{shift} = \delta$. The set of δ for which that test does not reject at level $1-$ `conf_level` is a confidence interval for the shift, and it is reported as `shift_ci_lower` and `shift_ci_upper`. Inverting a test this way is the general recipe (Ernst, 2004); the endpoints are located by bisection on the *p*-value, using the same reassignments throughout so the interval and the test agree.

The contrast with `normal_theory_ci_lower` and `normal_theory_ci_upper`, which are Welch's *t* limits on the same mean difference, is worth reading whenever both are printed. The randomization interval is exactly the set of shifts the test being run does not reject, so the test and the interval can never disagree. The normal theory interval instead assumes the sampling distribution of the mean difference has a known shape; it is smooth, symmetric about the point estimate, and can extend past the range the data can support. The randomization interval is discrete, need not be symmetric, and in a very small design is unbounded, a correct statement of how little information the design carries rather than a defect: with three observations per group the smallest attainable two-sided *p*-value is 2/20 = 0.10, so no shift can be rejected at the 5% level and the 95% interval is the whole real line. The randomization interval also inherits the shift model, so it answers a narrower question than the test does: the test needs only exchangeability, while the interval needs the treatment to move every score by the same amount.

Effect sizes. Every effect size reported here comes from the package function that owns it, so the numbers match a direct call. The one exception is a pooled within-group variance of zero, where the standardized mean difference and the common language effect size are reported as NA (see Value). smd and ci_smd supply the standardized mean difference and its noncentral t interval; cles supplies the common language effect size, the probability that a randomly drawn score from the first group exceeds one from the second, by transforming those limits through $\Phi(\cdot/\sqrt{2})$; and cliff_delta supplies Cliff's delta with its consistent interval. The standardized mean difference and the common language effect size are normal theory quantities, so their intervals lean on the assumption the test itself avoids. Cliff's delta does not: it is a function of the ordering of the observations alone, which makes it the natural effect size companion to a randomization test. Reporting all three lets a reader see whether the distribution-free and normal theory summaries tell the same story.

Ranks give the Wilcoxon test. Replacing the scores by their ranks and running this test with `statistic = "mean"` reproduces the exact Wilcoxon rank sum test, since the rank sum is a monotone function of the difference in mean ranks. That equivalence is a useful check and a reminder of what the rank test is: a randomization test on transformed data.

Value

A `data.frame` with a `term` column and a numeric `value` column, in three blocks.

The test: `mean_difference` (first group minus second), `statistic` (the statistic actually referred to the reference distribution, always finite), `p_value`, and `p_value_se` (the Monte Carlo standard error of the p-value, NA under exact enumeration, which has no Monte Carlo error).

The intervals and effect sizes: `shift_ci_lower` and `shift_ci_upper` (the randomization interval for the shift, obtained by inverting the test); `normal_theory_ci_lower` and `normal_theory_ci_upper` (Welch's t interval on the same mean difference, reported for contrast); `smd` with `smd_ci_lower` and `smd_ci_upper` from ci_smd; `cles` with `cles_ci_lower` and `cles_ci_upper` from cles; and `cliff_delta` with `cliff_delta_ci_lower` and `cliff_delta_ci_upper` from cliff_delta. When the pooled within-group variance is zero (each group constant) the standardized mean difference is undefined: the six rows `smd`, `smd_ci_lower`, `smd_ci_upper`, `cles`, `cles_ci_lower`, and `cles_ci_upper` are NA, a warning of class `dmar_smd_undefined` says so, and the test, the shift interval, and Cliff's delta are reported as usual.

The design: `n_1`, `n_2`, `N`, `n_evaluated` (how many reassignments were actually used), and `exact` (1 if every reassignment was enumerated, 0 if they were sampled).

Non-numeric information travels on attributes rather than in the `value` column: `statistic_name`, `method` ("exact enumeration" or "Monte Carlo"), `alternative`, `group_labels`, `response_name`, `group_name`, `seed`, `observed_statistic`, and `reference_distribution`, the vector of statistics over the reassignments that plot_randomization_test draws, which can hold `Inf` and `-Inf` under `statistic = "t"` (see `statistic`). `reference_extreme` is a logical vector of the same length that marks the reassignments the p-value counted as at least as extreme as the observed one.

No table is returned when `statistic = "t"` is refused; the condition, of class `dmar_studentized_statistic_undefined`, carries `reason` (`"constant_groups"`), `alternative`, `n_1`, `n_2`, and `p_value_mean`, the exact p-value `statistic = "mean"` gives for the same data and alternative.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Edgington, E. S., & Onghena, P. (2007). *Randomization tests* (4th ed.). Chapman & Hall/CRC.

Ernst, M. D. (2004). Permutation methods: A basis for exact inference. *Statistical Science, 19*(4), 676–685. doi:10.1214/088342304000000396

Fisher, R. A. (1935). *The design of experiments*. Oliver & Boyd.

Janssen, A. (1997). Studentized permutation tests for non-i.i.d. hypotheses and the generalized Behrens-Fisher problem. *Statistics & Probability Letters, 36*(1), 9–21. doi:10.1016/S01677152(97)00043-6

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 1 on the logic of randomization and the randomization test.)

Neuhaus, G. (1993). Conditional rank tests for the two-sample problem under random censorship. *The Annals of Statistics, 21*(4), 1760–1779. doi:10.1214/aos/1176349396

Phipson, B., & Smyth, G. K. (2010). Permutation *p*-values should never be zero: Calculating exact *p*-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology, 9*(1), Article 39. doi:10.2202/15446115.1585

Pitman, E. J. G. (1937). Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society, 4*(1), 119–130.

See Also

`plot_randomization_test` for the figure that shows the reference distribution, the observed statistic, and the rejection region; `randomization_test_paired` for the sign-flip sibling used with paired observations; `t.test` and `wilcox.test` for the parametric and rank-based alternatives; `smd`, `ci_smd`, `cles`, and `cliff_delta` for the effect sizes reported here.

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# 1. Ten observations, so every one of the choose(10, 5) = 252
#    reassignments is enumerated and the p-value is exact.
treatment <- c(80, 84, 79, 88, 83)
control   <- c(72, 75, 68, 81, 74)
randomization_test(group_1 = treatment, group_2 = control)

# 2. The studentized statistic. The groups are the same size here, so it
#    gives the same p-value as the difference in means; the two differ
#    when the groups are unequal in size.
randomization_test(group_1 = treatment, group_2 = control,
                   statistic = "t")

# 3. Formula interface: weekly drinking in the two comparable arms of
```

```
#    the drinks_trial data, a right-skewed outcome, which is exactly
#    where a distribution-free test earns its keep. With 37 and 32
#    participants there are far too many reassignments to enumerate, so
#    they are drawn at random and the p-value carries a Monte Carlo
#    standard error. n_resamples = 2000 keeps the example quick; a
#    reported p-value deserves the default of 10,000.
cra <- droplevels(subset(drinks_trial, treatment != "CRA + Disulfiram"))
set.seed(113)
randomization_test(drinks_per_week ~ treatment, data = cra,
                   n_resamples = 2000, seed = 113)

# 4. A one-sided test, and the one-sided interval that goes with it.
randomization_test(group_1 = treatment, group_2 = control,
                   alternative = "greater")

# 5. On ranks, the test is the exact Wilcoxon rank sum test.
y <- c(treatment, control)
g <- rep(c("treatment", "control"), each = 5)
res <- randomization_test(rank(y), g)
res$value[res$term == "p_value"]
wilcox.test(treatment, control, exact = TRUE)$p.value

# 6. Both observed groups are constant, at different values. The
#    difference in means is defined, and gives the smallest p-value
#    groups of 4 and 4 allow. The warnings say that the standardized
#    mean difference is undefined and that the groups do not overlap.
#    With statistic = "t" the same call is refused, with a message that
#    says why: the standard error is 0, so the studentized statistic is
#    undefined.
randomization_test(group_1 = rep(5, 4), group_2 = rep(3, 4))
```

`randomization_test_paired` *Paired-Samples Randomization (Sign-Flip) Test*

Description

Computes an exact (or Monte Carlo) sign-flip randomization test for paired observations (x_i, y_i), treating the within-pair sign of $d_i = y_i - x_i$ as the randomization mechanism (Fisher, 1971; Edgington & Onghena, 2007). Under the null hypothesis of exchangeability (H_0: the labeling of x and y within each pair is arbitrary), each of the 2^n sign patterns is equally likely.

Usage

```
randomization_test_paired(
  x,
  y,
  statistic = c("mean", "t"),
```

```
  alternative = c("two_sided", "less", "greater"),
  exact = NULL,
  n_resamples = 10000L,
  seed = NULL
)
```

Arguments

x, y	Paired numeric vectors of equal length. NAs are removed pairwise; the remaining values must be finite.
statistic	One of "mean" (default) or "t". "mean" reports the mean difference and "t" the paired *t* statistic. Over the sign patterns the paired *t* is a strictly increasing function of the mean difference, so the two give the same *p*-value and differ only in the statistic reported. "t" is refused when every difference has the same nonzero value (exactly, or to within the tolerance of the comparison), because the statistic then has a zero standard error and no value (see *When the studentized statistic is undefined*); when every difference is zero it is reported as 0 with a *p*-value of 1.
alternative	One of "two_sided" (default; the base-R spelling "two.sided" is accepted as an alias), "less", or "greater".
exact	Logical. If NULL (default), uses exact enumeration when $n \leq 20$ and Monte Carlo otherwise. If TRUE, forces exact enumeration (caps at $n \leq 25$; above that the 2^n space is too large). If FALSE, forces Monte Carlo.
n_resamples	A single whole number of at least 1: the number of Monte Carlo resamples when exact enumeration is not used. Default 10000L.
seed	Optional integer seed for reproducibility of the Monte Carlo branch. Default NULL, under which the randomizations are drawn from, and advance, the current random number stream; supply an integer for reproducibility, which is set for the call only, the caller's random number state being restored when it returns.

Details

For small n (default $n \leq 20$) the test enumerates all 2^n sign patterns exactly; for larger n a Monte Carlo approximation is used (default n_resamples = 10000L).

Why randomization. The randomization test makes no distributional assumption on d_i; it only assumes that under the null, the sign of each d_i is arbitrary. This is exactly the inference that pre-experimental random assignment licenses. Because the reference distribution is built from the observed differences, the Type I error rate under that null does not exceed the nominal level whatever the distribution of the differences, and for any continuous distribution, heavy-tailed, a mixture, or with outliers, it is the same attainable level: under exact enumeration, the largest multiple of $1/2^{n-1}$ (two-sided) or $1/2^n$ (one-sided) that does not exceed the nominal level (6/128 = .047 two-sided at .05 with 8 pairs). When different sign patterns can give the same statistic, as with discrete scores, it can be lower.

Exact enumeration. All 2^n sign patterns are enumerated for n of 20 or fewer by default, and for n up to 25 when exact = TRUE. The observed test statistic is compared with the full reference distribution. The exact two-sided *p*-value is the proportion of patterns yielding a test statistic at least as extreme (in absolute value) as the observed. A pattern whose statistic equals the observed

one counts as at least as extreme; the comparison allows for the rounding of the arithmetic, with a bound proportional to the size of the scores, so that patterns tied in exact arithmetic are counted as tied.

The two statistics order the patterns alike. Write S for the sum of the signed differences and $Q = \sum d_i^2$, which no sign pattern changes. The paired *t* is $S\sqrt{n-1}/\sqrt{nQ - S^2}$, which increases with S because Q is fixed. Both statistics therefore rank the patterns as S does, and the function counts on S for both.

When the studentized statistic is undefined. The studentized statistic is a ratio, the mean difference over its standard error, and it has a value only when that standard error is positive. If every paired difference is the same nonzero number (exactly, or to within the tolerance of the comparison), the standard error of the observed data is 0 and `statistic = "t"` is refused with a condition of class `dmar_studentized_statistic_undefined`. The observed data are then at the boundary of the reference set: among the 2^n sign patterns only the observed one and its mirror image have a zero standard error. The refusal concerns the statistic and not the evidence. `statistic = "mean"` is defined for every arrangement and gives the smallest *p*-value the design allows in the direction of the data, $2/2^n$ two-sided and $1/2^n$ one-sided; the message states the value, and the condition carries it as `p_value_mean`. The refusal is made before any sign pattern is enumerated or drawn, so a refused Monte Carlo call leaves the random number stream where it was.

When the observed statistic is defined, some arrangements in the reference set can still have a zero standard error: with paired differences of equal size and mixed signs, the two patterns that give them all one sign. Those patterns are the two most extreme, which is where the mean difference also places them, and since the function counts on the signed sum S for both statistics they are counted without a standard error being formed. Refusing them would refuse ordinary coarsely measured data. For differences of 2, 2, 2, -2, and 2 the paired *t* is 1.5 and the *p*-value, 12/32 = 0.375, is the one the mean difference gives. When every difference is zero, every sign pattern is the same data set; the statistic is reported as 0 and the *p*-value is 1.

Monte Carlo branch. For larger n, `n_resamples` random sign patterns are drawn uniformly from $\{-1, +1\}^n$; the Monte Carlo *p*-value uses the standard $(1 + \text{count}) / (1 + B)$ plug-in to avoid $p = 0$.

Value

A `data.frame` (class `dmar_tbl`) with columns `term` and `value` and five rows: `statistic` (the mean difference $\bar{d}$, or the paired *t* under `statistic = "t"`, always finite), `p_value`, `n_pairs` (the pairs with both scores present), `n_evaluated` (2^n under exact enumeration, `n_resamples` otherwise), and `exact` (1 when every sign pattern was enumerated, 0 when patterns were sampled). No table is returned when `statistic = "t"` is refused; the condition, of class `dmar_studentized_statistic_undefined`, carries `reason` (`"constant_differences"`), `alternative`, `n` (the number of pairs), and `p_value_mean`, the exact *p*-value `statistic = "mean"` gives for the same call.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Edgington, E. S., & Onghena, P. (2007). *Randomization tests* (4th ed.). Chapman & Hall/CRC.

Fisher, R. A. (1971). *The design of experiments* (9th ed., reprint). Hafner.

Pitman, E. J. G. (1937). Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society, 4*(1), 119–130.

See Also

t.test (parametric paired test), probability_of_superiority_paired

Other hypothesis tests: adjusted_means(), ancova(), anova_within(), ci_dunnett(), ci_dunnett_t3(), ci_games_howell(), ci_scheffe(), ci_tukey_kramer(), compare_cov_structures(), contrast_test(), correlations_test(), equivalence_proportion_paired(), equivalence_r(), equivalence_rmsea(), equivalence_smd(), factorial_anova(), manova_split_plot(), mauchly_test(), mixed_anova(), obrien_test(), pairwise_within(), randomization_test(), regions_of_significance(), simple_effects_AB(), summary_t_test(), welch_t()

Examples

```
# 1. Small-n exact: Bayley scores on twin pairs.
control <- c(95, 102,  98, 107, 105)
treat   <- c(102, 108, 100, 112, 109)
randomization_test_paired(control, treat)

# 2. Larger n: Monte Carlo branch.
set.seed(113)
x <- rnorm(50, 100, 15)
y <- x + rnorm(50,   5, 12)
randomization_test_paired(x, y, n_resamples = 10000L)

# 3. Every pair differs by the same amount. The mean difference is
#    defined, and gives the smallest p-value 5 pairs allow. With
#    statistic = "t" the same call is refused, with a message that says
#    why: the observed standard error is 0, so the studentized statistic
#    is undefined.
before <- c(10, 12, 9, 15, 11)
randomization_test_paired(before, before + 2)
```

---

regions_of_significance

Regions of Significance for a Covariate by Group Interaction

Description

Finds the values of a covariate at which two groups differ significantly when the within-group regression slopes are *not* equal, that is, when there is a covariate-by-group interaction (heterogeneity of regression). With more than two groups the calculation is carried out for every pair of groups.

Usage

```
regions_of_significance(
  object,
  data = NULL,
  conf_level = 0.95,
  method = c("simultaneous", "pointwise")
)
```

Arguments

`object`	Either a fitted `lm` or `aov` of the form `y ~ x * group`, or a model formula of that form. When a formula is supplied, `data` must be supplied too and the model is fit with `lm`.
`data`	A `data.frame` holding the variables in the formula. Used, and required, only when `object` is a formula; ignored with a warning when `object` is already a fitted model.
`conf_level`	Confidence level for the boundaries. Default `0.95`.
`method`	Character string naming the critical value. `"simultaneous"` (the default) uses Potthoff's (1964) simultaneous critical value $\sqrt{2F_{1-\alpha;2,\nu}}$, which holds the error rate over the whole covariate continuum at once. `"pointwise"` uses the classic critical value $t_{1-\alpha/2;\nu}$, which holds it at one covariate value chosen in advance.

Details

Why the procedure exists. An analysis of covariance that assumes a common within-group slope reports one adjusted mean difference, and that single number is a complete summary of the group comparison only if the slopes really are common. When the covariate interacts with the group factor the slopes are not common, the two fitted lines converge or cross, and there is no such thing as "the" treatment effect: the difference between the groups depends on where along the covariate you look. Reporting the adjusted mean difference anyway reports the difference at one covariate value, the covariate grand mean, and says nothing about the rest of the range. The question worth answering is instead *where* on the covariate the groups differ, and that is what this function answers.

The calculation. For two groups, write the estimated difference at covariate value x as the line

$$\hat{D}(x) = \hat{d}_0 + \hat{d}_1 x,$$

where $\hat{d}_0$ is the difference in intercepts and $\hat{d}_1$ the difference in slopes. Because $\hat{D}(x)$ is a linear combination of the regression coefficients, its sampling variance follows from their covariance matrix,

$$\mathrm{Var}[\hat{D}(x)] = \mathrm{Var}(\hat{d}_0) + 2x\,\mathrm{Cov}(\hat{d}_0, \hat{d}_1) + x^2 \mathrm{Var}(\hat{d}_1).$$

The groups differ significantly at x exactly when $\hat{D}(x)^2 > t_{crit}^2\,\mathrm{Var}[\hat{D}(x)]$. Setting the two sides equal gives a quadratic in x,

$$(\hat{d}_1^2 - t_{crit}^2 \mathrm{Var}(\hat{d}_1))x^2 + 2(\hat{d}_0\hat{d}_1 - t_{crit}^2 \mathrm{Cov}(\hat{d}_0, \hat{d}_1))x + (\hat{d}_0^2 - t_{crit}^2 \mathrm{Var}(\hat{d}_0)) = 0,$$

whose real roots are the boundaries of the region of significance.

Every geometry is possible, and the leading coefficient decides which. The coefficient on x^2 is positive exactly when the slope difference itself clears the critical value. When it is positive the

parabola opens upward and the groups differ *outside* the two boundaries, the familiar picture of two lines that cross somewhere in the middle of the covariate and separate at both ends. When it is negative the parabola opens downward and the groups differ *between* the boundaries, a middle band of covariate values where the two lines are far enough apart relative to the precision available there. When there are no real roots the sign never changes, so the groups differ either everywhere or nowhere. All of these are reported through `region_code` rather than being treated as failures.

Of those, "everywhere" is a case the code enumerates but the mathematics rules out whenever the slopes genuinely differ. The estimated difference $\hat{D}(x)$ is then a nonconstant line in x, so it crosses zero at some covariate value, and at that value the squared difference is zero while the critical bound is positive. The quadratic therefore always has two real roots, and the significant set is the pair of tails outside them or the band between them, never the whole line. Note this is a statement about the covariate axis extended without limit, not about the observed data: within the range actually observed the groups may well differ everywhere, which is why the next paragraph matters.

Boundaries outside the observed data. A boundary is a root of an equation, and the equation is happy to place it far outside the covariate values that were actually observed. Such a boundary is an extrapolation of two fitted lines into a region where there are no data to support them, and it should not be read as a covariate value at which anything can be claimed. The boundary is still reported, since suppressing it would hide the shape of the result, but it is flagged in `lower_bound_in_range` and `upper_bound_in_range` and noted in the `region` string.

Simultaneous versus pointwise. The classic critical value is $t_{1-\alpha/2}$, which controls the Type I error rate at a *single* covariate value fixed in advance. The procedure, however, scans the covariate continuum and reads off where the groups differ, which is a search over infinitely many tests. Potthoff (1964) gave the simultaneous critical value $\sqrt{2F_{1-\alpha;2,\nu}}$, which holds the error rate over the entire covariate range at once and is therefore the default here, and the value used in Maxwell, Delaney, and Kelley (2027, Chapter 9). The simultaneous critical value is always the larger of the two, so its region of significance is always the more conservative one. With more than two groups the simultaneous guarantee applies to each pair over the covariate range, not to the family of pairs. For a Bonferroni protection across the pairs, divide the Type I error rate by the number of pairs before choosing `conf_level`: with three groups (three pairs) and a familywise rate of .05, pass `conf_level = 1 - .05 / 3`.

What the model may contain. The model must contain exactly one interaction between a numeric covariate and a grouping factor, and the grouping factor may not appear in any other term. Additional predictors that do not interact with the group are allowed and drop out of the group difference, so `y ~ x * group + block` is fine while `y ~ x * group * block` is not. A grouping variable stored as a number (0/1, say) is a numeric predictor to R, not a factor, so convert it with `factor` first.

Value

A `data.frame` (class `dmar_tbl`) with one row per reported quantity per pair of groups and columns `pair`, `term`, and a numeric `value`. The terms, for each pair, are

`lower_bound`, `upper_bound` The boundaries of the region, sorted, on the scale of the covariate. `NA` when a boundary does not exist; when there is a single boundary it is reported as `lower_bound` and `upper_bound` is `NA`. Read them with `region_code`: the boundaries are the covariate values at which the group difference sits exactly on the critical value, and it is `region_code` that says on which side of them the groups differ.

`region_code` How to read the boundaries. `1`: two boundaries, the groups differ *outside* them. `2`: two boundaries, the groups differ *between* them. `3`: no boundary, the groups differ at every

covariate value. 4: no boundary, the groups differ at no covariate value. 5 and 6: one boundary, the groups differ above it (5) or below it (6).

`n_boundaries` How many real boundaries exist: 0, 1, or 2.

`lower_bound_in_range`, `upper_bound_in_range` 1 when the boundary falls inside the covariate values actually observed in the two groups, 0 when it falls outside them, NA when the boundary does not exist. A boundary outside the observed range is an extrapolation of the fitted lines and should not be interpreted as a covariate value at which anything was or could be seen.

`difference_intercept`, `difference_slope` The intercept d_0 and slope d_1 of the group difference as a function of the covariate, so that the estimated difference at covariate value x is $d_0 + d_1 x$.

`critical_value` The critical value used, t_{crit}.

`df_error` Error degrees of freedom of the fitted model.

`conf_level` The confidence level.

Everything that is not a number is carried on attributes rather than forced into `value`: `method`, `outcome`, `covariate`, `group`, the overall observed `covariate_range`, a `pairs` `data.frame` with the group labels, the per-pair observed covariate range, and a plain-language `region` string for each pair, and a `geometry` `data.frame` holding d_0, d_1, and the three elements of their covariance matrix, which is what `plot_regions_of_significance` draws.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Johnson, P. O., & Neyman, J. (1936). Tests of certain linear hypotheses and their application to some educational problems. *Statistical Research Memoirs, 1*, 57–93.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 and its extension on heterogeneity of regression.)

Potthoff, R. F. (1964). On the Johnson-Neyman technique and some extensions thereof. *Psychometrika, 29*(3), 241–256. doi:10.1007/BF02289721

Rogosa, D. (1980). Comparing nonparallel regression lines. *Psychological Bulletin, 88*(2), 307–321. doi:10.1037/00332909.88.2.307

See Also

`plot_regions_of_significance` to see the group difference and its confidence band across the covariate; `ancova` for the common-slope analysis and its homogeneity-of-regression test; `pygmalion` for the data used below.

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `simple_effects_AB()`, `summary_t_test()`, `welch_t()`

Examples

```
# ---- The Pygmalion teacher-expectancy data ----
# Post-test IQ, averaged over the two follow-up assessments, on
# pretest IQ, separately by condition. The slopes differ, so the
# expectancy effect depends on where the child started.
data(pygmalion)
pygmalion$iq_post <- (pygmalion$iq_4 + pygmalion$iq_8) / 2
fit <- lm(iq_post ~ iq_pre * treatment, data = pygmalion)

regions_of_significance(fit)

# The plain-language reading of each pair is on an attribute.
attr(regions_of_significance(fit), "pairs")$region

# The pointwise (classic) critical value gives a wider region,
# because it does not pay for scanning the whole covariate.
regions_of_significance(fit, method = "pointwise")

# ---- Formula interface, and more than two groups ----
set.seed(113)
n <- 150
g <- factor(rep(c("control", "low", "high"), each = n / 3))
x <- rnorm(n, 50, 10)
y <- 2 + 0.5 * x + (g == "high") * (0.4 * x - 15) + rnorm(n, 0, 5)
d <- data.frame(y, x, g)
regions_of_significance(y ~ x * g, data = d)
```

---

reliability *Reliability Coefficient With a Confidence Interval (General Dispatch)*

Description

General-purpose entry point for the reliability family. Dispatches to `reliability_alpha`, `reliability_kr20`, `reliability_omega`, `reliability_omega_categorical`, or `reliability_H` according to the requested `type`. When `type` is not specified, the function picks a reasonable default from the supplied input following the recommendations of Kelley and Pornprasertmanit (2016).

Usage

```
reliability(
  data = NULL,
  S = NULL,
  N = NULL,
  type = NULL,
  estimator = c("analytic", "model_implied"),
  denominator = c("observed", "model_implied"),
  missing = NULL,
  aux = NULL,
```

```
  ci_method = NULL,
  conf_level = 0.95,
  B = NULL,
  seed = NULL,
  ncpus = 1L,
  improper = c("keep", "drop"),
  impute = NULL,
  imputations = NULL,
  wald_df = Inf,
  progress = NULL
)
```

Arguments

`data`	A numeric matrix or data frame of item scores, or `NULL`.
`S`	A symmetric covariance matrix among the items, or `NULL`. If supplied, `N` must also be supplied; methods that require raw data are then unavailable.
`N`	Total sample size; required when `S` is supplied.
`type`	Character; one of `"alpha"`, `"kr20"`, `"omega"`, `"omega_categorical"` (`"omega_c"` is accepted as a shorthand), `"H"` (coefficient *H*, the reliability of the optimally weighted composite, from raw data or a covariance matrix; `reliability_H` also takes a fitted model or standardized loadings), or `NULL` for auto-detection. See *Details*.
`estimator`	For `type = "alpha"` only: how coefficient α is estimated, `"analytic"` (default; the closed-form equation applied to the observed covariance matrix) or `"model_implied"` (the reliability implied by the τ-equivalent single-factor model fit by maximum likelihood); forwarded to `reliability_alpha`, whose help page discusses the choice and which interval methods each estimator supports. Supplying it with any other `type` is an error, as supplying `denominator` is for any type but `"omega"` and `"omega_categorical"`.
`denominator`	For `type = "omega"` and `type = "omega_categorical"` only: how the total variance in the denominator is estimated, `"observed"` (default; robust omega, and for categorical omega the unrestricted polychoric correlations) or `"model_implied"` (for categorical omega, the coefficient as Green and Yang, 2009, defined it); forwarded to `reliability_omega` or `reliability_omega_categorical`, whose help pages discuss the choice.
`missing`	How incomplete rows of `data` are handled, forwarded to the family function when supplied, which checks it against its own choices and whose help page discusses them: for `type = "alpha"` and `type = "omega"`, `"listwise"`, `"fiml"` (full information maximum likelihood), or `"two_stage"` (two-stage maximum likelihood); for `type = "H"`, `"listwise"` or `"fiml"`; for `type = "omega_categorical"`, `"listwise"`, `"pairwise"`, or `"mi"` (multiple imputation). The default, `NULL`, leaves the family function's default in place: for alpha, omega, and H, full information maximum likelihood when an item value is missing and the complete data analysis when none is, since the complete-data intervals of alpha and omega need that analysis (`fa_confirmatory` fits complete data by FIML too; a row with every item missing does not count, since both treatments drop it),

and "listwise" for categorical omega. Supplying it with type = "kr20" is an error.

aux — Optional character vector naming auxiliary variable columns of data, forwarded to the family function: saturated correlates under full information maximum likelihood, and variables entered into the first stage under two-stage estimation, for type = "alpha", type = "omega", and type = "H" (supplying aux without missing implies missing = "fiml"); predictors in the imputation model for type = "omega_categorical" (it implies missing = "mi"). Supplying it with type = "kr20" is an error.

ci_method — Method for constructing the confidence interval, or NULL to use the chosen family function's default. See the help page for the chosen reliability_* function for the full list of accepted values.

conf_level — Confidence level. Defaults to 0.95.

B — Number of bootstrap replications when a bootstrap method is selected. The default, NULL, uses the family function's default: 10000, or 1000 for categorical omega under missing = "mi". The family function checks B against the number its interval method and level require (400 for "percentile", 2000 for "bca", 100 for the two bootstrap standard error methods, at the default conf_level) and refuses a smaller one (class dmar_count_invalid); see dmar_bootstrap.

seed — Random number seed used for bootstrap reproducibility. Defaults to NULL, under which the resamples are drawn from, and advance, the current random number stream; supply an integer for a reproducible interval, which is set for the call only, the caller's random number state being restored when it returns.

ncpus — The number of processes over which the bootstrap replications are spread; forwarded to every family function. The default 1 runs them in this R session, and a given seed returns the same interval whatever ncpus is (see reliability_omega).

improper — For type = "alpha" and type = "omega" only: how a bootstrap replication whose single-factor solution is improper but still yields a coefficient is treated, "keep" (the default) or "drop"; forwarded to reliability_alpha or reliability_omega, whose help pages describe it. Categorical omega is not defined for an improper solution, and KR-20 fits no model, so supplying it with another type is an error.

impute, imputations — For type = "omega_categorical" with missing = "mi" only: the imputation model and the number of imputations of each bootstrap replication, forwarded to reliability_omega_categorical, whose help page describes them. The default, NULL, leaves that function's defaults ("pmm" and 5) in place; supplying either with another type is an error.

wald_df — For type = "alpha" and type = "omega" only: the reference distribution of a Wald interval, Inf (the default, the normal reference), a single positive number of degrees of freedom of a *t* reference, or, for type = "omega" with denominator = "model_implied" only, "missing_information", the degrees of freedom of a *t* reference from omega's fraction of missing information; forwarded to reliability_alpha or reliability_omega, whose help pages describe it (the keyword is an error for alpha, for which no rule passed the calibration study's decision rule). A finite value or the keyword with another type is an error.

progress NULL (the default), or a function of two arguments that the bootstrap calls as it runs, so that an application can show how far a long bootstrap has gone and let its user stop it (a Shiny app, for example). It is called as progress(done, total), with the number of replications attempted so far and B: after each replication when ncpus = 1, and with more processes after each block of replications they complete together, about a fiftieth of B (ncpus * ceiling(B / (50 * ncpus)) replications, the last block possibly fewer), so that done advances by the block. A BCa interval takes its acceleration from a regression on the resampling frequencies when at least as many replications return a value as there are cases, and from the jackknife, one leave-one-out refit per case, when fewer do or when that regression fails or gives influence values that are not all finite; the jackknife runs at most once in a call, and not at all when no interval is formed or every returned replicate has the same value. While it runs, the callback is called again after each refit (with more processes, after each block of ncpus * ceiling(n / (50 * ncpus)) refits, n the number of cases resampled), with done and total both B, so it can be called more than B times, and done equal to total does not mean that the bootstrap has finished. If the callback returns FALSE, the bootstrap stops before the next replication or block with an error of class dmar_bootstrap_cancelled, whose elements done and total say how far it got (see dmar_metadata); any other value is ignored. An error inside the callback stops the call with an error of class dmar_progress_error, whose message names the callback and which carries done, total, and, as parent, the callback's own condition with its class intact, so that an application can tell a failing callback from a failing analysis. The callback runs outside the random number stream, so a given seed returns the same result with or without it, and a callback that draws random numbers leaves the caller's stream as it found it. With NULL nothing is called. Ignored unless a bootstrap method is selected.

Details

Auto-detection rules (used only when type = NULL):

- If raw items are integer-valued and every item has at most 10 distinct values, type = "omega_categorical" (categorical omega; appropriate when items are ordered categorical and the relationship between the underlying factor and the observed items is non-linear).
- Otherwise, type = "omega" (McDonald's coefficient ω from a single-factor CFA). This includes the case where only a covariance matrix S and sample size N are supplied; **lavaan** fits the CFA on the covariance matrix.

The rule reads the items only. The columns of data named in aux are auxiliary variables, which never enter the composite, so their scale says nothing about the items' response format and they take no part. Auto-detection emits a single message() indicating which type was chosen, so the choice is stated rather than made silently. It never chooses type = "H", which describes a different composite (the optimally weighted one) rather than a different set of assumptions about the summed one, and is computed only when asked for.

The selected family function determines which ci_method values are accepted; see the help page for the chosen function for the full list. When ci_method is left at its default (NULL), the family function's own default is used:

- `reliability_alpha`: `"bonett"`, or `"ml"` when incomplete items are analyzed by full information maximum likelihood (the default of `missing` for them).
- `reliability_alpha(estimator = "model_implied")`: `"mlr"` with raw data; `"ml"` with covariance input.
- `reliability_kr20`: `"feldt"`.
- `reliability_omega`: for `denominator = "observed"` (robust omega, the default), the point estimate with no interval, since its interval is bootstrap based and no bootstrap runs unless requested; for `denominator = "model_implied"`, `"mlr_logistic"` with raw data and `"ml_logistic"` with covariance input, the limits formed on the logit scale (`"mlr"` and `"ml"` under `wald_df = "missing_information"`).
- `reliability_omega_categorical`: the point estimate with no interval, for the same reason; request `"bca"`.
- `reliability_H`: `"mlr_logistic"` with raw data; `"ml_logistic"` with covariance input.

A bootstrap is never run by default anywhere in the family. When a bootstrap method is requested, `B = 10000` replications is the default, except for categorical omega under `missing = "mi"`, whose bootstrap then impute replications each impute the data and default to `B = 1000`.

Several reliability coefficients exist because their assumptions differ. Coefficient α (and its dichotomous specialization KR-20) equals the population reliability under essential τ-equivalence (equal loadings); McDonald's ω relaxes that assumption to a congeneric single-factor model; and `reliability_omega(denominator = "observed")` further relaxes the requirement that the single-factor model fit perfectly by using the observed composite variance in the denominator (see the `reliability_omega` help page for the properties of that choice). For well-behaved homogeneous measurement instruments these coefficients typically yield very similar values. For ordered categorical items the relationship between the latent factor and the observed responses is non-linear, and `reliability_omega_categorical` handles that case explicitly via a probit-link single-factor model.

Value

The `data.frame` returned by the dispatched `reliability_*` function (rows: `estimate`, `se`, `ci_lower`, `ci_upper`, `conf_level`, `N`, `N_complete`, `n_items`). The `se` row is on the coefficient scale; the transformation-based intervals (`"fisher"`, `"bonett"`, `"hakstian_whalen"`) add an `se_transformed` row carrying the transformation-scale standard error, with the scale named in the `se_transform_scale` attribute. The `coefficient` attribute identifies which coefficient was computed, coefficients alpha and omega record `wald_df`, and a bootstrap call also carries the `bootstrap` attribute with its replication counts (see `reliability_alpha`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika, 74*, 155–167. doi:10.1007/s11336-00890993

Kelley, K., & Cheng, Y. (2012). Estimation of and confidence interval formation for reliability coefficients of homogeneous measurement instruments. *Methodology, 8*, 39–50. doi:10.1027/1614-2241/a000036

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Terry, L. J., & Kelley, K. (2012). Sample size planning for composite reliability coefficients: Accuracy in parameter estimation via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 371–401. doi:10.1111/j.20448317.2011.02030.x

See Also

`reliability_alpha`, `reliability_kr20`, `reliability_omega`, `reliability_omega_categorical`, `ss_aipe_reliability`, `cfa_1`.

Other reliability: `cohen_kappa()`, `diagnosis_agreement`, `fleiss_kappa()`, `icc()`, `reliability_H()`, `reliability_alpha()`, `reliability_kr20()`, `reliability_omega()`, `reliability_omega_categorical()`

Examples

```
set.seed(113)
J <- 6
loadings <- seq(0.4, 0.8, length.out = J)
eta <- rnorm(200)
errors <- matrix(rnorm(200 * J), 200, J) %*% diag(sqrt(1 - loadings^2))
items <- sweep(matrix(rep(eta, J), 200, J), 2, loadings, `*`) + errors
colnames(items) <- paste0("y", seq_len(J))

# Auto-detection picks coefficient omega for continuous data.
reliability(data = items)

# Explicit type.
reliability(data = items, type = "alpha")

# A covariance matrix and its sample size stand in for raw data, and
# auto-detection again picks coefficient omega. The point estimate is
# the same as from the raw data; the message explains why no interval
# accompanies it, since the bootstrap behind robust omega resamples
# rows that a covariance matrix does not carry.
reliability(S = cov(items), N = 200)
```

`reliability_alpha` *Coefficient Alpha With a Confidence Interval*

Description

Estimates coefficient α for a homogeneous composite score and returns a confidence interval for the population coefficient using one of several documented methods.

Usage

```
reliability_alpha(
  data = NULL,
  S = NULL,
  N = NULL,
  estimator = c("analytic", "model_implied"),
  missing = c("fiml", "listwise", "two_stage"),
  aux = NULL,
  ci_method = NULL,
  conf_level = 0.95,
  B = 10000,
  seed = NULL,
  ncpus = 1L,
  improper = c("keep", "drop"),
  wald_df = Inf,
  progress = NULL
)
```

Arguments

data	A numeric matrix or data frame of item scores (rows are respondents, columns are items). Either data or S must be supplied. How incomplete rows are handled is governed by missing: full information maximum likelihood by default, listwise deletion with missing = "listwise", or two-stage maximum likelihood with missing = "two_stage". When aux is supplied, data contains both the item columns and the auxiliary columns, and the columns not named in aux are the items. An item column that is not numeric (or logical), and an item that takes a single value in the rows analyzed (the complete rows under listwise deletion, the rows that observe it otherwise), are refused by name before anything is computed, as is a zero variance on the diagonal of S: such an item adds a constant to the composite, whose reliability it leaves as it is, while the coefficient would count it among the J items (alpha with it is $J(J-2)/(J-1)^2$ times alpha without it). A composite that does not vary is refused too, as when the items are ipsative (their scores sum to the same total in every row): its reliability is not defined. Under missing = "fiml" and "two_stage" it is checked in the rows that observe every item when they outnumber the items, where a composite that does not vary leaves the likelihood of the saturated model without a maximum, and otherwise on the EM estimate of the covariance matrix. SPSS user-defined missing values, which haven::read_sav(user_na = TRUE) keeps as values of a haven_labelled_spss column, are missing, as SPSS treats them.
S	A symmetric covariance matrix among the items. If S is supplied, N must also be supplied; raw-data-only confidence interval methods ("adf", "bootstrap_*", "percentile", "bca") are then unavailable, as are missing = "fiml" and missing

= "two_stage" named explicitly, and aux, since a covariance matrix has no incomplete cases (the default of missing analyzes it as it stands).

N — Total sample size; required when S (rather than data) is supplied.

estimator — How α is estimated: "analytic" (the default) applies the closed-form equation to the observed covariance matrix, and "model_implied" takes the reliability implied by the τ-equivalent single-factor model fit by maximum likelihood. See *Details*.

missing — How incomplete rows of data are handled: "fiml" (the default; full information maximum likelihood, using every case with at least one observed item or observed auxiliary variable), "listwise" (complete-case analysis), or "two_stage" (two-stage maximum likelihood, using the same cases as FIML). The default analysis is FIML whenever an item value is missing and the complete data analysis otherwise, unlike fa_confirmatory, which fits complete data by FIML too: data in which no value is missing, data complete but for rows with every item missing (which both treatments drop), and S are analyzed as they stand, with the complete data default interval, since the complete-data intervals need that analysis; incomplete items are analyzed by FIML, or, when ci_method names an interval FIML cannot give, the call stops and names the choices. See *Details*. With missing = "listwise", when fewer than two rows are complete, listwise deletion leaves nothing to analyze, and the function stops before any fit, says so, and names the treatment that uses the incomplete rows. The analytic estimator needs only the trace and the sum of the covariance matrix, which any two rows give, so it is computed from two or more complete rows. The model implied estimator fits the single-factor model by maximum likelihood, which needs the covariance matrix positive definite, and N complete rows give a matrix of rank at most $N - 1$; with no more complete rows than items it stops before the fit, says why, and names the treatment that uses the incomplete rows. Ordinary FIML is evaluated by the observed values and missingness patterns, not a blanket count of complete cases. Analytic alpha and two-stage estimation need unrestricted moments. The likelihood diagnostic refuses a certified affine-collapse direction or a numerical affine instability, as described in cov_fiml; EM convergence alone does not establish existence or uniqueness. Model-implied alpha can retain a proper identified constrained FIML fit with H0-based errors when H1 diagnostics are unavailable; the returned result states this limit. The complete rows behind the model-implied listwise estimator are counted as distinct rows. Bootstrap resamples retain a separate distinct-case estimability screen, which is a policy, not a proof that every smaller constrained FIML likelihood is unbounded. The refusal message for the analytic estimator names listwise deletion when two or more rows are complete and listwise deletion would accept them, and otherwise says why it cannot help either. A refusal of listwise deletion names missing = "fiml" only when that method could use the data, so the two refusals never point at each other, and when two columns share a name (which listwise deletion without aux reads by position, and that method refuses) it says to name them apart first.

aux — Optional character vector naming auxiliary variable columns of data: entered as saturated correlates under full information maximum likelihood, and into the first stage under missing = "two_stage". Supplying aux without missing implies missing = "fiml". See *Details*.

`ci_method` Method for constructing the confidence interval. See *Details* for which methods each estimator supports. The default `NULL` resolves to `"bonett"` for the analytic estimator and to `"mlr"` for the model implied estimator (or `"ml"` when only a covariance matrix is supplied, since `"mlr"` needs raw data). Under FIML (the default of `missing` with incomplete items) and two-stage estimation the analytic default is `"ml"`, whose standard error comes from the information matrix of the saturated model; with complete data the default of `missing` keeps `"bonett"`.

`conf_level` Confidence level for the interval (1 - Type I error rate). Defaults to `0.95`.

`B` Number of bootstrap replications when a bootstrap method is selected, a single whole number. Defaults to `10000`. The fewest accepted depends on the interval method and the level: at the default `conf_level`, 400 for `"percentile"` (10 replications beyond each limit), 2000 for `"bca"` (50 beyond each nominal limit, because the adjusted limits sit farther into the tails), and 100 for `"bootstrap_se"` and `"bootstrap_se_logistic"` (a standard deviation of the replications); at 0.99 the first two are 2000 and 10000. A smaller `B` is refused before anything is computed (class `dmar_count_invalid`). The count is held to the number requested. Replications set aside afterward are counted and reported in a warning of class `dmar_bootstrap_replications_dropped`, which names the function and says to request more when those that return a value are fewer than the number required; when fewer than half of the `B` replications return a value, the interval is withheld (limits and bootstrap standard error `NA`) under a warning of class `dmar_bootstrap_insufficient`. See `dmar_bootstrap`, which also describes `options(dmar.bootstrap_exploratory = TRUE)` for a first look with a smaller `B`.

`seed` Random number seed used for bootstrap reproducibility. Defaults to `NULL`, under which the resamples are drawn from, and advance, the current random number stream; supply an integer for a reproducible interval. When set, the seed applies inside the function only, and the user's generator state is restored when the function returns.

`ncpus` The number of processes over which the bootstrap replications are spread. The default `1` runs them in this R session; a larger number uses forked workers (**boot**'s `parallel = "multicore"`) or, on Windows, which cannot fork, a socket cluster (`"snow"`), whose workers load the installed **DMAR**. **boot** draws every resample before any replication runs, so a given `seed` returns the same interval, counts, and replicates whatever `ncpus` is. A session that forks workers and also starts background processes through **callr** or **processx** needs the environment variable `PROCESSX_NOTIFY_OLD_SIGCHLD` set before **processx** is loaded (see `sem_exact_fit_bootstrap`). Ignored unless a bootstrap method is selected.

`improper` For `estimator = "model_implied"` with a bootstrap interval: how a replication whose tau-equivalent single-factor solution is improper (a negative error variance, a Heywood case) is treated when the solution still yields a coefficient. `"keep"` (the default) uses its estimate like any other, and `"drop"` sets it aside; either way such replications are counted, and one whose solution yields no coefficient is set aside under both. With `"drop"` an improper point estimate stops the call. See `reliability_omega` for the reasoning. The analytic estimator fits no model, so no replication of it can be improper, and the argument has no effect there beyond being recorded as the `improper_rule` of the `bootstrap` attribute.

`wald_df` The degrees of freedom of the *t* distribution to which a Wald interval refers: `Inf` (the default) is the normal reference, and a single positive number replaces the normal quantile with the *t* quantile on that many degrees of freedom. It applies to `"ml"`, `"mlr"`, `"adf"`, and their `_logistic` variants; a finite value with any other interval is an error. The keyword `"missing_information"` of `reliability_omega` is an error here: its calibration found that the *t* reference gives alpha's FIML interval no gain and too much coverage. See *Details*.

`progress` `NULL` (the default), or a function of two arguments that the bootstrap calls as it runs, so that an application can show how far a long bootstrap has gone and let its user stop it (a Shiny app, for example). It is called as `progress(done, total)`, with the number of replications attempted so far and `B`: after each replication when `ncpus = 1`, and with more processes after each block of replications they complete together, about a fiftieth of `B` (`ncpus * ceiling(B / (50 * ncpus))` replications, the last block possibly fewer), so that `done` advances by the block. A BCa interval takes its acceleration from a regression on the resampling frequencies when at least as many replications return a value as there are cases, and from the jackknife, one leave-one-out refit per case, when fewer do or when that regression fails or gives influence values that are not all finite; the jackknife runs at most once in a call, and not at all when no interval is formed or every returned replicate has the same value. While it runs, the callback is called again after each refit (with more processes, after each block of `ncpus * ceiling(n / (50 * ncpus))` refits, `n` the number of cases resampled), with `done` and `total` both `B`, so it can be called more than `B` times, and `done` equal to `total` does not mean that the bootstrap has finished. If the callback returns `FALSE`, the bootstrap stops before the next replication or block with an error of class `dmar_bootstrap_cancelled`, whose elements `done` and `total` say how far it got (see `dmar_metadata`); any other value is ignored. An error inside the callback stops the call with an error of class `dmar_progress_error`, whose message names the callback and which carries `done`, `total`, and, as `parent`, the callback's own condition with its class intact, so that an application can tell a failing callback from a failing analysis. The callback runs outside the random number stream, so a given `seed` returns the same result with or without it, and a callback that draws random numbers leaves the caller's stream as it found it. With `NULL` nothing is called. Ignored unless a bootstrap method is selected.

Details

Coefficient α was first derived by Guttman (1945) as his λ_3 and was subsequently popularized by Cronbach (1951), under whose name the coefficient is often informally cited. The attribution to Cronbach is historically incomplete; the modern literature increasingly refers to the coefficient simply as "coefficient alpha" (see, e.g., Sijtsma, 2009; Revelle & Zinbarg, 2009). DMAR follows that convention.

For a J-item composite $Y = \sum_j X_j$ the population coefficient is

$$\alpha = \frac{J}{J-1}\left(1 - \frac{\sum_j \sigma_j^2}{\sigma_Y^2}\right),$$

where σ_j^2 is the variance of item j and σ_Y^2 is the variance of the composite. The sample estimate substitutes sample variances. Under classical test theory, α equals the population reliability of the

composite when the items are essentially τ-equivalent (i.e., equal factor loadings); when loadings differ, α is a lower bound on the population reliability. For well-behaved homogeneous measurement instruments coefficient α and McDonald's (1999) coefficient ω (`reliability_omega`) typically yield very similar values; ω extends to congeneric items (heterogeneous loadings) without the lower-bound caveat. For ordered categorical items see `reliability_omega_categorical`.

Coefficient α is at most one but, as a function of the covariance matrix, has no lower bound: it falls below zero when the item covariances are negative on balance, which can arise when an item is scored in the opposite direction to the others and needs reverse scoring. A reliability coefficient cannot be negative, so a negative estimate is returned as computed, with a warning that says so, and each confidence limit is clamped into [0, 1]; the interval around an estimate far enough below zero is then [0, 0], which says that the data support no positive reliability for the items as keyed.

Two estimators of the same coefficient. The argument `estimator` selects how α is estimated from the data. Both target the same population quantity, and they agree in the population whenever the τ-equivalent model holds; they differ in a finite sample because they take different routes to it.

`"analytic"` The default. The closed-form equation above, applied to the observed covariance matrix. This is the classical coefficient, the number a hand calculation produces, and it makes no assumption beyond those of classical test theory.

`"model_implied"` The reliability implied by the τ-equivalent (equal loadings) single-factor model fit by maximum likelihood. With a shared loading λ and error variances ψ_j^2 the model implied reliability of the J-item composite is

$$\alpha = \frac{(J\lambda)^2}{(J\lambda)^2 + \sum_j \psi_j^2},$$

with maximum likelihood estimates substituted. Estimating through the model brings inference the formula cannot provide: a delta method standard error, a robust (Satorra-Bentler) variant under nonnormality, the profile likelihood interval, and a fit assessment of the τ-equivalence claim itself. When the equal-loadings claim is doubtful, the congeneric `reliability_omega` is the appropriate coefficient rather than either α.

Users of **MBESS** will recognize these as `ci.reliability(type = "alpha")` and `ci.reliability(type = "alpha-cfa")` respectively.

Missing data and auxiliary variables. Listwise deletion is unbiased only when the data are missing completely at random and is inefficient always; the `mlmr` vignette develops the argument at length. By default (`missing = "fiml"`) incomplete data are therefore analyzed by full information maximum likelihood (FIML), which keeps every case with at least one observed item and is consistent and efficient under the weaker missing at random (MAR) assumption; `missing = "listwise"` analyzes the complete rows alone. Data in which no value is missing, and a covariance matrix, are analyzed as they stand: FIML would give the same coefficient, and the interval is the complete data one, Bonett's by default for the analytic estimator. A row with every item missing does not count as incomplete data: without auxiliary variables, listwise deletion and FIML both drop it, so data that are complete but for such rows (a blank line at the end of a spreadsheet, say) are analyzed as they stand too. The default analysis is therefore FIML whenever an item value is missing and the complete data analysis otherwise. `fa_confirmatory` differs: it fits complete data by FIML as well. The complete data analysis is used here because the closed-form intervals (`"bonett"`, `"feldt"`, `"fisher"`, `"hakstian_whalen"`), `"adf"`, `"adf_logistic"`, and `"likelihood"` need it, FIML giving none of them. When the default meets incomplete items and one of those intervals, the

call stops rather than deleting cases unasked, and names the two choices: `missing = "listwise"`, which computes the interval from the complete rows alone, or an interval FIML gives for the estimator in use (`"ml"`, `"ml_logistic"`, with the model implied estimator `"mlr"` and `"mlr_logistic"`, or a bootstrap method); when `ci_method` is not supplied, the analytic estimator's interval under FIML is `"ml"`. The `aux` argument names auxiliary variables: columns of `data` that are not part of the composite but are correlated with the items or with the reasons values are missing. They are entered as *saturated correlates* (Graham, 2003): correlated freely with each other and with every item's residual, never loading on the factor and never entering the composite, so the measurement model is undisturbed while FIML uses their information. Beyond recovering information, a good auxiliary makes the MAR assumption itself more plausible, since missingness that depends on the auxiliary becomes MAR once the auxiliary is conditioned on (Collins, Schafer, & Kam, 2001). A case whose items are all missing is kept when an auxiliary variable is observed on it: the saturated model estimates the items and the auxiliaries jointly, so such a case informs the item moments through its auxiliaries, and setting it aside would make the analysis conditional on having an observed item, which biases the estimates when the items go missing together at random given the auxiliaries. Supplying `aux` without `missing` implies `missing = "fiml"`, whether or not a value is missing; combining it with an explicit `missing = "listwise"` is an error. How each estimator uses FIML: the model implied estimator simply fits its model with `missing = "ml"`; the analytic estimator applies the classical formula, unchanged, to the FIML estimate of the item covariance matrix (the EM estimate under a saturated model over the items and any auxiliaries, the matrix `cov_fiml` returns), so the estimand stays the classical coefficient and only the covariance matrix it is computed from improves. Under `missing = "fiml"` the available intervals are `"ml"` and `"ml_logistic"` (both estimators; the standard error comes from the FIML information matrix), `"mlr"` and `"mlr_logistic"` (model implied estimator; the Yuan-Bentler robust standard error), and the bootstrap methods, which resample rows (including the partially observed ones) and refit by FIML on each replication. The complete-data closed forms (`"feldt"`, `"fisher"`, `"bonett"`, `"hakstian_whalen"`), `"adf"`, and `"likelihood"` are errors with `missing = "fiml"` rather than silently reverting to listwise deletion.

`missing = "two_stage"` estimates in two stages (Yuan & Bentler, 2000; Savalei & Bentler, 2009): the means and covariances of the items are first estimated by maximum likelihood under the saturated normal model (the EM estimates, with any `aux` variables entering this stage only), and the model is then fit to them by the complete data maximum likelihood discrepancy function. The analytic coefficient is a function of the first stage alone, which is the saturated maximum likelihood estimate FIML uses too, so its value and its intervals are those of `missing = "fiml"` computed from the same cases. Those are the cases FIML uses: every case with an observed item or, with `aux`, an observed auxiliary variable, since the first stage estimates the items and the auxiliaries jointly and a case observed on an auxiliary alone informs the item moments through it. Setting such a case aside would condition the analysis on an observed item and bias the estimates when the items went missing together at random given the auxiliaries. **lavaan**'s two-stage estimator reads such a case in its first stage but leaves it out of its sample size, which would make its standard errors too large; the model implied fits go through `fa_confirmatory`, which computes them again with every case the first stage read, and a message counts such cases (when a fit cannot be computed again, a warning says that the standard error is too large and that a bootstrap interval counts every case). The model implied coefficient comes from the tau-equivalent model fit in the second stage, with the normal theory two-stage standard error (Savalei & Bentler, 2009, Eq. 6) for `"ml"` and the one robust to nonnormality (Yuan & Bentler, 2000; Savalei & Falk, 2014) for `"mlr"`. The intervals FIML refuses are refused here too, `"likelihood"` also because no likelihood of the observed data is maximized. Multiple imputation is a different feature with a different interface and is out of scope here.

Both methods need the covariance of every pair of items and auxiliary variables, and the data carry no information about the covariance of a pair that no case observes together: lavaan only warns, and its EM returns an arbitrary value for it. Nor do they carry any about how two variables vary together when the cases that observe both hold a single value of one of them (or are one case): the estimate of their covariance then rests on the form of the normal model rather than on the data, or is not determined at all. Either pair is refused by name before any fit, with a pointer to `missing_coverage`, and a refusal of listwise deletion offers `missing = "fiml"` only when the data hold no such pair. In a bootstrap, a resample that leaves such a pair (or, for the BCa acceleration, a jackknife deletion that does) is set aside with the same message and counted, under either method.

Available confidence interval methods (set via `ci_method`). Some belong to one estimator only, because a closed-form interval for the sample coefficient and a model-based interval are not interchangeable; requesting a method the chosen estimator cannot supply is an error that names the estimator to use instead:

`"feldt"` *Analytic estimator only.* The F-distribution interval of Feldt (1965), exact under multivariate normality and parallel items.

`"fisher"` *Analytic estimator only.* Fisher's z' transformation (Fisher, 1950). Tends to overcover (Padilla, Divers, & Newton, 2012) and is generally not recommended.

`"bonett"` *Analytic estimator only.* Bonett's (2002) log transformation. The default for that estimator; well-behaved under normality.

`"hakstian_whalen"` *Analytic estimator only.* Cube-root transformation of Hakstian and Whalen (1976).

`"mlr"`, `"mlr_logistic"` *Model implied estimator only.* Wald interval using the robust (Satorra & Bentler, 1994) standard error from the fitted model. The default for that estimator; recommended among the closed forms when item distributions deviate from normality (Kelley & Pornprasertmanit, 2016). Requires raw data.

`"likelihood"` *Model implied estimator only.* Profile likelihood interval: the set of population values not rejected by the likelihood ratio test under the τ-equivalent model, located by refitting under the nonlinear constraint that the model implied reliability equals each candidate value. Respects [0, 1], is not forced to be symmetric, and works from raw data or covariance input. Requires **lavaan**. It is unavailable with the analytic estimator because the interval and the point estimate would then refer to different quantities: the interval profiles the model implied coefficient while the estimate is the sample coefficient, so under a misspecified model the interval can exclude the estimate it accompanies.

`"ml"`, `"ml_logistic"` Available to both estimators, by the route each affords. With `"analytic"` it is the closed-form ML standard error of van Zyl, Neudecker, and Nel (2000), computed directly from the covariance matrix; with `"model_implied"` it is the delta method standard error from the fitted model. The `_logistic` variant applies Browne's (1982) logit transformation.

`"adf"`, `"adf_logistic"` Available to both estimators. With `"analytic"` it is the asymptotic distribution-free standard error of Maydeu-Olivares, Coffman, and Hartmann (2007); with `"model_implied"` the model is fit by weighted least squares (Browne, 1984) and the delta method applied. Requires raw data and a relatively large sample size.

`"bootstrap_se"`, `"bootstrap_se_logistic"`, `"percentile"`, `"bca"` Available to both estimators. Nonparametric bootstrap intervals (Efron & Tibshirani, 1993): the rows of `data` are resampled with replacement `B` times and the chosen estimator is recomputed on each replication. `"percentile"` takes the interval limits from the empirical quantiles of the bootstrap

estimates (for a 95 percent interval, the 2.5th and 97.5th percentiles); it respects [0, 1] and is not forced to be symmetric about the estimate, but its coverage degrades when the estimator is biased or its variance changes with the parameter. `"bca"` (bias-corrected and accelerated) adjusts the two quantile positions for exactly those features, estimating the median bias from the bootstrap distribution and the acceleration from the jackknife, and is second-order accurate where the percentile interval is first-order accurate (DiCiccio & Efron, 1996). The acceleration is estimated as **boot** estimates it, by regression on the resampling frequencies when at least as many replications as cases returned a value and by the jackknife otherwise, or when that regression fails; a leave-one-out refit that fails is counted and set aside rather than stopping the call, and when the acceleration or the bias correction cannot be estimated at all the percentile interval is returned in its place, with a warning and the substitution recorded in the `ci_method` attribute; see `reliability_omega_categorical` for the full account. `"bootstrap_se"` uses the standard deviation of the bootstrap estimates as a standard error in an ordinary normal-theory interval; the `_logistic` variant builds that interval on the logit scale so the endpoints respect [0, 1], with the standard deviation carried to that scale by the delta method, divided by $\hat{\alpha}(1 - \hat{\alpha})$, as in Equation 32 of Kelley and Pornprasertmanit (2016) and in `MBESS::ci.reliability()`. When every replication returns the same value, as when every item is an exact linear function of the same variable (an item entered twice beside other items does not do it), the bootstrap distribution is a single point and every method returns that point at both limits, with a warning; so does the `_logistic` variant at a value of one (identical items), where the logit is infinite. Replications on which the estimator cannot be computed (for example, a model refit that does not converge) are set aside, and so is a replication in which an item takes a single value (a resample that has left out every case on which a rare item differs from the rest, or the jackknife deletion of such a case), which is refused as the data would be, by name. A replication follows the rule on the number of cases as the data do (see `missing`): a resample repeats rows, and one with no more distinct cases than the rule asks for is set aside, so near the rule's bound many are (at 12 cases and eight variables, about three resamples in four). When more than half are, no interval is reported (the standard error and the limits are `NA`): the replications left hold more distinct cases than a resample typically does, so their spread understates the sampling variability of the coefficient, and at the bound itself each holds every distinct case and, without repeated rows, gives the estimate. Without repeated rows that happens at fewer cases than about 1.65 times the count the rule compares them with (fewer than 14 for eight variables). The interval is computed from the replications that return a value, whose quantiles the percentile and BCa limits are; the counts, and why the rest were set aside, travel in the `bootstrap` attribute (see *Value*), print under the table, and are stated in a warning whenever a replication was set aside. Requires raw data and the **boot** package. The default `B = 10000` is an accuracy choice: the BCa adjustment pushes the working quantiles farther into the tails of the bootstrap distribution than the percentile interval uses, and stabilizing them takes more replications than the customary 2000; reduce `B` for exploration, not for a reported analysis. Bootstrap results vary from run to run; supply `seed` for reproducibility, and `ncpus` to spread the replications over several processes, which leaves the interval for a given seed unchanged. Under `estimator = "model_implied"` each replication refits the tau-equivalent model by maximum likelihood, whose estimate robust maximum likelihood shares, and a replication whose solution is improper is kept or set aside according to `improper`, as `reliability_omega` describes; an improper point estimate whose error variances sum below zero, so that the model implies a negative error variance for the composite, stops with a message rather than returning a coefficient above one. (The likelihood of the tau-equivalent model has a maximum: one item's error variance cannot run off below zero while

its loading grows, since the loadings are equal and every covariance would grow with them, so the stop of class `dmar_no_maximum` that `reliability_omega` describes does not arise here.)

`"none"` Return only the point estimate.

Two complete rows. Coefficient α needs only the trace and the sum of the covariance matrix, so it is computed from any two rows, but the covariance matrix of two rows has rank one: every pair of items correlates +1 or -1, whatever the items, and the estimate is a function of the difference between the two rows alone. Feldt's interval is exact for any *N* of at least 2 under multivariate normality and parallel items (for 0/1 items, as in `reliability_kr20`, it is an approximation; Feldt, 1965), and it is the interval the refusals below name. The Hakstian-Whalen interval is defined there too, but at *N* = 2 its upper limit lies above one for every data set whose alpha is below one at any confidence level above about .90, and is clamped to one (at an alpha of one, which two rows can give, the interval is $[1, 1]$). The rest are refused at *N* = 2, each for its own reason, as `"fisher"` is below *N* = 4: Bonett's variance, $2J/((J-1)(N-2))$, is infinite; the `"ml"` and `"adf"` standard errors (and their `_logistic` variants), which treat that matrix as the population's, are zero for any data (the asymptotic variance of van Zyl, Neudecker, and Nel, 2000, vanishes at a matrix of rank one, and each case's centered cross products are then proportional to the covariance matrix, which alpha does not change with); and every bootstrap resample is one row twice or both rows, so the bootstrap distribution has no spread. The model implied estimator needs more complete rows than items (see `missing`).

A *t* reference distribution for the Wald intervals. The intervals `"ml"`, `"mlr"`, and `"adf"`, with their `_logistic` variants, are Wald intervals: the estimate plus or minus a normal quantile times its standard error. The argument `wald_df` replaces the normal quantile with the *t* quantile on `wald_df` degrees of freedom and leaves the estimate and the standard error as they are (von Hippel, 2016, Eq. 29); for the `_logistic` variants the *t* quantile is applied on the logit scale. The default, `Inf`, is the normal reference, and a finite value with any other interval is an error. A number is the analyst's own choice of degrees of freedom, a sensitivity check; `reliability_omega` discusses it further.

The keyword `wald_df = "missing_information"`, which `reliability_omega` and `fa_confirmatory` accept, is an error here. The calibration study behind it (in the package's source repository, `methodology/missing_data`; `fa_confirmatory` describes its design) evaluated alpha's FIML interval (`missing = "fiml"`, `ci_method = "ml"`) with normal data, one factor models of six items (*N* from 30 to 200) or 12 (*N* = 50 and 100), and 15 or 30 percent of the values of half the items missing. On the normal reference that interval was already close to .95: over the cells with $N \leq 100$ and missing values its coverage was .0043 from .95 on average (between .940 and .958 in every cell). The *t* reference on von Hippel's $\max\{3, N(1-\gamma)-1\}$ degrees of freedom came barely closer (.0042), and in one cell its coverage reached .966, above the .96 that the study's decision rule, fixed before the run, allowed, so no rule passed. That cell, with 12 items, *N* = 50, and 30 percent missing completely at random, is the one where the study's fit of the saturated model converged in only 53 percent of the replications (in 100 replications of that design this function's fit, which starts at the EM estimates, converged in 91 and the study's in 49), so the coverage there conditions on the samples the study's fit could handle; without it the *t* reference would have passed (.0035 against .0042, at most .9595). The verdict is the rule's as fixed in advance, and alpha keeps the normal reference unless a number is given; the *t* reference would change its interval little in any case. On the logit scale none of von Hippel's four rules passed either. The printed table names the reference distribution in a line under it, and the `wald_df` attribute records it.

Comparison with other packages. The **psych** package provides `alpha`, which reports coefficient α along with item-level diagnostics (item-total correlations, alpha-if-item-deleted, and several alternative coefficients). The emphasis in **psych** is broad exploratory psychometric reporting.

reliability_alpha in **DMAR** differs in emphasis: it returns a single point estimate alongside a confidence interval drawn from the methods compared in Kelley and Pornprasertmanit (2016).

Value

A data.frame with columns term and value and rows "estimate" (sample coefficient α), "se" (standard error on the coefficient scale, NA for methods that do not produce one; for the transformation-based intervals "fisher", "bonett", and "hakstian_whalen" it is the delta method back-transform of the transformation-scale standard error, evaluated at the estimate), "se_transformed" (only for those transformation-based intervals: the standard error on the transformation scale, the quantity the interval is built from, with the scale named in the attribute se_transform_scale: "fisher_z", "log(1-alpha)", or "cube_root"), "ci_lower" and "ci_upper" (each clamped into [0, 1], so the limits stay in order), "conf_level", "N" (the cases the analysis used: the complete cases under listwise deletion, and under "fiml" and "two_stage" every case with at least one observed item or observed auxiliary variable), "N_complete" (the complete cases, so the cost of listwise deletion is visible at a glance; equal to "N" under listwise deletion and for covariance input), and "n_items" (number of items). The selected coefficient, CI method, and missing-data treatment travel as the attributes coefficient, ci_method, missing, wald_df (the degrees of freedom of the Wald interval's *t* reference, Inf for the normal one and for intervals that are not Wald intervals), and (when supplied) aux; bootstrap calls also record B. A bootstrap call also carries the attribute bootstrap, a list recording the replications the interval conditions on: attempted (B), returned (the replications whose value the interval uses; the standard error is their standard deviation and the percentile and BCa limits are their quantiles), failed (the difference), reasons (a named count of why replications were set aside: error when the refit raised an error, non_finite when it was set aside without one, as a fit that did not converge is, and as an improper one is under improper = "drop" even though its estimate is finite), error_messages (the distinct error messages with their counts, empty when none), undetermined (the replications set aside because the resample left a polychoric correlation undetermined, which only reliability_omega_categorical counts; zero here), improper and improper_kept (the replications whose tau-equivalent single-factor solution, under estimator = "model_implied", was improper, with a negative error variance: those set aside, which are among the non_finite ones, and those whose estimate the interval uses, which are among the returned ones), no_solution (zero here: it counts the replications of three items with no congeneric single-factor solution, which reliability_omega meets), improper_rule ("keep" or "drop"; see the argument improper), not_converged (zero here: a fit that did not converge is counted among the non_finite ones), too_few_cases (the replications set aside because the resample held too few distinct cases for the rule on the number of cases, which are among the error ones; when they number more than half of B, no interval is formed), caution (TRUE when the replications set aside and the improper ones kept number at least $(1 - C)/2$ of B, with C the confidence level, so that how they were treated can decide where a limit falls), require_half (TRUE: an interval requires at least half of all attempted replications to return defined estimates; this is an estimability safeguard, not a coverage guarantee), required (the fewest replications a request for an interval of this method and level may ask for, a count held to B and not to the replications that return a value: 400 for "percentile", 2000 for "bca", 100 for the two bootstrap standard error methods at the 95 percent level; see dmar_bootstrap), bootstrap_status ("complete", "dropped", "exploratory", "insufficient_count", or "insufficient_share"; under the last two the limits are NA), B_used (the smallest count any reported limit rests on), target_returned and target_withheld (counts and withholding flags for each target in a vector bootstrap), acceleration and acceleration_method (the BCa acceleration used and whether it came from the regression or the jackknife; NA unless a BCa interval was formed), jackknife

(the same attempted, returned, failed, reasons, error_messages, undetermined, improper, improper_kept, no_solution, not_converged, and too_few_cases for the leave-one-out refits behind the acceleration, all zero when no jackknife ran), ncpus (the number of processes used), t (the estimate of every replication, in order, NA where there is none; an improper estimate set aside under improper = "drop" is kept here, so the interval under the other rule can be formed without refitting), and status (a factor giving the fate of every replication, with levels "proper", "improper", "error", "non_finite", "undetermined", and "not_converged"). The interval uses the replications whose status is "proper" and, under improper = "keep", those whose status is "improper" and whose t is finite. The attribute B_used is the number of replications the interval uses (returned). The attributes n_dropped (B minus B_used), B_required, and bootstrap_status repeat the counts in the names every bootstrap result of the package carries. The warning has class dmar_bootstrap_replications_dropped when replications were set aside and the interval was formed, and dmar_bootstrap_insufficient when the interval was withheld. A BCa interval is also withheld, with the second class, when fewer than 10 replications lie beyond a limit where the bias correction and the acceleration place it (see dmar_bootstrap). The printed table shows these counts in one line, and a warning states them whenever a replication was set aside or the caution applies.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G. (2002). Sample size requirements for testing and estimating coefficient alpha. *Journal of Educational and Behavioral Statistics, 27*(4), 335–340. doi:10.3102/10769986027004335

Browne, M. W. (1982). Covariance structures. In D. M. Hawkins (Ed.), *Topics in applied multivariate analysis* (pp. 72–141). Cambridge, UK: Cambridge University Press.

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*, 62–83.

Collins, L. M., Schafer, J. L., & Kam, C.-M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods, 6*, 330–351. doi:10.1037/1082989X.6.4.330

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334.

DiCiccio, T. J., & Efron, B. (1996). Bootstrap confidence intervals. *Statistical Science, 11*(3), 189–228.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Feldt, L. S. (1965). The approximate sampling distribution of Kuder-Richardson reliability coefficient twenty. *Psychometrika, 30*, 357–370.

Fisher, R. A. (1950). *Statistical methods for research workers*. Edinburgh, UK: Oliver & Boyd.

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

Guttman, L. (1945). A basis for analyzing test-retest reliability. *Psychometrika, 10*(4), 255–282.

Hakstian, A. R., & Whalen, T. E. (1976). A *k*-sample significance test for independent alpha coefficients. *Psychometrika, 41*, 219–231.

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Kelley, K., & Cheng, Y. (2012). Estimation of and confidence interval formation for reliability coefficients of homogeneous measurement instruments. *Methodology, 8*, 39–50. doi:10.1027/1614-2241/a000036

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis. In A. von Eye & C. C. Clogg (Eds.), *Latent variables analysis: Applications for developmental research* (pp. 399–419). Thousand Oaks, CA: Sage.

Savalei, V., & Bentler, P. M. (2009). A two-stage approach to missing data: Theory and application to auxiliary variables. *Structural Equation Modeling, 16*, 477–497. doi:10.1080/10705510903008238

Savalei, V., & Falk, C. F. (2014). Robust two-stage approach outperforms robust full information maximum likelihood with incomplete nonnormal data. *Structural Equation Modeling, 21*(2), 280–302. doi:10.1080/10705511.2014.882692

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*, 165–200.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Maydeu-Olivares, A., Coffman, D. L., & Hartmann, W. M. (2007). Asymptotically distribution-free (ADF) interval estimation of coefficient alpha. *Psychological Methods, 12*, 157–176. doi:10.1037/1082989X.12.2.157

McDonald, R. P. (1999). *Test theory: A unified treatment*. Mahwah, NJ: Lawrence Erlbaum Associates.

Padilla, M. A., Divers, J., & Newton, M. (2012). Coefficient alpha bootstrap confidence interval under nonnormality. *Applied Psychological Measurement, 36*, 331–348. doi:10.1177/0146621612445470

Revelle, W., & Zinbarg, R. E. (2009). Coefficients alpha, beta, omega, and the GLB: Comments on Sijtsma. *Psychometrika, 74*, 145–154. doi:10.1007/s113360089102z

Sijtsma, K. (2009). On the use, the misuse, and the very limited usefulness of Cronbach's alpha. *Psychometrika, 74*, 107–120. doi:10.1007/s1133600891010

Terry, L. J., & Kelley, K. (2012). Sample size planning for composite reliability coefficients: Accuracy in parameter estimation via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 371–401. doi:10.1111/j.20448317.2011.02030.x

van Zyl, J. M., Neudecker, H., & Nel, D. G. (2000). On the distribution of the maximum likelihood estimator of Cronbach's alpha. *Psychometrika, 65*(3), 271–280. doi:10.1007/BF02296146

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

See Also

reliability (general wrapper that dispatches by coefficient), reliability_omega, reliability_omega_categorical, reliability_kr20, cfa_1 (single-factor CFA used internally), ss_aipe_reliability, alpha.

Other reliability: cohen_kappa(), diagnosis_agreement, fleiss_kappa(), icc(), reliability(), reliability_H(), reliability_kr20(), reliability_omega(), reliability_omega_categorical()

Examples

```
set.seed(113)
# Simulate six tau-equivalent items with population reliability ~ .8.
J <- 6
loadings <- rep(0.6, J)
eta <- rnorm(200)
errors <- matrix(rnorm(200 * J, sd = sqrt(1 - 0.6^2)), 200, J)
items <- outer(eta, loadings) + errors
colnames(items) <- paste0("y", seq_len(J))

# Default (Bonett's transformation) CI from raw data.
reliability_alpha(data = items)

# Same point estimate from a covariance matrix; CI requires N.
S <- cov(items)
reliability_alpha(S = S, N = 200, ci_method = "feldt")

# The bootstrap intervals resample the rows and recompute the
# coefficient on each replication. The percentile interval reads its
# limits off the empirical quantiles of the bootstrap estimates.
# B = 500 keeps the example quick (a 95 percent percentile interval
# is reported from 400 or more); a reported interval deserves the
# default B = 10000, and seed makes the interval reproducible.
reliability_alpha(data = items, ci_method = "percentile", B = 500,
                  seed = 113)

# The bias-corrected and accelerated interval adjusts those two
# quantile positions for median bias and for acceleration, and is the
# better choice for a reported interval. The default B = 10000 is an
# accuracy choice, not a formality, since BCa works farther into the
# tails of the bootstrap distribution than the percentile interval
# does (see Details). For the same reason B = 2000 is the fewest
# replications a 95 percent BCa interval is run with.
reliability_alpha(data = items, ci_method = "bca", B = 2000, seed = 113)

# An application that runs a long bootstrap (a Shiny app, for example)
# can follow it through a progress callback, which is called with the
# replications attempted so far and B, and can stop it by returning
# FALSE. The callback changes no result: this is the percentile
# interval above.
reliability_alpha(data = items, ci_method = "percentile", B = 500,
                  seed = 113, progress = function(done, total) {
                    if (done %% 250 == 0) message(done, " of ", total)
                  })
```

```
# Full information maximum likelihood with an auxiliary variable.
# Missingness on y2 depends on an auxiliary z (missing at random given
# z), so listwise deletion is biased and FIML with z is not. Supplying
# aux implies missing = "fiml"; the N_complete row shows how many
# rows listwise deletion would have kept.
z <- eta + rnorm(200, sd = 0.5)
d <- data.frame(items, z = z)
d$y2[runif(200) < plogis(-1 + 1.5 * as.numeric(scale(z)))] <- NA
reliability_alpha(data = d, aux = "z")
```

`reliability_H` *Coefficient* H *(Maximal Reliability) With a Confidence Interval*

Description

Estimates coefficient *H* (Hancock & Mueller, 2001), the reliability of the optimally weighted composite of the items that measure one factor, and returns a confidence interval for the population coefficient. *H* is the largest reliability that any weighted sum of the items can reach under the factor model, whereas McDonald's ω (`reliability_omega`) is the reliability of the unit-weighted sum that people usually score. The coefficient can be computed from a fitted **lavaan** model (`fit`, the recommended input), from raw item scores (`data`), from a covariance matrix and its sample size (`S` and `N`), or from the standardized loadings a paper reports (`loadings`). The interval comes from the delta method with the full covariance matrix of the parameter estimates, built on the logit scale by default, or from a nonparametric bootstrap when one is requested.

Usage

```
reliability_H(
  loadings = NULL,
  se_loadings = NULL,
  conf_level = 0.95,
  fit = NULL,
  factor = NULL,
  group = NULL,
  data = NULL,
  S = NULL,
  N = NULL,
  vcov_loadings = NULL,
 ci_method = c("mlr_logistic", "mlr", "ml_logistic", "ml", "delta_logistic", "delta",
    "bootstrap_se", "bootstrap_se_logistic", "percentile", "bca", "none"),
  missing = c("fiml", "listwise"),
  aux = NULL,
  B = 10000,
  seed = NULL,
  ncpus = 1L,
  progress = NULL
)
```

Arguments

loadings	Numeric vector of the standardized loadings of one factor, each in $(-1, 1)$, at least two. A fitted **lavaan** model supplied in this position is taken as fit, so reliability_H(fit) works.
se_loadings	Optional vector of the marginal standard errors of the standardized loadings, the same length as loadings. The interval it gives ignores the covariances among the loading estimates and warns that it does; prefer vcov_loadings or fit.
conf_level	Confidence level for the interval. Defaults to 0.95.
fit	A fitted **lavaan** model (from lavaan::cfa(), lavaan::sem(), or fa_confirmatory(output = "fit")), the recommended input; see *Details*.
factor	With fit, the name of the factor whose *H* is wanted; required when the model has more than one factor measured by observed items.
group	With a multiple group fit, the label of the group (as lavaan::lavInspect(fit, "group.label") gives it) whose *H* is wanted; required for such a fit.
data	A numeric matrix or data frame of item scores of one scale (rows are respondents, columns are items, three or more), and, when aux is supplied, the auxiliary columns. An item column that is not numeric (or logical), and an item that takes a single value in the rows analyzed, are refused by name before any fit, as is a zero variance on the diagonal of S: the single-factor model cannot be fit with such an item (lavaan would stop on it with an error that names no variable). SPSS user-defined missing values, which haven::read_sav(user_na = TRUE) keeps as values of a haven_labelled_spss column, are missing, as SPSS treats them.
S	A symmetric covariance matrix among the items (three or more); N is then required, and only "ml_logistic", "ml", and "none" apply.
N	Sample size. Required with S; optional with loadings, where it is recorded in the table; not used with fit or data, which count their own cases.
vcov_loadings	With loadings, the $J \times J$ covariance matrix of the standardized loading estimates, for the delta method interval with the full covariance matrix.
ci_method	Method for the confidence interval. "delta_logistic" and "delta" use the delta method with the covariance matrix that comes with the input (the fit's, or vcov_loadings); "mlr_logistic", "mlr", "ml_logistic", and "ml" name the standard error of the single-factor model fitted to data or S (robust maximum likelihood or maximum likelihood); "percentile", "bca", "bootstrap_se", and "bootstrap_se_logistic" are nonparametric bootstraps (with data, or with a single-group fit that carries its raw data); "none" returns the point estimate. The _logistic methods build the interval on the logit scale. When not supplied, the default is "mlr_logistic" with data, "ml_logistic" with S, and "delta_logistic" with fit and with loadings that come with vcov_loadings or se_loadings; for loadings without either it is "none", since nothing is then known about their sampling variability. A method that does not apply to the input stops with a message naming the ones that do.
missing	With data, how incomplete rows are handled: "fiml" (the default; full information maximum likelihood) or "listwise". The default analysis is FIML whenever an item value is missing and the complete data analysis otherwise (data in

which no value is missing, or none but in rows with every item missing, which both treatments drop, are fitted as they stand), as in reliability_omega and unlike fa_confirmatory; see *Details*. Under listwise deletion the single-factor model needs more complete rows than items (with no more, their covariance matrix is singular and the maximum likelihood fit is not defined), and the function stops before the fit when there are not. Ordinary FIML instead evaluates the constrained measurement model. A failure of the unrestricted H1 diagnostic does not alone invalidate its proper identified estimate or H0-based uncertainty; the result carries an explicit qualification. Supplied two-stage fits require valid unrestricted first-stage moments even for a point estimate. See cov_fiml for the sufficient affine-collapse diagnostic and its limits. Bootstrap resamples retain the family's separate distinct-case screen.

aux — With data, an optional character vector naming auxiliary variable columns of data, entered as saturated correlates under full information maximum likelihood; supplying it implies missing = "fiml".

B — Number of bootstrap replications when a bootstrap method is selected, a single whole number. Defaults to 10000, the reliability family's default; every replication refits a model (see *Details* for the cost). The fewest accepted depends on the interval method and the level: at the default conf_level, 400 for "percentile" (10 replications beyond each limit), 2000 for "bca" (50 beyond each nominal limit, because the adjusted limits sit farther into the tails), and 100 for "bootstrap_se" and "bootstrap_se_logistic" (a standard deviation of the replications); at 0.99 the first two are 2000 and 10000. A smaller B is refused before anything is computed (class dmar_count_invalid). The count is held to the number requested. Replications set aside afterward are counted and reported in a warning of class dmar_bootstrap_replications_dropped, which names the function and says to request more when those that return a value are fewer than the number required; when fewer than half of the B replications return a value, the interval is withheld (limits and bootstrap standard error NA) under a warning of class dmar_bootstrap_insufficient. See dmar_bootstrap, which also describes options(dmar.bootstrap_exploratory = TRUE) for a first look with a smaller B.

seed — Random number seed for the bootstrap. Defaults to NULL, under which the resamples are drawn from, and advance, the current random number stream; a supplied seed is set for the call, and the caller's RNG state is restored when it returns.

ncpus — The number of processes over which the bootstrap replications are spread. The default 1 runs them in this R session; a larger number uses forked workers (**boot**'s parallel = "multicore") or, on Windows, which cannot fork, a socket cluster ("snow"), whose workers load the installed **DMAR**. **boot** draws every resample before any replication runs, so a given seed returns the same interval, counts, and replicates whatever ncpus is. A session that forks workers and also starts background processes through **callr** or **processx** needs the environment variable PROCESSX_NOTIFY_OLD_SIGCHLD set before **processx** is loaded (see sem_exact_fit_bootstrap). A replication whose solution is improper is always set aside, since H is not defined for it. Ignored unless a bootstrap method is selected.

Bootstrap intervals from supplied two-stage fits currently refuse separately stored auxiliary variables: each refit must resample and carry those values with the items. Use an eligible delta interval where available, or explicitly resample and refit them together.

`progress` `NULL` (the default), or a function of two arguments that the bootstrap calls as it runs, so that an application can show how far a long bootstrap has gone and let its user stop it (a Shiny app, for example). It is called as `progress(done, total)`, with the number of replications attempted so far and `B`: after each replication when `ncpus = 1`, and with more processes after each block of replications they complete together, about a fiftieth of `B` (`ncpus * ceiling(B / (50 * ncpus))` replications, the last block possibly fewer), so that `done` advances by the block. A BCa interval takes its acceleration from a regression on the resampling frequencies when at least as many replications return a value as there are cases, and from the jackknife, one leave-one-out refit per case, when fewer do or when that regression fails or gives influence values that are not all finite; the jackknife runs at most once in a call, and not at all when no interval is formed or every returned replicate has the same value. While it runs, the callback is called again after each refit (with more processes, after each block of `ncpus * ceiling(n / (50 * ncpus))` refits, `n` the number of cases resampled), with `done` and `total` both `B`, so it can be called more than `B` times, and `done` equal to `total` does not mean that the bootstrap has finished. If the callback returns `FALSE`, the bootstrap stops before the next replication or block with an error of class `dmar_bootstrap_cancelled`, whose elements `done` and `total` say how far it got (see `dmar_metadata`); any other value is ignored. An error inside the callback stops the call with an error of class `dmar_progress_error`, whose message names the callback and which carries `done`, `total`, and, as `parent`, the callback's own condition with its class intact, so that an application can tell a failing callback from a failing analysis. The callback runs outside the random number stream, so a given `seed` returns the same result with or without it, and a callback that draws random numbers leaves the caller's stream as it found it. With `NULL` nothing is called. Ignored unless a bootstrap method is selected.

Details

The coefficient. For a factor f measured by items b that load on it alone, with unstandardized loadings λ, residual covariance matrix Θ_{bb}, and model implied factor variance ϕ_f,

$$q = \phi_f \, \lambda' \Theta_{bb}^{-1} \lambda, \qquad H = \frac{q}{1+q}.$$

With uncorrelated residuals and standardized loadings ℓ_j this is the form Hancock and Mueller (2001) gave,

$$H = \frac{\sum_j \ell_j^2/(1-\ell_j^2)}{1+\sum_j \ell_j^2/(1-\ell_j^2)},$$

in which each item contributes its ratio of common to unique variance, $\ell_j^2/(1-\ell_j^2)$. The weights that reach *H* are proportional to $\Theta_{bb}^{-1}\lambda$, so an item counts in proportion to its loading divided by its residual variance, and adding an item can never lower *H*. Gabler and Raykov (2017, Eq. 9) showed that the same expression with a nondiagonal Θ_{bb} is the maximal reliability when residuals inside

the block are correlated; a fitted model with residual covariances among the factor's items gets that coefficient here, and the printed table says how many entered the weights. Rescaling the factor rescales λ and ϕ_f inversely, so H does not depend on whether the factor was identified by a marker item or by fixing its variance.

Two further readings of the number are useful. H is the squared correlation between the factor and its best linear predictor from the block's items, the regression factor score computed from those items alone, so it is the reliability of that score: for a one-factor model the determinacy that `factor_scores` reports is the square root of H. And because H is the largest reliability any weighting of the items can reach under the fitted model, it is at least the model implied ω of the unit-weighted sum from the same fit (Raykov, 1997), with equality when the weights $\Theta_{bb}^{-1}\lambda$ are all equal, which with uncorrelated residuals means equal standardized loadings. Robust omega, whose denominator is the observed variance of the composite, need not lie below it.

Reading *H* with care. The sample H is a positively biased estimator of the population coefficient, and a composite formed with weights estimated from the sample is on average less reliable than the population H. Aguirre-Urreta, Rönkkö, and McIntosh (2019) found both effects pronounced in small samples (around $N = 100$), diminishing as the sample grows and barely noticeable beyond about 1,000, although with three items of modest loadings even 1,000 cases left more than 1% bias; the reliability estimate of the unit-weighted composite was nearly unbiased in the same conditions. Report ω, with its interval, for a scale that people will score by summing or averaging its items, and H as the ceiling that optimal weights could reach, or as the reliability of scores computed from the factor model. A bootstrap call estimates the bias of the sample H (the `bias` element of the `bootstrap` attribute), and the printed table shows it.

The four inputs. Exactly one of `fit`, `data`, `S`, and `loadings` is supplied.

`fit` A fitted **lavaan** model, from `lavaan::cfa()`, `lavaan::sem()`, or `fa_confirmatory(output = "fit")`; the recommended input, because it carries everything the coefficient and its interval need, and the model can be the one the analysis reports. Any factor measured by observed items can be named with `factor` (a factor of a confirmatory model, or an exogenous or endogenous factor of a structural model), and any group of a multiple group model with `group`. H is read from the fitted matrices, and its standard error comes from the fit's own covariance matrix of the estimates, so it is the standard, robust, or FIML standard error the model was fitted with. With ordered items fitted by WLSMV (either parameterization) H is the maximal reliability of the latent responses behind the categories, not of the observed categories; the `metric` attribute and the printed table say so. A loading or a regression the model holds at zero is not one, whether it is fixed (`0*item`) or constrained (`a*item` with `a == 0`, directly or through a chain of equality constraints between labels, as in `a == b` with `b == 0`): the model written with that term is the model written without it and gets the same H, so an item listed under the factor with its loading held at zero is not one of the factor's items, and a cross-loading held at zero refuses nothing. A fit is refused, with a message naming the remedy, when an item of the factor also loads on another factor or is regressed on another variable (for the factors of a bifactor model, `bifactor_indices` reports the index that Rodriguez, Reise, and Haviland, 2016, call H, which counts the other factors' common variance as uncorrelated error and is not this maximal reliability, and, in its `"maximal_reliability"` attribute, the maximal reliability the fitted model implies for each factor, and `ci_bifactor_indices` gives both a confidence interval), when the factor is measured by other factors, when an observed covariate predicts the factor, when the solution is improper (an item's residual variance that is not positive, for which H is not defined), and when a factor or a group must be named and is not.

`data` Raw scores on the items of one scale. The single-factor congeneric model is fitted exactly

as reliability_omega fits it, so H and ω rest on the same fit, the same cases, and the same kind of standard error, and missing and aux work as they do there. Three or more items are required, since the model is not identified with two.

S A covariance matrix among the items, with its sample size N; the same single-factor model, fitted by maximum likelihood.

loadings The standardized loadings of one factor, as a paper reports them, with the closed form above (which assumes uncorrelated residuals). For an interval, supply vcov_loadings, the covariance matrix of the standardized loading estimates (for example the matching block of lavaan::lavInspect(fit, "vcov.std.all")); se_loadings, their marginal standard errors, is accepted, but it ignores the covariances and warns that it does. Loadings alone carry nothing an interval can be formed from, so without either the estimate is returned with se and the limits NA, and the printed table says what an interval would need.

The interval. The delta method variance of $\hat{H}$ is $g'Vg$, where V is the covariance matrix of all the free parameter estimates of the fit and g is the gradient of H with respect to them, computed by central differences of the closed form above. V is used only when it can be, by the check the factor analysis tables (fa_confirmatory, fa_bifactor, fa_second_order) make before they report a standard error: when **lavaan** could not compute it, or computed it from an information matrix, observed or expected, that is singular in the directions the fit's equality constraints leave free (scaled to a unit diagonal, its smallest eigenvalue is below 10^{-6}), a parameter of the fit is not identified at the estimates and $g'Vg$ means nothing. The estimate is then returned with se and the limits NA, and with the warning, of class dmar_result_qualified, that those tables give for the same fit: the same words for a singular matrix, and for one **lavaan** could not compute a sentence that ends by naming the standard error and interval of H rather than the tables' columns. The check is of the whole fit, as the tables make it, so a factor whose own parameters are identified loses its interval beside one whose parameters are not. An interval from se_loadings uses only the marginal standard errors of the standardized loadings, the diagonal of V. The loading estimates of a fitted factor model are correlated, and the diagonal can understate or overstate the variance: in seven blocks of three to six tests from holzinger_swineford, each fitted as one factor, the standard error from the full matrix ranged from about 0.7 to 1.3 times the one from the diagonal. By default the Wald interval is built on the logit scale and transformed back (Browne, 1982): the limits of a monotone transformation of a parameter are the transformation of its limits, the confidence interval transformation principle that Kelley (2007) sets out, so the interval stays inside (0, 1) and is asymmetric near the upper bound, as the sampling distribution of a high reliability is; "delta", "ml", and "mlr" give the untransformed Wald interval. The default standard error depends on the input: with data, the robust maximum likelihood standard error (Satorra-Bentler, or Yuan-Bentler under FIML), as for ω, because Kelley and Pornprasertmanit (2016) found the normal theory Wald interval wanting when items depart from normality; with S, the maximum likelihood standard error; and with fit, the fit's own. With fit, "ml" and "mlr" (and their _logistic forms) are accepted when they name the standard error the fit already carries, and otherwise stop with a message saying which one it carries.

The delta method interval is the fast default. For a reported analysis, the percentile or BCa bootstrap with the model refitted on every replication (by FIML when data are missing) is the safer choice, as it is for ω (Kelley & Pornprasertmanit, 2016), and it also estimates the bias of the sample H. With data the rows are resampled and the single-factor model refitted; with fit the rows the model was fitted to are resampled and the model itself refitted through lavaan::lavaan(), with the fitted estimates as starting values (for a single-group fit that carries its raw data). Before any resample is drawn, the model is refitted this way to the rows of fit itself, and the call stops unless that refit reproduces the estimate, so the replications are refits of the model the estimate describes. A fit with

sampling weights is refused, since a resample of its rows does not carry them and every replication would be an unweighted fit, and so is a fit with a `cluster` variable, whose cases are not independent and cannot be resampled one at a time; the delta method interval, built from the fit's own covariance matrix of the estimates, serves both. The bootstrap engine is the one the whole reliability family uses (Efron & Tibshirani, 1993): replications that fail to converge, give an improper solution (or, with three items, a resample whose correlations admit no single-factor solution), or raise an error are set aside and counted (a resample in which an item takes a single value is refused before the refit, by name, as the data would be; **lavaan** would stop on the item after printing its table of the variables; so is a resample with too few distinct cases, by the rule of `missing` with `data`, and with `fit` by the rule of the fit's own treatment of incomplete rows: under listwise deletion, or with complete data, one with no more distinct rows than the model's continuous variables, whose sample covariance matrix is singular, and under FIML or two-stage estimation one with no more distinct cases that observe a variable, while a fit by pairwise deletion is held to neither), and the counts travel in the `bootstrap` attribute, print under the table, and are stated in a warning. A replication whose likelihood has no maximum (an error variance running off below zero while the likelihood is flat or still rising; see `fa_bifactor`) is counted among those that raised an error, and a fit with no maximum behind the estimate itself stops with an error of class `dmar_no_maximum`, which with `data` prints the **lavaan** code of the single-factor fit, where `bounds = "pos.var"` or a different model are the choices. *H* is not defined for an improper solution, so the attribute records `improper_rule = "drop"`. Each replication is a model fit, a few hundredths of a second for a one-factor model on complete data and longer under FIML, so the default `B = 10000` (the family's default, chosen for the stability of the BCa quantiles) takes minutes. The examples therefore stop at the delta method; the bootstrap call for their five verbal tests is `reliability_H(data = holzinger_swineford[, verbal], ci_method = "bca", B = 2000, seed = 113)`.

Missing data. With `data`, incomplete items are fitted by default by full information maximum likelihood (FIML), which keeps every case with at least one observed item or observed auxiliary variable, with auxiliary variables (`aux`) entered as saturated correlates (Graham, 2003), and `missing = "listwise"` fits the complete rows alone; data in which no value is missing, and `S`, are fitted as they stand. A row with every item missing does not count as incomplete data, since without auxiliary variables both treatments drop it. The default analysis is therefore FIML whenever an item value is missing and the complete data analysis otherwise, unlike `fa_confirmatory`, which fits complete data by FIML too. That is the rule of `reliability_omega`, whose complete-data intervals need the complete data analysis, and it is followed here so that the two fit the same single-factor model the same way; on incomplete data it agrees with `fa_confirmatory`. With `fit`, the fit's own treatment applies: a model fitted with `missing = "ml"` gives *H* and its standard error by FIML. Either way `N` counts the cases the fit used and `N_complete` those complete on the factor's items, so the cost of listwise deletion is visible. Under `missing = "fiml"` a pair of items or auxiliary variables that no case observes together is refused by name before any fit, since the data carry no information about its covariance (lavaan only warns, and its EM returns an arbitrary value for it), and so is a pair the cases observe together only where one of them takes a single value, since those cases carry none about how the two vary together; in a bootstrap, a resample that leaves such a pair is set aside with the same message and counted.

From standardized loadings. With `loadings` alone the function returns the estimate, with `se` and the limits `NA`, because standardized loadings carry nothing an interval can be formed from (the default `ci_method` is then `"none"`). With `vcov_loadings` the interval is the delta method interval from the full covariance matrix of the loading estimates. With `se_loadings` it is the delta method interval from their marginal standard errors, and the call warns every time, with class `dmar_H_marginal_se`, that the interval ignores the covariances among the loading estimates. Every

input returns the reliability family table described under *Value*; the sampling variance of H is the square of the row `se`. `reliability(type = "H")` dispatches here.

The coefficient that `fa_confirmatory` reports as `H_<factor>` is the same quantity, with **lavaan**'s delta method standard error and an untransformed interval; `reliability_H(fit, factor = "speed", ci_method = "delta")` reproduces its estimate and standard error for the factor `speed`.

Value

The reliability family's table (class `c("dmar_reliability", "dmar_tbl", "data.frame")`), with columns `term` and `value` and rows `"estimate"` (H), `"se"` (its standard error: delta method, or the standard deviation of the bootstrap replicates; `NA` for `ci_method = "none"`), `"ci_lower"` and `"ci_upper"`, `"conf_level"`, `"N"` (the cases the fit used, in the named group of a multiple group fit; `NA` for `loadings` without `N`), `"N_complete"` (the cases complete on the factor's items), and `"n_items"` (the number of items). Attributes record the computation: `coefficient` (`"H"`), `ci_method` (the method used), `missing` (`"listwise"`, `"fiml"`, the fit's own label such as `"pairwise"`, or `"unknown"` for `loadings`), `vcov_basis` (`"full"` when the delta method used the full covariance matrix of the estimates, `"diagonal"` when only marginal standard errors were supplied, `"none"` when no delta method standard error was formed), `se_type` (the **lavaan** standard error type of the fit; `NA` for `loadings`), `factor` and `group` (with `fit`; `NA` otherwise), `metric` (`"observed"`, `"latent_response"` for ordered items, or `"unknown"` for `loadings`), `residual_covariances` (the number of residual covariances inside the block that entered the weights), and `loadings` (the standardized loadings of the block, named by item). With `loadings`, `vcov_supplied` records what came with them, whether or not an interval was formed: `"full"` for `vcov_loadings`, `"diagonal"` for `se_loadings`, and `"none"` for neither, in which case the printed table says what an interval would need. With `data`, `aux` is recorded when supplied. A bootstrap call also records `B` and the family's `bootstrap` attribute (the replications attempted, returned, and set aside, with the reasons; see `reliability_omega`), which here also carries `bias`, the mean of the returned replicates minus the estimate (`NA` when no interval is formed). Its `no_solution` counts the resamples of three items whose correlations admit no single-factor solution, which are among the `improper` ones (on the data path; zero with `fit`, whose refits are not checked for it), and `improper_kept` is zero, since H is not defined for an improper solution.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Aguirre-Urreta, M. I., Rönkkö, M., & McIntosh, C. N. (2019). A cautionary note on the finite sample behavior of maximal reliability. *Psychological Methods, 24*(2), 236–252. doi:10.1037/met0000176

Browne, M. W. (1982). Covariance structures. In D. M. Hawkins (Ed.), *Topics in applied multivariate analysis* (pp. 72–141). Cambridge, UK: Cambridge University Press.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Gabler, S., & Raykov, T. (2017). Evaluation of maximal reliability for unidimensional measuring instruments with correlated errors. *Structural Equation Modeling, 24*(1), 104–111. doi:10.1080/10705511.2016.1159916

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), *Structural equation modeling: Present and future* (pp. 195–216). Scientific Software International.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Raykov, T. (1997). Estimation of composite reliability for congeneric measures. *Applied Psychological Measurement, 21*(2), 173–184. doi:10.1177/01466216970212006

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016). Evaluating bifactor models: Calculating and interpreting statistical indices. *Psychological Methods, 21*(2), 137–150. doi:10.1037/met0000045

See Also

`reliability_omega` for the reliability of the unit-weighted composite, `reliability` for the family's general entry point, `factor_scores` for the scores whose reliability H is, `fa_confirmatory` for the fit, and `bifactor_indices` for the bifactor index of Rodriguez, Reise, and Haviland (2016) and the maximal reliability of each factor of a bifactor model.

Other reliability: `cohen_kappa()`, `diagnosis_agreement`, `fleiss_kappa()`, `icc()`, `reliability()`, `reliability_alpha()`, `reliability_kr20()`, `reliability_omega()`, `reliability_omega_categorical()`

Examples

```
verbal <- c("t5_general_information", "t6_paragraph_comprehension",
            "t7_sentence", "t8_word_classification", "t9_word_meaning")

# From raw item scores: the single-factor model is fitted here, and the
# interval uses its robust maximum likelihood standard error on the
# logit scale.
reliability_H(data = holzinger_swineford[, verbal])

# The omega of the unit-weighted sum from the same model, for
# comparison; H is the ceiling that optimal weights could reach.
reliability_omega(data = holzinger_swineford[, verbal],
                  denominator = "model_implied")

# From a fitted model with several factors, name the factor. The
# standard error is the fit's own, from the full covariance matrix of
# its estimates (requires lavaan).
fit <- lavaan::cfa(
  "visual  =~ t1_visual_perception + t2_cubes + t4_lozenges
   textual =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning
   speed   =~ t10_addition + t12_counting_groups_of_dots +
              t13_straight_and_curved_capitals",
  data = holzinger_swineford, std.lv = TRUE)
h <- reliability_H(fit, factor = "textual")
```

```
h
attr(h, "loadings")

# From the standardized loadings a paper reports, without an interval.
reliability_H(loadings = c(0.6, 0.7, 0.8), ci_method = "none")

# The broom verbs.
generics::tidy(h)
generics::glance(h)
```

`reliability_kr20` *Kuder-Richardson Formula 20 (KR-20) With a Confidence Interval*

Description

Estimates Kuder-Richardson formula 20 (Kuder & Richardson, 1937) for a homogeneous composite scored on dichotomous (0/1) items, and returns a confidence interval for the population coefficient.

Usage

```
reliability_kr20(
  data,
 ci_method = c("feldt", "bonett", "fisher", "hakstian_whalen", "ml", "ml_logistic",
  "adf", "adf_logistic", "bootstrap_se", "bootstrap_se_logistic", "percentile", "bca",
    "none"),
  conf_level = 0.95,
  B = 10000,
  seed = NULL,
  ncpus = 1L,
  progress = NULL
)
```

Arguments

`data`	A numeric matrix or data frame of 0/1 item scores (rows are respondents, columns are items). Rows with any missing values are listwise-deleted. Non-binary values trigger an error, and so do fewer than two complete rows, from which no variance can be computed. A column that is not numeric, and an item that takes one value in every complete row (one no respondent passes, or every respondent does), are refused by name: such an item adds a constant to the total score, whose reliability it leaves as it is, while KR-20 would count it among the J items, at $J(J-2)/(J-1)^2$ times the KR-20 of the other items, the coefficient of the same total. So is a total score that takes one value in every complete row (on two rows, as many items passed in one row as in the other), whose reliability is not defined. For incomplete data, `reliability_alpha` with `missing = "fiml"` computes the same coefficient (alpha of 0/1 items is KR-20) from the incomplete rows as well. SPSS user-defined missing values, which

haven::read_sav(user_na = TRUE) keeps as values of a haven_labelled_spss column, are missing, as SPSS treats them.

ci_method — Method for constructing the confidence interval; see reliability_alpha for the full list. Defaults to "feldt".

conf_level — Confidence level for the interval (1 - Type I error rate). Defaults to 0.95.

B — Number of bootstrap replications when a bootstrap method is selected, a single whole number. Defaults to 10000. The fewest accepted depends on the interval method and the level: at the default conf_level, 400 for "percentile" (10 replications beyond each limit), 2000 for "bca" (50 beyond each nominal limit, because the adjusted limits sit farther into the tails), and 100 for "bootstrap_se" and "bootstrap_se_logistic" (a standard deviation of the replications); at 0.99 the first two are 2000 and 10000. A smaller B is refused before anything is computed (class dmar_count_invalid). The count is held to the number requested. Replications set aside afterward are counted and reported in a warning of class dmar_bootstrap_replications_dropped, which names the function and says to request more when those that return a value are fewer than the number required; when fewer than half of the B replications return a value, the interval is withheld (limits and bootstrap standard error NA) under a warning of class dmar_bootstrap_insufficient. See dmar_bootstrap, which also describes options(dmar.bootstrap_exploratory = TRUE) for a first look with a smaller B.

seed — Random number seed used for bootstrap reproducibility. Defaults to NULL, under which the resamples are drawn from, and advance, the current random number stream; supply an integer for a reproducible interval, which is set for the call only, the caller's random number state being restored when it returns.

ncpus — The number of processes over which the bootstrap replications are spread. The default 1 runs them in this R session; a larger number uses forked workers (**boot**'s parallel = "multicore") or, on Windows, which cannot fork, a socket cluster ("snow"), whose workers load the installed **DMAR**. **boot** draws every resample before any replication runs, so a given seed returns the same interval, counts, and replicates whatever ncpus is. A session that forks workers and also starts background processes through **callr** or **processx** needs the environment variable PROCESSX_NOTIFY_OLD_SIGCHLD set before **processx** is loaded (see sem_exact_fit_bootstrap). Ignored unless a bootstrap method is selected.

progress — NULL (the default), or a function of two arguments that the bootstrap calls as it runs, so that an application can show how far a long bootstrap has gone and let its user stop it (a Shiny app, for example). It is called as progress(done, total), with the number of replications attempted so far and B: after each replication when ncpus = 1, and with more processes after each block of replications they complete together, about a fiftieth of B (ncpus * ceiling(B / (50 * ncpus)) replications, the last block possibly fewer), so that done advances by the block. A BCa interval takes its acceleration from a regression on the resampling frequencies when at least as many replications return a value as there are cases, and from the jackknife, one leave-one-out refit per case, when fewer do or when that regression fails or gives influence values that are not all finite; the jackknife runs at most once in a call, and not at all when no interval is formed or every returned replicate has the same value. While it runs, the callback is

called again after each refit (with more processes, after each block of `ncpus * ceiling(n / (50 * ncpus))` refits, `n` the number of cases resampled), with `done` and `total` both `B`, so it can be called more than `B` times, and `done` equal to `total` does not mean that the bootstrap has finished. If the callback returns `FALSE`, the bootstrap stops before the next replication or block with an error of class `dmar_bootstrap_cancelled`, whose elements `done` and `total` say how far it got (see `dmar_metadata`); any other value is ignored. An error inside the callback stops the call with an error of class `dmar_progress_error`, whose message names the callback and which carries `done`, `total`, and, as `parent`, the callback's own condition with its class intact, so that an application can tell a failing callback from a failing analysis. The callback runs outside the random number stream, so a given `seed` returns the same result with or without it, and a callback that draws random numbers leaves the caller's stream as it found it. With `NULL` nothing is called. Ignored unless a bootstrap method is selected.

Details

Kuder and Richardson's (1937) formula 20 for a J-item composite of binary items is

$$KR_{20} = \frac{J}{J-1}\left(1 - \frac{\sum_j p_j q_j}{s_Y^2}\right),$$

where p_j is the proportion of respondents endorsing item *j* (i.e., scoring 1), $q_j = 1 - p_j$, and s_Y^2 is the variance of the composite score. For dichotomous items $p_j q_j$ is the item variance, so KR-20 is algebraically identical to coefficient α (Guttman, 1945; Cronbach, 1951) computed from the item covariance matrix. KR-20 predates coefficient α by 14 years and Feldt's (1965) F-distribution interval was derived specifically for the sampling distribution of KR-20.

Because KR-20 is a special case of coefficient α (Guttman, 1945; Cronbach, 1951), the same considerations apply: KR-20 equals the population reliability of the composite under essential τ-equivalence (i.e., equal factor loadings) and serves as a lower bound otherwise. For dichotomous items see also `reliability_omega_categorical` (categorical omega), which treats the items via a probit-link single-factor model and does not assume equal loadings.

KR-20, like α, is at most one but can fall below zero, when the item covariances are negative on balance; on dichotomous items that can arise when an item is scored in the opposite direction to the others (a reversed key). A reliability coefficient cannot be negative, so a negative estimate is returned as computed with a warning that says so, and each confidence limit is clamped into [0, 1]; the interval around an estimate far enough below zero is then [0, 0], which says that the data support no positive reliability for the items as keyed.

Available confidence interval methods (set via `ci_method`) are the same as for `reliability_alpha`; see that function's *Details* for full descriptions. The default for KR-20 is `"feldt"`, the F-distribution interval originally developed for KR-20.

From two complete rows KR-20 is computed with Feldt's interval, which is defined for any *N* of at least 2 (every item must differ between the two rows, since an item they answer alike does not vary, and the two totals must differ, since a total that does not vary has no reliability). Feldt's *F* reference is exact under multivariate normality and parallel items, which 0/1 items cannot meet, and for them it is the approximation Feldt (1965) derived. The intervals that are not defined there (`"bonett"`, `"fisher"`, `"ml"`, `"adf"`, their `_logistic` variants, and the bootstrap) are refused, as

reliability_alpha explains, and "hakstian_whalen", which is defined, has an upper limit of one there whatever the data, at any confidence level above about .90.

The menu also includes the nonparametric bootstrap intervals "percentile", "bca", "bootstrap_se", and "bootstrap_se_logistic", which resample the rows of data with replacement B times and recompute KR-20 on each replication (Efron & Tibshirani, 1993). They are worth the cost when the closed forms are least trustworthy, which for dichotomous items means highly unbalanced item difficulties or a sample size too small for the normal-theory derivations behind "feldt" and "bonett". No bootstrap runs unless ci_method asks for one; when it does, the default is B = 10000 replications, and supplying seed makes the interval reproducible. Replications on which KR-20 cannot be computed are set aside and counted, among them a replication in which an item takes a single value, refused by name as the data would be: a resample leaves out a given respondent about 37 percent of the time ($(1 - 1/N)^N$, near e^{-1}), so an item passed by one respondent loses its variation in about a third of the resamples. The interval is computed from the replications that return a value; the counts travel in the bootstrap attribute described under reliability_alpha, print under the table, and are stated in a warning whenever a replication was set aside.

Comparison with other packages. The **psych** package computes the same quantity via alpha (since α on 0/1 data *is* KR-20). reliability_kr20 restricts input to raw 0/1 data so the dichotomous-items assumption cannot be quietly violated, presents the historical formula in the documentation, and accompanies the point estimate with a confidence interval drawn from the methods compared in Kelley and Pornprasertmanit (2016).

Value

A data.frame with columns term and value and rows "estimate" (sample KR-20), "se" (standard error on the coefficient scale, NA for methods that do not produce one; for the transformation-based intervals "fisher", "bonett", and "hakstian_whalen" it is the delta method back-transform of the transformation-scale standard error), "se_transformed" (only for those transformation-based intervals: the standard error on the transformation scale, with the scale named in the attribute se_transform_scale: "fisher_z", "log(1-alpha)", or "cube_root"), "ci_lower" and "ci_upper" (each clamped into [0, 1], so the limits stay in order), "conf_level", "N" (effective sample size after listwise deletion), "N_complete" (the complete cases; equal to "N" here, and carried so the whole reliability family returns one shape), and "n_items" (number of items). Attributes coefficient ("kr20") and ci_method record the computation; bootstrap calls also record B, the replication counts in the bootstrap attribute (see reliability_alpha; KR-20 fits no model, so no replication is improper), and the number of replications the interval uses in the attribute B_used, with n_dropped, B_required, and bootstrap_status (see dmar_bootstrap).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Feldt, L. S. (1965). The approximate sampling distribution of Kuder-Richardson reliability coefficient twenty. *Psychometrika, 30*, 357–370.

Guttman, L. (1945). A basis for analyzing test-retest reliability. *Psychometrika, 10*(4), 255–282.

Kelley, K., & Cheng, Y. (2012). Estimation of and confidence interval formation for reliability coefficients of homogeneous measurement instruments. *Methodology, 8*, 39–50. doi:10.1027/1614-2241/a000036

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Kuder, G. F., & Richardson, M. W. (1937). The theory of the estimation of test reliability. *Psychometrika, 2*, 151–160.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Terry, L. J., & Kelley, K. (2012). Sample size planning for composite reliability coefficients: Accuracy in parameter estimation via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 371–401. doi:10.1111/j.20448317.2011.02030.x

See Also

`reliability` (general wrapper), `reliability_alpha`, `reliability_omega_categorical`, `alpha`.

Other reliability: `cohen_kappa()`, `diagnosis_agreement`, `fleiss_kappa()`, `icc()`, `reliability()`, `reliability_H()`, `reliability_alpha()`, `reliability_omega()`, `reliability_omega_categorical()`

Examples

```
set.seed(113)
# Ten dichotomous items with a single underlying ability.
N <- 300
J <- 10
ability <- rnorm(N)
loadings <- rep(0.6, J)
latent <- outer(ability, loadings) +
          matrix(rnorm(N * J, sd = sqrt(1 - 0.6^2)), N, J)
items <- (latent > 0) * 1
colnames(items) <- paste0("y", seq_len(J))

reliability_kr20(data = items)
reliability_kr20(data = items, ci_method = "bonett")

# A bootstrap interval recomputes KR-20 on each of B resamples of the
# rows. B = 500 keeps the example quick; a reported interval deserves
# the default B = 10000, and seed makes the interval reproducible.
reliability_kr20(data = items, ci_method = "percentile", B = 500,
                 seed = 113)
```

`reliability_omega` *Coefficient Omega (McDonald) With a Confidence Interval*

Description

Estimates McDonald's (1999) coefficient ω for a homogeneous composite score from a single-factor confirmatory factor analysis model and returns a confidence interval for the population coefficient. The `denominator` argument selects whether the total variance in the denominator of ω is estimated directly from the data (`"observed"`, the default: robust omega) or taken from the fitted model (`"model_implied"`); see *Details* for the properties of each choice. A bootstrap confidence interval is never run unless requested; see `ci_method`.

Usage

```
reliability_omega(
  data = NULL,
  S = NULL,
  N = NULL,
  ci_method = c("mlr_logistic", "ml_logistic", "mlr", "ml", "likelihood", "adf",
   "adf_logistic", "feldt", "fisher", "bonett", "hakstian_whalen", "bootstrap_se",
    "bootstrap_se_logistic", "percentile", "bca", "none"),
  denominator = c("observed", "model_implied"),
  missing = c("fiml", "listwise", "two_stage"),
  aux = NULL,
  conf_level = 0.95,
  B = 10000,
  seed = NULL,
  ncpus = 1L,
  improper = c("keep", "drop"),
  wald_df = Inf,
  progress = NULL
)
```

Arguments

`data` A numeric matrix or data frame of item scores (rows are respondents, columns are items). Either `data` or `S` must be supplied. How incomplete rows are handled is governed by `missing`: full information maximum likelihood by default, listwise deletion with `missing = "listwise"`, or two-stage maximum likelihood with `missing = "two_stage"`. When `aux` is supplied, `data` contains both the item columns and the auxiliary columns, and the columns not named in `aux` are the items. An item column that is not numeric (or logical), and an item that takes a single value in the rows analyzed (the complete rows under listwise deletion, the rows that observe it otherwise), are refused by name before anything is computed, as is a zero variance on the diagonal of `S`: such an item adds a constant to the composite, whose reliability it leaves as it is, while the coefficient would count it among the J items (alpha with it is $J(J-2)/(J-1)^2$

times alpha without it). A composite that does not vary is refused too, as when the items are ipsative (their scores sum to the same total in every row): its reliability is not defined. Under `missing = "fiml"` and `"two_stage"` it is checked in the rows that observe every item when they outnumber the items, where a composite that does not vary leaves the likelihood of the saturated model without a maximum, and otherwise on the EM estimate of the covariance matrix. In `holzinger_swineford`, `t1_visual_perception`, `t2_cubes`, and `t4_lozenges`, each centered on the pupil's mean of the three, are ipsative items of this kind. SPSS user-defined missing values, which `haven::read_sav(user_na = TRUE)` keeps as values of a `haven_labelled_spss` column, are missing, as SPSS treats them.

`S` A symmetric covariance matrix among the items. If supplied, `N` must also be supplied; methods that require raw data (`"mlr*"`, `"adf*"`, `"bootstrap_*"`, `"percentile"`, `"bca"`) are then unavailable, as are `missing = "fiml"` and `missing = "two_stage"` named explicitly, and `aux`, since a covariance matrix has no incomplete cases (the default of `missing` analyzes it as it stands).

`N` Total sample size; required when `S` is supplied.

`ci_method` Method for constructing the confidence interval. See *Details*. When not supplied (or passed as `NULL`): for `denominator = "model_implied"` the default is `"mlr_logistic"` with raw data, the robust maximum likelihood standard error with the limits formed on the logit scale and transformed back (Browne, 1982; Kelley, 2007a), whatever the treatment of missing values, except under `wald_df = "missing_information"`, where it is `"mlr"`, the same standard error with the limits formed on the scale of omega, the interval that rule is calibrated for; with covariance input, which the robust standard error cannot use, it is `"ml_logistic"`, the maximum likelihood standard error with the limits formed the same way (`"ml"` under `wald_df = "missing_information"`); for `denominator = "observed"` (robust omega) no interval is computed by default, because its interval is bootstrap based and a bootstrap is never run unless requested. Ask for `"percentile"` or `"bca"` to obtain the recommended interval. The closed forms, the Wald intervals, and the profile likelihood apply to `denominator = "model_implied"` only; a combination that does not apply stops with a message naming the denominator that does (see *Details*).

`denominator` How the total variance in the denominator of ω is estimated: `"observed"` (default; robust omega) uses the variance of the composite estimated directly from the data; `"model_implied"` uses the total variance reproduced by the fitted single-factor model. See *Details*. With `"observed"`, `ci_method` must be a bootstrap method or `"none"`.

`missing` How incomplete rows of `data` are handled: `"fiml"` (the default; full information maximum likelihood, using every case with at least one observed item or observed auxiliary variable), `"listwise"` (complete-case analysis), or `"two_stage"` (two-stage maximum likelihood, using the same cases as FIML). The default analysis is FIML whenever an item value is missing and the complete data analysis otherwise, unlike `fa_confirmatory`, which fits complete data by FIML too: data in which no value is missing, data complete but for rows with every item missing (which both treatments drop), and `S` are analyzed as they stand, since the complete-data intervals need that analysis; incomplete items are analyzed by FIML, or, when `ci_method` names an interval FIML cannot give, the

call stops and names the choices. See *Details*. With `missing = "listwise"`, when fewer than two rows are complete, listwise deletion leaves nothing to analyze, and the function stops before any fit, says so, and names the treatment that uses the incomplete rows. The single-factor model is fit by maximum likelihood, which needs the covariance matrix positive definite, and N complete rows give a matrix of rank at most $N - 1$, so with no more complete rows than items the function stops the same way; `reliability_alpha`, which needs only the sums of the matrix, is computed from two or more complete rows. Full information maximum likelihood does not impose an ordinary complete-case count rule. Observed-denominator omega and two-stage estimation require unrestricted moments, checked for affine collapse or numerical instability as described in `cov_fiml`. EM convergence alone does not establish existence or uniqueness. Model-implied omega can retain a proper identified constrained FIML fit and H0-based uncertainty while withholding unavailable H1 diagnostics and residual correlations. The result states this qualification. Bootstrap resamples retain a distinct-case estimability screen as a separate policy, not a universal likelihood-existence theorem. Listwise complete rows are counted as distinct rows. Two columns of one name are refused before any of this, under every treatment, since the items are read by name.

`aux` Optional character vector naming auxiliary variable columns of `data`: entered as saturated correlates under full information maximum likelihood, and into the first stage under `missing = "two_stage"`. Supplying `aux` without `missing` implies `missing = "fiml"`. See *Details*.

`conf_level` Confidence level for the interval. Defaults to `0.95`.

`B` Number of bootstrap replications when a bootstrap method is selected, a single whole number. Defaults to `10000`. The fewest accepted depends on the interval method and the level: at the default `conf_level`, 400 for `"percentile"` (10 replications beyond each limit), 2000 for `"bca"` (50 beyond each nominal limit, because the adjusted limits sit farther into the tails), and 100 for `"bootstrap_se"` and `"bootstrap_se_logistic"` (a standard deviation of the replications); at 0.99 the first two are 2000 and 10000. A smaller `B` is refused before anything is computed (class `dmar_count_invalid`). The count is held to the number requested. Replications set aside afterward are counted and reported in a warning of class `dmar_bootstrap_replications_dropped`, which names the function and says to request more when those that return a value are fewer than the number required; when fewer than half of the `B` replications return a value, the interval is withheld (limits and bootstrap standard error `NA`) under a warning of class `dmar_bootstrap_insufficient`. See `dmar_bootstrap`, which also describes `options(dmar.bootstrap_exploratory = TRUE)` for a first look with a smaller `B`.

`seed` Random number seed used for bootstrap reproducibility. Defaults to `NULL`, under which the resamples are drawn from, and advance, the current random number stream; supply an integer for a reproducible interval, which is set for the call only, the caller's random number state being restored when it returns.

`ncpus` The number of processes over which the bootstrap replications are spread. The default `1` runs them in this R session; a larger number uses forked workers (**boot**'s `parallel = "multicore"`) or, on Windows, which cannot fork, a socket

cluster (`"snow"`), whose workers load the installed **DMAR**. **boot** draws every resample before any replication runs, so a given `seed` returns the same interval, counts, and replicates whatever `ncpus` is. A session that forks workers and also starts background processes through **callr** or **processx** needs the environment variable `PROCESSX_NOTIFY_OLD_SIGCHLD` set before **processx** is loaded (see `sem_exact_fit_bootstrap`). Ignored unless a bootstrap method is selected.

`improper` How a bootstrap replication whose single-factor solution is improper (a negative error variance, a Heywood case) is treated when the solution still yields a coefficient: `"keep"` (the default) uses its estimate like any other, and `"drop"` sets it aside. Either way such replications are counted in the `bootstrap` attribute and the printed line, and a replication whose solution yields no coefficient is set aside under both (see *Details*). With `"drop"` an improper point estimate stops the call, since an interval built from proper replications only would not describe it. Ignored unless a bootstrap method is selected.

`wald_df` The reference distribution of a Wald interval: `Inf` (the default), the normal distribution; a single positive number, the degrees of freedom of a t distribution; or `"missing_information"`, von Hippel's (2016) degrees of freedom $\max\{3, N(1-\gamma)-1\}$ from omega's fraction of missing information γ ($N-1$ with complete data). A number applies to the Wald intervals of the model implied coefficient (`"ml"`, `"mlr"`, `"adf"`, and their `_logistic` variants), the keyword to `"ml"` and `"mlr"` only, without two-stage estimation, the intervals its calibration covers; anything else is an error that says why. See *Details*.

`progress` `NULL` (the default), or a function of two arguments that the bootstrap calls as it runs, so that an application can show how far a long bootstrap has gone and let its user stop it (a Shiny app, for example). It is called as `progress(done, total)`, with the number of replications attempted so far and `B`: after each replication when `ncpus = 1`, and with more processes after each block of replications they complete together, about a fiftieth of `B` (`ncpus * ceiling(B / (50 * ncpus))` replications, the last block possibly fewer), so that `done` advances by the block. A BCa interval takes its acceleration from a regression on the resampling frequencies when at least as many replications return a value as there are cases, and from the jackknife, one leave-one-out refit per case, when fewer do or when that regression fails or gives influence values that are not all finite; the jackknife runs at most once in a call, and not at all when no interval is formed or every returned replicate has the same value. While it runs, the callback is called again after each refit (with more processes, after each block of `ncpus * ceiling(n / (50 * ncpus))` refits, `n` the number of cases resampled), with `done` and `total` both `B`, so it can be called more than `B` times, and `done` equal to `total` does not mean that the bootstrap has finished. If the callback returns `FALSE`, the bootstrap stops before the next replication or block with an error of class `dmar_bootstrap_cancelled`, whose elements `done` and `total` say how far it got (see `dmar_metadata`); any other value is ignored. An error inside the callback stops the call with an error of class `dmar_progress_error`, whose message names the callback and which carries `done`, `total`, and, as `parent`, the callback's own condition with its class intact, so that an application can tell a failing callback from a failing analysis. The callback runs outside the random number stream, so a given `seed` returns the same result with or without it, and a callback that draws random numbers leaves the caller's stream as it found it.

With NULL nothing is called. Ignored unless a bootstrap method is selected.

Details

Coefficient ω is a population coefficient of determination. Write the composite $Y = \sum_j X_j$ through its measurement decomposition, $Y = E[Y \mid \eta] + e$, the regression of the observed composite on the latent variable η it is intended to measure. By the law of total variance, the proportion of composite variance attributable to η is $\mathrm{Var}(E[Y \mid \eta])/\mathrm{Var}(Y)$ (McDonald, 1999, 2011). Under the congeneric single-factor model, in which item j has factor loading λ_j and error variance ψ_j^2, the numerator equals $\left(\sum_j \lambda_j\right)^2$ and the population coefficient is

$$\omega = \frac{\left(\sum_j \lambda_j\right)^2}{\sigma_Y^2},$$

with σ_Y^2 the variance of the composite. Coefficient ω relaxes the equal-loadings assumption underlying coefficient α (Guttman, 1945; Cronbach, 1951), so on congeneric scales ω estimates population reliability whereas α underestimates it.

The denominator argument. The sample estimate substitutes **lavaan** estimates of the loadings into the numerator; the two `denominator` settings differ in how σ_Y^2 is estimated.

`"observed"` **(default)** The variance of the composite estimated directly from the data (the sum of all elements of the unrestricted covariance matrix, on the same maximum likelihood divisor as the fitted loadings). This is the coefficient that Kelley and Pornprasertmanit (2016) call hierarchical omega (`MBESS::ci.reliability(type = "hierarchical")`); the DMAR documentation refers to it as *robust omega*, and it is the default because its denominator estimates the variance of the composite consistently whether or not the single-factor model is correctly specified.

`"model_implied"` The total variance reproduced by the fitted single-factor model, $\left(\sum_j \hat{\lambda}_j\right)^2 + \sum_j \hat{\psi}_j^2$. This is the textbook form of $\hat{\omega}$, correct exactly when the one-factor model reproduces the composite variance.

The choice matters only insofar as the single-factor model is misspecified, and the properties of each setting can be stated exactly.

- If the single-factor model is correctly specified, the two definitions coincide in the population, and both estimators converge to the same value. Nothing is given up, in that sense, by either choice.
- The observed total variance is a consistent estimator of $\mathrm{Var}(Y)$ whether or not the single-factor model is correctly specified; the model implied total variance is consistent for $\mathrm{Var}(Y)$ only when the model is correct.
- Under misspecification (for example, a minor unmodeled factor or correlated errors), only `"observed"` retains the coefficient of determination interpretation: the proportion of the variance of the composite users actually compute that is attributable to the fitted common factor. With `"model_implied"` the estimate becomes a ratio of two model derived quantities whose denominator is no longer the variance of any composite a user scores.

- Under a correctly specified model with normal items, the model implied denominator uses the model structure and can be a slightly more efficient estimator of σ^2_Y in finite samples. In the simulations of Kelley and Pornprasertmanit (2016) the practical differences between the two coefficients were negligible when the model held, while interval coverage under modest model error favored the observed denominator paired with a bootstrap interval.

Those simulation results are why Kelley and Pornprasertmanit (2016) recommend the observed denominator with a bootstrap confidence interval when unidimensionality is only approximate, which is common with real items.

Stated as one identity, the two denominators differ by $\mathbf{1}'(\mathbf{S} - \hat{\Sigma})\mathbf{1}$, the sum of every residual covariance of the single-factor fit. The model implied setting treats covariance the model omits as absent from the composite; the observed setting counts it as variance of the composite that the factor does not explain. The first describes the composite as the fitted model describes it, the second the composite users compute. Estimating the factor model from its own implied moments and reporting the reliability of a scored composite with the observed variance are therefore not in tension: the model is needed for the numerator, which partitions the variance of the composite into the part due to the factor and the rest, and is not needed for the denominator, which is the variance of the composite itself. The omega that `cfa_1` reports beside the loadings, and that `fa_confirmatory` reports for continuous items, is the model implied coefficient, a property of the fitted model.

Two cautions frame the choice. First, no denominator repairs a misspecified measurement model: the fitted loadings absorb part of whatever structure the single-factor model omits, so the numerator is affected under either setting. Assess the single-factor model (for example with `cfa_1`) before interpreting any ω variant, and model real multidimensionality directly rather than patching over it. A gap between the two denominators shows that the model does not reproduce the composite variance, but a small gap does not show that the loadings are free of such structure: a doublet or a method factor shared by a few same-signed items inflates their loadings while leaving the two denominators close. The residual correlations of the single-factor fit are where that structure usually shows, so the result carries them (attribute `residual_cor`; see *Value*) and the printed table names the largest. One limit applies: when the shared structure sits on the items that dominate the factor, the single factor absorbs it and the misfit appears as residual correlations among the other items, so the largest residual signals local dependence more reliably than it locates it. Second, the naming history is worth knowing. The observed denominator coefficient was introduced as hierarchical omega (Kelley & Pornprasertmanit, 2016; **MBESS** type `"hierarchical"`), a name motivated by a hierarchical factor logic: model misfit is viewed as a set of minor common factors (visible as residual correlations), the single factor is retained as an approximation, and the coefficient isolates the variance attributable to the general factor alone, expressed relative to the observed variance of the unweighted composite. It is not the bifactor coefficient ω_H of Zinbarg, Revelle, Yovel, and Li (2005), whose numerator comes from the general factor loadings of an explicitly multidimensional model. Upon reflection, the authors would have named the coefficient for its behavior rather than for the hierarchical motivation, as observed omega or robust omega; **DMAR** uses *robust omega*, with the qualifications that word requires. The robustness is to misspecification of the *total variance* only, since the numerator remains model based under either denominator; it is not the outlier robustness of Zhang and Yuan (2016), and it is separate from the robust maximum likelihood standard errors available through `ci_method`. Robust omega also shares a design principle with categorical omega as Kelley and Pornprasertmanit (2016) computed it: in both, the total variance in the denominator is not taken from the fitted factor model. Green and Yang (2009) built the denominator of their categorical coefficient from the model implied polychoric correlations instead, and `reliability_omega_categorical` offers both through its own `denominator` argument. The

reliability vignette develops this framing.

Available confidence interval methods (set via `ci_method`):

`"ml"`, `"ml_logistic"` Wald interval using the maximum likelihood standard error from **lavaan** (Raykov, 2002). The `_logistic` variant applies Browne's (1982) logit transformation (see below).

`"mlr"`, `"mlr_logistic"` Wald interval using the robust (Satorra & Bentler, 1994) standard error, recommended among closed-form methods when item distributions deviate from normality (Kelley & Pornprasertmanit, 2016). `"mlr_logistic"` is the default for `denominator = "model_implied"`.

`"likelihood"` Profile likelihood interval: the set of population values not rejected by the likelihood ratio test, located by refitting the model under the nonlinear constraint that the model implied reliability equals each candidate value. It respects the [0, 1] range and is not forced to be symmetric about the estimate. Maximum likelihood; available with raw data or covariance input, for `denominator = "model_implied"` only.

`"adf"`, `"adf_logistic"` Wald interval using weighted least squares ("ADF") estimation (Browne, 1984). Requires raw data and a relatively large sample size.

`"feldt"`, `"fisher"`, `"bonett"`, `"hakstian_whalen"` Closed-form intervals derived for coefficient α. They apply mechanically to ω but their coverage performance for ω is generally inferior to the maximum likelihood and bootstrap intervals (Kelley & Pornprasertmanit, 2016).

`"bootstrap_se"`, `"bootstrap_se_logistic"`, `"percentile"`, `"bca"` Nonparametric bootstrap intervals (Efron & Tibshirani, 1993): the rows of `data` are resampled with replacement `B` times and ω is recomputed, with the factor model refit, on each replication. `"percentile"` takes the interval limits from the empirical quantiles of the bootstrap estimates; it respects [0, 1] and is not forced to be symmetric about the estimate. `"bca"` (bias-corrected and accelerated) adjusts the two quantile positions for median bias, estimated from the bootstrap distribution, and for the rate at which the estimator's variance changes with the parameter, the acceleration, estimated by the jackknife; those two adjustments make it second-order accurate where the percentile interval is first-order accurate (DiCiccio & Efron, 1996). The acceleration is estimated as **boot** estimates it, by regression on the resampling frequencies when at least as many replications as cases returned a value and by the jackknife otherwise, or when that regression fails; a leave-one-out refit that fails is counted and set aside rather than stopping the call, and when the acceleration or the bias correction cannot be estimated at all the percentile interval is returned in its place, with a warning and the substitution recorded in the `ci_method` attribute; see `reliability_omega_categorical` for the full account. `"bootstrap_se"` uses the standard deviation of the bootstrap estimates as a standard error in a normal-theory interval, built on the logit scale for the `_logistic` variant so the endpoints respect [0, 1] (the standard deviation is carried to the logit scale by the delta method, divided by $\hat{\omega}(1 - \hat{\omega})$, as in Equation 32 of Kelley and Pornprasertmanit, 2016, and in `MBESS::ci.reliability()`). When every replication returns the same value, the bootstrap distribution is a single point and every method returns that point at both limits, with a warning (items that are exact linear functions of one variable, which give alpha such replications, do not reach the bootstrap here: their covariance matrix is singular, and the single-factor fit behind the estimate stops on it). Replications whose model refit does not converge are set aside, and so is a replication in which an item takes a single value (a resample that has left out every case on which a rare item differs from the rest), which is refused before the refit, by name, as the data would be, and so is one with no more distinct cases than the rule on the number of cases for the data asks for (see `missing`;

a resample repeats rows, so near the rule's bound many are, and when more than half are no interval is reported, since those left hold more distinct cases than a resample typically does and their spread understates the sampling variability, and at the bound each holds every distinct case; without repeated rows that happens at fewer cases than about 1.65 times the count the rule compares them with); the interval is computed from the replications that return a value, whose quantiles the percentile and BCa limits are; the counts, and why the rest were set aside, travel in the `bootstrap` attribute (see *Value*), print under the table, and are stated in a warning whenever a replication was set aside. The default `B = 10000` is an accuracy choice: the BCa adjustment pushes the working quantiles farther into the tails of the bootstrap distribution than the percentile interval uses, and stabilizing them takes more replications than the customary 2000; reduce `B` for exploration, not for a reported analysis. Recommended when assumptions of parametric methods are questionable. Each replication refits the single-factor model by maximum likelihood: robust maximum likelihood has the same estimates and differs only in the standard errors and the test statistic, which a replication does not use, so the robust fit is not repeated `B` times. Requires raw data and the **boot** package; supply `seed` for run-to-run reproducibility, and `ncpus` to spread the replications over several processes, which leaves the interval for a given seed unchanged. How replications with an improper solution are treated is described next.

`"none"` Return only the point estimate.

Which intervals apply to which denominator.

`denominator = "model_implied"` Every method on the menu: the Wald intervals `"ml"`, `"mlr"`, and `"adf"` with their `_logistic` variants, the profile `"likelihood"` interval, the alpha-derived closed forms `"feldt"`, `"fisher"`, `"bonett"`, and `"hakstian_whalen"`, the four bootstrap methods, and `"none"`.

`denominator = "observed"` **(robust omega, the default)** The four bootstrap methods (`"percentile"`, `"bca"`, `"bootstrap_se"`, `"bootstrap_se_logistic"`) and `"none"` only. The delta method standard errors, the profile likelihood, and the alpha-derived closed forms are derived under the model implied ratio and do not account for sampling of the observed denominator, so `"mlr_logistic"`, the default interval of the model implied coefficient, is not an interval for robust omega.

A request for a combination that does not apply, such as `ci_method = "mlr"` with the default denominator, stops with a message naming the denominator that does; nothing falls back silently. Covariance input (`S`) and the incomplete data methods (`missing = "fiml"` and `"two_stage"`) narrow the menu further, as the `S` argument and the paragraphs on missing data below describe, and each stops the same way. The pairing of robust omega with the bootstrap is not a limitation in practice, since the bootstrap is the interval Kelley and Pornprasertmanit (2016) recommend for the observed denominator coefficient in any case. Because no analysis in **DMAR** runs a bootstrap unless the user requests one, the default for robust omega is the point estimate with no interval, accompanied by a message naming the call that produces the recommended interval; request `ci_method = "percentile"` or `"bca"` to obtain it.

Limits on the scale of omega or on the logit scale. The Wald intervals come in two forms, both built from the same standard error. `"ml"`, `"mlr"`, and `"adf"` take the estimate plus or minus a normal quantile times the standard error, an interval symmetric about the estimate whose upper limit can pass one (the limits are then clamped into [0, 1]; see *Value*). Their `_logistic` variants form the limits on the logit scale, $\log\{\hat{\omega}/(1-\hat{\omega})\}$, whose standard error the delta method gives as $\mathrm{SE}/\{\hat{\omega}(1-\hat{\omega})\}$, and transform them back (Browne, 1982): the limits of a monotone transformation

of a parameter are the transformation of its limits, the confidence interval transformation principle that Kelley (2007a) sets out. The interval then lies within (0, 1) and is asymmetric near one, as the sampling distribution of a high reliability is; the `se` row is the standard error of $\hat{\omega}$ itself under either form. The default is `"mlr_logistic"` on the evidence of a Monte Carlo study (in the package's source repository, `methodology/reliability_limits`) of factors of three to six items with *N* from 50 to 500. The study formed both kinds of limits from the same standard error, the normal theory maximum likelihood one for normal data and the Huber-White one of MLR for nonnormal data, so what it compared is the scale on which the limits are formed. With normal data, under listwise deletion and FIML alike, the logit limits covered from .939 to .956 in every cell (a mean absolute departure from .95 of .003), where the limits on the scale of omega covered from .910 to .952 (.009 to .010), their misses falling mostly on one side, with the population value below the lower limit. With nonnormal data fitted by MLR both forms covered below .95, the logit form by less (a mean absolute departure of .013 against .022). No cell used the robust.sem standard error that `"mlr_logistic"` takes with complete data; it estimates the same asymptotic variance as the Huber-White one, which the default takes under FIML. `fa_confirmatory` forms the limits of its omega row the same way, and `reliability_H` those of *H* by default, so the three give limits of one form. With a covariance matrix, which the robust standard error cannot use, the default is `"ml_logistic"`, as in `reliability_H`. Under `wald_df = "missing_information"` the default is `"mlr"` (`"ml"` with a covariance matrix), since that rule is calibrated for the intervals on the scale of omega only (see below).

A *t* reference distribution for the Wald intervals. The Wald intervals (`"ml"`, `"mlr"`, `"adf"`, and their `_logistic` variants) are the estimate plus or minus a normal quantile times its standard error. The normal quantile treats the standard error as known, which it is only in the limit; in small samples, and more so when values are missing, such intervals are too short. Von Hippel (2016, pp. 431–433) showed this for maximum likelihood estimates from incomplete data and proposed keeping the estimate and its standard error while replacing the normal quantile with a *t* quantile (his Eq. 29). The argument `wald_df` does that: the interval becomes $\hat{\omega} \pm t_{1-\alpha/2,\nu}\mathrm{SE}$ with ν the value of `wald_df`, and for the `_logistic` variants the *t* quantile is applied on the logit scale before the limits are transformed back. The default, `Inf`, is the normal reference. A number is the analyst's own choice of degrees of freedom, best used to see how much a plausible range of them would lengthen the interval; with a few hundred cases the answer is very little (on 200 degrees of freedom the 95% quantile is 1.972 rather than 1.960).

The calibration study described next selected its simulated data sets by an eligibility rule that keeps some draws the affine-collapse diagnostic sets aside, so its results are conditional on that rule, and coverage under the diagnostic's selection has not been established by that study. A raw-data eligibility replay alone cannot validate the FMI refits and information diagnostics. The keyword is an approximate, opt-in rule, not a coverage guarantee.

The keyword `wald_df = "missing_information"` takes the degrees of freedom from omega's own fraction of missing information γ by von Hippel's (2016, Eqs. 31 and 34) effective sample size rule with $k = 1$,

$$\nu = \max\{3, N(1-\gamma) - 1\},$$

where N is the number of cases the fit used and γ is estimated from the fit as `fraction_missing_information` estimates it; with complete data (listwise deletion, or a covariance matrix) $\gamma = 0$ and $\nu = N - 1$. The rule comes from a calibration study of these intervals under full information maximum likelihood (in the package's source repository, `methodology/missing_data`; `fa_confirmatory` describes its design), which simulated the model implied omega of one factor models with six items (*N* from 30 to 200) or 12 (*N* = 50 and 100), normal data fit by ML and nonnormal data by MLR.

Over the cells with $N \leq 100$ and missing values the coverage of the *t* interval was .0180 from .95 on average, against .0242 for the normal one, and no higher than .9550 in any cell, with normal theory standard errors (`"ml"`) and Huber-White ones (`"mlr"`) alike; with nonnormal data both intervals still covered below .95, the *t* one less so. On the logit scale the rule did not pass: with normal data the `_logistic` interval on the normal reference was already within .0044 of .95 on average, and the *t* reference raised its coverage as high as .9645, above the .96 the study allowed. The keyword therefore applies to `ci_method = "ml"` and `"mlr"` only (with `ci_method` not supplied, the interval is `"mlr"`, or `"ml"` with a covariance matrix), under listwise deletion or `missing = "fiml"`, and is an error with the `_logistic` variants, with `"adf"` and `"adf_logistic"` (the study evaluated maximum likelihood standard errors only), with two-stage estimation (for which no fraction of missing information is computed), and with every interval that is not a Wald interval. The study fit no auxiliary variables: with `aux` the rule takes omega's fraction of missing information from the fit that carries them, which is smaller when they predict the missing values, and the study supports that case by analogy rather than directly. Under listwise deletion the standard errors here come from the expected information (`"ml"`, from a covariance matrix too) and the robust.sem sandwich (`"mlr"`, which needs raw data), where the study's cells without missing values used the observed information and the Huber-White sandwich; for omega at $N = 30$ and 50 they are shorter by less than 1 and by 1 to 2 percent, so the complete data intervals are, if anything, slightly shorter than those the study evaluated. The printed table names the reference distribution in a line under it, under the keyword with *N* and the FMI; the `wald_df` attribute records the degrees of freedom, and under the keyword the `wald_df_table` attribute holds the `df_missing` table they came from.

Missing data and auxiliary variables. By default (`missing = "fiml"`) incomplete data are analyzed by full information maximum likelihood (FIML), which keeps every case with at least one observed item or observed auxiliary variable and fits the single-factor model to all of them, consistent and efficient under the missing at random (MAR) assumption; `missing = "listwise"` fits the complete rows alone. Under the default, the model implied omega here and the omega `fa_confirmatory` reports for the same items agree on incomplete data, the estimate and, with `ci_method = "ml_logistic"`, the interval; the default interval, `"mlr_logistic"`, forms its limits the same way from the robust standard error. Data in which no value is missing, and a covariance matrix, are analyzed as they stand: FIML would give the same estimate, and the standard errors and intervals come from the expected information (and for `"mlr"` the robust.sem sandwich) where FIML takes the observed information (and the Huber-White sandwich). A row with every item missing does not count as incomplete data: without auxiliary variables, listwise deletion and FIML both drop it, so data that are complete but for such rows (a blank line at the end of a spreadsheet, say) are analyzed as they stand too. The default analysis is therefore FIML whenever an item value is missing and the complete data analysis otherwise. `fa_confirmatory` differs: it fits complete data by FIML as well. The complete data analysis is used here because the closed-form intervals (`"feldt"`, `"fisher"`, `"bonett"`, `"hakstian_whalen"`), `"adf"`, `"adf_logistic"`, and `"likelihood"` need it, FIML giving none of them. On complete data the standard error of omega from `fa_confirmatory` therefore differs slightly from the one here: by 0.4 percent for `t1_visual_perception`, `t2_cubes`, `t3_paper_form_board`, and `t4_lozenges` in `holzinger_swineford`. When the default meets incomplete items and one of those intervals, the call stops rather than deleting cases unasked, and names the two choices: `missing = "listwise"`, which computes the interval from the complete rows alone, or an interval FIML gives (`"ml"`, `"mlr"`, their `_logistic` variants, or a bootstrap method). The `aux` argument names auxiliary variables: columns of `data` that are not part of the composite but are correlated with the items or with the reasons values are missing. They enter as *saturated correlates* (Graham, 2003): correlated freely with each other and with every item's residual, never loading on the factor, so the measurement model is undisturbed while FIML uses their

information; a good auxiliary also makes MAR itself more plausible (Collins, Schafer, & Kam, 2001). A case whose items are all missing is kept when an auxiliary variable is observed on it: the model estimates the items and the auxiliaries jointly, so such a case informs the item moments through its auxiliaries, and setting it aside would make the analysis conditional on having an observed item, which biases the estimates when the items go missing together at random given the auxiliaries. Supplying `aux` without `missing` implies `missing = "fiml"`, whether or not a value is missing; combining it with an explicit `missing = "listwise"` is an error. Under `missing = "fiml"` the denominators are estimated as follows: with `"model_implied"` the fitted FIML model supplies the total variance directly, and with `"observed"` the total variance is the sum of the FIML estimate of the item covariance matrix (the EM estimate under a saturated model over the items and any auxiliaries, the matrix `cov_fiml` returns), which is the estimate of $\mathrm{Var}(Y)$ that uses the partially observed rows; it is already on the maximum likelihood divisor, matching the fitted loadings. The available intervals under `"fiml"` are `"ml"`, `"mlr"` (Yuan & Bentler, 2000, robust), their `_logistic` variants, and the bootstrap methods, which resample rows including the partially observed ones and refit by FIML; the complete-data closed forms, `"adf"`, and `"likelihood"` are errors rather than silent fallbacks to listwise deletion.

`missing = "two_stage"` estimates in two stages instead (Yuan & Bentler, 2000; Savalei & Bentler, 2009): the means and covariances of the items are first estimated by maximum likelihood under the saturated normal model (the EM estimates, the same saturated estimate FIML's observed denominator uses), and the single-factor model is then fit to them by the complete data maximum likelihood discrepancy function. The auxiliary variables named in `aux` enter the first stage only, where they inform the estimated moments, and the measurement model is fit as if they were absent, which is simpler than the saturated correlates FIML needs. Two-stage estimation uses the cases FIML uses, every case with an observed item or an observed auxiliary variable: the first stage estimates the items and the auxiliaries jointly, so a case observed on an auxiliary alone informs the item moments through it, and setting such a case aside would condition the analysis on an observed item and bias the estimates when the items went missing together at random given the auxiliaries. **lavaan**'s two-stage estimator reads such a case in its first stage but leaves it out of its sample size, which would make its standard errors too large; the fits go through `fa_confirmatory`, which computes them again with every case the first stage read, and a message counts such cases (when a fit cannot be computed again, a warning says that the standard error is too large and that a bootstrap interval counts every case). The estimates are consistent under MAR and only marginally less efficient than FIML's (Savalei & Bentler, 2009), and with incomplete nonnormal data Savalei and Falk (2014) found the coverage of robust two-stage intervals better than that of robust FIML intervals. With `"model_implied"` the `"ml"` interval uses the normal theory two-stage standard error (Savalei & Bentler, 2009, Eq. 6) and `"mlr"` the standard error robust to nonnormality (Yuan & Bentler, 2000; lavaan's `missing = "robust.two.stage"`); with `"observed"` the numerator comes from the two-stage loadings and the denominator from the EM covariance matrix, so the coefficient differs from its FIML counterpart only through the loadings, and its interval is a bootstrap that refits both stages on every replication. A delta method interval for that coefficient can be derived under two-stage estimation, but its behavior has not been studied, so it is not offered. With three items the single-factor model is saturated and two-stage estimation gives the FIML estimate and standard error. The intervals that FIML refuses are refused here too, and `"likelihood"` for a further reason: no likelihood of the observed data is maximized. Multiple imputation is a different feature with a different interface and is out of scope here.

Both methods need the covariance of every pair of items and auxiliary variables, and the data carry no information about the covariance of a pair that no case observes together: lavaan only warns, and its EM returns an arbitrary value for it. Nor do they carry any about how two variables vary

together when the cases that observe both hold a single value of one of them (or are one case): the estimate of their covariance then rests on the form of the normal model rather than on the data, or is not determined at all. Either pair is refused by name before any fit, with a pointer to `missing_coverage`, and a refusal of listwise deletion offers `missing = "fiml"` only when the data hold no such pair. In a bootstrap, a resample that leaves such a pair (or, for the BCa acceleration, a jackknife deletion that does) is set aside with the same message and counted, under either method.

Improper solutions. A single-factor fit can converge to an improper solution, one with a negative error variance (a Heywood case). McDonald (2004), following van Driel (1978), separates improper estimates that reflect sampling error around a proper structure, typically a negative error variance near zero in a small sample or for an item that nearly duplicates the factor, from those that reflect a misspecified or empirically underidentified model: the first kind needs no change of model, while the second calls for a different model rather than an estimate. One sample cannot always tell the two apart, so the function draws the line at what the coefficient can mean. When the error variances still sum to a nonnegative number, the model implies a nonnegative error variance for the composite and a model implied ω between zero and one, so the coefficient is returned with a warning. When they sum to less than zero, the model implies a negative error variance for the composite and a model implied ω above one, which no reliability coefficient can take; sampling error near a zero error variance does not produce this unless every item is nearly error free, and it is the signature of a solution that has run away, one error variance falling without bound as its loading grows. The function then stops with a message naming the item. So it does for three items whose correlations have a negative product: a single factor implies that the product of the three correlations equals the square of the product of the three loadings, so no real loadings reproduce them and the maximum likelihood estimate does not exist. Whatever the sum, a fit whose likelihood has no maximum, an error variance running off below zero while the likelihood is flat or still rising (checked by fitting the model again with the variance held further out, as `fa_bifactor` describes), stops the function with an error of class `dmar_no_maximum` that names the item and prints the **lavaan** code of the single-factor fit, where `bounds = "pos.var"` or a different model are the choices.

A bootstrap meets improper solutions more often than the sample does, since a resample can push an error variance that is near zero in the sample below it, which is a standardized loading past one (the fits are on the items' own metric, where a loading above one is routine). The argument `improper` governs the replications whose solution is improper but still yields a coefficient: `"keep"`, the default, uses their estimates like any other, and `"drop"` sets them aside. Keeping is the default because setting them aside conditions the bootstrap distribution on admissibility and so trims its upper tail, the replications in which an error variance fell below zero; Yuan and Hayashi (2003) make the parallel argument for test statistics, counting a replication that fails because its resample lies far from the model rather than discarding it. A replication whose solution yields no coefficient is set aside under either setting. When its error variances sum below zero, the solution has run away and implies an ω above one, beyond the top of the distribution like the improper replications kept. When three items have no single-factor solution, the side depends on the scale the resample is read on: on the estimator's own scale it continues the top of the distribution, since as a resample approaches that boundary one loading, and ω with it, grows without bound, while its correlations, one of which is negative or zero, point to low reliability. The argument of Yuan and Hayashi does not settle it, since their extreme value belongs to a statistic that measures misfit and so has a direction, so such a replication is counted, on its own as well as among the improper ones, rather than placed on either side. A replication whose likelihood has no maximum is set aside as well, and counted among those that raised an error, under a message that names the item whose error variance ran off; the run goes on. Every such replication is counted in the `bootstrap` attribute and the printed line, and when the replications set aside and the improper ones kept together reach the tail area of

the interval, $(1 - C)/2$ of B, a caution says that the limits depend on how they were treated. When some of them had no single-factor solution, the caution adds that the lower limit is likely too high: with three weakly related items the estimator itself runs high near that boundary, and the resamples beyond it are those whose correlations point to low reliability. When about half of the replications are improper, neither setting gives an interval worth reporting, and the counts are what the result can state.

Comparison with other packages. The **psych** package provides omega, which fits a Schmid-Leiman hierarchical factor model and reports several variants of ω (ω_t, ω_h) alongside extensive psychometric diagnostics. reliability_omega in **DMAR** differs in emphasis: it implements McDonald's ω from a single-factor (congeneric) model and accompanies the point estimate with a confidence interval drawn from the methods compared in Kelley and Pornprasertmanit (2016). The same denominator distinction appears in semTools' compRelSEM() as its obs.var argument.

Value

A data.frame with columns term and value and rows "estimate" (sample coefficient ω), "se" (standard error on the coefficient scale, NA for methods that do not produce one; for the transformation-based intervals "fisher", "bonett", and "hakstian_whalen" it is the delta method back-transform of the transformation-scale standard error), "se_transformed" (only for those transformation-based intervals: the standard error on the transformation scale, with the scale named in the attribute se_transform_scale: "fisher_z", "log(1-alpha)", or "cube_root"), "ci_lower" and "ci_upper" (each clamped into [0, 1], so the limits stay in order), "conf_level", "N" (the cases the analysis used: the complete cases under listwise deletion, and under "fiml" and "two_stage" every case with at least one observed item or observed auxiliary variable), "N_complete" (the complete cases; equal to "N" under listwise deletion and for covariance input), and "n_items" (number of items). Attributes coefficient ("omega"), ci_method, denominator, missing, wald_df (the degrees of freedom of the Wald interval's *t* reference, Inf for the normal one and for intervals that are not Wald intervals; under wald_df = "missing_information" the attribute wald_df_table holds the df_missing table they came from), and (when supplied) aux record the computation; bootstrap calls also record B. The attribute residual_cor carries the local dependence summary of a maximum likelihood fit of the single-factor model to the same cases, whatever the ci_method (the weighted least squares fit behind the ADF intervals has residuals of its own, so it is not used for the summary): a list with matrix (the observed minus the model implied item correlations, named by item), largest (the residual correlation largest in absolute value, with its sign), items (the pair it belongs to), and saturated. With three items the single-factor model is saturated (zero degrees of freedom) and a proper solution reproduces every correlation, so its residuals are zero and carry no information about local dependence: saturated is then TRUE, largest is NA, and items is NULL, rather than naming whichever pair carries the largest rounding error. The printed table names the denominator and the largest residual correlation (or says that the model is saturated) in two lines under the table. A bootstrap call also carries the attribute bootstrap, a list recording the replications the interval conditions on: attempted (B), returned (the replications whose value the interval uses; the standard error is their standard deviation and the percentile and BCa limits are their quantiles), failed (the difference), reasons (a named count of why replications were set aside: error when the refit raised an error, non_finite when it was set aside without one, as a fit that did not converge is, and as an improper one is under improper = "drop" even though its estimate is finite), error_messages (the distinct error messages with their counts, empty when none), undetermined (the replications set aside because the resample left a polychoric correlation undetermined, which only reliability_omega_categorical counts; zero

here), improper and improper_kept (the replications whose single-factor solution was improper, with a negative error variance, or with three items had no single-factor solution at all: those set aside, which are among the non_finite ones, and those whose estimate the interval uses, which are among the returned ones), no_solution (the replications of three items set aside because their correlations admit no single-factor solution, a subset of improper), improper_rule ("keep" or "drop"; see the argument improper), not_converged (zero here: a fit that did not converge is counted among the non_finite ones), too_few_cases (the replications set aside because the resample held too few distinct cases for the rule on the number of cases, which are among the error ones; when they number more than half of B, no interval is formed), caution (TRUE when the replications set aside and the improper ones kept number at least $(1 - C)/2$ of B, with C the confidence level, so that how they were treated can decide where a limit falls), require_half (TRUE: an interval requires at least half of all attempted replications to return defined estimates; this is an estimability safeguard, not a coverage guarantee), required (the fewest replications a request for an interval of this method and level may ask for, a count held to B and not to the replications that return a value: 400 for "percentile", 2000 for "bca", 100 for the two bootstrap standard error methods at the 95 percent level; see dmar_bootstrap), bootstrap_status ("complete", "dropped", "exploratory", "insufficient_count", or "insufficient_share"; under the last two the limits are NA), B_used (the smallest count any reported limit rests on), target_returned and target_withheld (counts and withholding flags for each target in a vector bootstrap), acceleration and acceleration_method (the BCa acceleration used and whether it came from the regression or the jackknife; NA unless a BCa interval was formed), jackknife (the same attempted, returned, failed, reasons, error_messages, undetermined, improper, improper_kept, no_solution, not_converged, and too_few_cases for the leave-one-out refits behind the acceleration, all zero when no jackknife ran), ncpus (the number of processes used), t (the estimate of every replication, in order, NA where there is none; an improper estimate set aside under improper = "drop" is kept here, so the interval under the other rule can be formed without refitting), and status (a factor giving the fate of every replication, with levels "proper", "improper", "error", "non_finite", "undetermined", and "not_converged"). The interval uses the replications whose status is "proper" and, under improper = "keep", those whose status is "improper" and whose t is finite. The attribute B_used is the number of replications the interval uses (returned). The attributes n_dropped (B minus B_used), B_required, and bootstrap_status repeat the counts in the names every bootstrap result of the package carries. The warning has class dmar_bootstrap_replications_dropped when replications were set aside and the interval was formed, and dmar_bootstrap_insufficient when the interval was withheld. A BCa interval is also withheld, with the second class, when fewer than 10 replications lie beyond a limit where the bias correction and the acceleration place it (see dmar_bootstrap). The printed table shows these counts in one line, and a warning states them whenever a replication was set aside or the caution applies.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G. (2002). Sample size requirements for testing and estimating coefficient alpha. *Journal of Educational and Behavioral Statistics, 27*(4), 335–340. doi:10.3102/10769986027004335

Browne, M. W. (1982). Covariance structures. In D. M. Hawkins (Ed.), *Topics in applied multivariate analysis* (pp. 72–141). Cambridge, UK: Cambridge University Press.

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*, 62–83.

Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis. In A. von Eye & C. C. Clogg (Eds.), *Latent variables analysis: Applications for developmental research* (pp. 399–419). Thousand Oaks, CA: Sage.

Savalei, V., & Bentler, P. M. (2009). A two-stage approach to missing data: Theory and application to auxiliary variables. *Structural Equation Modeling, 16*, 477–497. doi:10.1080/10705510903008238

Savalei, V., & Falk, C. F. (2014). Robust two-stage approach outperforms robust full information maximum likelihood with incomplete nonnormal data. *Structural Equation Modeling, 21*(2), 280–302. doi:10.1080/10705511.2014.882692

van Driel, O. P. (1978). On various causes of improper solutions in maximum likelihood factor analysis. *Psychometrika, 43*, 225–243.

von Hippel, P. T. (2016). New confidence intervals and bias comparisons show that maximum likelihood can beat multiple imputation in small samples. *Structural Equation Modeling, 23*(3), 422–437. doi:10.1080/10705511.2015.1047931

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*, 165–200.

Collins, L. M., Schafer, J. L., & Kam, C.-M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods, 6*, 330–351. doi:10.1037/1082989X.6.4.330

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334.

DiCiccio, T. J., & Efron, B. (1996). Bootstrap confidence intervals. *Statistical Science, 11*(3), 189–228.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Feldt, L. S. (1965). The approximate sampling distribution of Kuder-Richardson reliability coefficient twenty. *Psychometrika, 30*, 357–370.

Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika, 74*, 155–167. doi:10.1007/s11336-00890993

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80–100. doi:10.1207/S15328007SEM1001_4

Guttman, L. (1945). A basis for analyzing test-retest reliability. *Psychometrika, 10*(4), 255–282.

Hakstian, A. R., & Whalen, T. E. (1976). A *k*-sample significance test for independent alpha coefficients. *Psychometrika, 41*, 219–231.

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Kelley, K., & Cheng, Y. (2012). Estimation of and confidence interval formation for reliability coefficients of homogeneous measurement instruments. *Methodology, 8*, 39–50. doi:10.1027/1614-2241/a000036

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

McDonald, R. P. (1999). *Test theory: A unified treatment*. Mahwah, NJ: Lawrence Erlbaum Associates.

McDonald, R. P. (2004). Respecifying improper structures. *Structural Equation Modeling, 11*(2), 194–209.

McDonald, R. P. (2011). Measuring latent quantities. *Psychometrika, 76*, 511–536. doi:10.1007/s1133601192237

Raykov, T. (2002). Analytic estimation of standard error and confidence interval for scale reliability. *Multivariate Behavioral Research, 37*, 89–103. doi:10.1207/S15327906MBR3701_04

Satorra, A., & Bentler, P. M. (2001). A scaled difference chi-square test statistic for moment structure analysis. *Psychometrika, 66*(4), 507–514. doi:10.1007/BF02296192

Terry, L. J., & Kelley, K. (2012). Sample size planning for composite reliability coefficients: Accuracy in parameter estimation via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 371–401. doi:10.1111/j.20448317.2011.02030.x

Yuan, K.-H., & Hayashi, K. (2003). Bootstrap approach to inference and power analysis based on three test statistics for covariance structure models. *British Journal of Mathematical and Statistical Psychology, 56*, 93–110.

Zhang, Z., & Yuan, K.-H. (2016). Robust coefficients alpha and omega and confidence intervals with outlying observations and missing data: Methods and software. *Educational and Psychological Measurement, 76*, 387–411. doi:10.1177/0013164415594658

Zinbarg, R. E., Revelle, W., Yovel, I., & Li, W. (2005). Cronbach's α, Revelle's β, and McDonald's ω_H: Their relations with each other and two alternative conceptualizations of reliability. *Psychometrika, 70*, 123–133. doi:10.1007/s1133600309747

See Also

`reliability` (general wrapper), `reliability_omega_categorical` (categorical omega for ordered items), `reliability_alpha`, `cfa_1` (single-factor CFA used internally), `omega`.

Other reliability: `cohen_kappa()`, `diagnosis_agreement`, `fleiss_kappa()`, `icc()`, `reliability()`, `reliability_H()`, `reliability_alpha()`, `reliability_kr20()`, `reliability_omega_categorical()`

Examples

```
set.seed(113)
J <- 6
loadings <- seq(0.5, 0.8, length.out = J)
eta <- rnorm(200)
errors <- matrix(rnorm(200 * J), 200, J) %*% diag(sqrt(1 - loadings^2))
items <- sweep(matrix(rep(eta, J), 200, J), 2, loadings, `*`) + errors
colnames(items) <- paste0("y", seq_len(J))

# Default: robust omega, point estimate only (no bootstrap is run
```

```
# unless requested; a message names the call that produces the
# recommended interval).
reliability_omega(data = items)

# The closed-form standard errors are derived under the model implied
# ratio, so with the observed denominator the interval comes from a
# bootstrap, which refits the single factor model once per
# replication. B = 10 keeps the example quick and is below the 400 a
# reported interval needs; the option lets it run and the warning says
# so. A reported interval deserves the default B = 10000, an accuracy
# choice rather than a formality (see Details). The percentile
# interval is what Kelley and Pornprasertmanit (2016) recommend for
# this coefficient, with ci_method = "bca" as the alternative, and
# seed makes the interval reproducible.
old <- options(dmar.bootstrap_exploratory = TRUE)
reliability_omega(data = items, ci_method = "percentile", B = 10,
                  seed = 113)
options(old)

# Model implied denominator with its closed-form robust ML interval,
# whose limits are formed on the logit scale; ci_method = "mlr" gives
# limits formed on the scale of omega.
reliability_omega(data = items, denominator = "model_implied")

# Two further routes into the same coefficient. The first works from
# the summary statistics a paper reports, a covariance matrix and its
# sample size, with the maximum likelihood interval that summary
# input supports.
reliability_omega(S = cov(items), N = 200,
                  denominator = "model_implied", ci_method = "ml")

# How much would a t reference lengthen that interval? On 30 degrees of
# freedom, an arbitrary small value chosen to show the effect, the
# limits move out a little; the line under the table says so.
reliability_omega(S = cov(items), N = 200,
                  denominator = "model_implied", ci_method = "ml",
                  wald_df = 30)

# The second is full information maximum likelihood with an auxiliary
# variable, where missingness on y2 depends on an auxiliary z (missing
# at random given z). Supplying aux implies missing = "fiml"; the
# N_complete row shows how many rows listwise deletion would have
# kept. With values missing, the interval can also refer to a t
# distribution on degrees of freedom taken from omega's own fraction of
# missing information, here in the fit that carries z (the calibration
# study behind the rule had no auxiliary variable, so this case rests
# on analogy; see Details); the line under the table gives N, the FMI,
# and the degrees of freedom that result. The rule is calibrated for
# limits on the scale of omega, so the interval here is the "mlr" one.
z <- eta + rnorm(200, sd = 0.5)
d <- data.frame(items, z = z)
d$y2[runif(200) < plogis(-1 + 1.5 * as.numeric(scale(z)))] <- NA
reliability_omega(data = d, aux = "z", denominator = "model_implied",
```

```
                  wald_df = "missing_information")

# Two-stage estimation instead: z enters only the first stage, where the
# means and covariances are estimated, and the interval uses the robust
# two-stage standard error, its limits again formed on the logit scale.
reliability_omega(data = d, aux = "z", missing = "two_stage",
                  denominator = "model_implied")
```

`reliability_omega_categorical` *Categorical Omega for Ordered-Categorical Items, With a Confidence Interval*

Description

Estimates categorical omega (ω_C; Green & Yang, 2009; Kelley & Pornprasertmanit, 2016) for a homogeneous composite of ordered-categorical items and returns a bootstrap confidence interval. The `denominator` argument selects whether the variance of the summed categories is built from the unrestricted polychoric correlations (`"observed"`, the default) or from the polychoric correlations the fitted single-factor model reproduces (`"model_implied"`, the coefficient as Green and Yang, 2009, defined it). Incomplete rows are set aside by listwise deletion (the default) or used through pairwise deletion or multiple imputation (`missing`); under multiple imputation the interval is formed by bootstrap then impute (Schomaker & Heumann, 2018).

Usage

```
reliability_omega_categorical(
  data,
 ci_method = c("bca", "percentile", "bootstrap_se", "bootstrap_se_logistic", "none"),
  conf_level = 0.95,
  B = NULL,
  seed = NULL,
  denominator = c("observed", "model_implied"),
  ncpus = 1L,
  missing = c("listwise", "pairwise", "mi"),
  aux = NULL,
  impute = c("pmm", "polr"),
  imputations = 20L,
  progress = NULL
)
```

Arguments

`data`	A numeric matrix or data frame of three or more ordered-categorical items (integer codes for the categories), plus any auxiliary variables named in `aux`. Only the order of an item's codes matters: its categories are scored 0, 1, …, $C - 1$ (see *How the categories are scored* in *Details*). A factor whose labels are whole

numbers is read as those numbers, so a factor labeled 0 to 4 is the item coded 0 to 4, under every treatment of `missing`; a factor whose labels are words is refused, as is an ordered factor whose levels are not in the order of the numbers they name. A column read with **haven** is read as its values, with SPSS user-defined missing values made missing. Incomplete rows are treated as `missing` says; under the default they are set aside. With two items the single-factor model is not identified (their one polychoric correlation determines only the product of the two loadings), and the function stops.

`ci_method` Method for constructing the confidence interval. See *Details*. When not supplied, no interval is computed: every interval for categorical omega is bootstrap based, and a bootstrap is never run unless requested. Ask for `"bca"` (the recommended method) or `"percentile"` to obtain one.

`conf_level` Confidence level for the interval. Defaults to `0.95`.

`B` Number of bootstrap replications. The default, `NULL`, is 10000 under `missing = "listwise"` and `"pairwise"`, where BCa, the recommended interval, reads its limits from extreme quantiles, and 1000 under `missing = "mi"`, where each replication is imputed `imputations` times and refitted on each, and the percentile interval is the recommended one (see *Run time under multiple imputation*). When no item value is missing, `missing = "mi"` has nothing to impute and returns the complete data result, whose bootstrap refits once per replication, so the default there is 10000, as the message says. When a bootstrap is requested, `B` (supplied or default) must be a single whole number of at least the count the interval method and the level require, as in the rest of the family: at the default `conf_level`, 400 for `"percentile"` (10 replications beyond each limit), 2000 for `"bca"` (50 beyond each nominal limit), and 100 for the two bootstrap standard error methods; at 0.99 the first two are 2000 and 10000, so the default of 1000 under `missing = "mi"` does not serve a 99 percent percentile interval and the refusal names the number. A smaller `B` is refused before anything is computed (class `dmar_count_invalid`). When fewer replications than required return a value the interval is withheld under a warning of class `dmar_bootstrap_insufficient`. See `dmar_bootstrap`, which also describes `options(dmar.bootstrap_exploratory = TRUE)` for a first look with a smaller `B`.

`seed` Random number seed used for bootstrap reproducibility. Defaults to `NULL`, under which the resamples are drawn from, and advance, the current random number stream; supply an integer for a reproducible interval, which is set for the call only, the caller's random number state being restored when it returns.

`denominator` How the polychoric correlations behind the total variance in the denominator are estimated: `"observed"` (the default; the unrestricted polychoric correlations) or `"model_implied"` (the correlations reproduced by the fitted single-factor model, as in Green and Yang, 2009). See *Details*. Every interval method applies to both settings, since every interval here is bootstrap based and each replication recomputes the coefficient with the same denominator.

`ncpus` The number of processes over which the bootstrap replications are spread. The default `1` runs them in this R session; a larger number uses forked workers (**boot**'s `parallel = "multicore"`) or, on Windows, which cannot fork, a socket

cluster (`"snow"`), whose workers load the installed **DMAR**. **boot** draws every resample before any replication runs, so a given `seed` returns the same interval, counts, and replicates whatever `ncpus` is. A session that forks workers and also starts background processes through **callr** or **processx** needs the environment variable `PROCESSX_NOTIFY_OLD_SIGCHLD` set before **processx** is loaded (see `sem_exact_fit_bootstrap`). Ignored unless a bootstrap method is selected.

`missing` How incomplete rows of `data` are treated: `"listwise"` (the default; only complete rows are analyzed), `"pairwise"` (each threshold and polychoric correlation from the rows that observe its items), or `"mi"` (multiple imputation, with a bootstrap then impute interval). Listwise and pairwise deletion are consistent only when values are missing completely at random; multiple imputation when they are missing at random given the items and `aux`. See *Missing data* in *Details*.

`aux` Optional character vector naming columns of `data` that are auxiliary variables: numeric predictors in the imputation model, never items, so they inform the imputations without entering the composite (the inclusive strategy of Collins, Schafer, & Kam, 2001). Supplying `aux` implies `missing = "mi"`, as it implies full information maximum likelihood in `reliability_omega`; with `"listwise"` or `"pairwise"` named explicitly it is an error.

`impute` The imputation model under `missing = "mi"`: `"pmm"` (the default; predictive mean matching) or `"polr"` (the proportional odds model). See *The imputation model* in *Details*. Supplying it with another `missing` is an error.

`imputations` Under `missing = "mi"`, the number of imputations of each bootstrap replication (at least 2, default 20), which is also the least number of imputations behind the point estimate (whose number is at least 20 and may be raised by von Hippel's rule; see *The number of imputations*). Supplying it with another `missing` is an error.

`progress` `NULL` (the default), or a function of two arguments that the bootstrap calls as it runs, so that an application can show how far a long bootstrap has gone and let its user stop it (a Shiny app, for example). It is called as `progress(done, total)`, with the number of replications attempted so far and `B`: after each replication when `ncpus = 1`, and with more processes after each block of replications they complete together, about a fiftieth of `B` (`ncpus * ceiling(B / (50 * ncpus))` replications, the last block possibly fewer), so that `done` advances by the block. A BCa interval takes its acceleration from a regression on the resampling frequencies when at least as many replications return a value as there are cases, and from the jackknife, one leave-one-out refit per case, when fewer do or when that regression fails or gives influence values that are not all finite; the jackknife runs at most once in a call, and not at all when no interval is formed or every returned replicate has the same value. While it runs, the callback is called again after each refit (with more processes, after each block of `ncpus * ceiling(n / (50 * ncpus))` refits, `n` the number of cases resampled), with `done` and `total` both `B`, so it can be called more than `B` times, and `done` equal to `total` does not mean that the bootstrap has finished. If the callback returns `FALSE`, the bootstrap stops before the next replication or block with an error of class `dmar_bootstrap_cancelled`, whose elements `done` and `total` say how far it got (see `dmar_metadata`); any other value is ignored. An error inside the callback stops the call with an error of class `dmar_progress_error`, whose

message names the callback and which carries `done`, `total`, and, as `parent`, the callback's own condition with its class intact, so that an application can tell a failing callback from a failing analysis. The callback runs outside the random number stream, so a given `seed` returns the same result with or without it, and a callback that draws random numbers leaves the caller's stream as it found it. With `NULL` nothing is called. Ignored unless a bootstrap method is selected.

Details

Categorical omega is designed for items measured on an ordered categorical scale (e.g., Likert items). It uses a probit-link single-factor model in which each observed item X_j is modeled as a categorization of an underlying continuous response variable X_j^* via thresholds $t_{j,c}$ (Muthén, 1984; Millsap & Yun-Tein, 2004), fit by diagonally weighted least squares with mean- and variance-adjusted test statistic (WLSMV), the standard estimator for ordered categorical items. With the delta parameterization ($Var(X_j^*) = 1$), the population categorical omega is

$$\omega_C = \frac{\sum_{j=1}^{J}\sum_{j'=1}^{J}\sigma_{jj'}(\lambda_j\lambda_{j'})}{\sum_{j=1}^{J}\sum_{j'=1}^{J}\sigma_{jj'}\left(\rho_{X_j^*X_{j'}^*}\right)},$$

where $\sigma_{jj'}(r)$ is the covariance of $(X_j, X_{j'})$ computed from a bivariate normal CDF over pairs of category thresholds and a latent-response correlation r (Green & Yang, 2009, Eq. 13–14, with the full coefficient their Eq. 21; Kelley & Pornprasertmanit, 2016, Eq. 17–18). The numerator uses the correlations of the common part, $\lambda_j\lambda_{j'}$, under either setting of `denominator`; the two settings differ in the correlations ρ used for the total variance. The bivariate normal probabilities, one for each pair of thresholds of each pair of items, are computed by **DMAR** itself, by the method of Drezner and Wesolowsky (1990) as refined by Genz (2004), which is the algorithm **mvtnorm** applies in two dimensions. The two agree to within 4×10^{-16}, except within about 10^{-7} of a correlation of one, where **mvtnorm** loses accuracy and **DMAR**'s computation does not. Computed together, the probabilities take a small fraction of the time of the factor model's fit.

`"observed"` **(default)** The unrestricted polychoric correlations, estimated from the data by a saturated bivariate model (Kelley & Pornprasertmanit, 2016, Eq. 18). Structure that the single-factor model omits, such as a pair of items with shared wording, stays in the total variance, which makes this the categorical counterpart of robust omega (see `reliability_omega`). It is the coefficient `MBESS::ci.reliability(type = "categorical")` computes, apart from the samples described below in which the data do not determine a polychoric correlation.

`"model_implied"` The latent-response correlations reproduced by the fitted single-factor model. This is categorical omega as Green and Yang (2009) defined it: their appendix program takes "the reproduced (model-based) correlation matrix" as the input to the denominator, so correlated errors enter the total variance only when they are specified in the model.

When the single-factor model reproduces the polychoric correlations the two settings coincide, and they diverge as the model misfits. Neither protects the numerator: shared structure among a few items raises their loadings under either setting. The result therefore reports the residual polychoric correlations (the unrestricted minus the reproduced) and prints the largest under the table, so that a high coefficient resting on redundant items is visible beside it. When the shared structure sits on the items that dominate the factor, the factor absorbs it and the residuals appear among the other items, so the largest residual signals local dependence more reliably than it locates it.

What each denominator describes. Both settings share the numerator, the variance of the summed categories attributable to the single factor, and differ in what they take the total variance of the sum to be. The model implied setting takes the variance the fitted single-factor model reproduces, so the coefficient describes the sum score as the model describes it: covariance the model omits is absent from the numerator and from the denominator alike. The observed setting takes the variance implied by the polychoric correlations estimated without the single-factor restriction (still under the latent-response model that defines polychoric correlations), so covariance the model omits is counted as variance of the sum that the factor does not explain. The two estimate the same quantity when the model holds. Kelley and Pornprasertmanit (2016) used the observed setting because the total variance of the sum score is a property of the data and need not be taken from a model that may be wrong; Green and Yang (2009) used the model implied setting, which is the reliability of the sum score under the fitted model.

How the categories are scored. Categorical omega is the reliability of a sum in which each item's categories are scored as consecutive ranks, 0, 1, ..., $C - 1$, in the order of the item's codes: Green and Yang (2009, Eq. 11) and Kelley and Pornprasertmanit (2016, Eq. 16) define the observed item as $X_{ij} = c$ for $c = 0, 1, \ldots, C - 1$, and **MBESS** and **semTools** compute the same coefficient. The function therefore uses the order of the codes in `data`, not their values, and the coefficient is that of the sum of the codes as given whenever every item's codes are equally spaced with one step shared by all the items (codes 0, 2, 4, 6 on every item give the same result as 1, 2, 3, 4). When they are not, as with codes -2, -1, 1, 2, with items stepping by 2 beside items stepping by 1, or with a category between two used ones that no respondent chose, the sum of the codes is a different composite, and a message says so and names the items; the coefficient is still that of the ranks. In the examples behind this paragraph the coded sum's reliability was lower by 0.009 to 0.012 for codes -2, -1, 1, 2 on every item and by 0.034 for two three-category items coded 1, 3, 5 beside two coded 1, 2, 3, and an unused interior category changed it by about 0.001.

When the data do not determine a polychoric correlation. The observed setting needs every unrestricted polychoric correlation. When two items are skewed in opposite directions, as in Green and Yang's (2009) mixed-skew conditions, their bivariate table can have empty cells in a pattern (every respondent below the top category of one item in the bottom category of the other) that leaves the polychoric correlation without a finite standard error. The unrestricted value is then arbitrary, and the single-factor fit gives the correlation no weight, so the loadings rest on the correlations the data do determine. Those identify the loadings only when they connect every item to the others, directly or through other items, and include at least one cycle of an odd number of items, such as three items whose three correlations are all determined (the graph whose edges are the determined pairs must be connected and not bipartite); an item joined to the rest by a single determined correlation is then identified as well. When every correlation between `y1` or `y2` and `y3` or `y4` of the example below is undetermined, the determined ones fix only the products of the loadings of `y1` and `y2` and of `y3` and `y4`: the relative sign of the two pairs is free, and **lavaan**'s default start values can return loadings of opposite signs and a categorical omega near 0.05, where the population value is 0.716. In 60 samples of 300 from that design, 13 had determined correlations that did not identify the loadings: 2 with that sign reversal (0.05 and 0.09), 8 whose value moved with the start values by up to 0.06, and 3 whose default start values gave an improper solution. The function therefore checks the graph. When the determined correlations identify the loadings, the observed setting stops with an error naming the pair and pointing to the model implied setting, and the model implied setting proceeds, says which pair it set aside, and reports that pair's residual correlation as `NA`. When they do not, both settings stop with an error naming the pairs and the blocks of items no determined correlation joins, since the numerator takes the loadings under either setting. The unrestricted polychoric correlations come from the single-factor fit itself, which agrees with a second, saturated fit

whenever every correlation is determined. **MBESS** computes them from such a saturated fit, which returns 0 for an undetermined correlation and so removes the pair from the total variance without a message. The example below uses Green and Yang's design at a sample size where samples rarely have this problem; at $N = 500$ most of them do.

A bootstrap meets the problem far more often than the sample does. The few respondents in the sparse cells of an oppositely skewed pair are what determine its polychoric correlation, and a resample omits some of them as a matter of course, so a sample whose correlations are all determined still yields many resamples in which one is not. On the data of the example below, which have no such pair, about half of the replications of a bootstrap of the observed setting have one (80, 95, and 97 of 200 in percentile bootstraps with `seed` 1, 2, and 3). Those replications are set aside, counted in the `bootstrap` attribute (`undetermined`; see *Value*), named in the printed line and in a warning, and the interval then conditions on the replications that redrew the respondents who determine the correlation, which is not the bootstrap distribution of the coefficient; with more than half set aside, no interval is reported (see below), and on these data the observed setting is always near that point. The model implied setting needs no unrestricted polychoric correlation and sets aside only the replications whose determined correlations leave the loadings unidentified (counted as `undetermined` as well), so with items skewed in opposite directions it keeps more of them, but how many it keeps depends on the sample size. In a percentile bootstrap of the example's complete data ($N = 1000$, `B = 200`, `seed = 1`) it set aside 1 replication of 200, and over 20 samples of that size from the design (`B = 50`) none in 15 and at most 18%. Under pairwise deletion of the example's incomplete copy it set aside 47 of 200: 27 whose loadings were unidentified, 3 improper, and 17 whose fit failed, so the caution applies. At $N = 300$, in the 45 samples of 60 whose point estimate was identified and proper, it set aside a median of 42% of the replications (`B = 50`): 18% where the sample determined every correlation, 40% where it left one undetermined, and 64% where it left two, and more than half in 15 of the 45. When fewer than half of the replications return a value, no interval is reported (see below). Under either denominator, a replication whose determined correlations do not identify the loadings is set aside rather than evaluated at arbitrary starting values.

When the solution is improper. With few respondents in the extreme categories, or an item that nearly duplicates the factor, the single-factor fit can return a standardized loading above one, a negative residual variance (a Heywood case). Categorical omega is not defined for such a solution, since its assembly evaluates the bivariate normal distribution at the item's communality, and the function stops with an error naming the item and its loading under either setting of `denominator`. A bootstrap replication with an improper solution is set aside and counted (`improper` in the `bootstrap` attribute); these are the replications in which a standardized loading ran past one (an error variance fell below zero), so setting them aside can trim the upper tail of the bootstrap distribution, which is why their number is reported.

Kelley and Pornprasertmanit (2016) found in extensive Monte Carlo simulation that the bias-corrected and accelerated (BCa) bootstrap confidence interval for categorical omega achieved acceptable coverage across a wide variety of threshold patterns, sample sizes, item counts, and population reliability values, with one documented exception: coverage dipped somewhat below the acceptable range when the number of items and the population reliability were both high. They specifically recommend BCa for categorical omega. Because no bootstrap runs in **DMAR** unless the user requests one, the default output is the point estimate with a message naming the call that produces the recommended interval; request `ci_method = "bca"` to obtain it. The interval is not a quick one. Every replication refits the ordered-categorical factor model, whose sample statistics also supply the unrestricted polychoric correlations behind the denominator. That is a fraction of a second for a scale of moderate length, so the recommended interval at the default `B = 10000` can run for several minutes; when `B` is below the number of cases, the jackknife behind the BCa acceleration adds one

such refit per case (at the default B the acceleration is read from the replicates themselves at no extra cost; see the "bca" entry below). The example on this page therefore stops at the point estimate; the reported analysis adds ci_method = "bca" to the same call, with seed for reproducibility, and for these oppositely skewed items it bootstraps the model implied setting, which at this sample size keeps nearly all of the replications of the complete data (see above), or reports how many replications of the observed setting were set aside.

When to use. Use reliability_omega_categorical when items are ordered-categorical, especially when (a) the number of categories is small (e.g., two to five), (b) item distributions are skewed, or (c) threshold patterns differ markedly across items. In Kelley and Pornprasertmanit's (2016) Study 3, treating ordered items as continuous and using reliability_omega with the observed total variance in the denominator achieved acceptable coverage only when threshold patterns were similar across items; in their experience that condition is rare in practice.

Available confidence interval methods (set via ci_method). Every interval here is bootstrap based: the rows of data are resampled with replacement B times (10000 by default, 1000 under missing = "mi"; see *Missing data* below) and categorical omega is recomputed, with the full WLSMV model refit, on each replication (Efron & Tibshirani, 1993). Replications whose refit fails or does not converge, most common with small samples and sparse response categories, are set aside, as are replications with an improper solution, replications that leave a polychoric correlation undetermined under the observed setting, and replications whose determined correlations leave the loadings unidentified under either setting (all described above), and the interval is computed from the replications that return a value: the standard error is their standard deviation, and the percentile and BCa limits are quantiles of the returned replicates, not of all B. How many were attempted, how many returned a value, and why the rest were set aside is recorded in the bootstrap attribute of the result (see *Value*), printed in one line under the table, and stated in a warning whenever a replication was set aside, so an interval that conditions on a subset of the replications says so and by how much. When fewer than half of the replications return a value, no interval is reported: the standard error and the limits are NA, and the warning and the printed line say why. The rest would be a minority of the resamples, chosen by whether they returned a value, and their spread would describe those resamples rather than the sampling distribution of the coefficient; this is the criterion by which the point estimate under missing = "mi" stops when fewer than half of its completed data sets return a value (Lim & Cheung, 2022, p. 1069). The remedy is a larger sample, which fills the sparse cells of the bivariate tables. Bootstrap results vary from run to run; supply seed for reproducibility.

"bca" The bias-corrected and accelerated bootstrap, the default and the specific recommendation of Kelley and Pornprasertmanit (2016). Where the percentile interval reads its limits directly off the empirical quantiles of the bootstrap estimates, BCa adjusts the two quantile positions for median bias (estimated from the bootstrap distribution) and for the rate at which the estimator's variance changes with the parameter (the acceleration, estimated by the jackknife), making it second-order accurate where the percentile interval is first-order accurate (DiCiccio & Efron, 1996). The adjusted quantile positions sit farther into the tails than the percentile interval uses, which is why the default B = 10000 is larger than the customary 2000; reduce B for exploration, not for a reported analysis. The acceleration is estimated as **boot** estimates it: by regressing the bootstrap replicates on the resampling frequencies when at least as many replications as cases returned a value, at no extra cost, and otherwise, or when that regression fails, by the jackknife, one leave-one-out refit per case. A leave-one-out refit that fails (when a case is the only respondent in one of an item's categories, for example) is counted and set aside, and the acceleration is estimated from the refits that returned a value. When the acceleration cannot be estimated at all (fewer than two leave-one-out refits returned a value),

or the bias correction cannot be (every returned replicate on one side of the estimate), the percentile interval is returned in its place with a warning, and the `ci_method` attribute records the substitution as `"percentile (BCa acceleration could not be estimated: ...)"`. The counts of leave-one-out refits and the acceleration used are in the `bootstrap` attribute.

`"percentile"` Percentile bootstrap: the interval limits are the empirical quantiles of the bootstrap estimates.

`"bootstrap_se"`, `"bootstrap_se_logistic"` Wald intervals using the bootstrap standard deviation as a standard error, built on the logit scale for the `_logistic` variant so the endpoints respect [0, 1]. Under `missing = "mi"` the standard error is that of the point estimate (see *The number of imputations* below).

`"none"` Return only the point estimate.

Under `missing = "mi"` the percentile interval is the recommended one and BCa is refused (see *Missing data*).

Missing data. The `missing` argument sets how incomplete rows of `data` are treated. Under `"pairwise"` and `"mi"`, rows with no observed item, which carry no information about the items, are set aside, as under full information maximum likelihood elsewhere in the family, and when no item value is missing in the rows that remain both return the complete data result, with a message. Under multiple imputation a row with no observed item is kept when an auxiliary variable is observed on it, and its items are imputed from the auxiliaries, as full information maximum likelihood elsewhere in the family keeps such a row: setting it aside would make the analysis conditional on having an observed item, which biases the estimate when the items go missing together at random given the auxiliaries.

`"listwise"` **(default)** Only the rows observed on every column of `data` are analyzed. Listwise deletion is consistent when values are missing completely at random (MCAR), and even then it discards what the incomplete rows say about the items they observe. When fewer than two rows are complete, or an item takes a single value in the complete rows, the function stops, says so, and names the treatment that uses the incomplete rows.

`"pairwise"` Each item's thresholds are estimated from the rows that observe it and each polychoric correlation from the rows that observe both items (**lavaan**'s `missing = "pairwise"`); the single-factor fit, both denominators, and the assembly are otherwise unchanged. Pairwise deletion is the default treatment of missing data under WLSMV estimation in Mplus (Shi, Lee, Fairchild, & Maydeu-Olivares, 2020). Asparouhov and Muthén (2010) show it consistent when the missingness depends only on covariates in the model; a single-factor model of the items has no covariates, so here it is consistent only under MCAR, the assumption of listwise deletion. It uses more of the data than listwise deletion, not a weaker assumption: when values are missing at random (MAR), Shi et al. (2020) found the estimates of pairwise deletion severely biased, and they recommend multiple imputation. Every interval method applies, each replication resampling the rows and refitting by pairwise deletion.

`"mi"` Multiple imputation, consistent when values are MAR given the items and any auxiliary variables named in `aux`. The items are imputed by chained equations with **mice** (van Buuren & Groothuis-Oudshoorn, 2011), with every other item and every auxiliary variable as a predictor and 10 iterations of the chains, and categorical omega is computed on each completed data set with the chosen `denominator`; the point estimate is their mean (Rubin's rule for a point estimate), and the residual polychoric correlations are averaged the same way. An auxiliary variable is a predictor in the imputation model only, never an item, so it carries information about the missing values without entering the composite. The interval is the bootstrap

then impute interval of Schomaker and Heumann (2018, their Boot MI): `B` bootstrap samples are drawn from the incomplete rows, each is imputed `imputations` times, the estimate of a replication is the mean of its imputations, and the percentile interval of those means is the confidence interval. Because the imputation model is refitted on every bootstrap sample, the interval carries the sampling variability of the imputation model as well as that of the coefficient. Bartlett and Hughes (2020) found that imputing and then bootstrapping does not, in general, give valid variances when the imputation and analysis models are uncongenial, as a regression imputation model and a probit factor model are, while bootstrapping and then imputing can. Schomaker and Heumann found the coverage of Boot MI near the nominal level in their simulations, and found that pooling the estimates of every imputation of every replication into one distribution gives intervals that are too wide, which **DMAR** therefore never does. A completed data set of the observed data that returns no value is set aside and counted, and the point estimate averages the rest; the call stops only when fewer than half return a value. A replication computes the same statistic: the mean of those of its completed data sets that return a value, set aside when fewer than half of them return one or when its imputation fails (see *When a completed data set is improper* below).

When a completed data set is improper. Imputation can produce an improper single-factor solution (a standardized loading above one) in a completed data set when the observed data have none, and the example below shows how. The polychoric correlation of an item piled into its top category with one piled into its bottom category is determined by the few respondents in the cells where the two disagree, and the values missing at random given `y1` and `y2` are missing most often among the respondents low on them, who supply those cells. Predictive mean matching fills each missing value with the observed value of a donor whose predicted mean is close, and among the observed respondents low on `y1` and `y2` nearly all chose the bottom category of `y3` (all but one, in the example's sample). How many imputed respondents land in the rare cells therefore varies from one completed data set to the next; where almost none do, the polychoric correlation runs toward one and the loading of `y3` past it. In 100 completed data sets of the example's data, 13 were improper: 7 of the 13 in which the imputations put no respondent in the cells where `y3` is above its bottom category and `y1` or `y2` below its top, 6 of the 58 in which they put one or two there, and none of the 29 in which they put three or more. The standard deviation of the loading of `y3` over those imputations was 0.106, against 0.046 over 40 imputations under the proportional odds model. Neither that model nor more donors is a remedy: over 61 samples of the example's design the proportional odds model gave improper sets in fewer samples (5 of the 61, against 13) but estimates lower than the complete data value by 0.066 on average, and predictive mean matching with ten donors in place of five lowered the average share of improper sets only from 3% to 2%.

Categorical omega is not defined for an improper solution, so such a completed data set has no estimate, and Rubin's rules, which average an estimate from every completed data set, need one. Stopping whenever one completed data set fails, as Jia and Wu (2019) counted a replication of their simulation as failed unless all its imputations converged, makes the method fail by design: with a share p of such sets, M imputations include one with probability $1 - (1 - p)^M$, which approaches one as von Hippel's rule asks for more imputations. The function instead sets such a completed data set aside, as it sets aside a bootstrap replication, and the point estimate averages those that returned a value, as Lim and Cheung (2022) pooled the imputations that converged and as **lavaan.mi** does by default for imputations that did not converge. Setting aside conditions the estimate on a property of the estimate, so the number set aside, and why, is reported in a warning and printed under the table, and the call stops when fewer than half of the completed data sets return a value, the criterion by which Lim and Cheung counted an analysis as failed. The improper sets are those with the largest loadings, so setting them aside lowers the estimate. Over the 61 samples, 13 had at least

one improper completed data set among 20, with a share of 3% on average and 65% at most, and no sample's complete data or pairwise deletion fit was improper. Against refitting those sets with every loading bounded at one, setting them aside lowered the estimate by about 0.034 times the share set aside (0.0017 at 5%, a tenth of the complete data estimate's sampling standard deviation of 0.0165), and over the 13 samples the two were equally far from the complete data estimate (root mean square differences 0.0126 and 0.0122). No completed data set of the observed data is fitted after the bootstrap, so none can discard a bootstrap already run (see *The number of imputations*).

A bootstrap replication follows the same rule, set by set: it averages those of its M_b completed data sets that return a value and is set aside when fewer than half do, because the bootstrap distribution is that of the statistic computed as the point estimate is computed. The interval follows it too: when fewer than half of the replications return a value, none is reported, as for the other treatments (see the confidence interval methods above). Setting a replication aside whenever any one of its completed data sets returned no value would be a different statistic, and the argument above applies to it unchanged: a replication would be set aside with probability $1-(1-p)^{M_b}$, so raising `imputations` would set more replications aside, and the completed data sets of those replications that did return a value would be discarded. In a bootstrap of the example's incomplete data under the model implied setting (`B = 200`, `imputations = 5`, `seed = 1`), in which a quarter of the completed data sets returned no value, that rule, applied to the first two, three, four, and five imputations of each replication, would have set aside 35.5, 45.5, 48.5, and 52.5 percent of the replications, where the rule used sets aside 12.5, 21, 14.5, and 20 percent. Under the rule used the share set aside goes to zero as M_b grows when fewer than half of the completed data sets fail, but not steadily: an odd M_b needs $(M_b+1)/2$ of its sets to return a value, one more than the even number below it needs of one set fewer, so it sets aside more replications than either even neighbor (with a fifth of the sets failing independently, 4.0, 10.4, 2.7, and 5.8 percent at two to five imputations). Where completed data sets are set aside, an even number of imputations is the better choice, and the number the warning names is even. The warning and the `imputation` attribute count the completed data sets set aside within the replications kept (`replicate_sets_set_aside`).

The imputation model. `impute = "pmm"`, the default, is predictive mean matching: each missing value is replaced by the observed value of the same item of a donor whose predicted mean, from a linear regression on the other variables, is close, with the regression parameters drawn anew for each imputation. It never imputes a category the item does not show, though a category rare among the observed values is rarely drawn. `impute = "polr"` is the proportional odds model, with every other item entering as a factor (a binary item is imputed by logistic regression, the same model for two categories). **mice** draws its categories at the model's point estimates, without drawing the model's parameters, and Jia and Wu (2019), pooling its imputations in the usual way, found acceptable loading estimates but confidence intervals that were too narrow under missingness at random; under bootstrap then impute the refit on every bootstrap sample supplies the variability of the imputation model instead. The default rests on a Monte Carlo simulation run for this function: 10 items, one factor with loadings from .50 to .80, $n = 300$, and items 6 to 10 missing at random given the mean of items 1 to 5, about 21% of the values in all. Against the complete data estimate of the same sample, the mean categorical omega was off by -0.0035 with five categories (100 replications) and by +0.0015 with three skewed categories (60 replications) under predictive mean matching; by -0.019 and +0.035 under the proportional odds model; by -0.028 and -0.047 under pairwise deletion; and by -0.105 with five categories under listwise deletion, which with three categories failed in 55 of the 60 samples. A replication under predictive mean matching was also about ten times faster (see *Run time* below). The evidence is one factor structure and one missing data mechanism, so with very rare categories or another mechanism the comparison could differ.

Every incomplete item is imputed. By default **mice** stops imputing a variable whose correlation

with another, computed from the rows that observe both, reaches .999 in absolute value; two items observed together on only two or three rows, as in a planned missing design with a small overlap, reach it by arithmetic, and the item left with its missing values would then be fitted by listwise deletion inside each completed data set. That check is therefore turned off, an item with a single observed value is refused before anything is imputed, and an item **mice** still does not impute stops the call with its name.

The same check also set aside an auxiliary variable that duplicates another (a score and its rescaling, age in years and in months). Without it both reach every item's regression, and when the predictors carry a second exact dependency (a second such pair, or an auxiliary variable computed from complete items) **mice** 3.19.0 removes only one of them and falls back to a ridge penalty, after which its imputations are unstable: on the five-item example of the tests, an auxiliary variable computed from the first three items and its rescaling gave 0.768 with a between-imputation standard deviation of 0.062, against 0.800 and 0.008 without them. So the auxiliary variables are screened before anything is imputed: one whose multiple correlation with the columns observed on every row that observes it (the items and the auxiliary variables kept before it) reaches .999, **mice**'s own threshold for a pair, is a linear function of predictors already in the model, carries no information about the missing values, and is left out, with a message naming it; the items are not screened, so a planned missing design is still imputed in full. When **mice** still applies its ridge penalty, which it does when the predictors of an item are (nearly) linearly dependent, the warning says how often, and the `imputation` attribute counts it (`ridge`).

The items are imputed as plain integer codes, whatever their class: a column read with **haven**, which carries value labels, and a factor whose labels are whole numbers are both accepted (see the `data` argument), and the imputation model is the one for the codes. SPSS user-defined missing values, which `haven::read_sav(user_na = TRUE)` keeps as values of a `haven_labelled_spss` column, are missing under every treatment, as SPSS and **haven**'s `is.na()` treat them.

The number of imputations. Each bootstrap replication is imputed `imputations` times (M_b, 20 by default). The variance of a replicate mean is the sampling variance of the coefficient plus the imputation variance divided by M_b, so imputation noise widens the percentile interval by about $\sqrt{1 + \gamma/M_b}$, where γ is the fraction of missing information: by 1.5% at $\gamma = .15$ and by 4.9% at $\gamma = .50$ with five imputations. The function estimates γ from the replications themselves, by the one-way analysis of variance of the replicate by imputation estimates (the variance between replications net of the imputation variance within them, which is the sampling variance with infinitely many imputations, against the imputation variance), reports the widening, and warns when it exceeds 5%, naming the value of `imputations` that would bring it under; when no interval is reported (fewer than half of the replications returned a value) neither the warning nor the printed line gives a widening, and the printed fraction of missing information is marked as that of the replications that returned one. A replication that sets some of its completed data sets aside averages $n_b < M_b$ of them, and the analysis of variance is then the unbalanced one (Searle, Casella, & McCulloch, 1992, Sec. 3.6), whose interval for the ratio of the variances is approximate; the widening is $\sqrt{1 + \gamma\,\overline{1/n_b}}$, the mean taken over the replications, and the value of `imputations` named allows for the share of completed data sets set aside and is even when any were, whether in the replications kept or in those set aside for them, but not when only the imputation itself failed (see *When a completed data set is improper*). Schomaker and Heumann (2018) found that Boot MI can do well with fewer than five imputations per replication and covers too often with one. The point estimate averages M_0 imputations of the observed data: at least 20 and at least `imputations`, and more when von Hippel's (2020, Eq. 1) quadratic rule, $M = 1 + \frac{1}{2}(\gamma/\mathrm{CV})^2$ with $\mathrm{CV} = .05$, asks for more at the upper limit of a 95% confidence interval for γ, as his two stage procedure uses it (here the limit for the ratio of variance

components in the same analysis of variance). His target is the replicability of a standard error from Rubin's rules, which bootstrap then impute does not use; the rule is borrowed to keep the point estimate from changing much if the data were imputed again, with the bootstrap as the pilot. Since the rule asks for at most 201 imputations (at $\gamma = 1$), a call with a bootstrap makes that many imputations of the observed data, and fits and checks each, before the bootstrap runs, and the point estimate then uses the first M_0 of them, the number fixed by the rule and the floor and never by their estimates. Nothing found in a completed data set of the observed data can then discard a finished bootstrap; the imputations the rule does not use cost at most 181 imputations and fits, less than 1% of the 20000 of the default bootstrap. The estimate of γ is imprecise when `B` is small, and under `impute = "polr"`, whose imputations ignore the uncertainty of the imputation model's parameters, it is too low. Without a bootstrap the point estimate averages `max(imputations, 20)` imputations, which keeps Rubin's relative efficiency, $(1 + \gamma/M)^{-1}$, at .976 or above for γ up to .5. The `se` row is the standard error of the point estimate, $\sqrt{s^2_{boot} + s^2_{imp}/M_0}$, with s^2_{boot} the sampling variance and s^2_{imp} the imputation variance from the same analysis of variance, so it counts the noise of the point estimate's own imputations. The analysis of variance is a construction for this function, not a published procedure for omega.

Run time under multiple imputation. A replication imputes its resample `imputations` times and refits the factor model on each completed data set, so the run time grows with `B` times `imputations`. With 10 items, five categories, and 300 cases, a replication with five imputations took about half a second under predictive mean matching and about five seconds under the proportional odds model, so the default `B = 1000` runs for about eight minutes and well over an hour, respectively, on one core with five imputations. The default of 20 imputations per replication does about four times that imputation and fitting work. This larger default reduces imputation noise; it has not been separately calibrated for interval coverage. The default `B` is 1000 under `missing = "mi"` rather than the 10000 of the other two treatments (a percentile interval reads its limits from less extreme quantiles than BCa), and why BCa, whose jackknife would add `imputations` imputations per case, is refused. `ncpus` spreads the replications over processes; every resample, and a seed for each replication's imputations, is drawn before any replication runs, so a given `seed` returns the same result whatever `ncpus` is.

Comparison with other packages. The **psych** package's `omega` fits a Schmid-Leiman hierarchical factor model to Pearson correlations or, with `poly = TRUE`, to polychoric correlations, and does not implement categorical omega in the sense of Green and Yang (2009), which `reliability_omega_categorical` computes for ordered-categorical items; `polychoric` provides polychoric correlation estimation as a separate tool.

Value

A `data.frame` with columns `term` and `value` and rows `"estimate"` (sample ω_C), `"se"` (the bootstrap standard deviation across replications, already on the coefficient scale; `NA` for `ci_method = "none"`), `"ci_lower"` and `"ci_upper"` (clamped to [0, 1]), `"conf_level"`, `"N"` (the rows analyzed: the complete rows under listwise deletion, otherwise the rows with at least one observed item, and under multiple imputation with `aux` also the rows with an observed auxiliary variable), `"N_complete"` (the rows complete on the items; equal to `"N"` under listwise deletion, and carried so the whole reliability family returns one shape), and `"n_items"` (number of items); the rows are the same under every `missing`. Under `missing = "mi"` the estimate is the mean over the imputations of the observed data and `se` is the standard error of that mean from the bootstrap (see *The number of imputations*). Attributes `coefficient` (`"omega_categorical"`), `denominator` (`"observed"` or `"model_implied"`), `missing` (`"listwise"`, `"pairwise"`, or `"mi"`; `"listwise"` also when the

items are complete and another treatment had nothing to do), `aux` (when supplied), `ci_method` (under `missing = "mi"` the method followed by `"(bootstrap then impute)"`), and `B` record the computation. The attribute `residual_cor` carries the local dependence summary of the single-factor fit: a list with `matrix` (the unrestricted polychoric correlations minus those the model reproduces, named by item), `largest` (the residual correlation largest in absolute value, with its sign), `items` (the pair it belongs to), and `saturated`; a pair whose polychoric correlation the data do not determine has an `NA` residual and is passed over. With three items the single-factor model is saturated (zero degrees of freedom) and reproduces every polychoric correlation, so its residuals are zero and carry no information about local dependence: `saturated` is then `TRUE`, `largest` is `NA`, and `items` is `NULL`, rather than naming whichever pair carries the largest rounding error. The printed table names the denominator and the largest residual correlation (or says that the model is saturated) in two lines under the table. A bootstrap call also carries the attribute `bootstrap`, a list recording the replications the interval conditions on: `attempted` (`B`), `returned` (the replications whose value the interval uses; the standard error is their standard deviation and the percentile and BCa limits are their quantiles), `failed` (the difference), `reasons` (a named count of why replications were set aside: `error` when the refit raised an error, `non_finite` when it was set aside without one, as a fit that **lavaan** stopped or that did not converge is, as an improper one is, and, under the observed denominator, as one that left a polychoric correlation undetermined is, and, under either, as one whose determined correlations leave the loadings unidentified is), `error_messages` (the distinct error messages with their counts, empty when none), `undetermined` (the replications set aside because the resample left a polychoric correlation undetermined, which the observed setting needs, or left correlations undetermined whose absence leaves the loadings unidentified, under either setting; see *Details*), `improper` (the replications set aside because their single-factor solution was improper, with a standardized loading above one, which is an error variance below zero), `improper_kept` (always zero here: categorical omega is not defined for an improper solution, so such a replication has no estimate to keep), `improper_rule` (`"drop"`, for the same reason), `no_solution` (always zero here: it counts the replications that `reliability_omega` sets aside because the correlations of three items admit no single-factor solution, a check the categorical fit does not make), `not_converged` (zero here: a fit that did not converge is counted among the `non_finite` ones), `too_few_cases` (always zero here: it counts the replications that `reliability_omega` sets aside for too few distinct cases, a rule the categorical fit does not apply), `caution` (`TRUE` when the replications set aside number at least $(1 - C)/2$ of `B`, with C the confidence level, so that where they would have fallen can decide a limit), `require_half` (`TRUE`: no interval is formed when fewer than half of the replications return a value), `required`, `bootstrap_status`, and `B_used` (the fewest replications a request may ask for, the verdict of the rule, and the count the limits rest on, as in `reliability_alpha`), `acceleration` and `acceleration_method` (the BCa acceleration used and whether it came from the regression or the jackknife; `NA` unless a BCa interval was formed), `jackknife` (the same `attempted`, `returned`, `failed`, `reasons`, `error_messages`, `undetermined`, `improper`, `improper_kept`, `no_solution`, `not_converged`, and `too_few_cases` for the leave-one-out refits behind the acceleration, all zero when no jackknife ran), `ncpus` (the number of processes used), `t` (the estimate of every replication, in order, `NA` where there is none), and `status` (a factor giving the fate of every replication, with levels `"proper"`, `"improper"`, `"error"`, `"non_finite"`, `"undetermined"`, and `"not_converged"`; the interval uses those whose status is `"proper"`). The attribute `B_used` is the number of replications the interval uses (`returned`; when fewer than half returned a value, or fewer than `required`, no interval is formed). The attributes `n_dropped`, `B_required`, and `bootstrap_status` repeat the counts in the names every bootstrap result of the package carries (see `dmar_bootstrap`). The printed table shows these counts in one line, and a warning states them whenever a replication was set aside or the caution applies.

Under `missing = "mi"` the `bootstrap` attribute has the same elements, with three differences: `reasons` has a third count, `imputation`, for the replications whose imputation itself failed (not part of `non_finite`), whose messages are in `imputation_messages`; `status` has a seventh level, `"imputation"`; and a replication counted under another reason is one in which fewer than half of its completed data sets returned a value, the first that returned none deciding the reason. The jackknife counts are zero, since BCa is refused. The attribute `imputation` is a list: `impute`; `imputations_point` (M_0, the imputations of the observed data behind the point estimate); `imputations_returned` (those of them that returned a value, which the estimate averages); `imputations_set_aside` (the others, counted by reason: `improper`, `undetermined`, `error`, `non_converged`, and `non_finite`); `imputations_replicate` (`imputations`); `imputations_floor` (`max(imputations, 20)`); `imputations_rule` (the number von Hippel's rule asks for); `maxit` (10, the iterations of the chained equations); `estimates` (the M_0 completed data estimates, `NA` where one was set aside), `status` (the fate of each: `"proper"` or the reason it was set aside), and `between_sd` (the standard deviation of those that returned a value); `sd_replicates` (the standard deviation of the replicate means); `s2_boot`, `s2_imp`, `fmi`, and `fmi_ci_upper` (the sampling and imputation variances, the fraction of missing information, and its 95% upper limit); `widening` ($\sqrt{1+\gamma/M_b}$, the factor by which imputation noise widens the percentile interval); `replicate_estimates` (the `B` by `imputations` matrix of completed data estimates, `NA` where one returned no value; `NULL` without a bootstrap); `replicate_sets_set_aside` (the completed data sets set aside within the replications that returned a value, counted by the reasons of `imputations_set_aside`; `NULL` without a bootstrap); `polr_fallbacks` (how often the proportional odds model fell back to a multinomial model, in the point estimate's imputations and in the replications); `ridge` (how often **mice** applied its ridge penalty, counted the same way); and `aux_set_aside` (the auxiliary variables left out of the imputation model as linear functions of other predictors, `character(0)` when none). The components that need the bootstrap are `NA` without one. Under `missing = "pairwise"` or `"mi"` the printed table adds a line naming the treatment and its assumption, and under `"mi"` a line giving the number of imputations, the fraction of missing information, and the widening (none when no interval is reported).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Asparouhov, T., & Muthén, B. (2010). *Weighted least squares estimation with missing data* (Mplus technical appendix). Muthén & Muthén.

Bartlett, J. W., & Hughes, R. A. (2020). Bootstrap inference for multiple imputation under uncongeniality and misspecification. *Statistical Methods in Medical Research, 29*(12), 3533–3546. doi:10.1177/0962280220932189

Collins, L. M., Schafer, J. L., & Kam, C.-M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods, 6*(4), 330–351. doi:10.1037/1082989X.6.4.330

DiCiccio, T. J., & Efron, B. (1996). Bootstrap confidence intervals. *Statistical Science, 11*(3), 189–228.

Drezner, Z., & Wesolowsky, G. O. (1990). On the computation of the bivariate normal integral. *Journal of Statistical Computation and Simulation, 35*(1–2), 101–107.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York, NY: Chapman & Hall/CRC.

Genz, A. (2004). Numerical computation of rectangular bivariate and trivariate normal and t probabilities. *Statistics and Computing, 14*(3), 251–260. doi:10.1023/B:STCO.0000035304.20635.31

Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika, 74*, 155–167. doi:10.1007/s11336-00890993

Jia, F., & Wu, W. (2019). Evaluating methods for handling missing ordinal data in structural equation modeling. *Behavior Research Methods, 51*, 2337–2355. doi:10.3758/s1342801811874

Kelley, K., & Cheng, Y. (2012). Estimation of and confidence interval formation for reliability coefficients of homogeneous measurement instruments. *Methodology, 8*, 39–50. doi:10.1027/1614-2241/a000036

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Lim, A. J.-M., & Cheung, M. W.-L. (2022). Evaluating FIML and multiple imputation in joint ordinal-continuous measurements models with missing data. *Behavior Research Methods, 54*, 1063–1077. doi:10.3758/s1342802101582w

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Millsap, R. E., & Yun-Tein, J. (2004). Assessing factorial invariance in ordered-categorical measures. *Multivariate Behavioral Research, 39*(3), 479–515. doi:10.1207/s15327906mbr3903_4

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika, 49*(1), 115–132.

Schomaker, M., & Heumann, C. (2018). Bootstrap inference when using multiple imputation. *Statistics in Medicine, 37*(14), 2252–2266. doi:10.1002/sim.7654

Searle, S. R., Casella, G., & McCulloch, C. E. (1992). *Variance components*. Wiley.

Shi, D., Lee, T., Fairchild, A. J., & Maydeu-Olivares, A. (2020). Fitting ordinal factor analysis models with missing data: A comparison between pairwise deletion and multiple imputation. *Educational and Psychological Measurement, 80*(1), 41–66. doi:10.1177/0013164419845039

Terry, L. J., & Kelley, K. (2012). Sample size planning for composite reliability coefficients: Accuracy in parameter estimation via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 371–401. doi:10.1111/j.20448317.2011.02030.x

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software, 45*(3), 1–67. doi:10.18637/jss.v045.i03

von Hippel, P. T. (2020). How many imputations do you need? A two-stage calculation using a quadratic rule. *Sociological Methods & Research, 49*(3), 699–718. doi:10.1177/00491241177473303

See Also

`reliability` (general wrapper), `reliability_omega` (use for continuous items), `reliability_kr20` (for dichotomous items), `omega`, `polychoric`.

Other reliability: `cohen_kappa()`, `diagnosis_agreement`, `fleiss_kappa()`, `icc()`, `reliability()`, `reliability_H()`, `reliability_alpha()`, `reliability_kr20()`, `reliability_omega()`

Examples

```
set.seed(113)
# The mixed-skew design of Green and Yang (2009, Table 1): four items
# loading 0.8 on one factor, each cut into five categories, two piled
# into the top category and two into the bottom. Their population
# categorical omega for this design is 0.716.
N <- 1000
p_top <- c(.011, .027, .047, .118, .797)
cut_top <- c(-Inf, qnorm(cumsum(p_top)[-5]), Inf)
cut_bottom <- c(-Inf, qnorm(cumsum(rev(p_top))[-5]), Inf)
latent <- outer(rnorm(N), rep(0.8, 4)) +
          matrix(rnorm(N * 4, sd = 0.6), N, 4)
items <- cbind(y1 = cut(latent[, 1], cut_top, labels = FALSE),
               y2 = cut(latent[, 2], cut_top, labels = FALSE),
               y3 = cut(latent[, 3], cut_bottom, labels = FALSE),
               y4 = cut(latent[, 4], cut_bottom, labels = FALSE))

# Categorical omega as Green and Yang defined it: the total variance
# of the summed categories comes from the correlations the fitted
# single-factor model reproduces.
reliability_omega_categorical(data = items, denominator = "model_implied")

# The default, categorical omega as Kelley and Pornprasertmanit (2016)
# computed and recommended it: the total variance comes from the
# unrestricted polychoric correlations. The single factor holds in
# this design, so the two agree to within about 0.01; they part ways
# as the model misfits.
reliability_omega_categorical(data = items)

# The same sum score with the items treated as continuous. Robust
# omega falls far below 0.716 here, because items whose categories
# pile up at opposite ends relate to the factor far from linearly,
# which is the case Green and Yang designed their coefficient for.
reliability_omega(data = items)

# The same items with values missing at random: respondents low on y1
# and y2 leave y3 and y4 unanswered more often. Listwise deletion, the
# default, keeps only the complete rows, and here that leaves too
# little: the few respondents who determine the correlation of a
# top-piled item with a bottom-piled one (one to five of the 1000 for
# each pair) stand low on y1 and y2, the complete rows keep none of
# them for y2 with y3 or for y1 with y4, and the correlations left do
# not identify the loadings, so a call with the default stops and says
# so (see Details).
z <- as.numeric(scale(items[, "y1"] + items[, "y2"]))
items_mar <- items
items_mar[runif(N) < plogis(-1.5 - 1.2 * z), "y3"] <- NA
items_mar[runif(N) < plogis(-1.5 - 1.2 * z), "y4"] <- NA

# Pairwise deletion estimates each polychoric correlation from the rows
# that observe both items, so every row contributes. It still assumes
# values missing completely at random, as listwise deletion does.
```

```
# Multiple imputation, which assumes only that they are missing at
# random, takes longer than an example here can (see Details).
reliability_omega_categorical(data = items_mar,
                              denominator = "model_implied",
                              missing = "pairwise")
```

`responder_analysis` *Responder Analysis: Who Cleared the Threshold, by Group*

Description

Reports the proportion of each group whose outcome reaches a meaningful threshold (a minimal clinically important difference, a remission cut, a mastery criterion), a clinical and behavioral endpoint that a mean difference does not show. For each group the function reports the responder count and proportion with a Wilson confidence interval; with exactly two groups it adds the risk difference with the Newcombe (1998) score-based hybrid interval and the number needed to treat with its confidence set; and across any number of groups it reports the omnibus chi square test of equal responder proportions. An optional `sweep` repeats the analysis over a grid of thresholds, making explicit how conclusions depend on where the line is drawn, disclosing the threshold dependence that any single-threshold claim leaves implicit.

Usage

```
responder_analysis(
  x,
  group,
  threshold,
  direction = c("ge", "le"),
  conf_level = 0.95,
  sweep = NULL
)
```

Arguments

`x`	Numeric vector of outcomes (for example, change scores).
`group`	Group labels, one per observation (coerced to factor; the first level is the reference for the two-group difference). An observation with a missing outcome or a missing label is removed, and a level left with no observation is dropped.
`threshold`	The cut defining response: a single number, fixed before the data are seen (see *Assumptions*).
`direction`	`"ge"` (default): a responder has `x >= threshold`; `"le"`: `x <= threshold` (for outcomes where lower is better). An outcome exactly at the threshold is a responder under either direction.
`conf_level`	Confidence level for all intervals. Defaults to 0.95.
`sweep`	Optional numeric vector of additional thresholds; the analysis is repeated at each and stacked with a leading `threshold` column. `threshold` comes first, then the values of `sweep` in the order given, each distinct value once.

Details

What is estimated. For each group, the proportion of its population whose outcome is at or beyond the threshold, π_g. With two groups, the difference of the two proportions, $\Delta = \pi_2 - \pi_1$ (second level of `group` minus the first), and the number needed to treat, $1/|\Delta|$: the number of people who would have to be moved from the first condition to the second for one more of them, on average, to be a responder. This is the number needed to treat of evidence based medicine, a function of two response proportions at a stated threshold. It is not the quantity `nnt_from_smd` returns, which is computed from a standardized mean difference under a normal model, involves no threshold, and is smaller for a positive effect (see *Which NNT this is* on that page).

Assumptions. The observations are independent, and with two groups the groups are independent of each other (two arms of a randomized trial, not two occasions for the same people; for paired proportions see `ci_proportion_paired`). The threshold is fixed in advance. Every interval and the test condition on the threshold as a known constant, so a threshold chosen after the data are seen, for instance the one at which the difference is largest, makes the stated confidence level and the p-value wrong, and no adjustment is made for it here. `sweep` describes how the result depends on the threshold; it is not a search, and the rows of a sweep are not adjusted for one another.

The interval for each proportion. With r responders among n, $p = r/n$, and z the $1 - \alpha/2$ quantile of the standard normal distribution, the Wilson score limits are

$$\frac{p + z^2/(2n) \mp z\sqrt{p(1-p)/n + z^2/(4n^2)}}{1 + z^2/n},$$

the same limits `ci_proportion` returns. They lie inside [0, 1] for every count, the counts 0 and n included, where the limit at the boundary is exactly 0 or 1 (and can be computed as a number within about 10^{-16} of it).

The interval for the difference. The two-group risk difference uses Newcombe's method 10: the difference interval is assembled from the two Wilson limits, which keeps it inside [-1, 1] and well behaved at boundary counts. With (l_1, u_1) and (l_2, u_2) the Wilson limits of the first and second group and $d = p_2 - p_1$, the limits computed are

$$d - \sqrt{(p_2 - l_2)^2 + (u_1 - p_1)^2} \quad \text{and} \quad d + \sqrt{(u_2 - p_2)^2 + (p_1 - l_1)^2}.$$

The number needed to treat and its confidence set. The `nnt` row holds $1/|d|$, an absolute value; the direction is the sign of the `difference` row. Its limits come from the limits of the difference, and what they are depends on where zero falls:

- Zero outside the interval for the difference: the reciprocals of the two limits, in absolute value, are the limits of the number needed to treat, the smaller in `ci_lower` and the larger in `ci_upper`.
- Zero inside the interval: a difference of zero is a number needed to treat of infinity, so the confidence set is every value at least as large as the reciprocal of one limit in one direction, together with every value at least as large as the reciprocal of the other limit in the other direction. These are two intervals that meet at infinity (Altman, 1998), which one lower and one upper limit cannot hold, so the two limits of the `nnt` row are `NA`.

In both cases the set is recorded in the `nnt_confidence_set` attribute on the signed scale $1/\Delta$, where a negative value means that the first group responds more often, and when it is two intervals

a line under the table states it, as nnt_from_smd does. The set has the coverage of the interval for the difference, because it is the image of that interval under the reciprocal.

The omnibus test. Pearson's chi square test of equal responder proportions across the groups, without a continuity correction (chisq.test with correct = FALSE), on the number of groups minus 1 degrees of freedom. With two groups it tests $\Delta = 0$. It and the Newcombe interval are different constructions and can disagree near the boundary of significance.

Edge cases.

- Equal observed proportions: the difference is 0, the nnt row is Inf, its limits are NA, and the confidence set is two intervals.
- No responder, or only responders, in one group: the Wilson limits stay inside [0, 1] and every row is reported. With none in one group and all in the other the difference is 1 or -1 and the number needed to treat is 1.
- No responder, or only responders, in every group: the proportions and their limits are reported, the difference is 0, and the chi square statistic is 0/0, so chi_square and p_value are NaN.
- A group with one observation: its Wilson interval is reported (for one responder of one at the 95 percent level, 0.207 to 1), and so is everything built from it. The chi square approximation is poor with expected counts this small; no warning is given.
- Outcomes equal to the threshold count as responders under both directions, so the counts under "ge" and "le" sum to more than the group size when there are such ties.
- Missing values in x or group are removed before anything is counted. Fewer than two groups after that is an error, as are fewer than four values in x.
- More than two groups: one row per group and the omnibus row. There is no difference or nnt row and no nnt_confidence_set attribute.

A worked example. In the first example below, 30 of 60 control participants and 35 of 60 treated participants gained at least 10 points. The responder proportions are 0.500 (95 percent Wilson interval 0.377 to 0.623) and 0.583 (0.457 to 0.699). The difference is 0.083, with the Newcombe interval -0.093 to 0.252. The number needed to treat is 1/0.083 = 12. Because the interval for the difference includes zero, the nnt row has no limits; the confidence set is the two intervals whose finite ends are the reciprocals of the limits of the difference, $1/(-0.093) = -10.8$ and $1/0.252 = 3.97$: a number needed to treat of 3.97 or more in favor of treatment, or of 10.8 or more in favor of control. The chi square statistic is 0.839 on 1 degree of freedom, p = 0.360. The second example repeats the analysis at thresholds of 5 and 15: at 5 the difference is 0.183 (0.026 to 0.330), its interval excludes zero, and the number needed to treat is 5.45 with limits 3.03 and 37.98; at 15 the difference is 0.067 and its interval includes zero again. One trial thus gives three different answers at three thresholds, which is the reason to fix the threshold first and to show the sweep.

What the threshold costs. Dichotomizing an outcome discards information. For two variables that are bivariate normal, MacCallum, Zhang, Preacher, and Rucker (2002, pp. 24 to 26) show that splitting one of them at its mean multiplies their correlation by 0.798 and their squared correlation by 0.637, which for a test of zero correlation is about the same as discarding 36 percent of the sample, and that the loss grows as the split moves away from the mean. A responder threshold is such a split of the outcome. A responder analysis therefore complements, and never replaces, the analysis of the continuous outcome (Maxwell, Delaney, & Kelley, 2027).

Value

A tidy wide `data.frame` (class `dmar_responder`, a `dmar_tbl`) with columns `group`, `n`, `responders`, `estimate`, `ci_lower`, `ci_upper`, `chi_square`, `df`, and `p_value`, and these rows:

- One row per group, named by its level: the group size `n`, the number of `responders`, the responder proportion in `estimate`, and its Wilson limits. The three test columns are `NA`.
- `difference` (two groups only): the second proportion minus the first in `estimate`, with the Newcombe limits. `n`, `responders`, and the test columns are `NA`.
- `nnt` (two groups only): $1/|d|$ in `estimate` (`Inf` when the difference is 0), with its limits when the interval for the difference excludes zero and `NA` limits otherwise. The other columns are `NA`.
- `omnibus`: `chi_square`, `df`, and `p_value` of the test of equal proportions. The other columns are `NA`.

When `sweep` is supplied, the same table is stacked per threshold with a leading `threshold` column.

With two groups the result carries two attributes. `nnt_set` is `"interval"` or `"two_intervals"`, one value per threshold. `nnt_confidence_set` is a `data.frame` with columns `lower` and `upper` (preceded by `threshold` under `sweep`) and one row per interval of the set, on the signed scale $1/\Delta$: one row, (`1 / ci_upper`, `1 / ci_lower`) of the `difference` row, for `"interval"`; and for `"two_intervals"` the rows (`-Inf`, `1 / ci_lower`) and (`1 / ci_upper`, `Inf`). For two intervals a line printed under the table states the set, and names each side by the group that responds more often there, from the `group_labels` attribute (the two levels of `group`, in order). The attributes belong to the table itself: `tidy()` and `glance()` return the rows without them, and a table made by `rbind()` or by taking rows keeps the attributes of the first table, as every `dmar_tbl` does. The confidence level is in the `conf_level` attribute.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Altman, D. G. (1998). Confidence intervals for the number needed to treat. *BMJ, 317*(7168), 1309–1312. doi:10.1136/bmj.317.7168.1309

MacCallum, R. C., Zhang, S., Preacher, K. J., & Rucker, D. D. (2002). On the practice of dichotomization of quantitative variables. *Psychological Methods, 7*(1), 19–40. doi:10.1037/1082-989X.7.1.19

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Newcombe, R. G. (1998). Interval estimation for the difference between independent proportions: Comparison of eleven methods. *Statistics in Medicine, 17*(8), 873–890. doi:10.1002/(SICI)1097-0258(19980430)17:8<873::AIDSIM779>3.0.CO;2I

See Also

`ci_proportion` for the per-group interval; `nnt_from_smd` for the model-based route to the number needed to treat from a standardized mean difference; `cliff_delta` and `proportion_of_superiority` for dominance-style effect sizes on the continuous outcome.

Other effect size estimates: cles(), cliff_delta(), correction_for_attenuation(), eta_squared(), eta_squared_generalized(), expected_partial_r(), expected_r(), expected_smd(), nnt_from_smd(), omega_squared(), probability_of_superiority_paired(), proportion_of_superiority(), smd_trimmed()

Examples

```
# A two-arm trial: change scores, response defined as a gain of 10+.
# The interval for the difference includes zero, so the NNT row has no
# limits and the line under the table gives the two intervals.
set.seed(113)
change <- c(rnorm(60, 8, 9), rnorm(60, 13, 9))
arm    <- rep(c("control", "treatment"), each = 60)
res <- responder_analysis(change, arm, threshold = 10)
res
attr(res, "nnt_confidence_set")

# How threshold-dependent is that conclusion?
responder_analysis(change, arm, threshold = 10, sweep = c(5, 15))
```

---

sd_unbiased *Unbiased Estimate of the Population Standard Deviation Under Normality*

Description

Returns the unbiased estimate of the population standard deviation under normality. Although the sample variance s^2 is unbiased for σ^2 when computed with $N - 1$ in the denominator, its square root s is biased downward for σ because the square root function is concave (Jensen's inequality). sd_unbiased applies the classical Holtzman (1950) correction factor that exactly removes that bias under normality.

Usage

```
sd_unbiased(s = NULL, N = NULL, X = NULL)
```

Arguments

s	The usual estimate of the standard deviation (the square root of the unbiased variance s^2). Must not be negative.
N	The sample size on which s is based. Must exceed 1; it need not be a whole number.
X	Optional vector of raw scores from which s and N are inferred (s = sd(X), N = length(X)). Mutually exclusive with s and N.

Details

The sample variance computed with $N - 1$ in the denominator is unbiased for the population variance σ^2, but its square root s is biased downward for σ. Under normality, the multiplicative bias is

$$E[s] \;=\; \sigma \cdot c_N^{-1}, \qquad c_N \;=\; \sqrt{(N-1)/2}\cdot \Gamma((N-1)/2)\,/\,\Gamma(N/2),$$

(Holtzman, 1950). Multiplying s by c_N therefore yields an unbiased estimator of σ. The correction is non-trivial in small samples: c_N is about 1.064 at $N = 5$, 1.028 at $N = 10$, 1.009 at $N = 30$, and is essentially 1 by $N = 100$ (about 1.003). For most applied work the bias of s is small enough to ignore, but it matters when s feeds into downstream quantities (variance components, standardizers, planning calculations) at small N.

Implementation note: c_N is computed as the reciprocal of $c_4(N) = E[s]/\sigma$, and $c_4(N)$ as the exponential of its logarithm. For $N \geq 101$ that logarithm is its asymptotic expansion, $-1/[4(N-1)] + 1/[24(N-1)^3] - \cdots$; for a smaller N it is the expansion at the first of $N+2, N+4, \ldots$ that reaches 101, carried back down by the exact recurrence $\log c_4(N) = \log c_4(N+2) - \log(1 + 1/[(N-1)(N+1)])/2$, whose terms all have one sign. Nothing cancels at any N, so the computation avoids both the overflow of `gamma` above $N \approx 340$ and the loss of digits in a difference of two `lgamma` values, which grows with N. c_N is accurate to two units in the last place at any N, and it is never below 1.

Value

A 1-row `data.frame` with columns `term` and `value`. The `term` value is `"sd"` and `value` is the unbiased estimate of σ.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Holtzman, W. H. (1950). The unbiased estimate of the population variance and standard deviation. *American Journal of Psychology*, *63*, 615–617.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

sd, cv

Examples

```
set.seed(113)
X <- rnorm(10, 100, 15)

# The plug-in estimate sqrt(s^2) is biased downward for sigma.
var(X)^.5

# Holtzman (1950) bias-corrected estimate, supplying s and N.
sd_unbiased(s = var(X)^.5, N = length(X))
```

```
# Equivalent call from the raw vector.
sd_unbiased(X = X)
```

`sem_exact_fit_bootstrap` *Bootstrap Test of Exact Fit for a Structural Equation Model With Complete or Incomplete Data*

Description

Gives the chi square test of exact fit of a structural equation model a reference distribution that does not assume normal data: the model based bootstrap of Bollen and Stine (1992), in which the data are first transformed so that the model holds in them exactly, as estimated, and the statistic is then recomputed on `B` samples drawn with replacement from the transformed data. The share of bootstrap statistics at least as large as the observed one is the bootstrap *p*-value. For incomplete data fitted by full information maximum likelihood (FIML) the transformation is extended so that the FIML estimates of the saturated model on the transformed data are the model implied moments (Enders, 2002; Savalei & Yuan, 2009), and each bootstrap sample keeps the missing values of the rows it draws, so that the model is fitted to it by FIML as it was to the data. Requires **lavaan** (in Suggests).

Usage

```
sem_exact_fit_bootstrap(
  fit,
  B = 1000,
  transformation = c("common", "pattern"),
  improper = c("keep", "drop"),
  conf_level = 0.95,
  seed = NULL,
  ncpus = 1L
)
```

Arguments

`fit`	A fitted **lavaan** model: a maximum likelihood fit (estimator ML, MLM, MLMV, or MLR) of raw continuous data, complete or incomplete under `missing = "ml"` (with auxiliary variables through lavaan's `aux` or without), in one or several groups, that converged to an admissible solution. See Details for what is refused.
`B`	Number of bootstrap samples, a single whole number of at least 100. Defaults to `1000`: every sample refits the model. A Monte Carlo *p*-value is run only from a request of at least 100 samples, at which its standard error is at most 0.05; a smaller `B` is refused before anything is computed (class `dmar_count_invalid`). See `dmar_bootstrap`.

`transformation` For incomplete data, how the transformation makes the saturated FIML estimates equal the model implied moments: `"common"` (the default) finds one matrix that transforms every row on its observed variables, symmetric on the scale of the variables divided by their saturated standard deviations, so that the FIML estimating equations hold summed over the missing data patterns; `"pattern"` transforms the rows of each missing data pattern on their own, with the pattern's own covariance matrix and means, so that each pattern's covariance matrix and means equal the model implied ones exactly, which needs every pattern to have more rows than observed variables. Both are solved on that standardized scale, so the result does not depend on the units of the variables. See Details. Ignored with complete data.

`improper` Whether a bootstrap sample whose fit converged to an inadmissible solution counts in the p-value: `"keep"` (the default) or `"drop"`. A solution is inadmissible when lavaan's post-estimation check finds a negative variance estimate or a covariance matrix that is not positive definite, of the latent variables, of the residuals of the observed variables, or of the residuals of the structural part of the model; with auxiliary variables the check is made on the model's own parameters (see What Is Counted). Either way the row `B_improper` reports how many there were.

`conf_level` Confidence level of the interval for the Monte Carlo error of the bootstrap p-value. Defaults to 0.95.

`seed` Random number seed for the bootstrap samples. Defaults to `NULL`, under which the samples are drawn from, and advance, the current random number stream; supply an integer for a reproducible result, which is set for the call only, the caller's random number state being restored when it returns.

`ncpus` The number of processes over which the bootstrap refits are spread. The default `1` runs them in this R session; a larger number uses forked workers or, on Windows, a socket cluster. Every bootstrap sample is drawn before any refit runs, so a given `seed` gives the same result whatever `ncpus` is. A session that forks workers and also starts background processes through **callr** or **processx** (a Shiny app, a job runner) needs the environment variable `PROCESSX_NOTIFY_OLD_SIGCHLD` set to a nonempty value before **processx** is loaded: **processx** otherwise replaces the signal handler through which **parallel** learns that a forked worker has finished, and **parallel** then reports at exit that it could not terminate its workers (**processx** documents the interaction and the switch). The results are not affected.

Details

The Model Based Bootstrap: The chi square statistic T of a maximum likelihood fit follows its chi square distribution under the model only for normal data in large samples. Bollen and Stine (1992) proposed to find its distribution under the model by bootstrap instead: resampling the data as they are would reproduce the misfit of the model as well as the sampling variation, so the data are first transformed to

$$Z = (Y - \bar{Y})S^{-1/2}\hat{\Sigma}^{1/2}$$

(Enders, 2002, Eq. 1), where S is the sample covariance matrix, $\hat{\Sigma}$ the covariance matrix the fitted model implies, and the square roots are symmetric. The transformed data have covariance matrix

exactly $\hat{\Sigma}$, so the model fits them exactly, and their distribution otherwise keeps the shape of the data. Each bootstrap sample draws rows of Z with replacement, within each group as many as the group has (each group is transformed with its own moments), the model is fitted to it, and its statistic T^* is recorded. The bootstrap p-value is the share of the T^* at least as large as the observed T (row `p_bootstrap`); a small value says that the observed statistic is unusual for data in which the model holds and have this distribution. Under a mean structure the model implied means are added back, so the means hold too. The statistic is the standard (uncorrected) likelihood ratio statistic in every case: the bootstrap replaces the correction a scaled statistic applies, and a fit by MLM or MLR contributes its estimates and its standard statistic. This is the bootstrap `lavaan::bootstrapLavaan(type = "bollen.stine")` runs for complete data, with the same transformation and resamples: the `seed` given here, passed to lavaan as its `iseed` (with lavaan run serially), gives the same T^*. lavaan called without `iseed` draws its own seed from the caller's random number stream, so a `set.seed()` before the two calls does not make them agree.

Incomplete Data: With missing values the transformation cannot be applied row by row, since a row observes only some of the variables. Enders (2002, Eq. 4) first extended it by transforming each row with the submatrices, for the variables the row observes, of the saturated and the model implied covariance matrices estimated by FIML from the whole sample, and adding back the model implied means; the model then does not hold exactly in the transformed data, since the rows of a pattern keep their own covariance matrix, and that transformation is not offered here. Savalei and Yuan (2009) gave two transformations under which the FIML estimates of the saturated model on the transformed data are exactly $\hat{\mu}$ and $\hat{\Sigma}$, so that the model fits the transformed data exactly under FIML, as the complete data transformation makes it fit exactly under complete data. The `"pattern"` transformation (their first) transforms the rows of each missing data pattern with the pattern's own covariance matrix and means, so each pattern's rows take exactly the model implied moments of the variables they observe; it needs every pattern to have more rows than observed variables, since the pattern's covariance matrix is inverted, and a pattern that falls short is refused with its index and counts. The `"common"` transformation (their second, and the default) centers each pattern at its own means, transforms every row with one matrix, symmetric on the standardized scale described below, chosen so that the covariance estimating equations of the saturated FIML model, summed over the patterns, hold at $\hat{\Sigma}$, and shifts each pattern to $\hat{\mu}$; it does not need large patterns. DMAR finds the matrix by Newton's method on the standardized variables, starting from the complete data transformation applied to the EM moments, and records the number of steps and the largest estimating equation at the solution in the `transformation` attribute, where the pattern transformation records the same check computed afresh. Both transformations are solved on the variables divided by their saturated standard deviations, and the transformed values are multiplied back: the matrix of the common transformation and the square roots of the pattern transformation are symmetric on that standardized scale. The bootstrap therefore does not depend on the units of the variables (rescaling an item rescales its transformed values and changes nothing else). Symmetry belongs to the scale on which it is imposed, so the same transformations solved in raw units, as `semTools::bsBootMiss()` solves the common one (its transformation 2), give transformed data that differ element by element from these and depend on the units; both versions make the saturated FIML estimates exactly $\hat{\mu}$ and $\hat{\Sigma}$, so the model holds exactly in either. Each bootstrap sample draws rows of the transformed data with their missing values, so it has the missing data patterns of the data in bootstrap proportions, and the model is fitted to it by FIML. In Enders's (2002) simulation of his transformation, with 10 to 20 percent of the values missing completely at random from nine nonnormal indicators, the bootstrap held the Type I error rate near .05 at N = 250 and 500, where the uncorrected FIML statistic rejected far too often, and was conservative at N = 100, the more so the more nonnormal the data and the more missing

values; he suggests a sample of about 200 or more with incomplete data. The calibration of the two transformations offered here on the current code is planned, not done.

What Is Counted: A bootstrap sample whose fit did not converge, raised an error, or returned no finite statistic is set aside, and one that converged to an inadmissible solution is counted or set aside as `improper` says. That includes a sample whose likelihood has no maximum, an error variance running off below zero while the likelihood is flat or still rising (see `fa_bifactor`): its estimates are wherever **lavaan**'s optimizer stopped on that ridge, but its chi square is not, since the likelihood levels off along the ridge and the optimizer stops only where its steps no longer gain, which puts the statistic within the optimizer's tolerance of its limit. A sample's solution is judged as the fit's is (see What Is Refused); with auxiliary variables that is the check of the model's own parameters, since **lavaan**'s own check would also count as improper the samples in which only the residual covariances of the auxiliary variables are indefinite, which with an auxiliary variable related to a factor can be many samples whose solution is proper. The rows `B`, `B_used`, `B_improper`, and `B_failed` count them, and a single warning (class `dmar_bootstrap_replications_dropped`) reports the samples set aside. The *p*-value is formed from the `B_used` statistics; when fewer than half of the attempted samples were used, it is `NA` with a warning of class `dmar_bootstrap_insufficient`, since a *p*-value from the samples that happened to converge would describe a selected distribution. The 100 required are held to the number requested, `B`, and not to `B_used`: a request that met the count and then had samples set aside still reports its *p*-value, the warning says how many were used, and the Wilson interval shows the Monte Carlo error from that number. Where refits can fail, request more than the minimum. `options(dmar.bootstrap_exploratory = TRUE)` lets a smaller `B` (2 or more) run for a first look or a demonstration; every such call gives a warning of class `dmar_bootstrap_exploratory`, and the line under the table says so (see `dmar_bootstrap`). The rows `p_bootstrap_ci_lower` and `p_bootstrap_ci_upper` are the Wilson interval for the Monte Carlo error of the *p*-value at `conf_level`, the uncertainty that comes from using `B_used` samples rather than infinitely many (`ci_proportion`); it is not an interval for anything about the model. The row `bootstrap_mean` is the mean of the used statistics, the bootstrap estimate of the expected statistic under the model, which the Bollen-Stine adjustments of fit indices use in place of the degrees of freedom.

What Is Refused: A fit is refused when it did not converge (an error of class `dmar_not_converged`) or has an inadmissible solution (a refusal of class `dmar_inadmissible` that names the negative variance or the matrix at fault), as for every fit measure of the SEM family (see `sem_fit_measures`, which explains the check and how it treats auxiliary variables). It is refused with an error of class `dmar_exact_fit_bootstrap_scope`, whose `reason` element names the case (see `dmar_metadata`), when it was made from summary statistics rather than raw data (there are no rows to transform); when it has ordered indicators or was fitted by an estimator other than maximum likelihood (the transformation is derived for the normal theory fit function); when it is a two-level model, has a cluster variable (the resamples draw rows independently, which the refit could not undo) or sampling weights, uses two-stage estimation (whose statistic is not the likelihood ratio statistic the bootstrap refits), or conditions on observed exogenous variables (`fixed.x = TRUE` with such variables, or `conditional.x = TRUE`: the transformation moves every variable, and the conditional likelihood holds some fixed; refit with `fixed.x = FALSE`, and with `conditional.x = FALSE` too, since **lavaan** does not take the first beside `conditional.x = TRUE`); and when its analysis data keep missing values without FIML (as under pairwise deletion) or it has no standard test statistic. A missing data pattern with no more rows than observed variables is refused for the pattern transformation with an error of class `dmar_pattern_too_small`, which carries the pattern, its counts, and its group, and a transformation that is undefined for the data

(a moment that is not positive definite, a Newton iteration that does not converge) with one of class dmar_transformation_undefined. These three classes, and dmar_inadmissible, inherit dmar_refusal; dmar_not_converged is caught by its own name. The checks of B, conf_level, and ncpus are argument checks and give plain errors. A fit whose cases were removed by listwise deletion is bootstrapped on its complete cases, as it was fitted, with a warning. A fit by MLM, MLMV, or MLR is accepted: its estimates are the ML estimates, and the bootstrap refits and resamples its standard statistic.

Value

A dmar_tbl (class dmar_sem_fit) with columns term and value and, always in this order, the rows chi_square, df, p_chi_square (the asymptotic *p*-value of the standard statistic), p_bootstrap, p_bootstrap_ci_lower, p_bootstrap_ci_upper, bootstrap_mean, B, B_used, B_improper, B_failed, N, and n_groups. The attributes B, B_used, n_dropped, B_required (100), and bootstrap_status record the samples in the names every bootstrap result of the package carries. The attribute bootstrap is a data frame with one row per bootstrap sample, its chi_square (NA when it was set aside) and its status (proper, improper, not_converged, or error); the attribute transformation records, per group, the transformation applied, the number of missing data patterns, the Newton steps and the largest estimating equation at the solution (on the standardized scale), and the largest difference between the transformed means and the model implied ones; and the sem_fit attribute records the statistic and its degrees of freedom, the transformation requested and the ones applied, the counts, and the lines of the printed footer.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bollen, K. A., & Stine, R. A. (1992). Bootstrapping goodness-of-fit measures in structural equation models. *Sociological Methods & Research, 21*(2), 205–229. doi:10.1177/0049124192021002004

Enders, C. K. (2002). Applying the Bollen-Stine bootstrap for goodness-of-fit measures to structural equation models with missing data. *Multivariate Behavioral Research, 37*(3), 359–377. doi:10.1207/S15327906MBR3703_3

Savalei, V., & Yuan, K.-H. (2009). On the model-based bootstrap with missing data: Obtaining a *p*-value for a test of exact fit. *Multivariate Behavioral Research, 44*(6), 741–763. doi:10.1080/00273170903333590

See Also

sem_fit_measures for the asymptotic test of exact fit beside the other fit measures, and equivalence_rmsea for the test of not close fit.

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_fit_measures(), sem_information_criteria(), sem_residual_fit(), simple_structure()

Examples

```
# Two correlated factors for six of the holzinger_swineford tests, with
# the cube and sentence scores of about a third of the students missing
# at random given the general information score, which the FIML fit
# takes as an auxiliary variable. lavaan notes that it adds the
# auxiliary variable and warns that the residual covariance matrix,
# which includes it, is not positive definite; the model's own solution
# is proper (see What Is Refused). B = 10 keeps the example quick and
# is below the 100 a reported p-value needs; the option lets it run
# and the warning says so. Use the default of 1000 or more in an
# analysis.
d <- holzinger_swineford[c("t1_visual_perception", "t2_cubes", "t4_lozenges",
                           "t6_paragraph_comprehension", "t7_sentence",
                           "t9_word_meaning", "t5_general_information")]
set.seed(113)
drop <- runif(nrow(d)) < plogis((d$t5_general_information - 30) / 10) - 0.35
d[drop, c("t2_cubes", "t7_sentence")] <- NA
fit <- lavaan::cfa(
  "visual  =~ t1_visual_perception + t2_cubes + t4_lozenges
   textual =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning",
  data = d, missing = "ml", std.lv = TRUE, aux = "t5_general_information")
old <- options(dmar.bootstrap_exploratory = TRUE)
sem_exact_fit_bootstrap(fit, B = 10, seed = 113)
options(old)
```

---

sem_fit_measures *Fit Measures for a Structural Equation Model*

Description

Reports in one table the measures researchers use to judge the global fit of a structural equation model: the chi square test of exact fit, the root mean square error of approximation (RMSEA) with its confidence interval and the tests of close and not close fit, the incremental indices CFI, TLI, NFI, IFI, and RNI, the SRMR, and gamma hat. The function takes either a fitted **lavaan** model, whose own statistics, sample size convention, and corrected (robust) versions it reads, or the summary statistics a paper reports, so that a published fit table can be reproduced and extended with an interval for the RMSEA and the tests it implies. A fitted model requires **lavaan** (in Suggests).

Usage

```
sem_fit_measures(
  fit = NULL,
  chisq = NULL,
  df = NULL,
  N = NULL,
  baseline_chisq = NULL,
  baseline_df = NULL,
```

```
  p = NULL,
  n_groups = 1,
  conf_level = 0.9,
  close_fit = 0.05,
  not_close_fit = 0.05,
  version = c("auto", "standard", "scaled", "robust"),
  sample_size_convention = c("N-1", "N")
)
```

Arguments

`fit`	A fitted **lavaan** model. It must have converged to an admissible solution, as Details describes (lavaan's post-estimation check, made on the model's own parameters when the fit has auxiliary variables). Supply either `fit` or the summary statistics below, not both.
`chisq`, `df`	For summary statistic input, the model's chi square test statistic T and its degrees of freedom: the likelihood ratio statistic of a normal theory maximum likelihood (ML) fit. Summary input gives the standard measures only.
`N`	For summary statistic input, the total sample size, summed over groups: a whole number greater than `n_groups`.
`baseline_chisq`, `baseline_df`	For summary statistic input, the chi square statistic and degrees of freedom of the baseline (independence) model fitted to the same data. The incremental indices need them and are `NA` without them. The baseline model is nested in the target model, so `baseline_df` must be positive and at least `df`.
`p`	For summary statistic input, the number of observed variables (not the number of parameters). Gamma hat and adjusted gamma hat need it and are `NA` without it; they are defined here for a single group and a covariance structure (see Details), so they are `NA`, with a warning, when `df` exceeds the $p(p+1)/2$ covariance moments, and `df` can be at most $p(p+3)/2$. Below that bound summary input cannot show that a model restricts the means, and supplying `p` states that it does not.
`n_groups`	For summary statistic input, the number of independent groups of a multiple group model, whose `chisq`, `df`, and `N` are totals over the groups. Default 1. A fitted model supplies its own number of groups.
`conf_level`	Confidence level of the RMSEA interval and of the gamma hat and adjusted gamma hat intervals, which come from the same noncentrality limits. Defaults to 0.90, the conventional level of the RMSEA interval, whose two limits decide the tests of close and not close fit at .05 (see Details); `conf_level = 0.95` gives 95 percent intervals.
`close_fit`	The RMSEA value ϵ_0 under the null hypothesis of close fit, $H_0 : \epsilon \leq \epsilon_0$. Default 0.05.
`not_close_fit`	The RMSEA value ϵ_1 under the null hypothesis of not close fit, $H_0 : \epsilon \geq \epsilon_1$. Default 0.05.
`version`	Which version of the measures to report for a fitted model: `"auto"` (the default), `"standard"`, `"scaled"`, or `"robust"`. See Details. Summary statistic input accepts `"auto"` and `"standard"`, which coincide.

`sample_size_convention`
: For summary statistic input, the multiplier a in $T = a\hat{F}$ with which `chisq` was formed: `"N-1"` (the default), each group's sample size less one, so $a = N - G$ for G groups, or `"N"`, so $a = N$. lavaan's default normal likelihood and Mplus form T with N, so their statistics take `"N"`. A fitted model supplies its own convention.

Details

Exact Fit and the RMSEA: Write T for the model's chi square statistic on df degrees of freedom, a for the sample size multiplier with which it was formed ($T = a\hat{F}$, where $\hat{F}$ is the minimized fit function), G for the number of groups, and α for 1 minus `conf_level`. The p-value of the test of exact fit (row `p_chi_square`) is the upper tail of the central chi square distribution at T. Under a model that is wrong by an amount small relative to sampling error, T is approximately noncentral chi square with noncentrality $\lambda = a \cdot df \cdot \epsilon^2/G$, where ϵ is the population RMSEA, the misfit per degree of freedom on the scale of the fit function (Steiger & Lind, 1980; Browne & Cudeck, 1993). The point estimate is $\hat{\epsilon} = \sqrt{G \max(T - df, 0)/(a \cdot df)}$, and the confidence interval inverts the noncentral chi square distribution at T (`ci_nc_chisq` does the inversion) and maps its noncentrality limits to the RMSEA metric in the same way. The factor G is Steiger's (1998, Eqs. 23 and 24) multiple group RMSEA, which divides the discrepancy by the average degrees of freedom per group; applying the single group formula to a multiple group statistic gives a value too small by a factor of $\sqrt{G}$ (Steiger, 1998, Eq. 25).

When T falls at or below the upper $\alpha/2$ quantile of the central chi square distribution, the lower limit is 0. When it falls at or below the lower $\alpha/2$ quantile, no noncentrality is consistent with T at that level, and the upper limit is reported as 0, as lavaan reports it and as Yuan, Chan, Marcoulides, and Bentler (2016, Eq. 4) define the corresponding T-size RMSEA, so that the upper limit and the test of not close fit always agree. This happens with probability $\alpha/2$ for a model that fits exactly, and more often when the normal theory statistic is deflated, for example by data with light tails.

The RMSEA is conventionally reported with a 90 percent interval, the one Steiger and Lind (1980) formed from the .05 and .95 percentile points of the noncentrality, whose details Browne and Cudeck (1993) give and which lavaan's summary prints. Its two limits are the two one-sided tests at level .05 described below: a $100(1 - 2\alpha)$ percent interval settles both tests at level α, so the lower limit exceeds ϵ_0 exactly when close fit is rejected, and the upper limit falls below ϵ_1 exactly when not close fit is rejected (MacCallum, Browne, & Sugawara, 1996, p. 136 and Table 1, with one bound playing both roles). A lower limit of zero likewise says that the test of exact fit does not reject at .05. The default `conf_level = 0.90` therefore gives the conventional interval, which reproduces lavaan's printed limits, and every RMSEA interval in DMAR defaults to it. `conf_level = 0.95`, the level of the package's other intervals, gives a wider interval whose limits decide the two tests at .025 rather than at .05. The printed table states the level beneath it.

A statistic in the millions, as from a large model fitted to a very large sample, puts the noncentrality where R's noncentral chi square distribution function, which lavaan uses, loses its accuracy (above about 10^5) and then fails (above about 2×10^6, where it once returned an interval collapsed to a point below its estimate). The limits and the tests of close and not close fit are then computed here, from the `chi_square` and `df` rows, with the Poisson mixture described in `ci_nc_chisq`; for a robust version whose corrected statistic DMAR cannot rebuild (FIML, ordered data, or two-stage estimation) they are `NA`, with a warning, once the statistic exceeds 10^6.

Tests of Close and Not Close Fit: The test of close fit (Browne & Cudeck, 1993) asks whether the data can reject $H_0 : \epsilon \leq \epsilon_0$; its p-value is $P\{\chi^2_{df}(\lambda_0) \geq T\}$ with $\lambda_0 = a \cdot df \cdot \epsilon_0^2/G$ (row

`p_close_fit`). Failing to reject close fit does not establish it. The test of not close fit (MacCallum, Browne, & Sugawara, 1996) reverses the roles: it asks whether the data can reject $H_0 : \epsilon \geq \epsilon_1$, so that a small p-value, $P\{\chi^2_{df}(\lambda_1) \leq T\}$ with $\lambda_1 = a{\cdot}df{\cdot}\epsilon_1^2/G$ (row `p_not_close_fit`), is evidence that the population RMSEA is below ϵ_1. It is the equivalence test of `equivalence_rmsea`, which reports it with its interval and verdict. MacCallum et al. (1996) used 0.05 for both bounds, the defaults here; lavaan's default bound for not close fit is 0.08, which `not_close_fit = 0.08` reproduces. These values are conventions for framing a test, not cutoffs that certify a model.

Incremental Indices: The incremental indices compare the model with a baseline model, by default the independence model, whose statistic and degrees of freedom are T_b and df_b. The comparative fit index is $CFI = 1 - \max(T - df, 0)/\max(T - df, T_b - df_b, 0)$, equal to 1 when both are zero (Bentler, 1990). Without the truncation at zero it is the relative noncentrality index, $RNI = 1 - (T - df)/(T_b - df_b)$ (Bentler, 1990; McDonald & Marsh, 1990), which can fall outside $[0, 1]$. The Tucker-Lewis index (Tucker & Lewis, 1973), also called the nonnormed fit index (Bentler & Bonett, 1980), is $TLI = (T_b/df_b - T/df)/(T_b/df_b - 1)$; it rewards parsimony and can also exceed 1. The normed fit index is $NFI = (T_b - T)/T_b$ (Bentler & Bonett, 1980), and the incremental fit index is $IFI = (T_b - T)/(T_b - df)$ (Bollen, 1989, Eq. 3). No index is clipped to $[0, 1]$, and an index whose denominator is zero or negative is `NA`. Brosseau-Liard and Savalei (2014, Eqs. 3 to 10) give the two population values these sample indices estimate: the NFI, CFI, RNI, and IFI all converge to the population CFI, $1 - F_H/F_B$ for the population ML discrepancies of the model and the baseline (the NFI poorly, since it does not subtract the degrees of freedom), and the TLI to the population TLI, $1 - (F_H/F_B)(df_B/df_H)$. None of them has a confidence interval here: the usual asymptotic theory assumes small misfit, which the independence model violates.

A fitted model with no degrees of freedom reproduces the moments exactly, so its statistic is zero in every version, and so is the model term $T - c \cdot df$ of the robust indices, whatever the scaling constant c, which is 0/0 there (lavaan's robust and scaled indices of such a model come out `NA` or not by rounding). Its CFI is therefore 1, and its RNI, NFI, and IFI are their values at a statistic of zero, those of lavaan's standard version, in every version that has them; its TLI, like the RMSEA, divides by the degrees of freedom and is `NA`. A model with no degrees of freedom need not fit exactly: an inequality constraint that binds, such as an error variance held at zero by `bounds = "pos.var"`, fixes its parameter as an equality would, though lavaan does not count it among the degrees of freedom. Such a model's statistic is not zero (32.8 for one factor of three variables correlated .8, .8, and .5, N = 200), and its CFI, RNI, NFI, and IFI are formed from it (.925, .925, .926, and .926), with the standard NFI computed as $1 - T/T_b$ where lavaan reports 1 at zero degrees of freedom; the TLI and the RMSEA rows are `NA` as before, and the warning and the footer say that the model does not fit exactly.

SRMR and Gamma Hat: The standardized root mean square residual (row `srmr`) is the root mean square of the covariance residuals, each divided by the product of the two sample standard deviations (lavaan's `srmr_bentler_nomean`). It is read from a fitted model and is `NA` for summary input. It is the same quantity in every version, since no test statistic enters it. lavaan's own `srmr` also averages in the mean residuals when the model has a mean structure, which makes it smaller by about $\sqrt{t/(t+p)}$, for $t = p(p+1)/2$ covariance moments of p variables, when the means are saturated, as under FIML; this table keeps the covariance definition throughout. With auxiliary variables (**lavaan**'s `aux` under FIML) the SRMR is over the model's own variables, the auxiliary variables set aside as in the admissibility check below. **lavaan**'s measures average over them as well, and an auxiliary variable, a saturated correlate, adds residuals of about zero that lower the SRMR of the same model: by about 8 percent with one auxiliary variable and 15 percent with two

for a three-factor model of nine items. The test statistic, and the measures built from it, compare the model with the saturated model with the auxiliary variables in both, so they need no such change. For the unbiased estimate of the SRMR, its interval, and its tests see `sem_residual_fit`.

Gamma hat rescales the estimated noncentrality to a goodness of fit metric, $\hat{\gamma} = p/[p + 2(T - df)/a]$ for p observed variables, and adjusted gamma hat penalizes it for the degrees of freedom used, $1 - [p(p + 1)/(2df)](1 - \hat{\gamma})$, counting the $p(p + 1)/2$ covariance moments of one group. They estimate the population values of the goodness of fit index (GFI) and the adjusted GFI that LISREL reports, which Steiger (1989) showed to be these functions of the population noncentrality for a model invariant under a constant scaling factor (a result he restates in Steiger, 1990, p. 178). Neither is truncated, so both exceed 1 when $T < df$. Each is a monotone function of the noncentrality, so the noncentrality limits of the RMSEA interval give their limits, $p/(p + 2\lambda/a)$ at each limit λ and the adjusted index of that (Steiger & Fouladi, 1997, pp. 251–252), in the rows `gamma_hat_ci_lower` through `adjusted_gamma_hat_ci_upper`. The limits are at most 1, the upper one is 1 when the lower RMSEA limit is 0, and a point estimate above 1 lies above its interval.

This gamma hat is the index lavaan 0.7-3 reports as `gfi_lrt`, with the same limits; lavaan caps the point estimate at 1 and reports it only when it is requested by name. The `gfi` lavaan prints by default is the same function of Browne's residual based (reweighted least squares) statistic in place of T, which differs from gamma hat in a finite sample (0.959 against 0.957 for the three-factor model of the examples), and lavaan's `gfi_lisrel` and `agfi_lisrel` are the sample GFI and adjusted GFI of LISREL, not estimates of their population values. lavaan also reports robust versions, and so does this table: for a robust ML fit of complete continuous data with a Satorra-Bentler, Yuan-Bentler, scaled and shifted, or mean and variance adjusted statistic (MLM, MLR, MLMV, MLMVS), the robust version estimates the noncentrality as $s(T_s - df_s)$, for the corrected statistic T_s, its scaling constant s, and its degrees of freedom df_s, which is $T - c \cdot df$ in each case (see the robust version below), the estimate behind the robust RMSEA of Brosseau-Liard, Savalei, and Li (2012), and takes its limits from the robust RMSEA interval; for the first three this is lavaan's `gfi_lrt.robust` and its limits.

Steiger's result is for covariance structures. Neither index is computed for a fitted model that restricts the means, whose statistic and degrees of freedom include the misfit and the restrictions of the means while p and $p(p+1)/2$ count the covariances: a growth model, or intercepts constrained equal (by a shared label, with or without lavaan's `ceq.simple = TRUE`), constrained to a constant, entered in a linear constraint, tied to a loading, or held at the bound of an inequality constraint (which lavaan's degrees of freedom do not count). Whether a fit restricts its means is read from the derivatives of its model implied moments, with the constraints that bind at the estimates imposed (the equality constraints, and the inequality constraints lavaan treats as active when it computes the standard errors): the means are free when the parameter directions that leave every covariance unchanged move the means in as many independent directions as there are modeled means, so that the means add no degree of freedom (the means of exogenous variables under `fixed.x = TRUE` are fixed at their sample values and are not counted). A mean structure whose means are free, as lavaan's default parameterization leaves them or as one indicator's intercept fixed at zero with the latent mean free leaves them, changes neither index. Summary input carries no mean structure to read, so it shows restricted means only when `df` exceeds $p(p + 1)/2$, which only restricted means can produce, and the indices are then `NA`; below that bound they are computed from the summary statistics on the understanding that the model is a covariance structure (the statistic of a linear growth model of four waves, 12.2 on 5 df with N = 400, gives 0.991 and 0.982, which the fitted model does not). The two indices are computed for the standard and robust versions just described of a single group, single level ML fit of continuous variables without FIML or sampling weights,

and for summary input when p is given, and are NA otherwise.

Standard, Scaled, and Robust Versions: A fitted model can carry more than one test statistic, and lavaan computes the fit measures from each.

"standard" The measures above computed from the uncorrected statistic. They are the right ones for a normal theory ML fit of complete continuous data.

"scaled" The same formulas with a scaled (or mean and variance adjusted) statistic T_s on its degrees of freedom df_s put in place of T and df: the RMSEA and its interval invert the noncentral chi square at T_s on df_s, the chi_square and df rows of the same table (Brosseau-Liard, Savalei, & Li, 2012, Eqs. 11 and 13, where $df_s = df$ for a scaled statistic; Savalei, 2018, Table 1), and the tests of close and not close fit, which those sources do not treat, follow the same substitution so that they keep their duality with the interval. This is the substitution EQS and Mplus report, and the one lavaan reported when Brosseau-Liard et al. (2012, p. 911) wrote, and the one a published table of scaled measures was computed with. It is not lavaan's default for a Satorra-Bentler or Yuan-Bentler statistic (MLM, MLR) or a mean and variance adjusted one (MLMVS): lavaan 0.7-3 refers the unscaled T to the noncentral chi square on $c \cdot df$ degrees of freedom, the trace of $U\Gamma$ for scaling constant c, which gives the same point estimate but a different interval and different close fit *p*-values from those the rows beside them imply. A fit made with lavaan's rmsea.scaled.mplus = TRUE gives the values reported here, as lavaan's default does for a scaled and shifted statistic (MLMV, WLSMV). The scaled RMSEA and CFI do not consistently estimate the population values the standard formulas define (Brosseau-Liard et al., 2012; Brosseau-Liard & Savalei, 2014; Savalei, 2018, 2021); they are offered to reproduce a published table.

"robust" lavaan's corrected measures, which estimate the population values of the standard measures: for a robust ML fit of complete continuous data (MLM, MLR), the nonnormality corrections of Brosseau-Liard, Savalei, and Li (2012) for the RMSEA and of Brosseau-Liard and Savalei (2014) for the CFI, TLI, and RNI, with the correction of Savalei (2018) for a scaled and shifted statistic (MLMV). The point estimates do not depend on which corrected statistic accompanies them: each subtracts from the ML statistic T its asymptotic expectation $c \cdot df$, with c the trace of $U\Gamma$ over df (Brosseau-Liard et al., 2012, Appendix; Savalei, 2018, Eq. 6). lavaan computes no robust measures beside the older mean and variance adjusted statistic (MLMVS, test = "mean.var.adjusted", the statistic Savalei, 2018, Appendix B, writes T_{MVold} and no longer recommends; her mean and variance adjusted statistic is the scaled and shifted one), whose scaling constant s and degrees of freedom df_s give $c = s \cdot df_s/df$, so they are computed here, for the model and for its baseline, with the interval and the tests of close and not close fit from T/c on df (Brosseau-Liard et al., 2012, Eq. 10); they equal lavaan's robust measures for the same model fitted by MLM, and the footer says that DMAR computed them (for a saturated model, whose c is 0/0 and whose robust CFI and RNI are 1 at a statistic of zero, nothing is computed from c, and the footer names the corrections alone, as for MLM). For ordered categorical data (WLSMV), the correction of Savalei (2021); and for a FIML fit (missing = "ml"), the FIML corrected indices of Zhang and Savalei (2023). The last are not corrections for nonnormality: they evaluate the complete data ML fit function at the FIML estimates, with the saturated FIML estimates in place of the sample moments, and so approximate the RMSEA and CFI the model would have had with complete data, whereas the uncorrected FIML indices change with the amount of missing data. lavaan uses their version 3, which does not assume normality, when the fit has a scaled statistic (MLR), and their version 6, which does, otherwise. The corrected values rest on the corrected statistic, which under FIML differs from the chi square shown in the table. After two-stage estimation

they are the corrected indices of Zhang and Savalei (2023), which correct the complete data indices of the second stage for the missing data: after robust two-stage estimation (`missing = "robust.two.stage"`) their TS V2, the version they recommend, and after normal theory two-stage estimation (`"two.stage"`) its normal theory analog, which takes the asymptotic covariance matrix of the stage one moments to be the inverse of their observed information where TS V2 takes the sandwich of the observed and first order information, and which they did not study (see `fa_confirmatory`). Under FIML and two-stage estimation alike, Zhang and Savalei define only the RMSEA and the CFI; the robust TLI and RNI, the RMSEA interval, and the tests of close and not close fit are lavaan's extension of the correction. NFI and IFI have no robust version, in lavaan or here, and are `NA`; gamma hat has the one described above.

The default `"auto"` reports the robust version for a fit that carries a corrected test statistic (MLM, MLR, WLSMV, a sampling weighted fit with lavaan's default test, and the like) or uses FIML, and the standard version otherwise. A fit to ordered data by DWLS or ULS has only the standard statistic unless a corrected test is requested (it is the default test of `estimator = "DWLS"` and `"ULS"`, which `"WLSMV"` and `"ULSMV"` extend with a corrected one), and lavaan gives that statistic no reference distribution, so `"auto"` reports its standard measures with a warning that the statistic is not chi square distributed and that a corrected test is needed for the robust measures. The same holds for a DWLS fit of continuous data whose test was set to `"standard"` (lavaan's default test for continuous data adds Browne's residual based statistic, which is chi square under normality and is then the one reported; the warning says to leave out `test = "standard"` or to use a corrected test), for a ULS fit of continuous data whose test was set so (see *ULS Fits of Continuous Data*), and for a sampling weighted fit whose test was set to `"standard"`, whose statistic is a weighted pseudo likelihood ratio statistic. Full WLS is the exception among the estimators for ordered data: its weight matrix is the inverse of the asymptotic covariance matrix of the thresholds and polychoric correlations, its statistic is asymptotically chi square (Browne, 1984; Muthén, 1984), lavaan reports its *p*-value, and `"auto"` reports its standard measures without a warning (a request for the robust version says that they are the ones to report). The reason in the footer names the estimator. The measures of a ULS fit of continuous data are reported only where they do not depend on the units of the variables (see *ULS Fits of Continuous Data*). Browne's residual based statistic, which lavaan computes beside the standard one, is an alternative test and not a correction, and does not count. When lavaan provides a corrected statistic but no robust measures for it (and DMAR cannot compute them), `"auto"` reports, with a warning, the standard version for a continuous ML fit, whose standard measures estimate the population values of the ML measures as the robust ones do and the scaled ones do not (Savalei, 2018), though their interval and tests assume normal data, and the scaled version for any other fit. The `chi_square`, `df`, and `p_chi_square` rows show the scaled statistic under the scaled and robust versions when the fit has one, and the standard statistic otherwise. The `sem_fit` attribute and the printed footer name the version reported, why, and the statistic used.

ULS Fits of Continuous Data: The corrections of Brosseau-Liard and colleagues target the population discrepancy of the ML fit function. Their use with another fit function needs separate justification (Savalei, 2018). For continuous ULS, `version = "robust"` is therefore refused for the entire robust family, including CFI, TLI, and RNI. Invariance to multiplying every variable by the same constant is not enough to establish the population interpretation of an incremental index. This is a conservative supported-method policy, not a claim that Savalei proved every continuous-ULS index invalid. Ordered-data corrections with their own derivation are handled separately.

With ULSM or ULSMV, `version = "auto"` selects eligible scaled indices. These put a corrected test statistic into the index formulas; they are descriptive approximations, do not estimate ML-population fit, and have no established license to use conventional ML cutoffs. When the fit restricts means, incompatible units in the mean and covariance portions of the ULS discrepancy can make these indices change even under a common rescaling. Those rows are withheld, with the existing mean-structure qualification.

The standard statistic NF_{ULS} is in squared-covariance units. Its RMSEA and incremental indices that subtract degrees of freedom change with those units and are `NA`; the standard NFI, a ratio of two such statistics, remains available when eligible. This does not apply to a separately justified residual-based statistic. A fit by `estimator = "ULS"` alone is tested in lavaan 0.7-3 by Browne's normal theory residual based statistic, which weights the residuals by the inverse of their normal theory covariance matrix and has no units as long as the estimates at which it is evaluated do not change with the units either. They do not when the model leaves the means free, or restricts them through parameters the covariances do not share, as the equal intercepts above do, and the standard measures formed from that statistic are then reported as they are. When the model restricts the means through parameters the covariances share, as the loadings enter the means through the latent means of a multiple group model whose loadings and intercepts are equal (the scalar rung of `measurement_invariance`), the ULS fit function trades the mean residuals against the covariance residuals, which are in other units, so the estimates change with the units, and so do Browne's statistic and every measure formed from it: on the scalar rung of the first example of `measurement_invariance`, the CFI is .8874 for the raw scores and .8861 for the scores divided by 10, and the RMSEA .0680 and .0684. The RMSEA, its interval, the tests of close and not close fit, and the CFI, TLI, RNI, NFI, and IFI are then `NA`, with the same warning, element, and line; the `chi_square` row, whose chi square reference distribution holds at any root-n consistent estimates (Browne, 1984, Proposition 4), and the SRMR, which describe the fit at hand, are reported. Whether the two parts share a parameter is read from the same derivatives as whether the means are restricted: they share one when the rank of the derivatives of every model implied moment falls short of the rank of the covariance rows plus that of the mean rows. With ordered data the robust measures of a fit with `"ULSMV"` are the categorical correction of Savalei (2021), formed from polychoric correlations, which have no units, and none of this applies. Maximum likelihood is scale invariant; `estimator = "MLR"` is the estimator for nonnormal continuous data.

Two-Stage Fits: A fit by two-stage maximum likelihood (Yuan & Bentler, 2000; Savalei & Bentler, 2009) fits the model to the EM estimates of the means and covariances as if they were the moments of complete data. lavaan's standard statistic for it is that naive one, which is not chi square distributed, so its standard measures (TS V0 in the terms of Zhang & Savalei, 2023) are refused, and so are its scaled measures, which put an approximate statistic into the standard formulas; the default `"auto"` reports the robust version. The `chi_square` row is then the residual based statistic of Savalei and Bentler (2009, Eq. 8; Eq. 14 with auxiliaries) after normal theory two-stage estimation (`missing = "two.stage"`), computed by DMAR from lavaan's moments and the observed information of the saturated model, and lavaan's scaled statistic after robust two-stage estimation (Savalei & Falk, 2014): the statistics `fa_confirmatory` and `measurement_invariance` report for the same fits. The indices are not computed from either statistic but from the naive one and its expectation, as Zhang and Savalei (2023) define them: TS V2 after robust two-stage estimation and its normal theory analog after normal theory two-stage estimation.

A case observed on an auxiliary variable (lavaan's `aux`) but on none of the model's variables is read by the first stage, whose EM estimates of the means and covariances it moves, and is left out of the sample size lavaan 0.7-3 reports (`lavaan::lavInspect(fit,"nobs")`), which counts it as empty. The naive statistic, and with it lavaan's scaled statistic, the corrected indices, the RMSEA

multiplier, and the N row, then rest on fewer cases than the moments, while the residual based statistic of a normal theory two-stage fit uses every case, so the rows of the table do not describe one set of cases (and lavaan's two-stage standard errors are too large by the square root of the ratio of the two counts). sem_fit_measures() reports the rows of a fit as **lavaan** computed them, and warns when a fit has such cases and counts them. fa_confirmatory(missing = "two_stage") and measurement_invariance keep such cases and compute their fits again with the sample size set to every case the first stage read, so that the fits they return, and the tables of this function for them, count every case; FIML (missing = "ml") with the same auxiliary variables keeps them as well. Setting the cases aside instead would condition the analysis on an observed variable of the model, which biases the estimates when the items went missing together at random given the auxiliary variables.

The Sample Size Convention: The multiplier a must be the one with which T was formed. A fitted model supplies it: lavaan forms $T = N\hat{F}$ for ML under its default normal likelihood, and $T = (N - G)\hat{F}$ under likelihood = "wishart" and for the least squares estimators, and the table follows the fit, so its values equal lavaan's (for the scaled version, those lavaan gives under rmsea.scaled.mplus = TRUE; see above). For summary input the default "N-1" is the convention of Browne and Cudeck (1993), MacCallum et al. (1996), Kelley and Lai (2011), and DMAR's ci_rmsea, ss_aipe_rmsea, and ss_power_sem. A statistic reported by lavaan under its default likelihood or by Mplus was formed with N and takes sample_size_convention = "N". The wrong multiplier changes the RMSEA, its limits, the close fit tests, and gamma hat by an amount of order $1/N$.

What Is Refused: A fitted model is refused when it did not converge (an error of class dmar_not_converged) and when its solution is inadmissible (a refusal of class dmar_inadmissible): fit measures of such a solution describe estimates outside the parameter space and do not speak to the specified model. A solution is inadmissible when **lavaan**'s post-estimation check finds a negative variance estimate (a Heywood case among them) or a covariance matrix that is not positive definite: that of the latent variables, that of the residuals of the observed variables (theta), or that of the residuals of the structural part of the model (psi). The message names the variable or the matrix, and the condition carries them (see dmar_metadata). With auxiliary variables (lavaan's aux under FIML) the check is made on the model's own parameters, with the auxiliary variables set aside: lavaan models an auxiliary variable as a saturated correlate, whose covariances with the items run through their residuals, and its own check of theta, which includes those covariances, fails once an auxiliary variable is related to a factor strongly enough, however proper the model's own solution. For one factor whose items have uncorrelated residuals, and an auxiliary variable related to the items only through the factor, that happens when the squared correlation between the auxiliary variable and the factor exceeds $(1 - H)/H$, with H the maximal reliability of the factor's items (coefficient *H*; see reliability_H), which is a correlation above .33 when H is .9. Auxiliary variables are recognized when they are given through lavaan's aux argument; saturated correlates written into the model by hand, as semTools::cfa.auxiliary() writes them, stay in theta and are judged with it, so refit with aux to have them set aside. The same check serves sem_residual_fit, sem_information_criteria, compare_sem, ecvi, and sem_exact_fit_bootstrap, and ci_bifactor_indices judges its fit and its replications by it. Resolve the improper solution first, by changing the model or by judging that a negative variance is within sampling error of zero and fixing it, or pass the reported statistics as summary input if you accept that responsibility. Multilevel fits are refused, since one set of indices mixes the two levels; lavaan's level specific fit measures (lavaan::fitMeasures(fit, level = ...)) are the tool for them. A fit whose cases were removed by listwise deletion is evaluated on the complete

cases with a warning, since that assumes the data are missing completely at random; refit it with `missing = "ml"` to use every case.

For a fit by normal theory two-stage estimation (`missing = "two.stage"`) the residual based statistic, and with it the table, is refused when the fit has sampling weights or clustered cases, conditions on an observed exogenous variable (`fixed.x = TRUE`, the default of `lavaan::sem()` for a model that regresses on one, or `conditional.x = TRUE`), has inequality constraints or a parameter on a bound, or has auxiliary variables that the installed lavaan does not keep where DMAR reads them. Each message says what to do instead; robust two-stage estimation, whose scaled statistic lavaan computes, is the alternative throughout. A fit that conditions on an exogenous variable is refused with an error of class `dmar_two_stage_fixed_x`, which carries the exogenous variables, and the advice to refit with `fixed.x = FALSE`, and with `conditional.x = FALSE` too, since **lavaan** does not take the first beside `conditional.x = TRUE`. The refit leaves the estimates as they are when the exogenous variables are complete, and otherwise also keeps the cases **lavaan** deletes for a missing exogenous variable when it conditions on them (robust two-stage estimation deletes them too), so that its estimates differ; to keep the fit's estimates, refit on the rows the fit used, those on which every exogenous variable is observed, which `lavaan::lavInspect(fit, "case.idx")` lists.

Reading the Measures: No row of the table certifies a model, and the table reports no cutoff. The conventions in use are descriptions of a population RMSEA: MacCallum et al. (1996, p. 134) credit the view that values below .05 indicate close fit to Steiger (1989) and others, describe .05 to .08 as fair and above .10 as poor after Browne and Cudeck (1993), and call .08 to .10 mediocre themselves. Chen, Curran, Bollen, Kirby, and Paxton (2008) found no empirical support for .05, or any other value, as a universal cutoff: a fixed bound rejects too many correct models in samples of 100 or fewer, too few misspecified ones in samples of 800 or more, and the test of close fit with a fixed bound performs inconsistently across sample sizes, degrees of freedom, and specifications. They recommend reading the RMSEA beside the chi square statistic and the other indices rather than as a single accept or reject decision; the interval and the two tests above are how this table states the uncertainty, and the researcher chooses the bound.

The cutoffs in common use for the other indices, near .95 for the CFI and TLI, .06 for the RMSEA, and .08 for the SRMR, applied in pairs, come from the simulations of Hu and Bentler (1999), who themselves wrote that no single cutoff works equally well across conditions and that the trade-off between the two kinds of error at samples of 250 or fewer cannot be judged from the sample. Marsh, Hau, and Wen (2004) showed that the rationale behind those simulations treated a minor misspecification as a model to be rejected, so that the rules reward rejecting models that fit acceptably and cannot hold across sample sizes; they warn against reading the rules of thumb as golden rules. What a value of an index means depends on the model: on the size of the loadings and unique variances, which lower the chi square, RMSEA, and SRMR of a misspecified model when the loadings are low (Heene, Hilbert, Draxler, Ziegler, & Buehner, 2011); on the number of observed variables, which inflates the statistic and every index formed from it (Moshagen, 2012); and on the reliability of the factors, the number of items and factors, and the degrees of freedom, which is why McNeish and Wolf (2023) derive cutoffs for the model at hand by simulation, at several sizes of misspecification, so that an index reads as an effect size rather than as a verdict. This table reports the indices and their uncertainty; the bound a model is held to is a judgment about the model, stated in advance.

With few degrees of freedom the RMSEA is unstable (Kenny, Kaniskan, & McCoach, 2015): for correctly specified models with 1 to 3 degrees of freedom and $N = 400$, the sample RMSEA exceeded .05 in 13 to 15 percent of samples, against none at 50 degrees of freedom, and its interval

at N = 50 and 1 degree of freedom was more than sixteen times as wide as at N = 1,000 and 100 degrees of freedom. They advise against computing the RMSEA when the degrees of freedom are small and suggest diagnosing poor fit by fitting the models that free the omitted parameters instead (`compare_sem` compares them). The interval in this table makes the instability visible rather than hiding it.

The incremental indices measure fit against the baseline model, and their metric is the baseline's misfit: under maximum likelihood the independence baseline's statistic is $-(N-1)\log|R|$ for the sample correlation matrix R (van Laar & Braeken, 2022). Two correctly specified models with the same chi square and RMSEA distributions have different CFI distributions when their $|R|$ differ, and misspecified models can rank differently on the CFI and on the chi square, so a CFI is comparable across data sets only when the baselines are; van Laar and Braeken recommend reporting $|R|$ and the baseline statistic and reading an incremental index as the share of the baseline's misfit the model removes. A fitted model's baseline statistic and degrees of freedom are `lavaan::fitMeasures(fit, c("baseline.chisq", "baseline.df"))`; summary input takes them as given.

The raw-data H1 screen is sufficient and nonexhaustive. In models with fixed or conditional observed predictors, it screens response variables only, since predictor moments are not free. It does not assess conditional design rank or exact response-on-predictor relations. No detected obstacle is not a proof of likelihood existence or uniqueness.

Value

A `dmar_tbl` (class `dmar_sem_fit`) with columns `term` and `value` and, always in this order, the rows `chi_square`, `df`, `p_chi_square` (the test of exact fit); `rmsea`, `rmsea_ci_lower`, `rmsea_ci_upper`; `close_fit_bound`, `p_close_fit`, `not_close_fit_bound`, `p_not_close_fit`; `cfi`, `tli`, `nfi`, `ifi`, `rni`; `srmr`; `gamma_hat`, `gamma_hat_ci_lower`, `gamma_hat_ci_upper`, `adjusted_gamma_hat`, `adjusted_gamma_hat_ci_lo` `adjusted_gamma_hat_ci_upper`; `N` and `n_groups`. A row the input or the version does not compute is `NA`. The `sem_fit` attribute records the source, the version requested and reported and why, the statistic in the `chi_square` row, the correction behind the robust measures, the sample size convention and multiplier, why rows of a ULS fit of continuous data are `NA` (`uls`; `NA` otherwise), the boundary conventions that applied, and the lines of the printed footer.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bentler, P. M. (1990). Comparative fit indexes in structural models. *Psychological Bulletin, 107*(2), 238–246. doi:10.1037/00332909.107.2.238

Bentler, P. M., & Bonett, D. G. (1980). Significance tests and goodness of fit in the analysis of covariance structures. *Psychological Bulletin, 88*, 588–606.

Bollen, K. A. (1989). A new incremental fit index for general structural equation models. *Sociological Methods & Research, 17*(3), 303–316. doi:10.1177/0049124189017003004

Brosseau-Liard, P. E., & Savalei, V. (2014). Adjusting incremental fit indices for nonnormality. *Multivariate Behavioral Research, 49*(5), 460–470. doi:10.1080/00273171.2014.933697

Brosseau-Liard, P. E., Savalei, V., & Li, L. (2012). An investigation of the sample performance of two nonnormality corrections for RMSEA. *Multivariate Behavioral Research, 47*(6), 904–930. doi:10.1080/00273171.2012.715252

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology, 37*(1), 62–83.

Browne, M. W., & Cudeck, R. (1993). Alternative ways of assessing model fit. In K. A. Bollen & J. S. Long (Eds.), *Testing structural equation models* (pp. 136–162). Sage.

Chen, F., Curran, P. J., Bollen, K. A., Kirby, J., & Paxton, P. (2008). An empirical evaluation of the use of fixed cutoff points in RMSEA test statistic in structural equation models. *Sociological Methods & Research, 36*(4), 462–494. doi:10.1177/0049124108314720

Heene, M., Hilbert, S., Draxler, C., Ziegler, M., & Buehner, M. (2011). Masking misfit in confirmatory factor analysis by increasing unique variances: A cautionary note on the usefulness of cutoff values of fit indices. *Psychological Methods, 16*(3), 319–336. doi:10.1037/a0024917

Hu, L., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling, 6*(1), 1–55. doi:10.1080/10705519909540118

Kelley, K., & Lai, K. (2011). Accuracy in parameter estimation for the root mean square error of approximation: Sample size planning for narrow confidence intervals. *Multivariate Behavioral Research, 46*(1), 1–32. doi:10.1080/00273171.2011.543027

Kenny, D. A., Kaniskan, B., & McCoach, D. B. (2015). The performance of RMSEA in models with small degrees of freedom. *Sociological Methods & Research, 44*(3), 486–507. doi:10.1177/0049124114543236

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130–149. doi:10.1037/1082989X.1.2.130

Marsh, H. W., Hau, K.-T., & Wen, Z. (2004). In search of golden rules: Comment on hypothesis-testing approaches to setting cutoff values for fit indexes and dangers in overgeneralizing Hu and Bentler's (1999) findings. *Structural Equation Modeling, 11*(3), 320–341. doi:10.1207/s15328007sem1103_2

McDonald, R. P., & Marsh, H. W. (1990). Choosing a multivariate model: Noncentrality and goodness of fit. *Psychological Bulletin, 107*, 247–255.

McNeish, D., & Wolf, M. G. (2023). Dynamic fit index cutoffs for confirmatory factor analysis models. *Psychological Methods, 28*(1), 61–88. doi:10.1037/met0000425

Moshagen, M. (2012). The model size effect in SEM: Inflated goodness-of-fit statistics are due to the size of the covariance matrix. *Structural Equation Modeling, 19*(1), 86–98. doi:10.1080/10705511.2012.634724

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika, 49*(1), 115–132.

Savalei, V. (2018). On the computation of the RMSEA and CFI from the mean-and-variance corrected test statistic with nonnormal data in SEM. *Multivariate Behavioral Research, 53*(3), 419–429. doi:10.1080/00273171.2018.1455142

Savalei, V. (2021). Improving fit indices in structural equation modeling with categorical data. *Multivariate Behavioral Research, 56*(3), 390–407. doi:10.1080/00273171.2020.1717922

Savalei, V., & Bentler, P. M. (2009). A two-stage approach to missing data: Theory and application to auxiliary variables. *Structural Equation Modeling, 16*, 477–497. doi:10.1080/10705510903008238

Savalei, V., & Falk, C. F. (2014). Robust two-stage approach outperforms robust full information maximum likelihood with incomplete nonnormal data. *Structural Equation Modeling, 21*(2), 280–302. doi:10.1080/10705511.2014.882692

Steiger, J. H. (1989). *EzPATH: A supplementary module for SYSTAT and SYGRAPH*. SYSTAT.

Steiger, J. H. (1990). Structural model evaluation and modification: An interval estimation approach. *Multivariate Behavioral Research, 25*(2), 173–180.

Steiger, J. H. (1998). A note on multiple sample extensions of the RMSEA fit index. *Structural Equation Modeling, 5*(4), 411–419. doi:10.1080/10705519809540115

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

Steiger, J. H., & Lind, J. C. (1980). *Statistically-based tests for the number of common factors*. Paper presented at the annual Spring meeting of the Psychometric Society, Iowa City, IA.

Tucker, L. R., & Lewis, C. (1973). A reliability coefficient for maximum likelihood factor analysis. *Psychometrika, 38*(1), 1–10.

van Laar, S., & Braeken, J. (2022). Caught off base: A note on the interpretation of incremental fit indices. *Structural Equation Modeling, 29*(6), 935–943. doi:10.1080/10705511.2022.2050730

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*, 165–200.

Yuan, K.-H., Chan, W., Marcoulides, G. A., & Bentler, P. M. (2016). Assessing structural equation models by equivalence testing with adjusted fit indexes. *Structural Equation Modeling, 23*(3), 319–330. doi:10.1080/10705511.2015.1065414

Zhang, X., & Savalei, V. (2023). New computations for RMSEA and CFI following FIML and TS estimation with missing data. *Psychological Methods, 28*(2), 263–283. doi:10.1037/met0000445

See Also

`sem_residual_fit` for the unbiased SRMR with its interval and tests, `equivalence_rmsea` for the test of not close fit with its verdict, `ci_rmsea` for an interval from a reported RMSEA, `sem_information_criteria` and `compare_sem` for comparing models, and the vignette `vignette("sem_fit_measures", package = "DMAR")`.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_information_criteria()`, `sem_residual_fit()`, `simple_structure()`

Examples

```
# The statistics lavaan reports for the three-factor model of nine of
# the holzinger_swineford tests (N = 301), with its baseline.
# lavaan forms the statistic with N under its default likelihood.
sem_fit_measures(chisq = 85.3055, df = 24, N = 301,
                 baseline_chisq = 918.8516, baseline_df = 36, p = 9,
```

```
                sample_size_convention = "N")

# The same model fitted here. The interval is the conventional 90
# percent one, the default, whose limits are the two one-sided .05 tests
# of close and not close fit.
fit <- lavaan::cfa(
  "visual  =~ t1_visual_perception + t2_cubes + t4_lozenges
   textual =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning
   speed   =~ t10_addition + t12_counting_groups_of_dots +
              t13_straight_and_curved_capitals",
  data = holzinger_swineford, std.lv = TRUE)
sem_fit_measures(fit)

# The 95 percent interval, wider, decides the two tests at .025.
sem_fit_measures(fit, conf_level = 0.95)

# A robust ML fit reports the robust measures by default.
fit_mlr <- lavaan::cfa(
  "visual  =~ t1_visual_perception + t2_cubes + t4_lozenges
   textual =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning
   speed   =~ t10_addition + t12_counting_groups_of_dots +
              t13_straight_and_curved_capitals",
  data = holzinger_swineford, std.lv = TRUE, estimator = "MLR")
sem_fit_measures(fit_mlr)

# The model fitted in each of the two schools: the multiple group RMSEA
# needs the number of groups (Steiger, 1998).
sem_fit_measures(chisq = 115.8513, df = 48, N = 301, n_groups = 2,
                sample_size_convention = "N")
```

`sem_information_criteria`

Information Criteria for Structural Equation Models

Description

Computes the information criteria used to choose among structural equation models fitted to the same data: AIC (Akaike, 1974), BIC (Schwarz, 1978), the sample size adjusted BIC (SABIC; Sclove, 1987), Haughton's (1988) BIC (HBIC), and the information matrix based BIC (IBIC) and scaled unit information prior BIC (SPBIC) that Bollen, Ray, Zavisca, and Harden (2012) proposed and Bollen, Harden, Ray, and Zavisca (2014) evaluated for structural equation models. Each criterion is minus twice the maximized log likelihood plus a penalty for the free parameters, smaller values are preferred, and only the difference between two models fitted to the same data under the same likelihood carries meaning. The function takes either a fitted **lavaan** model, whose log likelihood, parameter count, sample size, and (for IBIC and SPBIC) information matrices it reads, or those quantities as summary input, so that a published comparison can be reproduced. `compare_sem` compares a set of fitted models by one criterion. A fitted model requires **lavaan** (in Suggests).

Usage

```
sem_information_criteria(
  fit = NULL,
  log_likelihood = NULL,
  npar = NULL,
  N = NULL,
  measures = c("AIC", "BIC", "SABIC", "HBIC"),
  parameter_estimate = NULL,
  prior_mean = NULL,
  zero_parameters = NULL,
  information_observed = NULL,
  information_expected = NULL,
  information_scale = c("total", "per_observation"),
  spbic_dimension = NULL
)
```

Arguments

`fit`	A fitted **lavaan** model: a maximum likelihood fit of continuous variables under the normal likelihood (estimator ML or its robust variants MLM, MLMV, and MLR) that converged to an admissible solution, as `sem_fit_measures` judges it. See Details for what is refused. Supply either `fit` or the summary input below, not both.
`log_likelihood`	For summary input, the maximized log likelihood ℓ, including its constants (lavaan's `logl`).
`npar`	For summary input, the number d of independent free parameters of the likelihood whose maximum is `log_likelihood`, a positive whole number, counted the same way for every model to be compared. A fitted model supplies lavaan's count, `attr(logLik(fit), "df")`, which leaves out the saturated means of a model without a mean structure and, under `fixed.x = TRUE`, the moments of the exogenous observed variables; a difference between two models with the same settings is unaffected.
`N`	For summary input, the sample size N in the penalties of the BIC family, greater than 1. A fitted model supplies its total over groups.
`measures`	The criteria to compute: any of `"AIC"`, `"BIC"`, `"SABIC"`, `"HBIC"`, `"IBIC"`, and `"SPBIC"`, in either case. The default computes the four that need only ℓ, d, and N. Every row of the table is reported whatever is requested, with `NA` for a criterion not computed.
`parameter_estimate`	For SPBIC from summary input, the vector $\hat{\theta}$ of the d free parameter estimates, in the order of the rows of `information_observed`.
`prior_mean`	For SPBIC, the vector θ^* of prior means, in the order of `parameter_estimate`. Supply this or `zero_parameters`. With a fitted model and neither, the rule of Bollen et al. (2014, p. 6) applies: free loadings and regression coefficients are centered at zero and every other free parameter at its estimate (see Details).

`zero_parameters`
: For SPBIC, the positions or unique names of the entries of `parameter_estimate` to center at zero, every other entry being centered at its estimate. A saturated model needs an explicit choice; Bollen et al. (2014, p. 6) center its covariances at zero and its variances (and means) at their estimates.

`information_observed`
: For SPBIC from summary input, the observed information matrix $I_O(\hat{\theta})$: the negative Hessian of the log likelihood at the estimates, in the free parameters, in the order of `parameter_estimate`.

`information_expected`
: For IBIC from summary input, the expected (Fisher) information matrix $I_E(\hat{\theta})$ at the estimates, in the free parameters. The inverse of a robust (sandwich) covariance matrix of the estimates is not an information matrix and is no substitute.

`information_scale`
: For summary input, whether the supplied information matrices are for the whole sample (`"total"`, the default; the inverse of the covariance matrix of the estimates under ML) or per observation (`"per_observation"`, what `lavaan::lavInspect()` reports), in which case they are multiplied by `N`. A fitted model supplies per observation matrices, which are multiplied by *N*, and the argument is refused with one.

`spbic_dimension`
: Optional: the dimension used in the penalty of SPBIC, a positive whole number no greater than `npar`. By default it is `npar`, as in the general expression. Bollen et al.'s (2014) Table 6 used smaller counts for SPBIC, printed in parentheses (15 for the saturated model rather than 20), and states no rule for them; supply such a count only to reproduce that table, never to tune a ranking. It changes neither `parameter_estimate` nor the information matrices.

Details

The Criteria: Write ℓ for the maximized log likelihood, d for the number of free parameters, and N for the sample size. Each criterion is -2ℓ plus a penalty: $AIC = -2\ell + 2d$ (Akaike, 1974), $BIC = -2\ell + d \log N$ (Schwarz, 1978), $SABIC = -2\ell + d \log\{(N + 2)/24\}$ (Sclove, 1987; lavaan's `bic2`), and $HBIC = -2\ell + d \log\{N/(2\pi)\}$ (Haughton, 1988).

Bollen et al. (2012) derive the BIC family from the Laplace approximation to the log of a model's marginal likelihood,

$$\log P(Y|M) = \ell + \log P(\hat{\theta}|M) + \frac{d}{2}\log(2\pi) - \frac{d}{2}\log N - \frac{1}{2}\log|\bar{I}_E(\hat{\theta})| + O(N^{-1/2})$$

(Bollen et al., 2014, Eq. 9), where $P(\hat{\theta}|M)$ is the prior density at the estimates and $\bar{I}_E = I_E/N$ the expected information per observation. Multiplied by -2, BIC keeps the first and fourth terms, HBIC the third as well, and IBIC every term but the prior: $IBIC = -2\ell + d \log\{N/(2\pi)\} + \log|\bar{I}_E(\hat{\theta})|$ (Bollen et al., 2012, Eq. 20; 2014, Eq. 16), which equals $-2\ell - d\log(2\pi) + \log|I_E(\hat{\theta})|$ for the whole sample information. Recomputing the worked example of Bollen et al. (2012, Table 1) from its printed inputs shows that the log determinant it adds to $d \log\{N/(2\pi)\}$ is that of the whole sample information (its elements are about N times the per observation ones the log likelihood implies), which counts $\log N$ twice relative to their Eq. 20; this function follows Eq. 20, which reproduces the IBIC row of Bollen et al. (2014, Table 6).

SPBIC replaces BIC's unit information prior with a normal prior centered at θ^* (`prior_mean`) whose variance, a multiple of the inverse unit information, is chosen to maximize the marginal likelihood, an empirical Bayes prior. With $Q = (\hat{\theta} - \theta^*)' I_O(\hat{\theta})(\hat{\theta} - \theta^*)$ for the estimates $\hat{\theta}$ (`parameter_estimate`) and the whole sample observed information I_O, $SPBIC = -2\ell + d\{1 + \log(Q/d)\}$ when $Q > d$ (Bollen et al., 2012, Eq. 14; 2014, Eq. 18). When $Q \leq d$ the maximizing prior variance is zero, the prior is a point mass at θ^*, and $SPBIC = -2\ell + Q$ (Bollen et al., 2012, Eq. 15); the two branches meet at $Q = d$. Bollen et al. (2014, note 7) never met that branch and caution that its behavior is little known, and the function warns when it applies.

SABIC replaces *N* in BIC's penalty with $(N + 2)/24$, the penalty of Rissanen's (1978) shortest data description criterion as Sclove (1987, criteria 4 and 5) gives it, which comes from the length of a coded description of the data rather than from the Laplace approximation; Lin, Huang, and Weng (2017) write it ABIC, and Mplus and lavaan's `bic2` call it the sample size adjusted BIC. Its penalty per parameter is smaller than BIC's, and it is not positive when $N \leq 22$, where the function warns. Lin et al. (2017) found that it improved on BIC in selecting path models mainly with small samples or small misspecifications, and that HBIC and SPBIC did best overall. AIC estimates the relative expected Kullback-Leibler discrepancy of a model rather than its marginal likelihood (Burnham & Anderson, 2004). AICc is not offered: its correction is derived for the univariate linear model and is not justified for structural equation models in general.

Units and Identification: AIC, BIC, SABIC, and HBIC depend on a model only through ℓ and d, which do not change when an observed variable is rescaled or a latent variable is identified another way. IBIC and SPBIC also depend on the information matrix in the free parameters as they are parameterized.

IBIC keeps $\log|\bar{I}_E|$ but drops the prior density of the Laplace approximation, and that is the term whose Jacobian would cancel a change of parameterization. When the units of a free parameter are multiplied by c^k, $\log|\bar{I}_E|$ changes by $-2k\log c$. Rescaling an observed variable by c therefore changes the IBIC of one model relative to another by $-2k\log c$ for each free parameter the first has and the second lacks whose units carry that variable's ($k = 1$ for a loading or for a covariance with an unscaled variable, $k = 2$ for its variance), and the ranking of two models fitted to the same data can reverse when one variable's units change. Identifying the latent variables by fixing their variances (`std.lv = TRUE`) rather than by a marker indicator changes IBIC differences too, since the map between the two parameterizations is nonlinear and its Jacobian differs between models. An IBIC comparison is defined only for a stated choice of units and identification. Standardizing every observed variable before fitting every model, or keeping their original metrics, fixes the units, but it is a convention rather than an invariant answer: report it, and read IBIC beside BIC, HBIC, and SPBIC.

SPBIC's *Q* is invariant to a linear reparameterization when the prior mean is transformed with the estimates, so rescaling an observed variable does not change it under the default prior. That prior, however, centers the free loadings and regression coefficients at zero and the other free parameters at their estimates, and which parameters are free loadings depends on the identification: under `std.lv = TRUE` every loading is centered at zero, while under a marker indicator, the convention of the example of Bollen et al. (2014), the scale sits in the factor variance, which stays at its estimate. A mean structure adds the saturated means to *d*, and their information to $\bar{I}_E$. Models compared by IBIC or SPBIC must therefore share their identification and mean structure; `compare_sem` requires every latent variable of every model to be identified the same way (each by a marker indicator, or each by a fixed variance) and the same mean structure, and refuses a set that does not, and for a fitted model the `sem_information` attribute records both.

What a Fitted Model Supplies: ℓ is lavaan's `logLik(fit)`, *d* its `df` attribute, and *N* the total

sample size. In a multiple group model ℓ is the sum over the groups and N the total, as in lavaan. Under `fixed.x = TRUE` (the default of `lavaan::sem()`) or `conditional.x = TRUE`, ℓ is the log likelihood conditional on the exogenous observed variables, and criteria compare only models with the same exogenous variables. Under FIML (`missing = "ml"`) ℓ is the observed data log likelihood. Under `missing = "two.stage"` or `"robust.two.stage"` it is the complete data normal log likelihood at the stage one EM estimates of the means and covariances, so that AIC and BIC are the two-stage criteria of Lai (2021, Eqs. 32 and 35), up to a constant shared by all models fitted to the same data; Lai shows that, unlike the FIML criteria, they select in large samples the model complete data would select when the values are missing at random, and that in samples of the usual size they miss it more often than complete data would. They are comparable only with other two-stage fits of the same variables with the same auxiliary variables in the first stage, and they are the `AIC` and `BIC` rows `fa_confirmatory` reports after two-stage estimation. For IBIC and SPBIC, the expected and observed information matrices are lavaan's per observation `lavInspect(fit, "information.expected")` and `lavInspect(fit, "information.observed")`, multiplied by N; they are the matrices the criteria are defined on under lavaan's defaults `h1.information = "structured"` (the expected information at the fitted moments) and `observed.information = "hessian"` (the negative Hessian), and other settings are refused.

A two-stage fit can have cases observed on an auxiliary variable (lavaan's `aux`) but on none of the model's variables. lavaan's first stage reads them, so they move the EM estimates, but its sample size (`lavaan::lavInspect(fit, "nobs")`) counts them as empty and leaves them out, and it forms the log likelihood as that sample size times the complete data log likelihood per case at the EM estimates. Its log likelihood, and the N of the penalties of BIC, SABIC, and HBIC, then rest on fewer cases than the moments. Both are computed here from the fit computed again with every case the first stage read: in one group, and in several groups that no constraint holds together, the estimates do not depend on the count, and the log likelihood of each group is its value per case times the cases its first stage read, with N their total. A line under the table says so, and the `sem_information` attribute records the counts. On six Holzinger and Swineford tests with the whole block missing for 77 of 301 pupils, at random given general information (the auxiliary), lavaan's N for the two-factor model is 224 and its log likelihood -4040.25, which is 224/301 of the -5429.09 reported here with N = 301 (BIC 10966.61 here, 8183.32 from lavaan's values). In several groups held together by a constraint, such as equal loadings or intercepts, the second stage weights each group by its count, so when the groups' counts grow in different ratios the estimates move as well, and the log likelihood is the one at the estimates with every case read (those of semTools' `twostage()`), which the line says. On the same tests by school with equal loadings and intercepts (130 and 94 cases with an observed test, 156 and 145 read), it is -5393.34 with BIC 10957.89, against -5393.52 and 10958.25 at lavaan's estimates and -4020.79 and 8203.92 from lavaan's values. `fa_confirmatory(missing = "two_stage")` and `measurement_invariance` return fits that count the cases already, and give the same criteria. FIML (`missing = "ml"`) keeps the cases in its likelihood and in N as well.

What Is Refused: A fitted model is refused when it did not converge or has an improper solution, when it has ordered categorical indicators or was fitted by an estimator other than ML (its fit function is then not a log likelihood), when it uses the Wishart likelihood, and when it was fitted with sampling weights: lavaan then maximizes a weighted pseudo log likelihood (Asparouhov, 2005), for which neither the penalties nor the information matrices have their likelihood justification. For a two-level model the sample size in the BIC family's penalty is ambiguous (Bollen et al., 2012, p. 295), so only AIC is computed from the fit, and the others need summary input with the N the user chooses. Automatic IBIC and SPBIC further need a single group, single level ML fit (not MLM or MLR) of complete data whose free parameters are unconstrained, with no

exploratory (rotated) blocks (their information matrices are singular) and with the exogenous observed variables modeled (`fixed.x = FALSE`). Each refusal says why and what to do; for another parameterization, supply the quantities as summary input.

Relative to the Saturated Model: Bollen et al. (2014) report each criterion as the difference between the hypothesized model and the saturated model, so that BIC becomes $T - df \log N$ for the likelihood ratio statistic T on df degrees of freedom. Subtracting the saturated model's criterion from the hypothesized model's gives that scale. A single model's criterion cannot be reconstructed from T and df alone.

Value

A `dmar_tbl` (class `dmar_sem_information`) with columns `term` and `value` and, always in this order, the rows `AIC`, `BIC`, `SABIC`, `HBIC`, `IBIC`, and `SPBIC` (`NA` for a criterion not requested), `log_likelihood`, `npar`, and `N`. The `sem_information` attribute records the source; for a fitted model the likelihood, the missing data method, the exogenous observed variables it conditions on, the mean structure, the number of groups, and, when IBIC or SPBIC is computed, the identification of each latent variable; the scale and provenance of the information matrices; the log determinant of the whole sample expected information for IBIC; for SPBIC the estimates, the prior means and the rule that set them, Q, the penalty dimension and its rule, the penalty, and whether the boundary branch applied; for a two-stage fit whose first stage read cases observed on an auxiliary variable alone, `two_stage_rows`, the cases read and **lavaan**'s count of them in each group, how many such cases there are, whether the fit was computed again with all of them (`corrected`), and whether its estimates depend on the counts, so that they were computed again too (`reestimated`); and the lines of the printed footer.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control, 19*(6), 716–723. doi:10.1109/TAC.1974.1100705

Asparouhov, T. (2005). Sampling weights in latent variable modeling. *Structural Equation Modeling, 12*(3), 411–434.

Bollen, K. A., Harden, J. J., Ray, S., & Zavisca, J. (2014). BIC and alternative Bayesian information criteria in the selection of structural equation models. *Structural Equation Modeling, 21*(1), 1–19. doi:10.1080/10705511.2014.856691

Bollen, K. A., Ray, S., Zavisca, J., & Harden, J. J. (2012). A comparison of Bayes factor approximation methods including two new methods. *Sociological Methods & Research, 41*(2), 294–324. doi:10.1177/0049124112452393

Burnham, K. P., & Anderson, D. R. (2004). Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research, 33*(2), 261–304. doi:10.1177/0049124104268644

Haughton, D. M. A. (1988). On the choice of a model to fit data from an exponential family. *The Annals of Statistics, 16*(1), 342–355. doi:10.1214/aos/1176350709

Lai, K. (2021). Using information criteria under missing data: Full information maximum likelihood versus two-stage estimation. *Structural Equation Modeling, 28*(2), 278–291. doi:10.1080/10705511.2020.1780925

Lin, L.-C., Huang, P.-H., & Weng, L.-J. (2017). Selecting path models in SEM: A comparison of model selection criteria. *Structural Equation Modeling, 24*(6), 855–869. doi:10.1080/10705511.2017.1363652

Rissanen, J. (1978). Modeling by shortest data description. *Automatica, 14*(5), 465–471. doi:10.1016/00051098(78)900055

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6*(2), 461–464. doi:10.1214/aos/1176344136

Sclove, S. L. (1987). Application of model-selection criteria to some problems in multivariate analysis. *Psychometrika, 52*(3), 333–343. doi:10.1007/BF02294360

See Also

compare_sem to compare fitted models by one of these criteria, sem_fit_measures for the fit measures of one model, and the vignette vignette("sem_fit_measures", package = "DMAR").

Other multivariate and latent variable methods: average_variance_extracted(), bifactor_indices(), cfa_1(), cfa_2(), ci_bifactor_indices(), ci_eigenvalue(), common_method_adjusted_r(), common_method_marker(), common_method_single_factor(), compare_sem(), dmacs(), ecvi(), equivalence_rmsea(), fa_bifactor(), fa_confirmatory(), fa_second_order(), factor_congruence(), factor_scores(), htmt(), irt_grm(), irt_information(), measurement_alignment(), measurement_invariance(), procrustes_phi(), sem_exact_fit_bootstrap(), sem_fit_measures(), sem_residual_fit(), simple_structure()

Examples

```
# The three-factor model of nine of the holzinger_swineford
# tests, identified by a marker indicator (lavaan's default).
fit <- lavaan::cfa(
  "visual  =~ t1_visual_perception + t2_cubes + t4_lozenges
   textual =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning
   speed   =~ t10_addition + t12_counting_groups_of_dots +
              t13_straight_and_curved_capitals",
  data = holzinger_swineford)
sem_information_criteria(fit)
sem_information_criteria(fit, measures = c("BIC", "IBIC", "SPBIC"))

# Summary input: the autoregressive model of Bollen et al. (2014,
# Table 6), whose BIC relative to the saturated model is T - df log N.
ar <- sem_information_criteria(log_likelihood = -534.27 / 2, npar = 14,
                               N = 3912)
saturated <- sem_information_criteria(log_likelihood = 0, npar = 20,
                                      N = 3912)
ar$value[ar$term == "BIC"] - saturated$value[saturated$term == "BIC"]

# SPBIC from summary input, with an explicit prior mean.
sem_information_criteria(log_likelihood = -120, npar = 2, N = 200,
                         measures = "SPBIC",
                         parameter_estimate = c(0.4, 1),
                         prior_mean = c(0, 1),
                         information_observed = diag(c(100, 50)))
```

`sem_residual_fit` *The SRMR With Its Unbiased Estimate, Confidence Interval, and Tests of Fit*

Description

Summarizes how far a fitted model's implied covariances are from the observed ones with the standardized root mean square residual (SRMR) and gives it the inferential treatment the RMSEA has long had: the asymptotically unbiased estimate of the population SRMR of Maydeu-Olivares (2017), its standard error and confidence interval, a test of exact fit, and an optional test of close fit, together with the largest standardized residual. The SRMR is an effect size of misfit in the metric of a correlation, and Maydeu-Olivares, Shi, and Rosseel (2018) found its interval and test of close fit more accurate than those of the RMSEA in models with many variables. The computations are lavaan's (`lavaan::lavResiduals()`), and the function requires **lavaan** 0.7-2 or newer (in Suggests).

Usage

```
sem_residual_fit(fit, conf_level = 0.9, close_fit = NULL)
```

Arguments

`fit`	A fitted **lavaan** model: a maximum likelihood fit (estimator ML, MLM, MLMV, or MLR) under lavaan's default normal likelihood, of one group of continuous variables at a single level, from complete raw data (fitted under listwise deletion, FIML, or pairwise deletion) or from a covariance matrix. It must have converged to an admissible solution, as `sem_fit_measures` judges it. See Details for what is refused and why.
`conf_level`	Confidence level of the interval for the population SRMR. Defaults to 0.90, as lavaan's does and as every RMSEA interval in DMAR does: a 90 percent interval pairs with a test of close fit at .05 (see Details). `conf_level = 0.95` gives a 95 percent interval.
`close_fit`	Optional bound c for the test of close fit, $H_0 : \mathrm{SRMR} \leq c$. Without it, no bound is imposed and the `close_fit_bound` and `p_close_fit` rows are `NA`. There is no universal value: Maydeu-Olivares, Shi, and Rosseel (2018) treat the bound as the researcher's choice. Shi, Maydeu-Olivares, and DiStefano (2018) proposed population reference values that scale with the average R^2 (communality) of the observed variables, in a criterion with two parts that must both hold: a model fits closely when its SRMR is at most .05 times the average R^2 and its largest absolute standardized residual covariance is at most .10 (and acceptably at .10 times the average R^2 and .15). They recommend applying these population values through an interval or a test of close fit rather than to the sample values, so `close_fit = 0.05` times the average R^2 tests the first part (`lavaan::lavInspect(fit, "rsquare")` gives the R^2 values). The second part is not tested here; the row `largest_absolute_residual` is its sample value, which they caution against comparing with .10 unless the sample is very

large. Ximénez, Maydeu-Olivares, Shi, and Revuelta (2022) found the bound of the first part, applied to the unbiased SRMR, to work reasonably well in finite samples, except that with small loadings (an average R^2 of .09) close fit called for samples of 500 or more, and recommend it.

Details

The SRMR and Its Unbiased Estimate: With p observed variables, sample covariances s_{ij}, and model implied covariances $\hat{\sigma}_{ij}$, the standardized residuals are $e_{ij} = (s_{ij} - \hat{\sigma}_{ij})/\sqrt{s_{ii}s_{jj}}$, and the SRMR is $\sqrt{\sum_{i \le j} e_{ij}^2/t}$ over the $t = p(p+1)/2$ variances and covariances (row `srmr`; lavaan's `srmr_bentler_nomean`). Its population counterpart is the same quantity computed from the population covariances, a measure of the size of misfit that does not depend on the sample size. The sample SRMR overestimates it, since sampling error adds to every residual. Maydeu-Olivares (2017) removes the bias on the squared scale: with Ξ the asymptotic covariance matrix of the standardized residuals and e their vector, the unbiased estimate (row `unbiased_srmr`) is $k^{-1}\sqrt{\max(e'e - \operatorname{tr}\Xi, 0)/t}$, with $k = 1 - [\operatorname{tr}(\Xi^2) + 2e'\Xi e]/[4(e'e)^2]$ (Maydeu-Olivares et al., 2018, Eq. 16). It is truncated at zero when the estimated sampling contribution exceeds $e'e$, and the square root and the truncation leave some bias in small samples.

The Interval and the Test of Close Fit: The interval for the population SRMR is the large sample normal reference interval $\hat{\rho}_u \pm z_{1-\alpha/2}\widehat{SE}(\hat{\rho}_u)$ for the unbiased estimate $\hat{\rho}_u$ (Maydeu-Olivares et al., 2018, Eqs. 20 and 22), where α is 1 minus `conf_level`. Its limits are reported as lavaan returns them, without truncation, so the lower limit can fall below zero although the population SRMR cannot, and when the estimate is truncated to zero the interval is centered at zero. In the simulations of Maydeu-Olivares et al. (2018) the interval computed without the normality assumption covered at close to the nominal rate in most conditions, including models of 60 variables, and did less well than the RMSEA interval only for very small misfit in small models, and then mostly in small samples; computed under normality, it covered poorly when the data were not normal. The test of close fit, $H_0 : \text{SRMR} \le c$, refers $z = (\hat{\rho}_u - c)/\widehat{SE}(\hat{\rho}_u)$ to the standard normal (Maydeu-Olivares et al., 2018, Eq. 21; row `p_close_fit`); it rejects at level α exactly when the lower limit of the $100(1-2\alpha)$ percent interval exceeds c, so the default `conf_level = 0.90` pairs with a test at .05, as the RMSEA interval of `sem_fit_measures` does with its tests, and `conf_level = 0.95` with a test at .025. The printed table states the level beneath it. Failing to reject close fit does not establish it.

The Test of Exact Fit: The test of exact fit (row `p_exact_fit`) asks whether the population SRMR is zero. It uses the sample (biased) SRMR, whose expected value and standard error under exact fit are known (Maydeu-Olivares, 2017): $z = \max(\text{SRMR} - E_0, 0)/SE_0$, referred to the upper tail of the standard normal. The row `srmr_se_exact_fit` is that standard error, $SE_0 = \sqrt{\operatorname{tr}(\Xi^2)/(2t \operatorname{tr}\Xi)}$, which holds only under exact fit: it is not the standard error of the sample SRMR of a misspecified model and must not be used to form an interval, which is what the unbiased estimate and its standard error are for. Because of the truncation at zero the *p*-value is at most 0.5, and it equals 0.5 whenever the sample SRMR is below its expectation under exact fit, as it is more than half the time when the model is correct.

Which Fits, and the Residual Covariance: The asymptotic covariance Ξ of the residuals, and so the unbiased estimate, its standard error, the interval, and both tests, depend on the fit, as lavaan computes them. For an ML fit with standard errors under normality it is normal theory. When the fit computed the asymptotic covariance matrix of the sample moments from their fourth order

moments (MLM, MLMV, or a Satorra-Bentler test), or uses sandwich standard errors (MLR), it is asymptotically distribution free, which is the version to use with nonnormal item data, for which normal theory intervals perform poorly (Maydeu-Olivares et al., 2018). The projection onto the residual space uses the fit's information setting, so observed information (the default under MLR) gives slightly different values from expected information; an MLR fit with `information = "expected"` reproduces MLM. The standardizing standard deviations are treated as fixed. The `sem_fit` attribute records which residual covariance and information were used.

The function refuses what these computations do not cover, each with a message that says why: estimators other than ML (least squares and categorical estimators), `likelihood = "wishart"`, multiple groups, multilevel models, ordered variables, sampling weights, `conditional.x = TRUE` (whose residuals are conditional on the exogenous covariates rather than the marginal covariances the SRMR summarizes), fits with incomplete data under FIML or pairwise deletion, and two-stage fits. A FIML or pairwise fit of data with no missing values is accepted: its residuals, their standard errors, and the tests are those of a listwise fit with the same mean structure and information setting (FIML's default is the observed information, which the footer names). A fit with auxiliary variables (**lavaan**'s `aux`) is refused: **lavaan**'s residual summary averages over them, and their residuals of about zero lower the SRMR, while with complete data they change nothing in the model's fit, so the fit without them is the one to summarize. A fit whose cases were removed by listwise deletion is evaluated on the complete cases with a warning, since that assumes the data are missing completely at random. A model fitted from a covariance matrix (`sample.cov`) gives the same values as one fitted to the raw data under normal theory, since these need only the moments. For a saturated model (zero degrees of freedom) the residuals are zero up to the convergence tolerance, and the standard errors, limits, and p-values are `NA`, with a warning that says so.

The SRMR here summarizes the covariance residuals only. For a model with a mean structure, row `srmr_mean` is the root mean square of the standardized mean residuals, zero when the means are saturated; when the model restricts the means, a warning points to it, since the covariance SRMR does not reflect misfit in the means. Always look at the residuals themselves as well as their average (the attribute `standardized_residuals`): a good average can hide a large local misfit.

Value

A `dmar_tbl` (class `dmar_sem_fit`) with columns `term` and `value` and, always in this order, the rows `srmr` (the sample SRMR), `srmr_se_exact_fit` (its standard error under exact fit only), `p_exact_fit`, `unbiased_srmr`, `unbiased_srmr_se`, `unbiased_srmr_ci_lower`, `unbiased_srmr_ci_upper`, `close_fit_bound` and `p_close_fit` (`NA` unless `close_fit` is given), `srmr_mean` (`NA` without a mean structure), `largest_absolute_residual`, and `N`. Attributes: `standardized_residuals`, the matrix of standardized covariance residuals; `largest_residual`, the pair of variables with the largest absolute residual and its signed value; and `sem_fit`, which records the residual covariance, the information setting, the cases used, and the lines of the printed footer.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maydeu-Olivares, A. (2017). Assessing the size of model misfit in structural equation models. *Psychometrika, 82*(3), 533–558. doi:10.1007/s1133601695527

Maydeu-Olivares, A., Shi, D., & Rosseel, Y. (2018). Assessing fit in structural equation models: A Monte-Carlo evaluation of RMSEA versus SRMR confidence intervals and tests of close fit. *Structural Equation Modeling, 25*(3), 389–402. doi:10.1080/10705511.2017.1389611

Shi, D., Maydeu-Olivares, A., & DiStefano, C. (2018). The relationship between the standardized root mean square residual and model misspecification in factor analysis models. *Multivariate Behavioral Research, 53*(5), 676–694. doi:10.1080/00273171.2018.1476221

Ximénez, C., Maydeu-Olivares, A., Shi, D., & Revuelta, J. (2022). Assessing cutoff values of SEM fit indices: Advantages of the unbiased SRMR index and its cutoff criterion based on communality. *Structural Equation Modeling, 29*(3), 368–380. doi:10.1080/10705511.2021.1992596

See Also

`sem_fit_measures` for the chi square, RMSEA, and incremental indices beside the sample SRMR, and `equivalence_rmsea` for the test of not close fit of the RMSEA.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `simple_structure()`

Examples

```
# The three-factor model of nine holzinger_swineford tests.
fit <- lavaan::cfa(
  "visual  =~ t1_visual_perception + t2_cubes + t4_lozenges
   textual =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning
   speed   =~ t10_addition + t12_counting_groups_of_dots +
              t13_straight_and_curved_capitals",
  data = holzinger_swineford, std.lv = TRUE)
sem_residual_fit(fit)

# The 90 percent interval, the default, pairs with a test of close fit
# at .05. The bound is the researcher's choice; here it is .05 times the
# average R squared of the nine tests, the population value for close
# fit of Shi, Maydeu-Olivares, and DiStefano (2018).
r2 <- lavaan::lavInspect(fit, "rsquare")
sem_residual_fit(fit, close_fit = 0.05 * mean(r2))

# A 95 percent interval instead.
sem_residual_fit(fit, conf_level = 0.95)
```

sem_sam *Structural Relations Estimated After the Measurement Models*

Description

Fits local structural-after-measurement (SAM) models using `lavaan::sam()`. Each measurement block is estimated separately, and the structural relations use a covariance matrix corrected for measurement error. Two-step standard errors account for estimating the measurement parameters. This offers an alternative to treating estimated factor scores as perfectly measured observations. A latent SEM remains available when this wrapper's narrower scope does not fit the study.

Usage

```
sem_sam(
  model,
  data,
  estimator = "ML",
  conf_level = 0.95,
  gamma_unbiased = TRUE
)
```

Arguments

`model`	A single-group lavaan model string with at least two simple-structure factors, at least three distinct continuous indicators per factor, and structural regressions among those factors. Indicators cannot cross-load. Derived parameters, explicit equality constraints, observed covariates, residual covariances, feedback paths, and latent interactions are outside this interface's validated scope.
`data`	Raw numeric observations on the model's indicators. Incomplete observations are refused; the function never drops cases automatically. Use a joint latent SEM with FIML for incomplete data. An indicator of `model` that `data` lacks is refused with an error of class `dmar_items_invalid` (see `dmar_metadata`), as are columns of `data` that share a name.
`estimator`	`"ML"` (default) with normal-theory `"twostep"` errors, or `"MLR"` with `"twostep.robust"` errors. Both use local ML mapping and the ML measurement estimators; the covariance assumptions for inference differ. Numerical optimization versus an analytic solution does not determine which SE is appropriate.
`conf_level`	Confidence level, default 0.95.
`gamma_unbiased`	Logical, default `TRUE`. For robust inference, use lavaan's finite-sample unbiased moment-covariance correction. `FALSE` uses the empirical moment covariance (HC0 convention).

Details

This interface currently supports unweighted, single-level, complete continuous data. It verifies the actual local mapping and two-step SE method after fitting and refuses a substitution. These

restrictions distinguish numerical agreement with independent complete-data oracles from evidence for other models or missing-data uncertainty calculations. Small or moderate samples can benefit from separating measurement and structural estimation (Rosseel & Loh, 2024), but SAM does not repair a misspecified or unidentified measurement model.

Returned intervals concern individual, original-unit parameters. The underlying fit's joint covariance matrix does not populate every cross-stage covariance. Consequently this function refuses user-defined parameters and does not report standardized structural intervals or intervals for effects combining measurement and structural estimates. The attached fit is retained for inspection with that limitation, not as a complete covariance matrix for arbitrary delta-method calculations.

Value

A `dmar_tbl` with parameter labels, `estimate`, `se`, `statistic`, `p_value`, confidence limits, `conf_level`, and `stage`. Attributes retain the `fit`, requested and actual `method`, and the covariance limitation. `tidy()` retains the rows and `glance()` gives the method and sample size.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Rosseel, Y., & Loh, W. W. (2024). A structural after measurement (SAM) approach to structural equation modeling. *Psychological Methods, 29*(3), 561–588. doi:10.1037/met0000503

Examples

```
model <- "visual =~ t1_visual_perception + t2_cubes + t4_lozenges
          verbal =~ t6_paragraph_comprehension + t7_sentence + t9_word_meaning
          verbal ~ visual"
sem_sam(model, holzinger_swineford)
```

---

signal_to_noise_R2 *Signal to Noise Estimators for the Squared Multiple Correlation Coefficient*

Description

Computes five estimators of the population signal to noise ratio $\phi^2 = \rho^2/(1-\rho^2)$ associated with the squared multiple correlation coefficient ρ^2. Two are functions of the unadjusted and the Wherry-adjusted sample R^2; the other three are the Muirhead (1985) unique minimum variance unbiased estimators that improve substantially on the plug-in estimator at small N and modest numbers of predictors.

Usage

```
signal_to_noise_R2(R2, N, p)
```

Arguments

R2	The usual sample estimate of the squared multiple correlation coefficient (no degrees of freedom adjustment). Numeric scalar in $(0, 1)$.
N	Sample size.
p	Number of predictor variables.

Details

The signal to noise ratio $\phi^2 = \rho^2/(1 - \rho^2)$ is a natural reparameterization of ρ^2 that is bounded only below (at zero) and so behaves more like a variance ratio than a proportion. It is also the noncentrality parameter (up to a factor of *N*) for the omnibus *F*-test of $\rho^2 = 0$ under fixed predictors; see `convert_R2_f`.

The five estimators returned, in increasing order of bias-correction machinery, are:

- `phi2_hat`: the plug-in estimator $\hat{\phi}^2 = R^2/(1-R^2)$. Biased upward in small samples because the sample R^2 is itself biased upward.
- `phi2_adj_hat`: the plug-in estimator applied to the Wherry-adjusted R^2. Removes the leading-order bias in R^2 but is not itself unbiased for ϕ^2.
- `phi2_umvue`: Muirhead's (1985) unique minimum variance unbiased estimator (his θ_U, their Eq. 4); equivalent to Stuart, Ord, and Arnold's (1999) equation 28.97. Requires $N \geq p + 6$ (the gate on all three Muirhead estimators); for smaller N the value is `NA`.
- `phi2_umvue_l`: Muirhead's (1985) linearly-improved unique minimum variance unbiased estimator (his θ_L); equivalent to Stuart et al.\ (1999) equation 28.98. Dominates `phi2_umvue` in mean squared error.
- `phi2_umvue_nl`: Muirhead's (1985) nonlinearly-improved estimator (his θ_{NL}). Dominates the linear improvement in MSE but requires $p \geq 5$; for smaller p the value is `NA`.

The nonlinear estimator dominates the linear one in risk when $p \geq 5$, though Muirhead notes the two perform very similarly in practice; for smaller p the linear estimator is preferred over the plug-in and adjusted-R^2 forms. All three Muirhead estimators are reported truncated at zero, so the returned value is on the same scale as ϕ^2 (the truncation introduces a negligible bias only when the population ϕ^2 is near zero).

As N grows with p fixed, the five estimators converge to a common value (the population ϕ^2); the difference between them is the small-sample bias machinery in operation. The `@examples` block illustrates that convergence.

Value

A `data.frame` with columns `term` and `value` and one row per estimator: `phi2_hat`, `phi2_adj_hat`, `phi2_umvue`, `phi2_umvue_l`, and `phi2_umvue_nl`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*, 524–555. doi:10.1080/00273170802490632

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Muirhead, R. J. (1985). Estimating a particular function of the multiple correlation coefficient. *Journal of the American Statistical Association, 80*, 923–925.

Stuart, A., Ord, J. K., & Arnold, S. (1999). *Kendall's advanced theory of statistics, volume 2A: Classical inference and the linear model* (6th ed.). Arnold.

See Also

ci_R2, ss_aipe_R2, convert_R2_f

Examples

```
# 1. Fixed R^2 = 0.5 and p = 2, growing N: the five estimators agree
#    to within a small fraction once N is moderate.
signal_to_noise_R2(R2 = .5, N = 50,   p = 2)
signal_to_noise_R2(R2 = .5, N = 100,  p = 2)
signal_to_noise_R2(R2 = .5, N = 500,  p = 2)

# 2. With p = 5 the nonlinear estimator is available; it differs
#    most from the plug-in at small N.
signal_to_noise_R2(R2 = .5, N = 50,  p = 5)
signal_to_noise_R2(R2 = .5, N = 500, p = 5)
```

---

simple_effects_AB *Simple Effect F Tests for a Two-Factor Between-Subjects Design*

Description

Given a fitted aov or lm object for a two-factor between-subjects design, conventionally written $Y \sim A \times B$, where A and B are crossed fixed factors, computes the family of *simple main effects*: the effect of A at each level of B and/or the effect of B at each level of A. Each row carries the simple effect F test, its (optionally adjusted) p-value, and the partial η^2 with a noncentrality-based confidence interval under the pooled error term. The error term can be either the full-model pooled MS_W (the textbook default) or a Welch–Satterthwaite test refitted within each conditioning level (robust to within-level heteroscedasticity).

Usage

```
simple_effects_AB(
  object,
  which = "both",
  error_term = "pooled",
  adjust = "none",
  conf_level = 0.95
)
```

Arguments

`object`	A fitted `aov` or `lm` object whose right-hand side has *exactly two* crossed factors (e.g.\ `iq_gain ~ treatment * grade`). The interaction term is strongly recommended so that the pooled error is the pure within-cell MS_W; the function still runs without it but issues a warning (the additive-model residual includes interaction variance and inflates the error term used for the pooled simple effect F).
`which`	Which family of simple effects to report: `"both"` **(default)** The b tests of A at each level of B followed by the a tests of B at each level of A ($a + b$ rows). `"A_at_B"` Only the b tests of the first factor at each level of the second. `"B_at_A"` Only the a tests of the second factor at each level of the first. The first factor on the right-hand side of the model formula is treated as A; the second as B.
`error_term`	Error-term strategy for the simple effect F: `"pooled"` **(default)** Use the full factorial model's MS_W and its residual *df*. This is Maxwell, Delaney, and Kelley's preferred default and gives the simple effect F maximum denominator *df*. Validity rests on homogeneity of variance across all $a \times b$ cells. `"welch"` Refit a Welch–Satterthwaite one-way test on only the data at the conditioning level (using `oneway.test` with `var.equal = FALSE`). Robust to heteroscedasticity within the conditioning level at the cost of fewer (and fractional) denominator *df*.
`adjust`	Multiple-comparison adjustment applied to the p-values of the entire family of simple effects returned ($a + b$ for `which = "both"`). One of `"none"` (default), `"bonferroni"`, or any sequential method supported by `p.adjust`: `"holm"`, `"hochberg"`, `"BH"`, `"BY"`.
`conf_level`	Confidence level for each row's partial η^2 interval. Default `0.95`.

Details

What a simple effect is. The simple main effect of A at level $B = b_j$ tests whether the a cell means at that single level of B differ. It is the one-way analysis of variance of Y on A restricted to observations with $B = b_j$. The counterpart, the simple effect of B at $A = a_i$, is defined symmetrically.

Test statistic. For the simple effect of A at $B = b_j$, let $SS_{A \mid b_j} = \sum_i n_{ij} (\bar{Y}_{ij\cdot} - \bar{Y}_{\cdot j\cdot})^2$ be the between-A sum of squares computed at that level, and let $MS_{A \mid b_j} = SS_{A \mid b_j}/(a-1)$.

- *Pooled* MS_W: $F = MS_{A \mid b_j}/MS_W$ with degrees of freedom $(a-1,\ N-ab)$, where MS_W and its *df* come from the fitted full factorial model.
- *Welch*: F and its (fractional) denominator *df* come from `oneway.test(y ~ A, subset = (B == b_j), var.equal = FALSE)`.

Test statistics for B at a_i are computed by interchanging the two factors.

Choosing an error term. The pooled denominator borrows strength from all N observations and is the textbook default in Maxwell, Delaney, and Kelley's treatment. Its validity rests on homogeneity of variance across *all* $a \times b$ cells, not merely within the conditioning level. When that assumption is doubtful , for example, if Levene's or the Brown–Forsythe test flags heteroscedasticity, or if cell variances visibly differ, the Welch option provides a level-conditional test that does not require homogeneity across cells. The trade-off is denominator *df*: pooled carries the full $N - ab$ residual *df*, whereas Welch carries the Welch–Satterthwaite *df* based on the a cell variances at that level only.

Partial η^2 and its CI. The point estimate is

$$\hat{\eta}_p^2 = \frac{df_{\text{effect}}\, F}{df_{\text{effect}}\, F + df_{\text{error}}},$$

computed from the pooled-error *F* and *df*. The confidence interval is built by Steiger's (2004) transformation principle: a CI for the noncentrality parameter λ of the *F* distribution is obtained via `ci_nc_F` and then mapped through $\eta_p^2 = \lambda/(\lambda + N_{\text{ref}})$, with N_{ref} taken to be the total study *N* for the pooled error term (treating the simple effect as a contrast within the full factorial design). When the lower limit on λ is not identified (i.e., the observed *F* is below its one-sided critical value), the lower limit on η_p^2 is set to 0; when the upper limit is at infinity, the upper limit on η_p^2 is set to 1.

Under `error_term = "welch"`, partial eta squared and both limits are `NA`. The transformation of Welch's statistic and fractional degrees of freedom does not estimate ordinary partial eta squared, and transforming a noncentral-F interval with the level's sample size targets a different quantity under unequal variances. The Welch F reference distribution is itself approximate. The tests remain available; a warning, the `effect_size_inference` attribute, and the printed explanation identify the unavailable effect-size inference.

Multiplicity across the family. With `which = "both"` the family is the $a+b$ simple effects returned in a single call; the adjustment is applied to that entire family. If only one direction is wanted, call the function twice with `which = "A_at_B"` and `which = "B_at_A"` so each family is adjusted on its own. The `"bonferroni"` adjustment is $p_{\text{adj}} = \min(1, m\,p)$ for m rows; the sequential methods (`"holm"`, `"hochberg"`, `"BH"`, `"BY"`) are computed via `p.adjust`.

When to perform simple effects. A significant omnibus interaction need not precede simple effect testing (Maxwell, Delaney, & Kelley, 2027). Simple effects are informative whenever the substantive question is conditional on a level of the other factor, and the multiplicity adjustment controls the family-wise error rate independently of any interaction screen.

Scope. Only fixed-effects between-subjects designs with *exactly two* crossed factors are supported. Within-subjects or mixed designs (`aovlist` fits) and three- or higher-way designs are out of scope for this function. Per-cell sample sizes may be unequal.

Value

A `data.frame` with one row per simple effect test and columns `effect`, `focal_factor`, `conditioning_factor`, `conditioning_level`, `F_value`, `df_effect`, `df_error`, `p_value`, `p_adjusted`, `eta_squared_partial`, `ci_lower`, `ci_upper`, `n_at_level`. Attributes `error_term`, `adjust`, `conf_level`, `factor_A`, and `factor_B` record the call options.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

Welch, B. L. (1951). On the comparison of several mean values: An alternative approach. *Biometrika, 38*, 330–336.

See Also

`contrast_test` for within-level pairwise or custom contrasts, `eta_squared` and `ci_eta_squared` for the omnibus effect size counterparts, `ci_nc_F` for the noncentrality machinery, `ss_power_factorial_anova` for power calculations on the omnibus factorial effects.

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `summary_t_test()`, `welch_t()`

Examples

```
# 2 x 3 factorial: expectancy treatment (A) x grade (B) on the
# pygmalion data. Grades 4 through 6 are omitted so the family of
# simple effects stays short enough to read at a glance.
pyg <- pygmalion[pygmalion$grade <= 3, ]
pyg$grade <- factor(pyg$grade)
fit <- aov(iq_gain ~ treatment * grade, data = pyg)

# Default: pooled MS_W, both families, no adjustment. The expectancy
# effect on IQ gain is concentrated in grades 1 and 2; at grade 3 the
# F is 0.004, so the lower limit on partial eta squared is clamped to
# 0 and the function notes the clamp in a warning.
simple_effects_AB(fit)
```

```
# Only the simple effects of grade within each treatment level, with
# a Holm adjustment across that family of two tests.
simple_effects_AB(fit, which = "B_at_A", adjust = "holm")

# Welch error term: refits a Welch one-way at each conditioning
# level. The Welch denominator df fall well below the pooled 157, so
# more of the lower limits are clamped to 0.
simple_effects_AB(fit, error_term = "welch")

# Bonferroni across the full a + b = 5-test family.
simple_effects_AB(fit, adjust = "bonferroni")

# Simulated 2 x 2 design with a known interaction pattern.
set.seed(113)
d <- simulate_ancova_factorial_data(
  a = 2, b = 2,
  true_mean_Y = c(50, 60, 55, 50),   # crossover at B = 2
  true_mean_X = matrix(10, nrow = 4, ncol = 1),
  true_sd_Y = 8, true_sd_X = 3, true_cor_YX = 0,
  n_per_cell = 30
)
fit_sim <- aov(y ~ A * B, data = d)
simple_effects_AB(fit_sim, conf_level = 0.95)
```

`simple_structure` *Quantify Simple Structure in a Factor Loading Matrix*

Description

Summarizes how closely a rotated loading matrix approaches Thurstone's simple structure, in which each item loads on as few factors as possible so that the factors are interpretable. Three complementary quantities are reported: the mean item complexity (the average number of factors an item effectively loads on, one for a perfectly simple item), the hyperplane proportion (the share of loadings near zero, which Thurstone sought to maximize), and the counts of pure versus complex items at a salience cutoff. Together they turn a visual impression of a loading matrix into numbers.

Usage

```
simple_structure(factor_loading_matrix, salient = 0.3, hyperplane = 0.1)
```

Arguments

`factor_loading_matrix`
: A numeric matrix of factor loadings, items in rows and factors in columns (for example `unclass(psych::fa(...)$loadings)` or a lavaan standardized loading matrix). Row names, if present, label the items.

`salient`
: Absolute loading at or above which an item is counted as loading saliently on a factor. Defaults to 0.30 (about ten percent of an item's variance), a common floor for a meaningful loading.

`hyperplane` Absolute loading below which a loading is treated as lying in the hyperplane (effectively zero). Defaults to 0.10.

Details

Item complexity is Hofmann's complexity index, proposed in Hofmann (1977) and given as Equation 1 of Hofmann (1978), $c_i = (\sum_j \lambda_{ij}^2)^2 / \sum_j \lambda_{ij}^4$, with λ_{ij} the loading of item i on factor j in `factor_loading_matrix`, which equals one when an item loads on a single factor and rises toward the number of factors as the loadings spread out; it is the same complexity that `psych::fa` reports. The `"complexity"` attribute holds these per-item values, and the `mean_complexity` row is their arithmetic average, what Hofmann (1978) calls the total matrix complexity. These are complexities, not Kaiser's (1974) simplicity index; Hofmann (1978) shows that either can be derived from the other at the item level, with the simplicity of item i in an m-factor solution given by his Equation 3, $s_i = [1/(m-1)][(m/c_i) - 1]$. An item is *pure* when exactly one of its loadings is salient and *complex* when more than one is. The hyperplane proportion is the fraction of all loadings whose absolute value is below `hyperplane`; a clean simple structure is mostly such near-zero loadings.

Value

A `data.frame` (class `dmar_tbl`) with one row per summary quantity (`term`, `value`): the number of items and factors, the mean and median item complexity, the hyperplane proportion, and the counts and proportion of pure items. The per-item complexities are attached as the `"complexity"` attribute (a named numeric vector), and the salience and hyperplane cutoffs as the `"salient"` and `"hyperplane"` attributes.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Hofmann, R. J. (1977). Indices descriptive of factor complexity. *The Journal of General Psychology, 96*(1), 103–110. doi:10.1080/00221309.1977.9920803

Hofmann, R. J. (1978). Complexity and simplicity as objective indices descriptive of factor solutions. *Multivariate Behavioral Research, 13*(2), 247–250. doi:10.1207/s15327906mbr1302_9

Kaiser, H. F. (1974). An index of factorial simplicity. *Psychometrika, 39*(1), 31–36.

Thurstone, L. L. (1947). *Multiple-factor analysis*. University of Chicago Press.

See Also

`average_variance_extracted` and `htmt` for the convergent and discriminant sides of an exploratory solution.

Other multivariate and latent variable methods: `average_variance_extracted()`, `bifactor_indices()`, `cfa_1()`, `cfa_2()`, `ci_bifactor_indices()`, `ci_eigenvalue()`, `common_method_adjusted_r()`, `common_method_marker()`, `common_method_single_factor()`, `compare_sem()`, `dmacs()`, `ecvi()`, `equivalence_rmsea()`, `fa_bifactor()`, `fa_confirmatory()`, `fa_second_order()`, `factor_congruence()`, `factor_scores()`, `htmt()`, `irt_grm()`, `irt_information()`, `measurement_alignment()`, `measurement_invariance()`, `procrustes_phi()`, `sem_exact_fit_bootstrap()`, `sem_fit_measures()`, `sem_information_criteria()`, `sem_residual_fit()`

Examples

```
# A nearly simple two-factor structure: six items, three per factor.
loading_matrix <- rbind(
  i1 = c(0.80, 0.05), i2 = c(0.75, 0.10), i3 = c(0.70, -0.05),
  i4 = c(0.08, 0.78), i5 = c(-0.04, 0.72), i6 = c(0.30, 0.60))
simple_structure(factor_loading_matrix = loading_matrix)
attr(simple_structure(loading_matrix), "complexity")
```

---

simulate_ancova_data *Simulate Data From a One-Covariate ANCOVA Model*

Description

Generates random data appropriate for an analysis of covariance with one continuous outcome (Y) and one continuous covariate (X) crossed with a fixed groups. The covariate is treated as a random variable; Y and X are jointly multivariate normal within each group. Both the randomized-design case (population covariate mean common across groups) and the non-randomized / preexisting-groups case (population covariate means differ across groups; Y-on-X correlation may also differ) are supported. Per-group sample sizes may be equal or unequal.

Usage

```
simulate_ancova_data(
  true_mean_Y,
  true_mean_X,
  true_sd_Y,
  true_sd_X,
  true_cor,
  a,
  n_per_group,
  randomized = TRUE,
  seed = NULL
)
```

Arguments

`true_mean_Y`	A numeric vector of length `a` giving the population mean of Y in each group.
`true_mean_X`	When `randomized = TRUE`, a single number giving the common population mean of X across all groups. When `randomized = FALSE`, a numeric vector of length `a` giving the population covariate mean in each group.
`true_sd_Y`	The population standard deviation of Y (assumed common across groups).
`true_sd_X`	The population standard deviation of X (assumed common across groups).
`true_cor`	The population correlation between Y and X. When `randomized = TRUE`, must be a single number (see Details). When `randomized = FALSE`, may be a single number (recycled to all `a` groups) or a numeric vector of length `a` giving a distinct correlation in each group.

a	The number of fixed levels of the grouping factor (i.e., the number of conditions in a fixed-effects ANCOVA design). Use this argument when groups are the design levels of interest. (For sample-selected groups, e.g., classrooms or schools randomly drawn from a population, the convention in this package is to use J instead.)
n_per_group	A single number (equal sample size per group) or a numeric vector of length a giving the sample size in each group.
randomized	Logical. TRUE (the default) for a randomized design (random assignment of subjects to groups, so that the population covariate mean is the same in every group). FALSE for a non-randomized / preexisting-groups design.
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Each group's (Y, X) pairs are drawn from a bivariate normal distribution with mean $(\mu_{Y,j}, \mu_{X,j})$ and covariance

$$\Sigma_j = \begin{pmatrix} \sigma_Y^2 & \rho_j\, \sigma_Y\, \sigma_X \\ \rho_j\, \sigma_Y\, \sigma_X & \sigma_X^2 \end{pmatrix}.$$

Why true_cor **must be a single number when** randomized = TRUE**.** Random assignment forms each group as an exchangeable random sample from the same population. The bivariate distribution of (Y, X) is therefore the same in every group, including the correlation. Allowing true_cor to differ across groups would silently break that interpretation and produce data that no randomized design could plausibly have generated. The function therefore stops with an error in that case; use randomized = FALSE if you genuinely want group-specific correlations.

Convention on group labels. The argument a is used here (and throughout DMAR's experimental-design functions) for the number of *fixed* levels of a designed factor, the levels you intend to compare. The letter J is reserved for the number of *sample- selected* groups, e.g., when classrooms or schools are randomly sampled from a population (a random-effects context).

Value

A long-format data.frame with one row per simulated subject and three columns:

group A factor with a levels ("1", ..., as.character(a)) identifying each subject's group.

y Numeric simulated outcome.

x Numeric simulated covariate.

This format is directly usable with aov(), lm(), and other model-fitting functions.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

mvrnorm, ci_c_ancova, ss_aipe_c_ancova

Other data simulators: simulate_ancova_factorial_data(), simulate_anova_data(), simulate_longitudinal_gomp
simulate_longitudinal_logistic(), simulate_longitudinal_negative_exponential(), simulate_longitudinal_
simulate_longitudinal_richards(), simulate_regression_data()

Examples

```
# 1. Randomized design, two groups, equal n.
set.seed(113)
simple <- simulate_ancova_data(
  true_mean_Y = c(3, 5),
  true_mean_X = 10,
  true_sd_Y    = 1,
  true_sd_X    = 2,
  true_cor   = 0.8,
  a          = 2,
  n_per_group = 20
)
head(simple)

# 2. Four preexisting groups with different correlations and unequal n.
#    The first two groups share true_cor = 0.30; the second two share a larger
#    true_cor = 0.60. The four groups enter the data generating mechanism
#    only through their per-group means and correlations, as four
#    distinct populations being sampled. In a downstream
#    analysis these four groups could be cross-classified as a 2 x 2
#    factorial design (e.g., the first factor distinguishing groups 1-2
#    from groups 3-4, and the second factor distinguishing groups 1, 3
#    from groups 2, 4) and analyzed via factorial ANCOVA.
set.seed(113)
preexisting <- simulate_ancova_data(
  true_mean_Y = c(50, 55, 60, 65),
  true_mean_X = c(10, 12, 11, 13),
  true_sd_Y    = 8,
  true_sd_X    = 3,
  true_cor   = c(0.30, 0.30, 0.60, 0.60),
  a          = 4,
  n_per_group = c(40, 35, 45, 30),
  randomized = FALSE
)
aggregate(cbind(y, x) ~ group, data = preexisting,
          FUN = function(z) round(c(mean = mean(z), sd = sd(z)), 2))
```

---

simulate_ancova_factorial_data

Simulate Data From a Factorial ANCOVA Design (up to Four Factors, Any Number of Covariates)

Description

Generates random data appropriate for an analysis of covariance with up to four crossed fixed factors (A, B, C, D) and one or more continuous covariates. Within every cell the outcome Y and the covariates $X_1, \ldots, X_q$ are jointly multivariate normal with a common (homogeneous) within-cell covariance structure, the standard assumption underlying classical ANCOVA. Per-cell sample sizes may be equal or unequal.

Usage

```
simulate_ancova_factorial_data(
  a,
  b = 1,
  c = 1,
  d = 1,
  n_covariates = 1,
  true_mean_Y,
  true_mean_X,
  true_sd_Y,
  true_sd_X,
  true_cor_YX,
  true_cor_XX = NULL,
  n_per_cell,
  randomized = TRUE,
  seed = NULL
)
```

Arguments

a	Number of levels of the first factor (must be at least 2).
b	Number of levels of the second factor (default 1; use 1 if absent).
c	Number of levels of the third factor (default 1; use 1 if absent).
d	Number of levels of the fourth factor (default 1; use 1 if absent). So a = 3, b = 2, c = 1, d = 1 specifies a 3×2 design.
n_covariates	Integer ≥ 1: the number of continuous covariates. Default 1.
true_mean_Y	Numeric vector of length $a \cdot b \cdot c \cdot d$ giving the population cell means of Y, in the same row order as expand.grid(A, B, C, D) (which varies factor A fastest, then B, then C, then D).
true_mean_X	Numeric matrix of dimension $(a \cdot b \cdot c \cdot d) \times q$ giving the population cell means of each covariate (rows = cells in the same order as true_mean_Y; columns = covariates). When randomized = TRUE every column must be constant across cells.
true_sd_Y	Within-cell standard deviation of Y (a single positive number, common across cells).
true_sd_X	Within-cell standard deviations of the covariates. Either a single number (recycled to all n_covariates) or a numeric vector of length n_covariates.

`true_cor_YX` Within-cell correlations between Y and each covariate. Either a single number (recycled to all `n_covariates`) or a numeric vector of length `n_covariates`.

`true_cor_XX` Within-cell correlation matrix among the covariates, $q \times q$. Default `NULL`, interpreted as the `n_covariates`-dimensional identity.

`n_per_cell` A single number (equal sample size per cell) or a numeric vector of length $a \cdot b \cdot c \cdot d$ giving per-cell sample sizes (in the same row order as `true_mean_Y`).

`randomized` Logical. `TRUE` (the default) for a randomized design, each cell is sampled from the same population covariate distribution, so `true_mean_X` must be constant across cells. `FALSE` for non-randomized / preexisting-groups designs in which the cells' population covariate means may differ.

`seed` `NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

This is the factorial generalization of `simulate_ancova_data` (which is the special case `b = c = d = 1, n_covariates = 1`). All cells share the same within-cell covariance structure (homogeneity of regression, the classical ANCOVA assumption); the difference between randomized and non-randomized designs lies entirely in whether the cell covariate means are constrained to be equal.

Why `true_mean_X` must be constant across cells when `randomized = TRUE`. Random assignment forms each cell as an exchangeable random sample from the same joint distribution of covariates and outcome. Cell-specific covariate means would silently break that interpretation. The function checks the constraint and stops with an informative error if it is violated. (The same logic that powers `simulate_ancova_data`.)

Cell ordering. The function uses `expand.grid`'s convention, factor `A` varies fastest, then `B`, then `C`, then `D`. So for a 2×3 design, the six cells of `true_mean_Y` are $(A_1B_1), (A_2B_1), (A_1B_2), (A_2B_2), (A_1B_3), (A_2B_3)$. If you build the cell specification by passing the factor levels to `expand.grid` in the same order, the row indexing automatically matches.

Value

A long-format `data.frame` with one row per simulated subject and the following columns:

`A`, `B`, `C`, `D` Factor columns for each present design factor (omitted when the factor is absent, i.e., its corresponding `a`/`b`/`c`/`d` argument is `1`).

`x1`, `x2`, ..., `x<q>` Numeric simulated covariates.

`y` Numeric simulated outcome.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

`simulate_ancova_data` (one factor, one covariate special case), `simulate_anova_data`, `simulate_regression_data`

Other data simulators: simulate_ancova_data(), simulate_anova_data(), simulate_longitudinal_gompertz(), simulate_longitudinal_logistic(), simulate_longitudinal_negative_exponential(), simulate_longitudinal_ simulate_longitudinal_richards(), simulate_regression_data()

Examples

```
# 1. 2 x 2 randomized design, single covariate.
set.seed(113)
design_2x2 <- expand.grid(A = factor(1:2), B = factor(1:2))
design_2x2$true_mean_Y <- c(50, 60, 55, 65)   # cell means in expand.grid order

d1 <- simulate_ancova_factorial_data(
  a            = 2, b = 2,
  true_mean_Y  = design_2x2$true_mean_Y,
  true_mean_X  = matrix(10, nrow = 4, ncol = 1),  # constant covariate mean
  true_sd_Y      = 8,
  true_sd_X      = 3,
  true_cor_YX  = 0.40,
  n_per_cell   = 30
)
aggregate(y ~ A + B, data = d1, FUN = mean)

# 2. 3 x 2 nonrandomized design with two covariates and unequal n.
set.seed(113)
a <- 3; b <- 2; q <- 2
n_cells <- a * b
d2 <- simulate_ancova_factorial_data(
  a            = a, b = b,
  n_covariates = q,
  true_mean_Y  = c(50, 55, 60,  52, 58, 64),
  true_mean_X  = matrix(c(10, 11, 12,  9, 10, 11,
                          5,  6,  7,  4,  5,  6),
                        nrow = n_cells, ncol = q),
  true_sd_Y      = 8,
  true_sd_X      = c(3, 2),
  true_cor_YX  = c(0.40, 0.25),
  true_cor_XX  = matrix(c(1,   0.3,
                          0.3, 1),
                        nrow = 2),
  n_per_cell   = c(30, 25, 20,  35, 30, 25),
  randomized   = FALSE
)
head(d2)

# 3. 2 x 2 x 2 randomized design with one covariate.
set.seed(113)
d3 <- simulate_ancova_factorial_data(
  a       = 2, b = 2, c = 2,
  true_mean_Y = c(50, 55, 52, 57,  53, 58, 55, 60),  # 8 cells in A-fastest order
  true_mean_X = matrix(10, nrow = 8, ncol = 1),
  true_sd_Y = 8,
  true_sd_X = 3,
  true_cor_YX = 0.40,
```

```
  n_per_cell = 25
)
table(d3$A, d3$B, d3$C)  # 25 per cell, 200 total
```

`simulate_anova_data` *Simulate Data From a One-Way Fixed-Effects ANOVA Model*

Description

Generates random data appropriate for a one-way fixed-effects analysis of variance. Each group's observations are drawn from a normal distribution with that group's population mean and a common (or per-group) standard deviation. Per-group sample sizes may be equal or unequal.

Usage

```
simulate_anova_data(true_mean, true_sd, a, n_per_group, seed = NULL)
```

Arguments

`true_mean`	A numeric vector of length `a` giving the population mean in each group.
`true_sd`	Within-group population standard deviation. Either a single number (homoscedastic; common across groups) or a numeric vector of length `a` (heteroscedastic; one SD per group).
`a`	The number of fixed levels of the grouping factor (per the convention used throughout DMAR for fixed-factor designs).
`n_per_group`	A single number (equal sample size per group) or a numeric vector of length `a` giving the sample size in each group.
`seed`	Optional integer random seed for reproducibility (default `NULL`; supply an integer such as `113` for reproducible output).

Details

The fixed-effects ANOVA model assumes group-specific means and a common within-group variance. Setting `true_sd` to a vector relaxes the homoscedasticity assumption; in that case the simulated data violate the standard ANOVA assumption, which allows studying the robustness of the ANOVA *F* test or the performance of Welch-style alternatives.

Value

A long-format `data.frame` with one row per simulated subject and two columns:

`group` A factor with `a` levels.

`y` Numeric simulated outcome.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

simulate_ancova_data, simulate_regression_data, contrast_test, ss_power_contrast

Other data simulators: simulate_ancova_data(), simulate_ancova_factorial_data(), simulate_longitudinal_gom
simulate_longitudinal_logistic(), simulate_longitudinal_negative_exponential(), simulate_longitudinal_
simulate_longitudinal_richards(), simulate_regression_data()

Examples

```
# Three-group ANOVA, equal n per group.
set.seed(113)
d <- simulate_anova_data(true_mean = c(50, 55, 60), true_sd = 8, a = 3, n_per_group = 30)
aggregate(y ~ group, data = d, FUN = mean)

# Same design with unequal n per group.
simulate_anova_data(true_mean = c(50, 55, 60), true_sd = 8, a = 3,
                    n_per_group = c(40, 30, 20), seed = 113)

# Heteroscedastic case: each group has its own SD.
simulate_anova_data(true_mean = c(50, 55, 60), true_sd = c(5, 8, 12),
                    a = 3, n_per_group = 30, seed = 113)
```

---

simulate_longitudinal_gompertz

Simulate Data From a Gompertz Change (Growth) Model

Description

Generates longitudinal data from a random-coefficients Gompertz change model: each unit (a person, an animal, a tree) follows an S-shaped curve that, unlike the logistic, is not symmetric about its point of inflection, the parameters vary randomly across units, and each measurement adds level-one error. The deterministic part of the four parameter Gompertz curve is

$$\mu(t) = \alpha \exp\left(-\exp(-\gamma(t-\beta))\right) + \zeta,$$

the parameterization of Kelley (2005, 2008), which generalizes the three parameter Gompertz of the literature (Winsor, 1932; Ratkowsky, 1983) by adding ζ so the lower asymptote is itself a modeled quantity rather than fixed at zero.

Usage

```
simulate_longitudinal_gompertz(
  n_per_group,
  target_times = NULL,
  fixed_parameters,
  time_range = NULL,
  n_occasions = NULL,
  time_distribution = "uniform",
```

```
  random_variances = 0,
  random_correlation = NULL,
  error_variance = NULL,
  reliability = NULL,
  error_structure = c("independent", "ar1", "compound_symmetry", "toeplitz"),
  error_correlation = NULL,
  timing_sd = 0,
  seed = NULL
)
```

Arguments

`n_per_group` The number of units in each population (group), never a total over populations: a single positive integer, the number of units (persons, animals, trees, classrooms) whose trajectories are drawn, or a vector giving the number of units for each population, one entry per parameter vector in `fixed_parameters`.

`target_times` Numeric vector of the nominal measurement times, one shared schedule for every unit. Give either this or `time_range`.

`fixed_parameters`
The population parameters `c(alpha = , beta = , gamma = , zeta = )` (an unnamed length-4 vector is taken in that order), or a list of such vectors, one per population (distinct data generating parameter vectors):

- `alpha` The total change: the curve travels `alpha` units from the lower asymptote `zeta` to the upper asymptote $\alpha + \zeta$ (for $\gamma > 0$).
- `beta` The point of inflection on the time axis. The Gompertz inflection is early and asymmetric: it occurs where the curve passes through $\alpha/e + \zeta$, about 36.8% of the total change, so growth accelerates briefly and decelerates over a long approach to the ceiling.
- `gamma` The curvature: how sharply the curve rises through its inflection. Negative values flip the curve to decreasing.
- `zeta` The lower asymptote (for $\gamma > 0$). The intercept is $\phi = \alpha \exp(-\exp(\gamma\beta)) + \zeta$.

`time_range` Alternative to `target_times`: `c(lower,upper)` bounds from which each unit draws its own measurement times, so no two units share a schedule (e.g., age in weeks at testing rather than a fixed grade). Requires `n_occasions`; with `time_range`, the level-one error must be a single `error_variance` with the default independent structure, and `timing_sd` does not apply.

`n_occasions` The number of measurement occasions per unit. With `time_range`: a single positive integer (every unit measured the same number of times) or `c(min,max)`, from which each unit's number of measurement times is drawn uniformly.

`time_distribution`
Distribution of the unit-specific times over `time_range`; currently `"uniform"`.

`random_variances`
Between-unit variances of `(alpha, beta, gamma, zeta)`, a single number recycled to all four or a length-4 vector. Default `0`. A named vector over any subset of the parameter names (e.g., `c(beta = 1.2)`) varies only those named and leaves the rest fixed.

`random_correlation`
: Optional 4-by-4 correlation matrix among the random parameters; default uncorrelated.

`error_variance`, `reliability`, `error_structure`, `error_correlation`, `timing_sd`
: The level-one error and assessment-time machinery, with the same meaning as in `simulate_longitudinal_polynomial`: specify exactly one of `error_variance` or `reliability` (solved here through the first-order delta method true-score variance; because the curve is nonlinear in its parameters, that approximation, and the `reliability_by_occasion` attribute with it, can drift from the realized variance ratio when the random variances are large relative to the mean curve); `error_structure` and `error_correlation` set the across-occasion error correlation; `timing_sd` jitters the actual assessment times.

`seed`
: `NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

The choice between the Gompertz and the logistic is substantive, not cosmetic: both are S-shaped, but the logistic spends equal time approaching floor and ceiling while the Gompertz commits to an early inflection (36.8% of total change) followed by a long deceleration. Processes with rapid early gains and slow consolidation, common in learning and development, are natural Gompertz candidates. The Gompertz is the $\delta \to 0$ limit of the Richards curve (`simulate_longitudinal_richards`), which frees the inflection entirely (Kelley, 2005, 2008).

Value

A long-format `data.frame` with columns `id`, `population`, `occasion`, `target_time`, `time`, `true_score`, and `y`, directly usable with `plot_trajectories` and nonlinear mixed-model fitters such as `nlme::nlme()`. Attributes carry the `model`, `fixed_parameters`, `random_covariance`, `error_variance`, `error_covariance`, `reliability_by_occasion`, and `schedule` (`"shared"` or `"unit_specific"`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2005). *Estimating nonlinear change models in heterogeneous populations when class membership is unknown: Defining and developing the latent classification differential change model* (Doctoral dissertation). University of Notre Dame.

Kelley, K. (2008). Nonlinear change models in populations with unobserved heterogeneity. *Methodology, 4*(3), 97–112.

Ratkowsky, D. A. (1983). *Nonlinear regression modeling: A unified practical approach*. Marcel Dekker.

Winsor, C. P. (1932). The Gompertz curve as a growth curve. *Proceedings of the National Academy of Sciences, 18*(1), 1–8.

See Also

simulate_longitudinal_logistic for the symmetric sibling; simulate_longitudinal_richards for the family that subsumes both; simulate_longitudinal_negative_exponential; simulate_longitudinal_polynom plot_trajectories.

Other data simulators: simulate_ancova_data(), simulate_ancova_factorial_data(), simulate_anova_data(), simulate_longitudinal_logistic(), simulate_longitudinal_negative_exponential(), simulate_longitudinal_ simulate_longitudinal_richards(), simulate_regression_data()

Examples

```
# The six-curve illustration from Kelley (2005): alpha = 0.75 and
# zeta = 0.25 throughout, so every curve crosses its inflection at
# the same height, 0.75 / exp(1) + 0.25, about 0.526. The three
# rising curves share the inflection time beta = 2 and differ only
# in curvature; the three falling curves share beta = 3. Each curve
# is its own population of size one, so a single call draws the whole panel.
panel <- simulate_longitudinal_gompertz(
  n_per_group = 1, target_times = seq(0, 6, by = 0.1),
  fixed_parameters = list(
    c(alpha = 0.75, beta = 2, gamma =  1.75, zeta = 0.25),
    c(alpha = 0.75, beta = 2, gamma =  1.00, zeta = 0.25),
    c(alpha = 0.75, beta = 2, gamma =  0.45, zeta = 0.25),
    c(alpha = 0.75, beta = 3, gamma = -0.35, zeta = 0.25),
    c(alpha = 0.75, beta = 3, gamma = -0.60, zeta = 0.25),
    c(alpha = 0.75, beta = 3, gamma = -2.00, zeta = 0.25)),
  error_variance = 0
)
plot_trajectories(panel, id = "id", time = "time",
                  outcome = "true_score", group = "population")

# Individual differences in a single parameter: only the inflection
# time varies (a named entry leaves every other variance at zero),
# so every trajectory shares the floor and the ceiling but reaches
# its fastest growth at its own moment.
set.seed(113)
d_beta <- simulate_longitudinal_gompertz(
  n_per_group = 25, target_times = seq(0, 8, by = 0.5),
  fixed_parameters = c(alpha = 75, beta = 3, gamma = 0.55, zeta = 10),
  random_variances = c(beta = 0.8), error_variance = 0
)
plot_trajectories(d_beta, id = "id", time = "time",
                  outcome = "true_score")

# Individual differences in every parameter at once, plus level-one
# error: the realistic sampling model.
set.seed(113)
d <- simulate_longitudinal_gompertz(
  n_per_group = 30, target_times = 0:12,
  fixed_parameters = c(alpha = 75, beta = 3, gamma = 0.55, zeta = 10),
  random_variances = c(alpha = 36, beta = 0.8, gamma = 0.01, zeta = 9),
  error_variance = 16
```

```
)
plot_trajectories(d, id = "id", time = "time", outcome = "y")
```

`simulate_longitudinal_logistic` *Simulate Data From a Logistic Change (Growth) Model*

Description

Generates longitudinal data from a random-coefficients logistic change model: each unit (a person, an animal, a tree) follows an S-shaped (sigmoidal) curve with a lower and an upper asymptote and a symmetric point of inflection, the parameters vary randomly across units, and each measurement adds level-one error. The deterministic part of the four parameter logistic curve is

$$\mu(t) = \frac{\alpha}{1 + \exp(-\gamma(t - \beta))} + \zeta,$$

the parameterization of Kelley (2005, 2008), which generalizes the three parameter logistic of the literature (Ratkowsky, 1983) by adding ζ so the lower asymptote is itself a modeled quantity rather than fixed at zero.

Usage

```
simulate_longitudinal_logistic(
  n_per_group,
  target_times = NULL,
  fixed_parameters,
  time_range = NULL,
  n_occasions = NULL,
  time_distribution = "uniform",
  random_variances = 0,
  random_correlation = NULL,
  error_variance = NULL,
  reliability = NULL,
  error_structure = c("independent", "ar1", "compound_symmetry", "toeplitz"),
  error_correlation = NULL,
  timing_sd = 0,
  seed = NULL
)
```

Arguments

`n_per_group`	The number of units in each population (group), never a total over populations: a single positive integer, the number of units (persons, animals, trees, classrooms) whose trajectories are drawn, or a vector giving the number of units for each population, one entry per parameter vector in `fixed_parameters`.

`target_times` Numeric vector of the nominal measurement times, one shared schedule for every unit. Give either this or `time_range`.

`fixed_parameters`
The population parameters `c(alpha = , beta = , gamma = , zeta = )` (an unnamed length-4 vector is taken in that order), or a list of such vectors, one per population (distinct data generating parameter vectors):

- `alpha` The total change: the curve travels `alpha` units from the lower asymptote `zeta` to the upper asymptote $\alpha + \zeta$ (for $\gamma > 0$).
- `beta` The point of inflection on the time axis: the moment of fastest change, at which exactly half the total change has occurred (the curve passes through $\alpha/2 + \zeta$). The inflection of the logistic is symmetric: the approach to the ceiling mirrors the departure from the floor.
- `gamma` The curvature: how sharply the curve rises through its inflection. Negative values flip the curve to decreasing.
- `zeta` The lower asymptote (for $\gamma > 0$), freeing the curve's floor from the fixed zero of the three parameter logistic. The intercept is $\phi = \alpha/(1+\exp(\gamma\beta))+\zeta$.

`time_range` Alternative to `target_times`: `c(lower,upper)` bounds from which each unit draws its own measurement times, so no two units share a schedule (e.g., age in weeks at testing rather than a fixed grade). Requires `n_occasions`; with `time_range`, the level-one error must be a single `error_variance` with the default independent structure, and `timing_sd` does not apply.

`n_occasions` The number of measurement occasions per unit. With `time_range`: a single positive integer (every unit measured the same number of times) or `c(min,max)`, from which each unit's number of measurement times is drawn uniformly.

`time_distribution`
Distribution of the unit-specific times over `time_range`; currently `"uniform"`.

`random_variances`
Between-unit variances of `(alpha, beta, gamma, zeta)`, a single number recycled to all four or a length-4 vector. Default `0`. A named vector over any subset of the parameter names (e.g., `c(beta = 1.2)`) varies only those named and leaves the rest fixed.

`random_correlation`
Optional 4-by-4 correlation matrix among the random parameters; default uncorrelated.

`error_variance`, `reliability`, `error_structure`, `error_correlation`, `timing_sd`
The level-one error and assessment-time machinery, with the same meaning as in `simulate_longitudinal_polynomial`: specify exactly one of `error_variance` or `reliability` (solved here through the first-order delta method true-score variance; because the curve is nonlinear in its parameters, that approximation, and the `reliability_by_occasion` attribute with it, can drift from the realized variance ratio when the random variances are large relative to the mean curve); `error_structure` and `error_correlation` set the across-occasion error correlation; `timing_sd` jitters the actual assessment times.

seed — NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Every parameter is a landmark of the change process: the floor (zeta), the ceiling ($\alpha + \zeta$), when change is fastest (beta), and how concentrated the change is around that moment (gamma). The logistic is the $\delta = 1$ special case of the Richards curve (simulate_longitudinal_richards); its inflection always sits at 50% of the total change, which is the substantive assumption a researcher accepts in choosing it (Kelley, 2005, 2008).

Value

A long-format data.frame with columns id, population, occasion, target_time, time, true_score, and y, directly usable with plot_trajectories and nonlinear mixed-model fitters such as nlme::nlme(). Attributes carry the model, fixed_parameters, random_covariance, error_variance, error_covariance, reliability_by_occasion, and schedule ("shared" or "unit_specific").

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2005). *Estimating nonlinear change models in heterogeneous populations when class membership is unknown: Defining and developing the latent classification differential change model* (Doctoral dissertation). University of Notre Dame.

Kelley, K. (2008). Nonlinear change models in populations with unobserved heterogeneity. *Methodology, 4*(3), 97–112.

Ratkowsky, D. A. (1983). *Nonlinear regression modeling: A unified practical approach*. Marcel Dekker.

See Also

simulate_longitudinal_gompertz for the asymmetric sibling; simulate_longitudinal_richards for the family that subsumes both; simulate_longitudinal_negative_exponential; simulate_longitudinal_polynom plot_trajectories.

Other data simulators: simulate_ancova_data(), simulate_ancova_factorial_data(), simulate_anova_data(), simulate_longitudinal_gompertz(), simulate_longitudinal_negative_exponential(), simulate_longitudinal_ simulate_longitudinal_richards(), simulate_regression_data()

Examples

```
# A six-curve panel in the style of the growth-curve illustrations
# in Kelley (2005): floor 10 and ceiling 90 throughout, so every
# curve crosses its inflection at the same height (50, half the
# total change). The rising curves share beta = 6 and differ only in
# curvature; the falling curves mirror them. Each curve is its own population
```

```
# of size one, so a single call draws the whole panel.
panel <- simulate_longitudinal_logistic(
  n_per_group = 1, target_times = seq(0, 12, by = 0.1),
  fixed_parameters = list(
    c(alpha = 80, beta = 6, gamma =  2.0, zeta = 10),
    c(alpha = 80, beta = 6, gamma =  0.9, zeta = 10),
    c(alpha = 80, beta = 6, gamma =  0.5, zeta = 10),
    c(alpha = 80, beta = 6, gamma = -0.5, zeta = 10),
    c(alpha = 80, beta = 6, gamma = -0.9, zeta = 10),
    c(alpha = 80, beta = 6, gamma = -2.0, zeta = 10)),
  error_variance = 0
)
plot_trajectories(panel, id = "id", time = "time",
                  outcome = "true_score", group = "population")

# Individual differences in a single parameter: only the inflection
# time varies (a named entry leaves the other variances at zero), so
# every learner shares the floor and the ceiling but hits fastest
# growth on a different week.
set.seed(113)
d_beta <- simulate_longitudinal_logistic(
  n_per_group = 25, target_times = seq(0, 12, by = 0.5),
  fixed_parameters = c(alpha = 80, beta = 6, gamma = 0.9, zeta = 10),
  random_variances = c(beta = 1.2), error_variance = 0
)
plot_trajectories(d_beta, id = "id", time = "time",
                  outcome = "true_score")

# Unit-specific measurement times: each child is tested at their own
# ages, drawn uniformly between 40 and 90 weeks with five to nine
# visits each, rather than on one shared schedule.
set.seed(113)
d_ages <- simulate_longitudinal_logistic(
  n_per_group = 12, time_range = c(40, 90), n_occasions = c(5, 9),
  fixed_parameters = c(alpha = 80, beta = 65, gamma = 0.15, zeta = 10),
  random_variances = c(beta = 16), error_variance = 4
)
plot_trajectories(d_ages, id = "id", time = "time", outcome = "y")

# Individual differences in every parameter, plus level-one error:
# skill acquisition from a floor near 10 to a ceiling near 90,
# fastest around week 6.
set.seed(113)
d <- simulate_longitudinal_logistic(
  n_per_group = 30, target_times = 0:12,
  fixed_parameters = c(alpha = 80, beta = 6, gamma = 0.9, zeta = 10),
  random_variances = c(alpha = 36, beta = 1, gamma = 0.01, zeta = 9),
  error_variance = 16
)
plot_trajectories(d, id = "id", time = "time", outcome = "y")
```

`simulate_longitudinal_negative_exponential` *Simulate Data From a Negative Exponential (Asymptotic Regression) Change Model*

Description

Generates longitudinal data from a random-coefficients negative exponential change model, also called asymptotic regression (Stevens, 1951): each unit (a person, an animal, a tree) approaches an asymptote at a rate set by a curvature parameter, the parameters vary randomly across units, and each measurement adds level-one error. The deterministic part of the curve is

$$\mu(t) = \alpha + \zeta \exp(-\gamma t),$$

the parameterization of Kelley (2005, 2008). The negative exponential is the simplest of the package's nonlinear change curves: it has one asymptote and no point of inflection, so it describes change that is fastest at the first assessment and decelerates thereafter.

Usage

```
simulate_longitudinal_negative_exponential(
  n_per_group,
  target_times = NULL,
  fixed_parameters,
  time_range = NULL,
  n_occasions = NULL,
  time_distribution = "uniform",
  random_variances = 0,
  random_correlation = NULL,
  error_variance = NULL,
  reliability = NULL,
  error_structure = c("independent", "ar1", "compound_symmetry", "toeplitz"),
  error_correlation = NULL,
  timing_sd = 0,
  seed = NULL
)
```

Arguments

`n_per_group`	The number of units in each population (group), never a total over populations: a single positive integer, the number of units (persons, animals, trees, classrooms) whose trajectories are drawn, or a vector giving the number of units for each population, one entry per parameter vector in `fixed_parameters`.
`target_times`	Numeric vector of the nominal measurement times, one shared schedule for every unit. Give either this or `time_range`.

`fixed_parameters`
: The population parameters `c(alpha = , zeta = , gamma = )` (an unnamed length-3 vector is taken in that order), or a list of such vectors, one per population (distinct parameter vectors):

  `alpha` The asymptote: the value $\mu(t)$ approaches as t grows.

  `zeta` The negative of the total change: the curve starts at the intercept $\phi = \alpha + \zeta$ and travels $-\zeta$ units to the asymptote. Positive `zeta` gives asymptotic decay toward `alpha` from above; negative `zeta` gives asymptotic growth from below.

  `gamma` The curvature ($\gamma > 0$): the rate at which the remaining distance to the asymptote closes per unit time. Larger values reach the asymptote sooner.

`time_range`
: Alternative to `target_times`: `c(lower,upper)` bounds from which each unit draws its own measurement times, so no two units share a schedule (e.g., age in weeks at testing rather than a fixed grade). Requires `n_occasions`; with `time_range`, the level-one error must be a single `error_variance` with the default independent structure, and `timing_sd` does not apply.

`n_occasions`
: The number of measurement occasions per unit. With `time_range`: a single positive integer (every unit measured the same number of times) or `c(min,max)`, from which each unit's number of measurement times is drawn uniformly.

`time_distribution`
: Distribution of the unit-specific times over `time_range`; currently `"uniform"`.

`random_variances`
: Between-unit variances of `(alpha, zeta, gamma)`, a single number recycled to all three or a length-3 vector. Default `0` (a common curve for everyone). A named vector over any subset of the parameter names (e.g., `c(gamma = 0.02)`) varies only those named and leaves the rest fixed.

`random_correlation`
: Optional 3-by-3 correlation matrix among the random parameters; default uncorrelated.

`error_variance`, `reliability`, `error_structure`, `error_correlation`, `timing_sd`
: The level-one error and assessment-time machinery, with the same meaning as in `simulate_longitudinal_polynomial`: specify exactly one of `error_variance` (scalar, per-occasion vector, or full covariance matrix) or `reliability` (a target average per-occasion reliability, solved by the delta method here); `error_structure` and `error_correlation` set the across-occasion error correlation; `timing_sd` jitters each unit's actual assessment times around the nominal targets.

`seed`
: `NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Every parameter answers a substantive question: where does change end (`alpha`), where does it start ($\phi = \alpha + \zeta$), and how fast does the gap close (`gamma`)? That interpretability is the argument for nonlinear change models over polynomials, whose coefficients describe no landmark of the process (Kelley, 2005, 2008); the package vignette on nonlinear growth develops the comparison.

Value

A long-format `data.frame` with columns `id`, `population`, `occasion`, `target_time`, `time`, `true_score`, and `y`, directly usable with `plot_trajectories` and nonlinear mixed-model fitters such as `nlme::nlme()`. Attributes carry the `model`, the `fixed_parameters`, the between-unit covariance `random_covariance`, the level-one `error_variance` and `error_covariance`, and `reliability_by_occasion` (from the first-order delta method true-score variance, which can drift from the realized variance ratio when the random variances are large relative to the mean curve). The `schedule` attribute records `"shared"` or `"unit_specific"`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2005). *Estimating nonlinear change models in heterogeneous populations when class membership is unknown: Defining and developing the latent classification differential change model* (Doctoral dissertation). University of Notre Dame.

Kelley, K. (2008). Nonlinear change models in populations with unobserved heterogeneity. *Methodology, 4*(3), 97–112.

Stevens, W. L. (1951). Asymptotic regression. *Biometrics, 7*(3), 247–267.

See Also

`simulate_longitudinal_logistic`, `simulate_longitudinal_gompertz`, `simulate_longitudinal_richards` for the sigmoidal members of the family; `simulate_longitudinal_polynomial` for the polynomial counterpart; `plot_trajectories` for plotting the result.

Other data simulators: `simulate_ancova_data()`, `simulate_ancova_factorial_data()`, `simulate_anova_data()`, `simulate_longitudinal_gompertz()`, `simulate_longitudinal_logistic()`, `simulate_longitudinal_polynomial()`, `simulate_longitudinal_richards()`, `simulate_regression_data()`

Examples

```
# The six-curve illustration from Kelley (2005): three growth curves
# (intercept 0, asymptote 1) that differ only in curvature, and
# three decay curves (intercept 1, asymptote 0) that mirror them.
# Each curve is its own population of size one, so a single call draws the
# whole panel.
panel <- simulate_longitudinal_negative_exponential(
  n_per_group = 1, target_times = seq(0, 10, by = 0.1),
  fixed_parameters = list(
    c(alpha = 1, zeta = -1, gamma = 0.9),
    c(alpha = 1, zeta = -1, gamma = 0.4),
    c(alpha = 1, zeta = -1, gamma = 0.2),
    c(alpha = 0, zeta =  1, gamma = 1.2),
    c(alpha = 0, zeta =  1, gamma = 0.5),
    c(alpha = 0, zeta =  1, gamma = 0.3)),
  error_variance = 0
)
```

```
plot_trajectories(panel, id = "id", time = "time",
                  outcome = "true_score", group = "population")

# Individual differences in a single parameter: only the curvature
# varies (a named entry leaves the other variances at zero), so all
# trajectories share their start and their destination but close the
# gap at their own rates.
set.seed(113)
d_gamma <- simulate_longitudinal_negative_exponential(
  n_per_group = 25, target_times = seq(0, 8, by = 0.5),
  fixed_parameters = c(alpha = 100, zeta = -80, gamma = 0.5),
  random_variances = c(gamma = 0.02), error_variance = 0
)
plot_trajectories(d_gamma, id = "id", time = "time",
                  outcome = "true_score")

# Individual differences in every parameter, plus level-one error:
# vocabulary learning that starts near 20 words (phi = alpha + zeta),
# climbs toward an asymptote near 100, and closes about 40% of the
# remaining gap per month (gamma = 0.5).
set.seed(113)
d <- simulate_longitudinal_negative_exponential(
  n_per_group = 30, target_times = 0:8,
  fixed_parameters = c(alpha = 100, zeta = -80, gamma = 0.5),
  random_variances = c(alpha = 25, zeta = 16, gamma = 0.01),
  error_variance = 9
)
head(d)
plot_trajectories(d, id = "id", time = "time", outcome = "y")
```

`simulate_longitudinal_polynomial` *Simulate Data From a Polynomial Change (Growth) Model*

Description

Generates longitudinal data from a random-coefficients polynomial change model: each subject follows a degree-P polynomial in time whose coefficients vary randomly across subjects, and each measurement adds independent level-one error. The polynomial order is general (order 0 is a flat line, 1 a straight line, 2 a quadratic, and so on), one or several populations may differ in their mean trajectories, the level-one error can be set directly or pinned to a target measurement reliability, and the actual time of each assessment may jitter away from its nominal target. This is the Monte Carlo companion to `ss_power_pcm`, which plans power for the same model under the closed-form assumptions of Raudenbush and Liu (2001); the simulator can relax those assumptions (notably the assumption of fixed, error-free, equally reliable assessment times) and study what happens.

Usage

```
simulate_longitudinal_polynomial(
  n_per_group,
  target_times = NULL,
  fixed_coefficients,
  time_range = NULL,
  n_occasions = NULL,
  time_distribution = "uniform",
  random_variances = 0,
  random_correlation = NULL,
  error_variance = NULL,
  reliability = NULL,
  error_structure = c("independent", "ar1", "compound_symmetry", "toeplitz"),
  error_correlation = NULL,
  timing_sd = 0,
  seed = NULL
)
```

Arguments

`n_per_group`	The number of units in each population (group), never a total over populations: a single positive integer (equal number of units in every population) or a numeric vector of length `G` giving the number of subjects in each of the `G` populations, where `G` is the number of mean trajectories supplied through `fixed_coefficients`.
`target_times`	A numeric vector of the nominal (planned) measurement times, length M, one shared schedule for every unit. They need not be equally spaced. A degree-P model requires $M \geq P + 1$ occasions. Give either this or `time_range`.
`fixed_coefficients`	The population mean trajectory, as the coefficients of a polynomial in time, ordered from the intercept upward: `c(b0, b1, ..., bP)` encodes $b_0 + b_1 t + b_2 t^2 + \cdots + b_P t^P$. The length sets the polynomial order P (length 1 is order 0, a flat line at `b0`). For several populations, pass a *list* of equal-length coefficient vectors, one per population; the populations then differ in their mean trajectories but share the variance components below (the Raudenbush-Liu two-group setup).
`time_range`	Alternative to `target_times`: `c(lower,upper)` bounds from which each unit draws its own measurement times, so no two units share a schedule (e.g., age in weeks at testing rather than a fixed grade). Requires `n_occasions`; with `time_range`, the level-one error must be a single `error_variance` with the default independent structure, and `timing_sd` does not apply.
`n_occasions`	The number of measurement occasions per unit. With `time_range`: a single positive integer (every unit measured the same number of times) or `c(min,max)`, from which each unit's number of measurement times is drawn uniformly. Every value must be at least $P + 1$ so each unit's trajectory identifies the polynomial.
`time_distribution`	Distribution of the unit-specific times over `time_range`; currently `"uniform"`.

`random_variances`
: The between-subject variances of the polynomial coefficients (the diagonal of the level-two covariance matrix), as a single number recycled to all $P + 1$ coefficients or a vector of length $P + 1$. Default `0`, a fixed common trajectory with no between-subject heterogeneity. Entries may be zero coefficient by coefficient (e.g., a random intercept with a fixed slope).

`random_correlation`
: Optional $(P + 1) \times (P + 1)$ correlation matrix among the random coefficients. Default `NULL` treats them as uncorrelated. Combined with `random_variances` it forms the level-two covariance $T = DRD$, with D the diagonal matrix of coefficient standard deviations.

`error_variance`
: The level-one (within-subject) measurement error variance σ_e^2. Most simply a single non-negative number (the same error variance at every occasion). It may instead be a length-M vector of per-occasion (heteroscedastic) error variances, or a full $M \times M$ error covariance matrix when the errors are correlated across occasions in an arbitrary (unstructured) way. For the common structured cases, give a scalar or vector here and set `error_structure` and `error_correlation` instead of building the matrix by hand. Specify exactly one of `error_variance` or `reliability`.

`reliability`
: A target measurement reliability in $(0, 1)$ from which σ_e^2 is derived (see Details). The value is the *average* per-occasion reliability across `target_times`; the per-occasion reliabilities, which generally differ from one another, are returned in the `"reliability_by_occasion"` attribute. Requires at least one positive entry in `random_variances`. The solved error variance is homoscedastic and may still be given an across-occasion correlation through `error_structure`. Specify exactly one of `error_variance` or `reliability`.

`error_structure`
: The correlation pattern of the level-one errors across occasions: `"independent"` (the default, uncorrelated errors), `"ar1"` (a first-order autoregressive decay $\rho^{|j-k|}$, so errors at occasions closer in time are more alike), `"compound_symmetry"` (a constant correlation ρ between every pair of occasions), or `"toeplitz"` (a banded structure set by the lag-1 through lag-$(M - 1)$ correlations). Ignored when `error_variance` is a full covariance matrix, which already fixes the structure.

`error_correlation`
: The correlation parameter(s) for `error_structure`: a single number ρ for `"ar1"` and `"compound_symmetry"`, or a vector of the lag-1 to lag-$(M - 1)$ correlations for `"toeplitz"`. Left `NULL` for `"independent"`. The implied correlation matrix must be positive semidefinite.

`timing_sd`
: The standard deviation of the difference between a subject's actual and nominal assessment time, as a single number recycled to all occasions or a vector of length M. Default `0`, every subject measured exactly on schedule. A positive value draws each subject's actual time at occasion m as $\tau_m + N(0, \text{timing_sd}_m^2)$ and evaluates that subject's true score at the actual time, while the nominal target is retained in a separate column (see Details).

`seed`
: `NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

The model. Subject i in population g has a random coefficient vector $\pi_i = (\pi_{i0}, \ldots, \pi_{iP})$ drawn from a multivariate normal with mean the population's `fixed_coefficients` β_g and covariance T. The latent trajectory is the polynomial $\mu_i(t) = \sum_{k=0}^{P} \pi_{ik}\, t^k$, and the observed score at a measurement time t is $y = \mu_i(t) + e$, with $e \sim N(0, \sigma_e^2)$ independent across occasions. Order 0 collapses to a flat line $\mu_i(t) = \pi_{i0}$; order 1 is the straight-line growth model underlying Raudenbush and Liu (2001) and `ss_power_pcm`.

Coefficient metric. The coefficients here are the ordinary (raw) polynomial coefficients on t^k, which is the most transparent metric for specifying a trajectory. The derivative-scaled change coefficient used by `ss_power_pcm` and the Raudenbush-Liu power formulas is $P!$ times the leading (highest-order) coefficient supplied here, so a quadratic with `fixed_coefficients = c(b0, b1, b2)` corresponds to a Raudenbush-Liu quadratic change coefficient of $2!\, b_2 = 2b_2$.

Reliability varies by occasion. At a measurement time t the implied between-subject (true-score) variance is the quadratic form $c(t)^\top T\, c(t)$ with $c(t) = (1, t, t^2, \ldots, t^P)^\top$, so the classical reliability of the observed score,

$$\rho_{XX}(t) = \frac{c(t)^\top T\, c(t)}{c(t)^\top T\, c(t) + \sigma_e^2},$$

generally *changes from occasion to occasion*: a growth measurement is not equally reliable everywhere, because the spread of true scores depends on where in time you measure relative to the centering of the polynomial and the random-effect structure. When `reliability` is supplied, σ_e^2 is solved (by `uniroot`) so that the average of $\rho_{XX}(t)$ over the nominal `target_times` equals the requested value; the occasion-by-occasion reliabilities are returned in the `"reliability_by_occasion"` attribute so the variation is visible rather than hidden behind a single number. Reliability is only meaningful when there is true-score variance to detect, so this route requires `random_variances` to be positive for at least one coefficient.

Measurement errors need not be independent or equal. The simplest model adds an independent, equal-variance error at every occasion, but repeated measurements of the same person are often correlated (an unmodeled state, a rater, or an instrument carries over from one wave to the next) and may be more or less variable at different waves. The level-one errors are drawn from $N(0, \Sigma_e)$, and Σ_e can be set three ways: a scalar or per-occasion `error_variance` combined with an `error_structure` (`"ar1"` for autoregressive decay, the natural choice when occasions are ordered in time; `"compound_symmetry"` for an equicorrelated error; `"toeplitz"` for a general banded pattern), or a full covariance matrix passed directly as `error_variance`. Because classical reliability at an occasion is a marginal quantity, it depends only on the diagonal of Σ_e; the across-occasion error correlation leaves `"reliability_by_occasion"` unchanged but does affect how a mixed model that assumes independent errors performs, which is exactly the kind of misspecification this simulator is meant to let a user study.

Assessment timing is rarely exact. Designs are written as if every subject is measured at the same fixed times ("the 7-day follow-up"), but in practice people arrive early or late, so the actual time differs from the nominal target. Setting `timing_sd > 0` draws each subject's actual time per occasion and evaluates the true score *at the time the measurement really happened*, while `target_time` keeps the nominal value an analyst would typically use. Analyzing on the nominal time when the data were in fact collected on jittered times biases estimates of the change coefficients, and the bias grows with the order of the trend and with the size of the timing variability. The two time columns let a user quantify that bias by fitting the same model on `time` versus `target_time`.

Why the closed-form Raudenbush-Liu planner does not cover all of this. The power formulas in Raudenbush and Liu (2001), carried by `ss_power_pcm`, are exact under three assumptions this simulator can relax: every subject is measured at the *same*, equally spaced, *error-free* occasion times; the level-one error variance is a single constant (so the closed form needs no notion of an occasion-varying reliability); and the within-subject sampling variance of the change coefficient has the known form $V = \sigma_e^2 f^{2p}(M-p-1)!/[K_p(M+p)!]$. Those assumptions buy a clean formula, but real designs violate them: assessments drift in time, and reliability is not the same at every wave. This function is the Monte Carlo complement that lets a researcher generate data under the messier reality and check how far the closed-form power and the fitted estimates can be trusted.

Value

A long-format `data.frame` with one row per subject-occasion and the columns

`id` Factor uniquely identifying each subject.

`population` Factor with `G` levels (`"1"`, ...) giving each unit's population (its data generating parameter vector). With one parameter vector there is one level.

`occasion` Integer occasion index, `1` to M.

`target_time` The nominal (planned) measurement time.

`time` The actual measurement time (equal to `target_time` when `timing_sd = 0`, otherwise jittered).

`true_score` The subject's latent trajectory value at the actual time, before level-one error.

`y` The observed score, `true_score` plus level-one error.

The returned object carries attributes `"error_variance"` (the σ_e^2 used, a scalar when the errors are homoscedastic and independent, otherwise the vector of per-occasion error variances), `"error_covariance"` (the full $M \times M$ level-one error covariance actually used), `"reliability_by_occasion"` (the per-occasion reliabilities at the nominal times), `"random_covariance"` (the level-two covariance T), `"polynomial_order"` (P), and `"schedule"` (`"shared"` or `"unit_specific"`). With `time_range` there is no shared occasion grid, so `"error_covariance"` is `NA` and `"reliability_by_occasion"` is `NA`. The format is directly usable with `plot_trajectories` and with mixed-model fitters such as `nlme::lme()` or `lme4::lmer()`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Rausch, J. R. (2011). Sample size planning for longitudinal models: Accuracy in parameter estimation for polynomial change parameters. *Psychological Methods, 16*(4), 391–405. doi:10.1037/a0023352

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 15 on the analysis of repeated measures and growth.)

Raudenbush, S. W., & Liu, X.-F. (2001). Effects of study duration, frequency of observation, and sample size on power in studies of group differences in polynomial change. *Psychological Methods, 6*(4), 387–401. doi:10.1037/1082989X.6.4.387

See Also

ss_power_pcm for closed-form power planning on the same model, plot_trajectories to visualize the simulated curves, and mvrnorm for the random-coefficient draw.

Other data simulators: simulate_ancova_data(), simulate_ancova_factorial_data(), simulate_anova_data(), simulate_longitudinal_gompertz(), simulate_longitudinal_logistic(), simulate_longitudinal_negative_exp simulate_longitudinal_richards(), simulate_regression_data()

Examples

```
# 1. One population of linear growers with a random intercept and a random slope,
#    measured yearly for four years (five occasions), with the level-one
#    error set directly.
set.seed(113)
d <- simulate_longitudinal_polynomial(
  n_per_group        = 50,
  target_times       = 0:4,
  fixed_coefficients = c(10, 1.5),          # intercept 10, slope 1.5 per year
  random_variances   = c(4, 0.25),          # var(intercept) = 4, var(slope) = .25
  error_variance     = 1
)
head(d)

# 2. Two populations that differ only in their slope (a treatment that changes
#    the rate of growth). Pass a list of coefficient vectors, one per population.
set.seed(113)
two <- simulate_longitudinal_polynomial(
  n_per_group        = c(40, 40),
  target_times       = 0:4,
  fixed_coefficients = list(control = c(10, 1.0), treatment = c(10, 1.8)),
  random_variances   = c(4, 0.25),
  error_variance     = 1
)
aggregate(y ~ population + occasion, data = two, FUN = mean)

# 3. Pin the level-one error to a target reliability instead of setting it
#    directly. The single number is the average reliability across occasions;
#    the per-occasion values differ and are returned as an attribute.
set.seed(113)
rel <- simulate_longitudinal_polynomial(
  n_per_group        = 100,
  target_times       = 0:4,
  fixed_coefficients = c(10, 1.5),
  random_variances   = c(4, 0.25),
  reliability        = 0.80
)
attr(rel, "error_variance")
round(attr(rel, "reliability_by_occasion"), 3)   # not constant across waves

# 4. Assessment-time jitter: the nominal "yearly" schedule, but subjects
#    actually arrive a little early or late (SD of about six weeks on a
#    one-year scale). The nominal and actual times are kept in separate
```

```
#    columns so the consequences of analyzing on the nominal time can be
#    studied.
set.seed(113)
jit <- simulate_longitudinal_polynomial(
  n_per_group        = 30,
  target_times       = 0:4,
  fixed_coefficients = c(10, 1.5),
  random_variances   = c(4, 0.25),
  error_variance     = 1,
  timing_sd          = 0.12
)
head(jit[, c("id", "occasion", "target_time", "time")])

# 4b. Unit-specific measurement times: each of 12 children is tested
#     between 40 and 90 weeks of age, five to nine times, no two on the
#     same schedule. The level-one error is a single variance; the
#     "schedule" attribute records the design.
set.seed(113)
ages <- simulate_longitudinal_polynomial(
  n_per_group        = 12,
  time_range         = c(40, 90),
  n_occasions        = c(5, 9),
  fixed_coefficients = c(10, 0.5),
  random_variances   = c(4, 0.01),
  error_variance     = 2
)
attr(ages, "schedule")
table(table(ages$id))   # units per occasion count

# 5. A flat line (order 0): no growth, only a random subject level and
#    measurement error. The coefficient vector has length one.
set.seed(113)
flat <- simulate_longitudinal_polynomial(
  n_per_group        = 20,
  target_times       = 0:4,
  fixed_coefficients = 5,
  random_variances   = 2,
  error_variance     = 1
)
head(flat)

# 6. Autocorrelated measurement error: the same error variance at each wave,
#    but the level-one errors decay as an AR(1) process (errors at adjacent
#    occasions correlate 0.5), the kind of dependence a model assuming
#    independent errors would miss. The full error covariance is returned.
set.seed(113)
ar <- simulate_longitudinal_polynomial(
  n_per_group        = 40,
  target_times       = 0:4,
  fixed_coefficients = c(10, 1.5),
  random_variances   = c(4, 0.25),
  error_variance     = 1,
  error_structure    = "ar1",
```

```
  error_correlation  = 0.5
)
round(attr(ar, "error_covariance"), 3)
```

`simulate_longitudinal_richards` *Simulate Data From a Richards Change (Growth) Model*

Description

Generates longitudinal data from a random-coefficients Richards change model, the flexible sigmoidal family whose point of inflection is itself a parameter rather than a fixed fraction of total change (Richards, 1959). The deterministic part of the five parameter Richards curve is

$$\mu(t) = \frac{\alpha}{\left(1 + \delta \exp(-\gamma(t - \beta))\right)^{1/\delta}} + \zeta,$$

the parameterization of Kelley (2005, 2008). The Richards family subsumes the package's other sigmoidal curves as special cases: at $\delta = 1$ it is exactly the logistic, and in the limit $\delta \to 0$ it is the Gompertz, so δ lets the data, or the theory, choose where in its course the process turns from acceleration to deceleration. Guo, Cheng, and Kelley (2016) use this flexibility to model self-replicating malware propagation, where the network structure moves the inflection of the outbreak.

Usage

```
simulate_longitudinal_richards(
  n_per_group,
  target_times = NULL,
  fixed_parameters,
  time_range = NULL,
  n_occasions = NULL,
  time_distribution = "uniform",
  random_variances = 0,
  random_correlation = NULL,
  error_variance = NULL,
  reliability = NULL,
  error_structure = c("independent", "ar1", "compound_symmetry", "toeplitz"),
  error_correlation = NULL,
  timing_sd = 0,
  seed = NULL
)
```

Arguments

`n_per_group`	The number of units in each population (group), never a total over populations: a single positive integer, the number of units (persons, animals, trees, classrooms) whose trajectories are drawn, or a vector giving the number of units for each population, one entry per parameter vector in `fixed_parameters`.

`target_times` Numeric vector of the nominal measurement times, one shared schedule for every unit. Give either this or `time_range`.

`fixed_parameters`

The population parameters `c(alpha = , beta = , gamma = , delta = , zeta = )` (an unnamed length-5 vector is taken in that order), or a list of such vectors, one per population (distinct data generating parameter vectors):

`alpha` The total change: the curve travels `alpha` units from the lower asymptote `zeta` to the upper asymptote $\alpha + \zeta$ (for $\gamma > 0$).

`beta` The point of inflection on the time axis.

`gamma` The curvature: how sharply the curve rises through its inflection. Negative values flip the curve to decreasing.

`delta` The shape parameter ($\delta > 0$): where the inflection falls on the outcome axis, $y^* = \alpha(1+\delta)^{-1/\delta} + \zeta$. At $\delta = 1$ the inflection sits at half the total change (the logistic); as $\delta \to 0$ it slides down to $\alpha/e + \zeta$, about 36.8% (the Gompertz); larger δ pushes it later than halfway.

`zeta` The lower asymptote (for $\gamma > 0$). The intercept is $\phi = \alpha(1+\delta\exp(\gamma\beta))^{-1/\delta} + \zeta$.

`time_range` Alternative to `target_times`: `c(lower,upper)` bounds from which each unit draws its own measurement times, so no two units share a schedule (e.g., age in weeks at testing rather than a fixed grade). Requires `n_occasions`; with `time_range`, the level-one error must be a single `error_variance` with the default independent structure, and `timing_sd` does not apply.

`n_occasions` The number of measurement occasions per unit. With `time_range`: a single positive integer (every unit measured the same number of times) or `c(min,max)`, from which each unit's number of measurement times is drawn uniformly.

`time_distribution`

Distribution of the unit-specific times over `time_range`; currently `"uniform"`.

`random_variances`

Between-unit variances of `(alpha, beta, gamma, delta, zeta)`, a single number recycled to all five or a length-5 vector. Default `0`. A named vector over any subset of the parameter names (e.g., `c(delta = 0.04)`) varies only those named and leaves the rest fixed. A positive variance on `delta` must be small enough that no unit's drawn δ falls at or below zero, where the curve is undefined; the function stops with a count if that happens.

`random_correlation`

Optional 5-by-5 correlation matrix among the random parameters; default uncorrelated.

`error_variance`, `reliability`, `error_structure`, `error_correlation`, `timing_sd`

The level-one error and assessment-time machinery, with the same meaning as in `simulate_longitudinal_polynomial`: specify exactly one of `error_variance` or `reliability` (solved here through the first-order delta method true-score variance; because the curve is nonlinear in its parameters, that approximation, and the `reliability_by_occasion` attribute with it, can drift from the realized variance ratio when the random variances are large relative to the mean curve); `error_structure` and `error_correlation` set the across-occasion error correlation; `timing_sd` jitters the actual assessment times.

seed — NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

The logistic and the Gompertz fix where the inflection falls as a fraction of total change (50% and 36.8%); choosing between them is choosing that fraction by assumption. The Richards curve makes the fraction estimable through δ, at the price of one more parameter and a harder estimation problem, since δ and γ carry overlapping information in finite samples (Richards, 1959; Kelley, 2005). Simulating from the Richards family at several δ values is the natural way to study whether a design can tell those shapes apart.

Value

A long-format `data.frame` with columns `id`, `population`, `occasion`, `target_time`, `time`, `true_score`, and `y`, directly usable with `plot_trajectories` and nonlinear mixed-model fitters such as `nlme::nlme()`. Attributes carry the `model`, `fixed_parameters`, `random_covariance`, `error_variance`, `error_covariance`, `reliability_by_occasion`, and `schedule` (`"shared"` or `"unit_specific"`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Guo, H., Cheng, H. K., & Kelley, K. (2016). Impact of network structure on malware propagation: A growth curve perspective. *Journal of Management Information Systems, 33*(1), 296–325.

Kelley, K. (2005). *Estimating nonlinear change models in heterogeneous populations when class membership is unknown: Defining and developing the latent classification differential change model* (Doctoral dissertation). University of Notre Dame.

Kelley, K. (2008). Nonlinear change models in populations with unobserved heterogeneity. *Methodology, 4*(3), 97–112.

Richards, F. J. (1959). A flexible growth function for empirical use. *Journal of Experimental Botany, 10*(2), 290–301.

See Also

`simulate_longitudinal_logistic` (the $\delta = 1$ special case), `simulate_longitudinal_gompertz` (the $\delta \to 0$ limit), `simulate_longitudinal_negative_exponential`, `simulate_longitudinal_polynomial`, `plot_trajectories`.

Other data simulators: `simulate_ancova_data()`, `simulate_ancova_factorial_data()`, `simulate_anova_data()`, `simulate_longitudinal_gompertz()`, `simulate_longitudinal_logistic()`, `simulate_longitudinal_negative_exp`
`simulate_longitudinal_polynomial()`, `simulate_regression_data()`

Examples

```
# The family in one panel: five Richards curves sharing alpha, beta,
# gamma, and zeta and differing only in the shape parameter delta.
# delta near 0 is the Gompertz, delta = 1 is the logistic, and
# larger delta pushes the inflection later than halfway. Each curve
# is its own population of size one, so a single call draws the whole panel.
panel <- simulate_longitudinal_richards(
  n_per_group = 1, target_times = seq(0, 12, by = 0.1),
  fixed_parameters = list(
    c(alpha = 80, beta = 6, gamma = 0.9, delta = 0.02, zeta = 10),
    c(alpha = 80, beta = 6, gamma = 0.9, delta = 0.25, zeta = 10),
    c(alpha = 80, beta = 6, gamma = 0.9, delta = 1.00, zeta = 10),
    c(alpha = 80, beta = 6, gamma = 0.9, delta = 3.00, zeta = 10),
    c(alpha = 80, beta = 6, gamma = 0.9, delta = 8.00, zeta = 10)),
  error_variance = 0
)
plot_trajectories(panel, id = "id", time = "time",
                  outcome = "true_score", group = "population")

# Individual differences in a single parameter: only the shape
# varies (a named entry leaves the other variances at zero), so the
# curves agree on floor, ceiling, timing, and curvature yet turn
# from acceleration to deceleration at different heights.
set.seed(113)
d_delta <- simulate_longitudinal_richards(
  n_per_group = 25, target_times = seq(0, 12, by = 0.5),
  fixed_parameters = c(alpha = 80, beta = 6, gamma = 0.9,
                       delta = 1, zeta = 10),
  random_variances = c(delta = 0.04), error_variance = 0
)
plot_trajectories(d_delta, id = "id", time = "time",
                  outcome = "true_score")

# Individual differences in every parameter, plus level-one error:
# a late-inflecting outbreak-style curve, with delta = 3 placing the
# inflection at about 63% of total change, (1 + 3)^(-1/3).
set.seed(113)
d <- simulate_longitudinal_richards(
  n_per_group = 30, target_times = 0:12,
  fixed_parameters = c(alpha = 80, beta = 6, gamma = 0.9,
                       delta = 3, zeta = 10),
  random_variances = c(alpha = 36, beta = 1, gamma = 0.01,
                       delta = 0.04, zeta = 9),
  error_variance = 16
)
plot_trajectories(d, id = "id", time = "time", outcome = "y")

# delta = 1 reproduces the logistic exactly: with no randomness and
# no error, the two simulators return identical true scores.
d_r <- simulate_longitudinal_richards(
  n_per_group = 1, target_times = 0:5,
  fixed_parameters = c(alpha = 80, beta = 3, gamma = 1,
```

```
                        delta = 1, zeta = 10),
  error_variance = 0)
d_l <- simulate_longitudinal_logistic(
  n_per_group = 1, target_times = 0:5,
  fixed_parameters = c(alpha = 80, beta = 3, gamma = 1, zeta = 10),
  error_variance = 0)
all.equal(d_r$true_score, d_l$true_score)
```

`simulate_regression_data` *Simulate Data From a Multivariate Normal Multiple-Regression Model*

Description

Generates random data $(Y, X_1, \ldots, X_p)$ jointly multivariate normal with user-specified marginal means, marginal SDs, and full correlation structure. Useful for sensitivity analyses, Monte Carlo studies of regression sample size methods, and teaching demonstrations.

Usage

```
simulate_regression_data(
  N,
  p,
  true_cor_YX,
  true_cor_XX = NULL,
  true_mean_Y = 0,
  true_mean_X = 0,
  true_sd_Y = 1,
  true_sd_X = 1,
  seed = NULL,
  column_names = NULL
)
```

Arguments

`N`	The total sample size (a positive integer $\geq p + 2$).
`p`	The number of predictor variables.
`true_cor_YX`	A numeric vector of length `p` giving the population correlations between Y and each predictor X_j.
`true_cor_XX`	A $p \times p$ symmetric correlation matrix for the predictors. Defaults to the identity matrix (orthogonal predictors).
`true_mean_Y`	The population mean of Y (default `0`).
`true_mean_X`	A numeric vector of length `p` giving the population mean of each predictor (default `0`, recycled across predictors).

true_sd_Y	The population standard deviation of Y (default 1, in which case the simulated Y is on the standardized scale).
true_sd_X	A single number or a numeric vector of length p giving the population standard deviation of each predictor (default 1, standardized).
seed	Optional integer random seed for reproducibility (default NULL).
column_names	Optional character vector of length p + 1 giving column names for the returned data.frame; defaults to c("y", "x1", "x2", ..., "xp").

Details

Internally the joint correlation matrix is assembled as

$$R = \begin{pmatrix} 1 & \rho_{YX}^{\top} \\ \rho_{YX} & R_{XX} \end{pmatrix},$$

where ρ_{YX} is true_cor_YX and R_{XX} is true_cor_XX, converted to a covariance matrix via the supplied SDs, and N draws are taken using mvrnorm. The resulting Y and predictors satisfy the requested marginal means and standard deviations and (in expectation) the requested correlation structure.

Value

A data.frame with N rows and p + 1 columns: the outcome Y (first column) followed by predictors $X_1, \ldots, X_p$.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

simulate_ancova_data, simulate_anova_data, ss_aipe_R2, ss_aipe_reg_coef, ci_R2

Other data simulators: simulate_ancova_data(), simulate_ancova_factorial_data(), simulate_anova_data(), simulate_longitudinal_gompertz(), simulate_longitudinal_logistic(), simulate_longitudinal_negative_exp simulate_longitudinal_polynomial(), simulate_longitudinal_richards()

Examples

```
# Five orthogonal predictors, each correlating .30 with Y.
set.seed(113)
d <- simulate_regression_data(
  N      = 200,
  p      = 5,
  true_cor_YX = rep(0.30, 5)
)
summary(lm(y ~ ., data = d))$r.squared   # about 0.45: five predictors, each 0.30^2

# Predictors with shared structure (exchangeable correlation matrix).
true_cor_XX <- matrix(0.5, nrow = 5, ncol = 5); diag(true_cor_XX) <- 1
simulate_regression_data(
```

```
  N      = 300,
  p      = 5,
  true_cor_YX = c(.50, .40, .30, .20, .10),
  true_cor_XX = true_cor_XX,
  seed   = 113
)[1:3, ]
```

`skewness` *Bias-Corrected Sample Skewness*

Description

Computes the sample skewness of a numeric vector using the bias-corrected (SAS/SPSS Type 2) formula. Skewness measures asymmetry of the distribution: zero is symmetric, positive values indicate a right-tail heavier than the left, negative values the reverse.

Usage

```
skewness(x, na_rm = TRUE)
```

Arguments

`x`	A numeric vector.
`na_rm`	Logical. If `TRUE` (the default), missing values are removed before computation. If `FALSE`, the result is `NA` when `x` contains any `NA`.

Details

The reported value is

$$\hat{\gamma}_1^{(2)} = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^3 ,$$

where s is the (divisor-$n-1$) sample standard deviation. This is sometimes called the "Type 2" or SAS/SPSS-default form; it is approximately unbiased under normality.

Three formulas. Three measures of sample skewness are in use. With $m_r = \sum_{i=1}^{n} (x_i - \bar{x})^r / n$, they are $g_1 = m_3 / m_2^{3/2}$ (Type 1), $G_1 = g_1 \sqrt{n(n-1)}/(n-2)$ (Type 2, the value above), and $b_1 = g_1((n-1)/n)^{3/2}$ (Type 3). The three differ by factors that depend only on n and approach 1 as n grows. `skewness()` reports G_1, which is also the skewness that `descriptives` reports. `kurtosis` describes the three corresponding measures of kurtosis.

Value

A single numeric value: the bias-corrected sample skewness, or `NA_real_` when fewer than three non-missing observations are available or when the sample standard deviation is zero.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Joanes, D. N., & Gill, C. A. (1998). Comparing measures of sample skewness and kurtosis. *The Statistician, 47*(1), 183–189. doi:10.1111/14679884.00122

See Also

kurtosis, descriptives

Other descriptive statistics: descriptives(), kurtosis()

Examples

```
# Symmetric data: skewness near zero.
set.seed(113)
skewness(rnorm(1000))

# Right-skewed data: positive value.
skewness(rexp(1000, rate = 1))

# The classic 1:5 example: exactly symmetric (returns 0).
skewness(1:5)
```

---

smd *Standardized Mean Difference*

Description

Estimates the standardized mean difference (Cohen's *d*), the difference between two group means divided by the pooled standard deviation, from either raw data or summary statistics. Expressing the difference in standard deviation units frees the comparison from the raw measurement units, so effects can be compared across measures and studies; either the ordinary or the unbiased (Hedges, 1981) estimate can be returned.

Usage

```
smd(
  group_1 = NULL,
  group_2 = NULL,
  mean_1 = NULL,
  mean_2 = NULL,
  s_1 = NULL,
  s_2 = NULL,
  s = NULL,
```

```
  n_1 = NULL,
  n_2 = NULL,
  unbiased = FALSE
)
```

Arguments

group_1	Raw data for group 1
group_2	Raw data for group 2
mean_1	The mean of group 1
mean_2	The mean of group 2
s_1	The standard deviation of group 1 (i.e., the square root of the unbiased estimator of the population variance)
s_2	The standard deviation of group 2 (i.e., the square root of the unbiased estimator of the population variance)
s	The pooled group standard deviation (i.e., the square root of the unbiased estimator of the population variance)
n_1	The sample size within group 1
n_2	The sample size within group 2
unbiased	Returns the unbiased estimate of the standardized mean difference

Details

When unbiased=TRUE, the unbiased estimate of the standardized mean difference is returned (Hedges, 1981). The correction multiplies the estimate by $J(\nu) = \Gamma(\nu/2)/\{\sqrt{\nu/2}\,\Gamma((\nu-1)/2)\}$ with $\nu = n_1 + n_2 - 2$ degrees of freedom. $J(1) = 0$: on 1 degree of freedom the mean of the estimate does not exist, so no multiple of it is unbiased, and the corrected value would be 0 whatever the data. A call with unbiased = TRUE and $n_1 + n_2 = 3$ (possible only from summary statistics) is therefore refused, with a condition of class dmar_hedges_correction_undefined (see dmar_metadata). With $n_1 + n_2 = 2$ there are no degrees of freedom, $J(0)$ is not defined, and the call is refused with the same class.

From summary statistics with s_1 and s_2, each of n_1 and n_2 must be at least 2, because a group of one observation has no standard deviation; a smaller value is refused with a condition of class dmar_argument_invalid, as raw data with a group of one are refused.

Value

A 1-row data.frame with columns term ("smd") and value (the estimated standardized mean difference).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K. (2005) The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals, *Educational and Psychological Measurement, 65*, 51–69. doi:10.1177/0013164404264850

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

smd_c, ci_smd, ci_smd_c, ss_aipe_smd, ss_power_smd, plot_smd, ci_nc_t

Examples

```
# Generate sample data.
set.seed(113)
g.1 <- rnorm(n = 25, mean = .5, sd = 1)
g.2 <- rnorm(n = 25, mean = 0, sd = 1)
smd(group_1 = g.1, group_2 = g.2)

M.x <- .66745
M.y <- .24878
sd <- 1.048
smd(mean_1 = M.x, mean_2 = M.y, s = sd)

M.x <- .66745
M.y <- .24878
n1 <- 25
n2 <- 25
sd.1 <- .95817
sd.2 <- 1.1311
smd(mean_1 = M.x, mean_2 = M.y, s_1 = sd.1, s_2 = sd.2, n_1 = n1, n_2 = n2)
```

```
smd(mean_1 = M.x, mean_2 = M.y, s_1 = sd.1, s_2 = sd.2, n_1 = n1, n_2 = n2,
    unbiased = TRUE)
```

`smd_c` *Standardized Mean Difference Using the Control Group as the Basis of Standardization*

Description

Estimates the standardized mean difference using the control group standard deviation as the basis of standardization (Glass's *g*), from either raw data or summary statistics, in ordinary or unbiased form. Standardizing by the control group alone keeps the scale of the effect free of any treatment effect on variability.

Usage

```
smd_c(
  group_T = NULL,
  group_C = NULL,
  mean_T = NULL,
  mean_C = NULL,
  s_C = NULL,
  n_C = NULL,
  unbiased = FALSE
)
```

Arguments

`group_T`	Raw data for the treatment group
`group_C`	Raw data for the control group
`mean_T`	The mean of the treatment group
`mean_C`	The mean of the control group
`s_C`	The standard deviation of the control group (i.e., the square root of the unbiased estimator of the population variance)
`n_C`	The sample size of the control group
`unbiased`	Returns the unbiased estimate of the standardized mean difference using the standard deviation of the control group

Details

When `unbiased=TRUE`, the unbiased estimate of the standardized mean difference (using the control group as the basis of standardization) is returned (Hedges, 1981). Although the unbiased estimate of the standardized mean difference is not often reported, at least at the present time, it is nevertheless made available to those who are interested in calculating this quantity.

The correction multiplies the estimate by $J(\nu) = \Gamma(\nu/2)/\{\sqrt{\nu/2}\,\Gamma((\nu-1)/2)\}$ with $\nu = n_C - 1$ degrees of freedom, the degrees of freedom of the control group standard deviation. $J(1) = 0$: on 1 degree of freedom the mean of the estimate does not exist, so no multiple of it is unbiased, and the corrected value would be 0 whatever the data. A call with `unbiased = TRUE` and a control group of 2 is therefore refused, with a condition of class `dmar_hedges_correction_undefined` (see `dmar_metadata`). A control group of 1 given with summary statistics is refused with the same class for another reason: $J(0)$ is not defined, because one observation has no standard deviation. Use `unbiased = FALSE`, or a control group of at least 3. `unbiased = TRUE` from summary statistics without `n_C` is refused with a condition of class `dmar_argument_invalid`.

Value

A 1-row `data.frame` with columns `term` (`"smd_c"`) and `value` (the estimated standardized mean difference using the control group standard deviation as the basis of standardization).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Glass, G. V. (1976). Primary, secondary, and meta-analysis of research. *Educational Researcher, 5*, 3–8.

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

See Also

`smd`, `ci_nc_t`

Examples

```
# Generate sample data.
set.seed(113)
g.T <- rnorm(n = 25, mean = .5, sd = 1)
g.C <- rnorm(n = 25, mean = 0, sd = 1)
smd_c(group_T = g.T, group_C = g.C)

M.T <- .66745
M.C <- .24878
sd.c <- 1.1311
n.c <- 25
smd_c(mean_T = M.T, mean_C = M.C, s_C = sd.c)
smd_c(mean_T = M.T, mean_C = M.C, s_C = sd.c, n_C = n.c, unbiased = TRUE)
```

`smd_trimmed` *Robust Standardized Mean Difference (Algina-Keselman-Penfield)*

Description

Computes the Algina, Keselman, and Penfield (2005) robust standardized mean difference, which replaces the sample means and pooled SD in Cohen's d with their trimmed-mean and Winsorized-SD counterparts:

$$d_R \;=\; 0.642 \cdot \frac{\bar{X}_{t,\,1} - \bar{X}_{t,\,2}}{s_{W,\,p}},$$

where $\bar{X}_{t,\,j}$ is the trimmed mean of group j, $s_{W,\,p}$ is the pooled Winsorized standard deviation, and 0.642 is the Algina-Keselman-Penfield (2005) constant chosen so that d_R equals Cohen's δ when the data are normal. Returns the point estimate, a noncentral *t* confidence interval, and the trimmed / Winsorized summary statistics.

Usage

```
smd_trimmed(x, y, trim = 0.2, conf_level = 0.95)
```

Arguments

`x, y`	Numeric vectors of observations from the two groups.
`trim`	Proportion to trim from each tail (and Winsorize from each tail). Must be in $[0, 0.5)$. Default `0.20` (Wilcox's 2017 recommended setting).
`conf_level`	Confidence level for the CI. Default `0.95`.

Details

Why robust. Under heavy-tailed or skewed marginal distributions, the conventional Cohen's d has very large standard error and biased coverage. Kelley (2005) documents the coverage distortion of parametric confidence intervals for the standardized mean difference under nonnormal distributions. Replacing means by 20%- trimmed means and SD by 20%-Winsorized SD yields an estimator whose efficiency under normality is roughly 96% (Wilcox, 2017, ch. 5) and whose efficiency under heavy-tailed contamination is substantially higher than Cohen's d.

The 0.642 constant. $0.642 = \mathrm{SD}(X_W)/\mathrm{SD}(X) = \sqrt{\mathrm{Var}(X_W)/\mathrm{Var}(X)}$ when $X \sim N(0,1)$ and X_W is the 20%-Winsorized version. Choosing this constant makes $d_R = \delta$ when the data are normal, so the new estimator is on the same scale as Cohen's d.

CI. The CI follows the construction of Keselman, Algina, Lix, Wilcox, and Deering (2008): Yuen's (1974) *t*-statistic on the trimmed-mean difference (their Equation 8) is referred to a noncentral *t* distribution with the Yuen-Welch approximate degrees of freedom (their Equation 9), the noncentrality parameters whose tail probabilities bracket the observed statistic are located with `ci_nc_t`, and those limits are rescaled to the d_R metric. The degrees of freedom are reported in the `df_yuen` row of the returned table. At `trim = 0` the construction reduces to the Welch approximate degrees of freedom interval; for the exact equal-variance interval on the untrimmed standardized mean difference use `ci_smd`.

Value

A `data.frame` with rows for the robust *d* estimate, the lower/upper CI bounds, the per-group trimmed means, the per-group Winsorized SDs, the pooled Winsorized SD, and the effective sample sizes (after trimming).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J., Keselman, H. J., & Penfield, R. D. (2005). An alternative to Cohen's standardized mean difference effect size: A robust parameter and confidence interval in the two independent groups case. *Psychological Methods, 10*(3), 317–328. doi:10.1037/1082989X.10.3.317

Kelley, K. (2005). The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals. *Educational and Psychological Measurement, 65*(1), 51–69. doi:10.1177/0013164404264850

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Keselman, H. J., Algina, J., Lix, L. M., Wilcox, R. R., & Deering, K. N. (2008). A generally robust approach for testing hypotheses and setting confidence intervals for effect sizes. *Psychological Methods, 13*(2), 110–129. doi:10.1037/1082989X.13.2.110

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

Wilcox, R. R. (2017). *Introduction to robust estimation and hypothesis testing* (4th ed.). Academic Press.

Yuen, K. K. (1974). The two-sample trimmed *t* for unequal population variances. *Biometrika, 61*(1), 165–170.

See Also

`smd`, `var_smd_trimmed`, `ci_smd`, `ci_nc_t`

Other effect size estimates: `cles()`, `cliff_delta()`, `correction_for_attenuation()`, `eta_squared()`, `eta_squared_generalized()`, `expected_partial_r()`, `expected_r()`, `expected_smd()`, `nnt_from_smd()`, `omega_squared()`, `probability_of_superiority_paired()`, `proportion_of_superiority()`, `responder_analysis()`

Examples

```
# 1. Two normal groups: robust d agrees closely with Cohen's d.
set.seed(113)
x <- rnorm(40, 0,   1); y <- rnorm(40, 0.5, 1)
smd_trimmed(x, y)

# 2. Contaminated y: a few outliers; robust d shifts much less
```

```
#        than Cohen's d.
set.seed(113)
x <- rnorm(40, 0, 1)
y <- c(rnorm(38, 0.5, 1), 30, -25)
smd_trimmed(x, y)
smd(x, y)
```

`ss_aipe_c` *Sample Size Planning for an ANOVA Contrast From the Accuracy in Parameter Estimation (AIPE) Perspective*

Description

Plans the sample size *per group* so that the confidence interval for an unstandardized contrast of means in a fixed effects analysis of variance is sufficiently narrow, following the accuracy in parameter estimation (AIPE) approach: the design goal is a contrast estimated with the precision the research question requires, not merely one detected as nonzero. AIPE sample size planning for ANOVA and ANCOVA contrasts is developed in Lai and Kelley (2012).

Usage

```
ss_aipe_c(
  true_var_error,
  c_weights,
  width,
  conf_level = 0.95,
  assurance = NULL,
  true_kurtosis = 0
)
```

Arguments

`true_var_error`	The population error variance: the variance of the errors common to the groups, which the mean square error (the mean square within groups) of an analysis of variance estimates. It is a variance, not a standard deviation: a planning value stated as an error standard deviation is squared before it is given. Must be a single finite positive number. This is the one way to state the error variability, as in `ss_aipe_c_sensitivity` and `ss_power_contrast`.
`c_weights`	The contrast weights
`width`	The desired full width of the obtained confidence interval
`conf_level`	The desired confidence interval coverage, (i.e., 1 - Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in `ci_nc_t`).
`assurance`	Parameter to ensure that the obtained confidence interval width is narrower than the desired width with a specified degree of certainty (must be NULL or between zero and unity)

`true_kurtosis` The population excess kurtosis of the errors (their fourth standardized moment less 3), on the scale `kurtosis` reports, so the value it gives for the residuals of a pilot study can be passed directly. It is used only with `assurance`, through the effective degrees of freedom described in Details: a negative value (lighter tails than the normal, as bounded outcomes such as proportions and rating scale scores often have) makes the sample variance less variable than normal theory implies, and a positive value (heavier tails) makes it more variable. The default `0` is the normal theory value. Must be a single finite number greater than -2; no distribution has a smaller excess kurtosis, and only a symmetric two-point distribution attains -2.

Details

Without `assurance`, the planned *n* per group is the smallest integer (and at least 2) satisfying

$$n \geq \frac{4\, t^2_{1-\alpha/2,\, J(n-1)}\, \sigma^2 \sum_j c_j^2}{\omega^2},$$

where σ^2 is the error variance (`true_var_error`), c_j are the contrast weights of the J groups, and ω is the desired full width. That is the width the interval would have if the pooled sample variance S^2 equaled σ^2; because S^2 varies from sample to sample, the realized width exceeds ω in a large proportion of studies, a proportion that approaches one half as *n* grows. With `assurance` equal to γ, the right side is multiplied by $\chi^2_{\gamma,\,\nu}/\nu$, the γ quantile of a chi square on ν degrees of freedom divided by ν, which is the γ quantile of S^2/σ^2 when $\nu S^2/\sigma^2$ is chi square on ν degrees of freedom; the realized width is then at most ω with probability γ. For normal errors $\nu = J(n-1)$ and the chi square law is exact.

For errors that are not normal, the variability of S^2 depends on their excess kurtosis κ. With J groups of n, the pooled variance has $\mathrm{var}(S^2) = \sigma^4[2/\{J(n-1)\} + \kappa/(nJ)]$, so, for an assurance above one half, a plan made under normality delivers more assurance than requested when $\kappa < 0$ and less when $\kappa > 0$. The `true_kurtosis` argument replaces $J(n-1)$ in the chi square quantile with the effective degrees of freedom

$$\nu = \frac{2}{2/\{J(n-1)\} + \kappa/(nJ)},$$

the degrees of freedom for which χ^2_ν/ν has the variance of S^2/σ^2; $\kappa = 0$ gives $\nu = J(n-1)$ and the normal theory plan, and the degrees of freedom of the *t* quantile are not changed. This is the approximation of Box (1953, p. 319, Eq. 3): a variance estimate on ϕ degrees of freedom from a population with excess kurtosis γ_2 has variance $2\sigma^4/(d\phi)$, with $d = [1 + \phi\gamma_2/\{2(\phi+1)\}]^{-1}$, and is distributed approximately as a normal theory estimate on $d\phi$ degrees of freedom; with $\phi = n - 1$ in each group, ν is d times $J(n-1)$. The approximation matches two moments and leaves the shape to the chi square. It requires errors with a finite fourth moment, and in Box's sampling experiments (p. 325) the large sample result was approached more slowly when the kurtosis was large. The kurtosis of a pilot study's residuals is itself an imprecise estimate in samples of the size pilots have, so the value supplied is a planning value, as the error variance is. The sensitivity companion, `ss_aipe_c_sensitivity`, simulates normal errors and has no `true_kurtosis` argument, so it evaluates a plan under the normal theory law, not under the errors a nonzero `true_kurtosis` allows for.

A Monte Carlo check of the approximation used two groups, `c_weights = c(1, -1)`, an error variance of 1, a width of .8, and an assurance of .80, with 10,000 replications per condition. With

uniform errors (excess kurtosis -1.2), the plan made with the kurtosis supplied (53 per group) met the width target in a proportion .811 of the replications, where the plan made under normality (55 per group) met it in .915; with *t* errors on 10 degrees of freedom (excess kurtosis 1), the plan with the kurtosis supplied (56 per group) met it in .811, where the normal theory plan (55 per group) met it in .776, short of the .80 requested. The study's script and results are kept with the package's development records and are not distributed with the package.

Value

A 1-row `data.frame` with columns `term` and `value`:

`necessary_n_per_group`
: the necessary sample size *per group*

A plan made with `assurance` and a nonzero `true_kurtosis` keeps that one row, carries the value as the attribute `true_kurtosis`, prints a line under the table saying that the assurance allows for that excess kurtosis of the errors, and reports it as a `true_kurtosis` column of `generics::glance()`. With the default `true_kurtosis = 0`, or without `assurance`, no attribute is added.

Note

Be sure to use the error variance and not its square root (i.e., the standard deviation of the errors).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Box, G. E. P. (1953). Non-normality and tests on variances. *Biometrika, 40*(3–4), 318–335. doi:10.1093/biomet/40.34.318

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

`ss_aipe_sc`, `ss_aipe_c_ancova`, `ci_c`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Suppose the population error variance of some three-group ANOVA model
# is believed to be 40. The researcher is interested in the difference
# between the mean of group 1 and the average of means of group 2 and 3.
# To plan the sample size so that, with 90 percent certainty, the
# obtained 95 percent full confidence interval width is no wider than 3:

ss_aipe_c(true_var_error = 40, c_weights = c(1, -0.5, -0.5),
          width = 3, assurance = .90)

# The planning values can come from a pilot study. The posttest scores in
# depression_bdi have an error variance near 40, and their residuals are
# lighter tailed than the normal, as scores on a bounded inventory often
# are. Thirty residuals estimate the kurtosis imprecisely, so it is a
# planning value to weigh, as the error variance is.
pilot <- lm(bdi_post ~ condition, data = depression_bdi)
sigma(pilot)^2
kurtosis(residuals(pilot))
ss_aipe_c(true_var_error = sigma(pilot)^2, c_weights = c(1, -0.5, -0.5),
          width = 3, assurance = .90, true_kurtosis = kurtosis(residuals(pilot)))
```

---

ss_aipe_cliff_delta *Sample Size for AIPE on Cliff's δ*

Description

Determines the sample size needed for the confidence interval on Cliff's (1993) δ (and equivalently Vargha-Delaney's $A = (\delta + 1)/2$) to have a desired width, using the maximum-variance bound on $\hat{\delta}$ (Feng & Cliff, 2004, Equation 6, p. 324).

Usage

```
ss_aipe_cliff_delta(
  true_delta,
  width,
  which_width = c("Full", "Lower", "Upper"),
  conf_level = 0.95,
  ratio = 1,
  assurance = NULL
)
```

Arguments

`true_delta`	Planning value of the population Cliff's δ; numeric scalar in $(-1, 1)$.
`width`	Desired full width of the CI on δ.
`which_width`	`"Full"` (default), `"Lower"`, or `"Upper"`.

`conf_level`	Desired confidence level. Default `0.95`.
`ratio`	Ratio n_1/n_2 of the two group sample sizes. Default `1` (balanced).
`assurance`	Optional. Probability that the realized CI is no wider than `width`.

Details

Maximum-variance bound. The variance of $\hat{\delta}$ at a given δ is largest in the bimodal configuration, where it equals $(1-\delta^2)/n_b$ with n_b the bimodal group's size; for unequal groups the smaller sample size is used conservatively (Feng & Cliff, 2004, Equation 6 and following text, p. 324):

$$\mathrm{Var}(\hat{\delta}) \;\le\; \frac{(1-\delta^2)}{\min(n_1, n_2)}.$$

Setting the half-width of a Wald-style CI $z_{1-\alpha/2}\sqrt{\mathrm{Var}(\hat{\delta})}$ equal to the target half-width and solving gives the recommended per-group sample size. The bound is conservative; the realized CI is generally narrower than the target.

Allocation. The bound is dominated by $\min(n_1, n_2)$, so balanced allocation (`ratio = 1`) is approximately optimal under standard conditions; unbalanced allocations require the larger total *N* to achieve the same precision.

Value

A `data.frame` with rows for the recommended group sample sizes n_1, n_2, the expected CI width, and the inputs echoed back.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin, 114*(3), 494–509. doi:10.1037/00332909.114.3.494

Feng, D., & Cliff, N. (2004). Monte Carlo evaluation of ordinal d with improved confidence interval. *Journal of Modern Applied Statistical Methods, 3*(2), 322–332. doi:10.22237/jmasm/1099267560

Vargha, A., & Delaney, H. D. (2000). A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics, 25*(2), 101–132. doi:10.3102/10769986025002101

See Also

`cliff_delta`, `ss_aipe_partial_r`, `ss_aipe_semipartial_r`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta_sensitivity()`, `ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,

ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(), ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivit ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(), ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# 1. Plan a balanced design so the 95% CI on delta has full width
#        <= 0.20 when anticipating delta = 0.30.
ss_aipe_cliff_delta(true_delta = 0.30, width = 0.20)

# 2. Unbalanced: twice as many in group 1.
ss_aipe_cliff_delta(true_delta = 0.30, width = 0.20, ratio = 2)
```

---

ss_aipe_cliff_delta_sensitivity

Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for Cliff's Delta

---

Description

Quantifies how much misspecification of the population Cliff's delta ($\delta = \Pr(X > Y) - \Pr(X < Y)$) distorts an AIPE-based sample size plan. On each replication the function simulates two independent samples whose population Cliff's delta equals `true_delta`, computes the sample `cliff_delta` and its CI, and summarizes the realized widths and coverage.

Replications without an interval. When the two simulated groups are completely separated (every score of one above every score of the other), or every score is equal, the estimated variance of Cliff's delta is 0 and `cliff_delta` reports both limits equal to the estimate, with a warning that they are not a confidence interval. Such a replication has no interval here. Its estimate still enters `mean_cliff_delta`, `median_cliff_delta`, and `sd_cliff_delta`. The width, width-target, and coverage rows (`mean_ci_width`, `median_ci_width`, `sd_ci_width`, `pct_ci_less_width`, `pct_ci_miss_low`, `pct_ci_miss_high`, `total_type_I_error`) describe the replications that have an interval, and are `NA` when none has; `pct_ci_less_width` is the number of intervals no wider than `width` divided by the number of replications with an interval. The warning of each replication is not passed on. One warning of class `dmar_replications_without_interval`, with the fields `n_without_interval` and `G`, states how many there were, and the `simulation` attribute records the counts. Separation is common when `true_delta` is large in absolute value and the groups are small: with `true_delta = 0.9` and six per group, about four replications in ten are separated.

Data generating mechanism. The simulator draws each sample from a normal distribution and chooses the mean shift so that the implied Cliff's delta equals `true_delta`. For normal samples $\delta = 2\Phi(\Delta/\sqrt{2}) - 1$ where Δ is the standardized mean difference, so the simulator sets $\Delta = \sqrt{2} \cdot \Phi^{-1}((1+\delta)/2)$.

Usage

```
ss_aipe_cliff_delta_sensitivity(
  true_delta = NULL,
  supposed_delta = NULL,
  ratio = 1,
  width,
  specified_N = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_delta`	Population Cliff's delta (the data generating value); in $(-1, 1)$.
`supposed_delta`	The supposed population Cliff's delta, the planning value passed to `ss_aipe_cliff_delta` (not an estimate computed from data); supply this or `specified_N` but not both.
`ratio`	Allocation ratio n_1/n_2 (default 1).
`width`	Desired full width of the CI on Cliff's delta.
`specified_N`	Total sample size to evaluate (split per `ratio`).
`conf_level`	Confidence level (default `0.95`).
`assurance`	Optional assurance probability.
`G`	Number of Monte Carlo replications.
`print_iter`	Logical.
`filename`	Optional path for a comma separated file recording every replication (the sample Cliff's delta, the two confidence limits, the interval width, and two indicators of whether the interval missed `true_delta` below or above): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`. A replication without an interval is recorded with its estimate and with `NA` in the other columns.
`seed`	Optional integer seed. The default `NULL` draws from, and advances, the current random number stream; an integer makes the call reproducible, and the caller's random number state is restored when the call returns.

Value

A `data.frame` (class `dmar_tbl`) with columns `term` and `value` and these rows: `mean_cliff_delta`, `median_cliff_delta`, and `sd_cliff_delta` (summaries of the `G` estimates); `mean_ci_width`, `median_ci_width`, and `sd_ci_width` (summaries of the widths of the replications with an interval); `pct_ci_less_width` (the proportion of all `G` replications whose interval is no wider than `width`); `pct_ci_miss_low`, `pct_ci_miss_high`, and `total_type_I_error` (among the replications with an interval, the proportions whose interval lies wholly above `true_delta`, wholly below

it, and either); `n_1`, `n_2`, and `total_N` (the sample sizes simulated); and the inputs `true_delta`, `supposed_delta` (`NA` when `specified_N` was given), `ratio`, `width`, `conf_level`, and `assurance` (present only when an assurance was supplied). The `simulation` attribute records the numbers of replications attempted, with an interval, and without one (see the section on the record of the simulation).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin, 114*(3), 494–509. doi:10.1037/00332909.114.3.494

Feng, D., & Cliff, N. (2004). Monte Carlo evaluation of ordinal d with improved confidence interval. *Journal of Modern Applied Statistical Methods, 3*(2), 322–332. doi:10.22237/jmasm/1099267560

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ss_aipe_cliff_delta, cliff_delta

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(), ss_aipe_equivalence_r_sensit ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(), ss_aipe_icc(), ss_aipe_icc_sensitivit ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivity(), ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(), ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(), ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# Small G keeps the Monte Carlo sweep fast; raise G for a real plan.
ss_aipe_cliff_delta_sensitivity(
  true_delta = 0.30, supposed_delta = 0.30,
  width = 0.30, G = 25, print_iter = FALSE, seed = 113
)

# A large effect with six per group: some replications are completely
# separated and have no interval. The width and coverage rows describe
# the others, and one warning says how many were without an interval.
ss_aipe_cliff_delta_sensitivity(
  true_delta = 0.90, specified_N = 12, width = 0.40, G = 100,
  seed = 113
)
```

---

ss_aipe_composite_sem *Sample Size for Accurate Estimation of a Set of SEM Parameters*

Description

Determine the necessary sample size for a structural equation model study so that the confidence interval for every parameter of interest is sufficiently narrow in the same study, or, given a sample size, return how narrow the set of intervals can be expected to be. This is the accuracy in parameter estimation (AIPE) counterpart of ss_power_composite_sem: where that function plans for every parameter to be statistically significant jointly, this one plans for every parameter to be estimated with a confidence interval no wider than desired, the goal when the research questions concern the magnitudes of the effects rather than their existence. The parameters of interest are any labeled parameters of a lavaan analysis model, structural paths, loadings, covariances, or quantities defined with := such as an indirect effect, and any subset of them can make up the set.

Usage

```
ss_aipe_composite_sem(
  model,
  true_cov_matrix = NULL,
  true_model = NULL,
  true_mean = NULL,
  parameters = NULL,
  width,
  conf_level = 0.95,
  assurance = NULL,
  specified_N = NULL,
  G = 1000,
  seed = NULL,
  ...
)
```

Arguments

model	A single character string giving the free analysis model in lavaan model syntax (see model.syntax), the model that would be fit to the data. Each parameter of interest must carry a parameter label so it can be referred to by name, for example "f2 ~ b*f1" labels the structural path b, and "ab := a*b" defines an indirect effect from the labeled paths a and b.
true_cov_matrix	Population covariance matrix of the observed variables, with row and column names matching the observed variables in model. It is typically obtained from a fully fixed population model via cov_sem. Supply exactly one of true_cov_matrix or true_model.
true_model	A single character string giving the population model in lavaan model syntax with every parameter fixed to its population value, from which cov_sem derives true_cov_matrix (and the population means, when the model has a mean structure). The fixed values are values the researcher posits (from theory, prior studies, or pilot data), never sample estimates. Supply exactly one of true_cov_matrix or true_model.
true_mean	Optional population means of the observed variables, used with true_cov_matrix: a named numeric vector with one entry per observed variable, or an unnamed vector in the row order of true_cov_matrix. The default NULL is zero means. Means matter only when the analysis model has a mean structure (an intercept term such as s ~ 1, as in a latent growth curve model); when true_model is supplied its mean structure provides the means and true_mean must not also be given.
parameters	Character vector of the parameter labels that make up the set. The default NULL uses every user-labeled parameter in model, in order of appearance, so labeling exactly the parameters of interest is the simplest way to state the set.
width	The desired full confidence interval width for each parameter of interest: a single value applied to every parameter, or a named numeric vector with one entry per

parameter label. An unnamed vector of several widths is not accepted, so a width can never silently attach to the wrong parameter.

`conf_level` Confidence level of each interval (default 0.95).

`assurance` The desired probability that a single study yields confidence intervals no wider than desired for every parameter of interest simultaneously (a value in [0.5, 1)), or `NULL` (the default) to plan against the expected widths instead; see Details.

`specified_N` Total sample size *N*; if supplied, the realized interval widths at that *N* are summarized rather than a sample size planned.

`G` Number of converged Monte Carlo replications per evaluated sample size (default 1000). The simulation error of each estimated proportion is about $\sqrt{p(1-p)/G}$; raise `G` for a sharper answer.

`seed` Optional integer seed for reproducibility. The default `NULL` uses the current state of the random number generator; a supplied seed is set internally and the prior state restored on exit.

`...` Additional arguments passed to `sem`, both when the population values are resolved and for every Monte Carlo fit (for example `std.lv = TRUE` or `missing = "listwise"`). A robust estimator such as `"MLM"` cannot be used here: the same arguments reach the setup fit, which is always from summary statistics, and lavaan refuses a robust estimator there. The Monte Carlo data are multivariate normal by construction, so a robust estimator would buy nothing. Every name given here must be an argument or an option of **lavaan** (see `lavOptions`); any other name, and any argument without a name, is refused before a model is fitted, with a condition of class `dmar_argument_invalid` that gives the name as it was typed

Details

AIPE planning for a single targeted SEM parameter is available in closed form (Lai & Kelley, 2011; `ss_aipe_sem_path`), but most studies estimate several effects and report all of them; a design is only as informative as its widest interval of interest. This function plans for the set by a priori Monte Carlo simulation (Muthén & Muthén, 2002; Maxwell, Kelley, & Rausch, 2008): for a candidate *N*, `G` data sets are drawn from the multivariate normal population with covariance matrix `true_cov_matrix`, the analysis model is fit to each, and each parameter's Wald confidence interval width, twice $z_{1-\alpha/2}$ times its standard error, is recorded. Because the estimates share one fitted model, the widths are dependent; the simulation reflects that dependence exactly, at the stated *N*, with no asymptotic shortcut.

Two planning criteria are available. With `assurance = NULL` the necessary sample size is the smallest *N* at which the mean simulated width of every parameter's interval is at or below its desired width, the expected-width criterion of the AIPE framework applied to each member of the set. Widths vary from sample to sample around their means, so each interval separately lands at or below its desired width in roughly half of the realizations. That is a statement about one interval at a time, not about the set: the probability that *every* interval is narrow enough at once falls well below one half as soon as more than one parameter binds, and falls further the more parameters are targeted and the more weakly their widths move together. Planning the whole set to a stated probability is exactly what `assurance` is for. Supplying `assurance` plans against the joint event instead: the smallest *N* at which the proportion of replications where every interval is simultaneously within its desired width reaches the assurance. The joint event is contained in each marginal event, so its

probability is at most the smallest per-parameter proportion, and the pct_ci_less_width_<label> rows show which parameter binds the design.

When specified_N is NULL the search starts at the largest of the per-parameter closed-form sample sizes (the no-assurance approximation ss_aipe_sem_path uses, computed from the asymptotic variances before any simulation), brackets the crossing geometrically, and bisects to adjacent integers, each candidate evaluated with its own G replications. A planning call therefore fits the analysis model several thousand times at the default G, and even at the smallest admissible G the search runs for several seconds, so the example below evaluates a stated specified_N, which is the cheap half of the method. A planning call is the same call with specified_N left out: supplying assurance plans against the joint event, and leaving assurance out as well plans against the expected widths; either way the first row of the result is necessary_N rather than specified_N. The vignette vignette("composite_sem_planning", package = "DMAR") works through both planning calls for a mediation model and a latent growth curve model, with reference values computed at G = 10000.

Each reported proportion carries a simulation standard error of about $\sqrt{p(1-p)/G}$, and the necessary sample size inherits that uncertainty; raising G narrows it, and reporting the seed makes a plan reproducible.

Value

A data.frame (a dmar_tbl) with term and value columns: the necessary_N (or supplied specified_N), the composite_assurance (the proportion of replications in which every interval was simultaneously within its desired width, reported under both criteria), then for each parameter its mean_width_<label>, its marginal pct_ci_less_width_<label> proportion, its width_<label>, and its purported true_<label> value under the analysis model, followed by conf_level, the requested replications, the converged_replications the summary is based on, and, when supplied, the assurance.

Note

A replication whose fit does not converge, or converges without a usable standard error for some parameter of interest, is discarded and fresh data are drawn, up to 20 * G attempts per evaluated sample size; the reported summaries condition on convergence. A replication whose likelihood has no maximum (an error variance running off below zero while the likelihood is flat or still rising; see fa_bifactor), which **lavaan** reports as converged where its optimizer stopped, is discarded the same way, and an analysis model with no maximum at the population moments stops the function with an error of class dmar_no_maximum that prints the **lavaan** code of that fit. When fewer than G replications converge within the cap, a single warning is issued and the summary is based on the converged replications (their count is the converged_replications row).

Because the planner itself is a Monte Carlo study, it has no separate _sensitivity sibling; to study misspecification of the population values, rerun the planner with the alternative true_cov_matrix or true_model values under consideration and compare the plans.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2011). Accuracy in parameter estimation for targeted effects in structural equation modeling: Sample size planning for narrow confidence intervals. *Psychological Methods, 16*(2), 127–148. doi:10.1037/a0021764

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Muthén, L. K., & Muthén, B. O. (2002). How to use a Monte Carlo study to decide on sample size and determine power. *Structural Equation Modeling, 9*(4), 599–620. doi:10.1207/S15328007SEM0904_8

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

See Also

`ss_power_composite_sem` for the same set of parameters planned for joint statistical significance; `cov_sem` for deriving `true_cov_matrix` from a fully fixed population model; `ss_aipe_sem_path` and `ss_aipe_sem_path_sensitivity` for a single targeted path.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`, `ss_aipe_equivalence_r_sensit`
`ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`, `ss_aipe_icc()`, `ss_aipe_icc_sensitivit`
`ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivity()`, `ss_aipe_mean()`,
`ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared()`,
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r()`, `ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`,
`ss_aipe_two_phase_cost()`, `ss_aipe_two_phase_sensitivity()`

Examples

```
# A mediation model whose research questions concern the magnitudes of
# both individual paths and the indirect effect. The population model
# fixes every parameter to its purported population value.
true_model <- "
  f1 =~ 1*y1 + 0.8*y2 + 0.8*y3
  f2 =~ 1*y4 + 0.8*y5 + 0.8*y6
  f3 =~ 1*y7 + 0.8*y8 + 0.8*y9
  f2 ~ 0.4*f1
  f3 ~ 0.5*f2 + 0.2*f1
  f1 ~~ 1*f1
  f2 ~~ 0.84*f2
  f3 ~~ 0.7*f3
  y1 ~~ 0.5*y1; y2 ~~ 0.5*y2; y3 ~~ 0.5*y3
  y4 ~~ 0.5*y4; y5 ~~ 0.5*y5; y6 ~~ 0.5*y6
  y7 ~~ 0.5*y7; y8 ~~ 0.5*y8; y9 ~~ 0.5*y9
"
```

```
# The analysis model labels the two paths and defines the indirect
# effect; all three make up the set of interest.
analysis_model <- "
  f1 =~ y1 + y2 + y3
  f2 =~ y4 + y5 + y6
  f3 =~ y7 + y8 + y9
  f2 ~ a*f1
  f3 ~ b*f2 + cp*f1
  ab := a*b
"

# Realized interval widths at N = 200, with the indirect effect held to a
# narrower interval than the paths through a named vector of widths. Each
# interval lands within its desired width in most of the replications, yet
# all three do so together in far fewer of them: that joint proportion,
# reported as composite_assurance, is what a design of this kind has to be
# planned against. G = 10 keeps the example quick; a reported plan
# deserves the default G = 1000 or more.
ss_aipe_composite_sem(model = analysis_model, true_model = true_model,
                      parameters = c("a", "b", "ab"),
                      width = c(a = 0.35, b = 0.40, ab = 0.25),
                      specified_N = 200, G = 10, seed = 113)
```

`ss_aipe_crd` *Sample Sizes for a Cluster Randomized Design: Accuracy in Parameter Estimation for the Difference Between Condition Means*

Description

Finds the number of clusters, the number of individuals per cluster, or both, for a two-condition cluster randomized design (CRD), so that the confidence interval for the difference between the condition means has a desired width or is as narrow as a budget allows. Four of the functions fix one of the two sizes and take a desired width or a budget; the two `both` functions take a budget and return the sizes that give the narrowest interval, or take a desired width and return the least costly sizes.

Usage

```
ss_aipe_crd_n_clusters_fixed_width(
  width,
  cluster_size,
  pr_treat,
  true_var_between = NULL,
  true_var_within = NULL,
  true_total_var = NULL,
  true_icc = NULL,
  true_R2_between = 0,
  true_R2_within = 0,
```

```
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  cluster_cost = NULL,
  individual_cost = NULL,
  diff_size = NULL
)

ss_aipe_crd_n_individuals_fixed_width(
  width,
  n_clusters,
  pr_treat,
  true_var_between = NULL,
  true_var_within = NULL,
  true_total_var = NULL,
  true_icc = NULL,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  cluster_cost = NULL,
  individual_cost = NULL,
  diff_size = NULL
)

ss_aipe_crd_n_clusters_fixed_budget(
  budget,
  cluster_size,
  cluster_cost = 0,
  individual_cost = 1,
  pr_treat = NULL,
  true_var_between = NULL,
  true_var_within = NULL,
  true_total_var = NULL,
  true_icc = NULL,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  diff_size = NULL
)

ss_aipe_crd_n_individuals_fixed_budget(
  budget,
  n_clusters,
  cluster_cost = 0,
```

```
  individual_cost = 1,
  pr_treat = NULL,
  true_var_between = NULL,
  true_var_within = NULL,
  true_total_var = NULL,
  true_icc = NULL,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  diff_size = NULL
)

ss_aipe_crd_both_fixed_budget(
  budget,
  cluster_cost = 0,
  individual_cost = 1,
  pr_treat,
  true_var_between = NULL,
  true_var_within = NULL,
  true_total_var = NULL,
  true_icc = NULL,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  diff_size = NULL
)

ss_aipe_crd_both_fixed_width(
  width,
  cluster_cost = 0,
  individual_cost = 1,
  pr_treat,
  true_var_between = NULL,
  true_var_within = NULL,
  true_total_var = NULL,
  true_icc = NULL,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  diff_size = NULL
)
```

Arguments

width	The desired width of the confidence interval of the unstandardized means difference
cluster_size	The number of individuals in each cluster: a single whole number of at least 2. The searches over the cluster size start at 2, and with one individual in a cluster the variance between clusters cannot be told from the variance within them
pr_treat	The proportion of treatment clusters: a single number strictly between 0 and 1
true_var_between	The population residual variance in the between level before accounting for the covariate (a planning value), 0 or more
true_var_within	The population residual variance in the within level before accounting for the covariate (a planning value), 0 or more
true_total_var	The population total residual variance before accounting for the covariate (a planning value), positive. Any two of true_var_between, true_var_within, true_total_var, and true_icc determine the other two; a third or fourth must agree with them (see the section on refused calls)
true_icc	The population intraclass correlation of the dependent variable (a planning value), the share of true_total_var that lies between clusters, in [0, 1]
true_R2_between	The population proportion of the between-level variance that the cluster-level covariates explain (a planning value in [0, 1); default 0). It must be 0 when n_covariates = 0, since there is then no covariate at the cluster level to explain it
true_R2_within	The population proportion of the within-level variance that the covariates explain (a planning value in [0, 1); default 0, no covariate)
n_covariates	The number of covariates at the cluster (between) level, not counting the treatment indicator: a single whole number of 0 or more. The degrees of freedom of the interval are n_clusters - 2 - n_covariates. The default, 0, is a model with the treatment indicator alone. It is not p, the total number of predictors, which here is p = n_covariates + 1.
assurance	The degree of assurance, which is the value with which confidence can be placed that describes the likelihood of obtaining a confidence interval less than the value specified (e.g., .80, .90, .95)
conf_level	The desired level of confidence for the confidence interval
cluster_cost	The cost of adding one cluster, whatever the number of individuals in it: a single finite number of 0 or more. Where the costs choose the design (the two budget functions that fix a size and the two both functions), cluster_cost and individual_cost cannot both be 0
individual_cost	The cost of adding one individual to a cluster: a single finite number of 0 or more, and positive where the costs choose the cluster size (ss_aipe_crd_n_individuals_fixed_budget and the two both functions), since otherwise clusters of every size cost the same

`diff_size` Difference cluster size specification. The differences in cluster sizes can be specified in two ways, and the specified vector is recycled across the clusters. First, users may specify differences as integers, which can be negative or positive; the resulting cluster sizes add the specified values to the estimated cluster size. For example, if the cluster size is 25, the number of clusters is 10, and `diff_size = c(-1, 0, 1)`, the cluster sizes will be 24, 25, 26, 24, 25, 26, 24, 25, 26, and 24. Second, users may specify multipliers of the cluster size as positive decimals; at least one value must be non-integer, which is what selects the multiplicative form. The resulting cluster sizes multiply the estimated cluster size by the specified values and round to the nearest integer. For example, if the cluster size is 25, the number of clusters is 10, and `diff_size = c(0.8, 1, 1.2)`, the cluster sizes will be 20, 25, 30, 20, 25, 30, 20, 25, 30, and 20. In either form a resulting cluster size below 1 is set to 1. If `NULL`, the cluster size is equal across clusters. Anything other than `NULL` or a vector of one or more finite numbers is refused

`n_clusters` The number of clusters: a single whole number of at least `3 + n_covariates`, since the interval has `n_clusters - 2 - n_covariates` degrees of freedom. With `pr_treat` it must put a cluster in each condition (`round(n_clusters * pr_treat)` of them are treated)

`budget` The amount available for the study, in the units of `cluster_cost` and `individual_cost`: a single finite number of 0 or more. A budget that does not buy the smallest design the function considers is refused (see the section on refused calls)

Details

Here are the functions' descriptions:

`ss_aipe_crd_n_clusters_fixed_width` Find the number of clusters given a specified width of the confidence interval and the cluster size

`ss_aipe_crd_n_individuals_fixed_width` Find the cluster size given a specified width of the confidence interval and the number of clusters

`ss_aipe_crd_n_clusters_fixed_budget` Find the number of clusters given a budget and the cluster size

`ss_aipe_crd_n_individuals_fixed_budget` Find the cluster size given a budget and the number of clusters

`ss_aipe_crd_both_fixed_budget` Find the sample size combinations (the number of clusters and that cluster size) providing the narrowest confidence interval given the fixed budget

`ss_aipe_crd_both_fixed_width` Find the sample size combinations (the number of clusters and that cluster size) providing the lowest cost given the specified width of the confidence interval

The searches over the number of clusters consider at least `ceiling(2 / min(pr_treat, 1 - pr_treat)) + n_covariates` clusters, which puts at least two clusters in each condition after rounding (the rule of **MBESS**; Kelley, 2007a, 2007b) and gives the interval at least two degrees of freedom: 4 clusters at `pr_treat = 0.5`, 20 at 0.1, and 200 at 0.01. From there `ss_aipe_crd_n_clusters_fixed_width` returns the smallest number of clusters whose width is no larger than `width` (a width equal to the target meets it, so asking for the width a plan reports returns that plan), and the two `both` functions compare numbers of clusters from there with cluster sizes of 2 or more. The searches consider at most 1,000,000 clusters, the bound the power planners use for a sample size: the width falls toward

0 as clusters are added, so every positive target is met by some number, but a target too narrow for any study would otherwise be searched for without end. The widths at 100, 200, ..., 1000 clusters bracket the target, and the bracket is searched one cluster at a time, which gives the smallest number when the width falls as clusters are added, as it does with equal cluster sizes and no `assurance`. At an `assurance` below .5 the width need not fall as the number of clusters grows, so the search may return more than the smallest number that meets the target; the same can happen with unequal cluster sizes (`diff_size`), whose width depends on how the sizes fall between the conditions. The cluster size searches consider 2 to 100,000 individuals per cluster.

Value

The `ss_aipe_crd_n_clusters_fixed_width` and `ss_aipe_crd_n_clusters_fixed_budget` functions provide the number of clusters. The `ss_aipe_crd_n_individuals_fixed_width` and `ss_aipe_crd_n_individuals_` functions provide the cluster size. The `ss_aipe_crd_both_fixed_budget` and `ss_aipe_crd_both_fixed_width` provide the number of clusters and the cluster size, respectively.

Each returns a table with the columns `term` and `value`. The rows, where they apply, are `necessary_n_clusters` (the number of clusters, where it is solved for), `necessary_cluster_size` (the number of individuals in each cluster, where it is solved for; with `diff_size` the cluster sizes and their frequencies are printed ahead of the table instead), `width` (the desired width, in the functions that plan from one), the width of the interval the design gives, and `budget` (the cost of the design, when the two costs are supplied). Without `assurance` the width row is `expected_width`, the expected width of the interval. With `assurance` it is `assured_width`, the width that the interval does not exceed with the requested probability, followed by the row `assurance` that echoes that probability.

A size that was given is an argument of the call and is not repeated in the table: only a size the function solved for has a row, and its name begins with `necessary_`. `tidy` and `glance` report the planned size as `estimate` (the number of clusters where it was solved for, otherwise the cluster size) and the desired width as `width`, which is `NA` for a plan made from a budget; `glance()` carries `expected_width` or `assured_width` as a column of that name.

Refused calls

Each refusal below is an error of class `dmar_argument_invalid` that names the argument or arguments as `arg` and, where the cause needs a name, gives it as `reason`.

- `n_covariates` must be a whole number of 0 or more, `cluster_size` a whole number of at least 2, and `n_clusters` a whole number of at least `3 + n_covariates`; `pr_treat` must be strictly between 0 and 1, and `true_R2_between` and `true_R2_within` in [0, 1). A nonzero `true_R2_between` with `n_covariates = 0` is refused with `reason "R2_between_without_covariates"`; `true_R2_within` may be nonzero without them. A number of clusters that, at `pr_treat`, leaves a condition without a cluster is refused with `reason "arm_without_clusters"`. `diff_size` must be `NULL` or a vector of one or more finite numbers.
- Each cost, and the budget, must be a single finite number of 0 or more. Costs that are both 0 are refused where the costs choose the design (`reason "costs_all_zero"`), and `individual_cost = 0` where they choose the cluster size (`reason "individual_cost_zero"`): a search over designs that all cost the same has no end. In the two functions that plan from a width and a fixed size the costs only price the design, and costs of 0 give a budget of 0.
- A budget that does not buy the smallest design the function can return is refused with `reason "budget_infeasible"` and the cost of that design as `min_cost`: in `ss_aipe_crd_n_clusters_fixed_budget`

4 clusters of cluster_size, or, with pr_treat, the fewest clusters that put a cluster in each condition and leave a degree of freedom (6 at pr_treat = 0.1); in ss_aipe_crd_n_individuals_fixed_budget n_clusters clusters of 2; and in ss_aipe_crd_both_fixed_budget the fewest clusters its search considers, given above, each of 2. When ss_aipe_crd_n_clusters_fixed_budget is given pr_treat and the budget buys 4 clusters or more but a design with no finite width, it is refused with reason "arm_without_clusters" (a condition without a cluster) or "too_few_clusters" (no degrees of freedom), with the budget that buys the fewest clusters that would do.

- ss_aipe_crd_n_individuals_fixed_width refuses a width that no cluster size up to 100,000 reaches with n_clusters clusters, with reason "width_not_reachable"; it gives as width_limit the width the interval falls toward as the clusters grow, which the variance between clusters sets, and says whether a larger cluster size would reach the target. ss_aipe_crd_n_clusters_fixed_width and ss_aipe_crd_both_fixed_width refuse a width that no design of up to 1,000,000 clusters meets (with clusters of up to 100,000 individuals in the latter), with reason "n_clusters_beyond_reach" and the width there as width_at_max.
- The variances are needed where a width is computed: two of true_var_between, true_var_within, true_total_var, and true_icc, any two (reason "underidentified_variances" otherwise, as for a between-cluster variance of 0 with an intraclass correlation of 0). Each must be NULL or a single finite number (reason "variance_input_invalid"). A variance must be 0 or more, the total variance positive (whether it is given or is the sum of the two variances, so two variances of 0 are refused), and the intraclass correlation in [0, 1] (reason "variance_out_of_range"). A third or fourth value must agree with the two that determine the model to within about 1e-14 of its scale (reason "inconsistent_variances", with the value they imply in the message), so a rounded value is refused; give two and leave the others NULL. With diff_size the variance within clusters must be positive (reason "within_variance_zero"). These refusals carry the four values as given in supplied.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Pornprasertmanit, S., & Schneider, W. J. (2014). Accuracy in parameter estimation in cluster randomized designs. *Psychological Methods, 19*(3), 356–379. doi:10.1037/a0037036

See Also

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Examples for each function
ss_aipe_crd_n_clusters_fixed_width(width = 0.3, cluster_size = 30,
```

```
  pr_treat = 0.5, true_var_between = 0.25, true_var_within = 0.75)

ss_aipe_crd_n_individuals_fixed_width(width = 0.3, n_clusters = 250,
  pr_treat = 0.5, true_var_between = 0.25, true_var_within = 0.75)

ss_aipe_crd_n_clusters_fixed_budget(budget = 10000, cluster_size = 20,
  cluster_cost = 20, individual_cost = 1)

ss_aipe_crd_n_individuals_fixed_budget(budget = 10000, n_clusters = 30,
  cluster_cost = 20, individual_cost = 1,
  pr_treat = 0.5, true_var_between = 0.05, true_var_within = 0.95, assurance = 0.8)

ss_aipe_crd_both_fixed_budget(budget = 10000, cluster_cost = 30, individual_cost = 1,
  pr_treat = 0.5, true_var_between = 0.25, true_var_within = 0.75)

ss_aipe_crd_both_fixed_width(width = 0.3, cluster_cost = 0, individual_cost = 1,
  pr_treat = 0.5, true_var_between = 0.25, true_var_within = 0.75)

# Examples for different cluster size
set.seed(113)
ss_aipe_crd_n_clusters_fixed_width(width = 0.3, cluster_size = 30,
  pr_treat = 0.5, true_var_between = 0.25, true_var_within = 0.75,
  diff_size = c(-2, 1, 0, 2, -1, 3, -3, 0, 0))

# Examples for different number of clusters
ss_aipe_crd_n_individuals_fixed_width(width = 0.3, n_clusters = 250,
  pr_treat = 0.5, true_var_between = 0.25, true_var_within = 0.75,
  diff_size = c(0.6, 1.2, 0.8, 1.4, 1, 1, 1.1, 0.9))
```

`ss_aipe_crd_es` *Find Target Sample Sizes for the Accuracy in Standardized Conditions Means Estimation in CRD*

Description

Finds the number of clusters, the number of individuals per cluster, or both, for a two-condition cluster randomized design (CRD), so that the confidence interval for the standardized difference between the condition means has a desired width or is as narrow as a budget allows. Four of the functions fix one of the two sizes and take a desired width or a budget; the two `both` functions take a budget and return the sizes that give the narrowest interval, or take a desired width and return the least costly sizes. Each returns a table with the size or sizes found, the cost, and the expected width of the interval where it was computed.

The interval is the likelihood-based confidence interval (Cheung, 2009) computed by the `OpenMx` package (Boker et al., 2011). See Pornprasertmanit and Schneider (2014) for the method.

Usage

```
ss_aipe_crd_es_n_clusters_fixed_width(
```

```
  width,
  cluster_size,
  true_es,
  es_type = 1,
  true_icc,
  pr_treat,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  G = 1000,
  true_icc_covariate = NULL,
  seed = NULL,
  ncpus = 1,
  cluster_cost = NULL,
  individual_cost = NULL,
  diff_size = NULL
)

ss_aipe_crd_es_n_individuals_fixed_width(
  width,
  n_clusters,
  true_es,
  es_type = 1,
  true_icc,
  pr_treat,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  G = 1000,
  true_icc_covariate = NULL,
  seed = NULL,
  ncpus = 1,
  cluster_cost = NULL,
  individual_cost = NULL,
  diff_size = NULL
)

ss_aipe_crd_es_n_clusters_fixed_budget(
  budget,
  cluster_size,
  cluster_cost,
  individual_cost,
  G = NULL,
  pr_treat = NULL,
```

```
  true_icc = NULL,
  true_es = NULL,
  es_type = 1,
  n_covariates = 0,
  true_icc_covariate = NULL,
  true_R2_within = 0,
  true_R2_between = 0,
  assurance = NULL,
  seed = NULL,
  ncpus = 1,
  conf_level = 0.95,
  diff_size = NULL
)

ss_aipe_crd_es_n_individuals_fixed_budget(
  budget,
  n_clusters,
  cluster_cost,
  individual_cost,
  G = NULL,
  pr_treat = NULL,
  true_icc = NULL,
  true_es = NULL,
  es_type = 1,
  n_covariates = 0,
  true_icc_covariate = NULL,
  true_R2_within = 0,
  true_R2_between = 0,
  assurance = NULL,
  seed = NULL,
  ncpus = 1,
  conf_level = 0.95,
  diff_size = NULL
)

ss_aipe_crd_es_both_fixed_budget(
  budget,
  cluster_cost = 0,
  individual_cost = 1,
  true_es,
  es_type = 1,
  true_icc,
  pr_treat,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
```

```
  G = 1000,
  true_icc_covariate = NULL,
  seed = NULL,
  ncpus = 1,
  diff_size = NULL
)

ss_aipe_crd_es_both_fixed_width(
  width,
  cluster_cost = 0,
  individual_cost = 1,
  true_es,
  es_type = 1,
  true_icc,
  pr_treat,
  true_R2_between = 0,
  true_R2_within = 0,
  n_covariates = 0,
  assurance = NULL,
  conf_level = 0.95,
  G = 1000,
  true_icc_covariate = NULL,
  seed = NULL,
  ncpus = 1,
  diff_size = NULL
)
```

Arguments

`width`	The desired width of the confidence interval of the standardized difference between the condition means
`cluster_size`	The number of individuals in each cluster: a single whole number of at least 2
`true_es`	The population effect size the interval is planned for (on the scale chosen by `es_type`): a single finite number, such as `0.5`. It is required wherever a width is computed, because the width is read from data simulated at that value: in the four planners that take or return a width, and in the two planners that fix a size and a budget when `G` is supplied. A call that asks for a width without it is refused with an error of class `dmar_argument_invalid` that names what is missing
`es_type`	The type of effect size. There are only three possible options: 0 = the effect size using total standard deviation, 1 = the effect size using the individual-level standard deviation (level 1), 2 = the effect size using the cluster-level standard deviation (level 2)
`true_icc`	The population intraclass correlation of the dependent variable (a planning value): a single number in [0, 1), and above 0 with `es_type = 2`. At 1 no variance lies within clusters, and the model fitted to each simulated data set has no within-cluster variance to estimate; with `es_type = 2` the effect is in units of the between-cluster standard deviation, which an intraclass correlation of 0 makes 0

pr_treat The proportion of treatment clusters: a single number strictly between 0 and 1

true_R2_between
The population proportion of the between-level variance that the covariate explains (a planning value in [0, 1), used when n_covariates = 1; the default is 0 in all six functions). It must be 0 when n_covariates = 0

true_R2_within The population proportion of the within-level variance that the covariate explains (a planning value in [0, 1), used when n_covariates = 1; the default is 0 in all six functions)

n_covariates The number of covariates at the cluster (between) level, not counting the treatment indicator: 0 (the default, the treatment indicator alone) or 1. A larger value is refused. It is not p, the total number of predictors, which here is p = n_covariates + 1.

assurance The assurance: the probability, over repeated studies of the design, that the study gives a confidence interval and that the interval is no wider than the width planned (e.g., .80, .90, .95). With NULL (the default) the plan is for the expected width instead. A simulated data set whose interval cannot be formed counts as one whose interval is wider than any width (see the section on simulated data sets without a usable interval)

conf_level The desired level of confidence for the confidence interval

G The number of replications used in the a priori Monte Carlo simulation: a single whole number from 1 to 999,999 (default 1000 in the four planners that need a simulation). Each simulated data set takes its own seed, drawn without replacement from the whole numbers 1 to 999,999, so no more can be simulated. Anything else (a fractional number, 0, a negative number, a number above 999,999, NA, a character string, or more than one value) is refused with an error of class dmar_count_invalid before any simulation starts. In the two planners that fix a size and a budget the default is NULL, which skips the simulation: the size the budget buys is returned without an expected width. Supplying G there asks for the expected width, which also needs pr_treat, true_icc, and true_es; a call with G and without any of them is refused, with a message that names the ones missing, in place of a table without the width that was asked for.

true_icc_covariate
The population intraclass correlation of the covariate (a planning value, used when n_covariates = 1; when it is NULL the value of true_icc is used). If true_icc_covariate = 0, the covariate varies within clusters only. If true_icc_covariate = 1, the covariate varies between clusters only

seed An optional integer seed for the a priori Monte Carlo simulation. The default NULL draws the seeds of the simulated data sets from the current random number stream, which advances that stream, so repeated calls reflect the sampling variability of the simulation. An integer makes the result reproducible: the seed is set for the call only, and the caller's random number state is restored when the function returns

ncpus The number of processes over which the G simulated data sets are spread, the name this count has in the bootstrap functions of the package (reliability_omega and its relatives). The default 1 runs them in this R session; a larger whole number uses forked workers or, on Windows, which cannot fork, a socket cluster.

Each simulated data set has its own seed, drawn before any is run, so a given `seed` returns the same result whatever `ncpus` is. A socket cluster is stopped when the function exits, whether it returns, stops with an error, or is interrupted.

`cluster_cost` The cost of adding one cluster, whatever the number of individuals in it: a single finite number of 0 or more. Where the costs choose the design (the two budget planners that fix a size and the two `both` planners), `cluster_cost` and `individual_cost` cannot both be 0

`individual_cost`
The cost of adding one individual to a cluster: a single finite number of 0 or more, and positive where the costs choose the cluster size (`ss_aipe_crd_es_n_individuals_fixed_buc` and the two `both` planners)

`diff_size` Difference cluster size specification. The differences in cluster sizes can be specified in two ways, and the specified vector is recycled across the clusters. First, users may specify differences as integers, which can be negative or positive; the resulting cluster sizes add the specified values to the estimated cluster size. For example, if the cluster size is 25, the number of clusters is 10, and `diff_size = c(-1, 0, 1)`, the cluster sizes will be 24, 25, 26, 24, 25, 26, 24, 25, 26, and 24. Second, users may specify multipliers of the cluster size as positive decimals; at least one value must be non-integer, which is what selects the multiplicative form. The resulting cluster sizes multiply the estimated cluster size by the specified values and round to the nearest integer. For example, if the cluster size is 25, the number of clusters is 10, and `diff_size = c(0.8, 1, 1.2)`, the cluster sizes will be 20, 25, 30, 20, 25, 30, 20, 25, 30, and 20. In either form a resulting cluster size below 1 is set to 1. If `NULL`, the cluster size is equal across clusters

`n_clusters` The number of clusters: a single whole number of at least `3 + n_covariates`, which with `pr_treat` puts a cluster in each condition

`budget` The amount available for the study, in the units of `cluster_cost` and `individual_cost`: a single finite number of 0 or more

Details

Here are the functions' descriptions:

`ss_aipe_crd_es_n_clusters_fixed_width` Find the number of clusters given a specified width of the confidence interval and the cluster size

`ss_aipe_crd_es_n_individuals_fixed_width` Find the cluster size given a specified width of the confidence interval and the number of clusters

`ss_aipe_crd_es_n_clusters_fixed_budget` Find the number of clusters given a budget and the cluster size

`ss_aipe_crd_es_n_individuals_fixed_budget` Find the cluster size given a budget and the number of clusters

`ss_aipe_crd_es_both_fixed_budget` Find the sample size combinations (the number of clusters and that cluster size) providing the narrowest confidence interval given the fixed budget

`ss_aipe_crd_es_both_fixed_width` Find the sample size combinations (the number of clusters and that cluster size) providing the lowest cost given the specified width of the confidence interval

Every answer that targets a width, and the expected width a budget planner reports when G and the population values are supplied, rests on an a priori Monte Carlo simulation: a candidate design is evaluated by generating G data sets and reading the likelihood-based confidence interval on the standardized effect size from OpenMx. The planners whose answer needs one such evaluation, or a handful, run in about a second at a small G and are the ones shown in the examples. The two that search over the number of clusters, ss_aipe_crd_es_n_clusters_fixed_width at a fixed cluster size and ss_aipe_crd_es_both_fixed_width, which repeats that search across candidate cluster sizes and keeps the least costly combination that reaches the width, evaluate many candidates in turn and run for minutes at the default G = 1000, so they are not among the examples. A call such as ss_aipe_crd_es_n_clusters_fixed_width(width = 0.3, cluster_size = 20, true_es = 0.5,es_type = 1, true_icc = 0.25, pr_treat = 0.5, G = 1000, seed = 113) returns the number of clusters of 20 individuals that brings the expected width of the interval on the individual-level standardized effect size to 0.3, and ss_aipe_crd_es_both_fixed_width(width = 0.5, cluster_cost = 5, individual_cost = 1, true_es = 0.5,es_type = 1, true_icc = 0.25, pr_treat = 0.5, G = 1000, seed = 113) returns the least costly pairing of clusters and cluster size at those costs that reaches a width of 0.5. Both accept diff_size for unequal cluster sizes, in which case the simulated data sets carry the unequal sizes and the confidence interval comes from a multiple-group model.

The width of a simulated interval is its upper limit minus its lower limit.

Value

The ss_aipe_crd_es_n_clusters_fixed_width and ss_aipe_crd_es_n_clusters_fixed_budget functions provide the number of clusters. The ss_aipe_crd_es_n_individuals_fixed_width and ss_aipe_crd_es_n_individuals_fixed_budget functions provide the cluster size. The ss_aipe_crd_es_both_fixe and ss_aipe_crd_es_both_fixed_width provide the number of clusters and the cluster size, respectively.

Each returns a table with the columns term and value. The rows, where they apply, are necessary_n_clusters (the number of clusters, where it is solved for), necessary_cluster_size (the number of individuals in each cluster, where it is solved for; with diff_size the cluster sizes and their frequencies are printed ahead of the table instead), width (the desired width, in the functions that plan from one), the width of the interval for the standardized effect size chosen by es_type, and budget (the cost of the design, when the two costs are supplied). Without assurance the width row is expected_width, the mean width of the interval over the G simulated data sets. With assurance it is assured_width, the width that the interval does not exceed with the requested probability, followed by the row assurance that echoes that probability.

A size that was given is an argument of the call and is not repeated in the table: only a size the function solved for has a row, and its name begins with necessary_. tidy and glance report the planned size as estimate (the number of clusters where it was solved for, otherwise the cluster size) and the desired width as width, which is NA for a plan made from a budget; glance() carries expected_width or assured_width as a column of that name.

A table whose width was simulated carries the record of the simulation of the design reported in its attribute simulation, a list read with attr(x, "simulation"): attempted (G), returned (the data sets with a usable interval), failed (those without one), denominator (the number of data sets the width row is taken over: returned for expected_width, attempted for assured_width), failure_reasons (the data sets without a usable interval by reason, a named integer vector: "fit_error", "optimizer_status_<code>", "limit_not_found", "limit_status_<code>", "limit_not_finite",

"limits_not_ordered"), failure_messages (for a fit that stopped with an error, the first error message, named by reason), width (the width reported), width_mc_ci (the Monte Carlo interval of that width at conf_level, which describes the error of the simulation only: the mean plus and minus the normal quantile times its standard error, or for the assured width the interval of two order statistics of the G widths), n_clusters and cluster_size (the design), and search, the totals over every design the call simulated (n_designs, attempted, returned, failed, failure_reasons, failure_messages) and the warnings the fits gave (warnings, one element per kind, with its class, first message, all its messages, and n, the number of data sets that gave it). The line "Replications" beneath the printed table gives G, and the three counts when a data set gave no usable interval.

Simulated data sets without a usable interval

Each simulated data set is fitted with **OpenMx**, and its interval is used only when the fit and the search for each limit succeeded: the optimizer status code of the fit and of each limit's search is 0 or 1 (the codes that **OpenMx**'s as.statusCode() describes as "OK" and "OK/green", optimization succeeded; with a code such as 5, a Hessian that does not appear to be convex, or 6, a solution that does not meet the first-order optimality conditions, the fit may not be at the maximum the interval is measured from, and that help page advises against trusting it), **OpenMx**'s diagnostic for each limit is "success", and the two limits are finite and bracket the estimate. A data set that fails any of these, or whose fit stops with an error, gives no usable interval. How often that happens depends on the design. With **OpenMx** 2.22.11 and a standardized effect of 0.5 (the intraclass correlation in parentheses), the share of 1000 simulated data sets without a usable interval, with its Monte Carlo standard error, was 0 with 20 clusters of 10 (0.2) and es_type = 1, 0.002 (0.0014) with six clusters of 5 (0.1), 0 with ten clusters of 2 (0.01), 0 with eight clusters of 4 (0.1) and es_type = 0, 0.415 (0.016) with six clusters of two to four individuals (0.05), and 0.208 (0.013) with 20 clusters of 10 (0.2) and es_type = 2; a second run of 400 data sets gave 0.378 (0.024) and 0.237 (0.021) for the last two. These are single runs, and other versions of **OpenMx** can give other shares. With the unequal clusters 412 of the 415 failed fits ended with status 5 or 6, and in 200 further data sets each of the 86 such fits had estimated the between-cluster variance below 0. The effect of es_type = 2 is divided by the between-cluster standard deviation, which can be near 0: for some data sets the likelihood does not fall far enough on one side for a finite limit to exist, for others **OpenMx**'s search does not reach a limit that exists, and some of the intervals it does find are very wide (with 20 clusters of 10, the usable intervals of 300 data sets had a median width of 2.1, 10 of 242 were wider than 10, and 2 wider than 100), so that their mean rests on a few data sets.

Without assurance, the expected width is the mean width over the data sets that gave a usable interval, so it describes studies in which the interval could be formed. With assurance, a data set without a usable interval counts as one whose interval is wider than any width, and the assured width is the assurance quantile of the widths of all G data sets; the assurance is then the probability that a study gives an interval and that the interval is no wider than the assured width. When more than about 1 - assurance of the data sets give no usable interval, no width is reached with that probability.

When some of the data sets simulated at the design reported gave no usable interval, a warning of class dmar_replications_without_interval (with scope "design") gives their number, the reasons, and the number of data sets the width is taken over. When every data set at the design reported gave an interval but some at other designs the search simulated did not, a warning of the same class (with scope "search") gives the totals of the search, since those data sets bear on where it stopped. When none gave one, or the assurance cannot be reached, the function stops with an

error of class dmar_computation_failed that carries the counts (attempted, returned, failed, failure_reasons), and no table is returned. Inside a search such a design counts as one whose width does not meet the target. For a fit that stopped with an error, the warning and the error give the first error message, which names the cause when every data set fails for one reason.

A warning that a fit gives is not discarded: each kind of warning is given once when the function exits, as a warning of class dmar_result_qualified that says how many simulated data sets gave it and carries its own class as warning_class. Warnings of one class whose messages differ only in their numbers count as one kind; the first message is shown, and all of them are in the field messages. The warning **OpenMx** gives for an optimizer status other than 0 is not passed on: a fit of status 1 is used, as **OpenMx**'s help describes it, and any other status is counted as the reason the data set has no usable interval.

The searches

The planners that search for a size start from the size the unstandardized difference between the condition means needs (the cluster size search with es_type = 1 can start lower; see below), and each size they try is a simulation of G data sets. A width meets the target when it is no larger than the target. From a start whose width meets the target, the search steps down one cluster or one individual at a time to the smallest size that still meets it. From a start whose width is above the target, the search steps up 1, 3, 7, 15, and so on, past the start until the width meets the target, and then halves the gap between the last size that did not meet it and the first that did until the two are adjacent. When the simulated width falls as the size grows, this gives the smallest size that meets the target, the size that stepping up one at a time gives, with far fewer sizes simulated. With a small G the simulated width can rise and fall from one size to the next, and then the two can stop at different sizes near the target. The same holds when designs without a finite width (see below) lie among designs with one, as can happen near the share of failed data sets an assurance allows: the search returns a size that meets the target next to one that does not, which need not be the smallest size that meets it.

The search for the number of clusters has no upper bound, since the width falls toward 0 as clusters are added wherever it is finite. A design at which no simulated data set gives a usable interval, or at which too few do for the assurance, has no finite width; after such a design the next size tried is at least twice as large, and three such designs in a row stop the search with an error of class dmar_computation_failed, with reason "no_finite_width", that gives the sizes, the counts, and the reasons.

With a fixed number of clusters the width levels off as the clusters grow, because the clusters themselves vary, so a target can be out of reach at any cluster size. For es_type = 1 with no covariate at the cluster level, the value it levels off at is known: as every cluster grows, the within-cluster standard deviation becomes known and the cluster means exact, and the likelihood-based interval for the difference between K cluster means with a common unknown variance has a width whose mean (or, with assurance, whose quantile) has a closed form. A target at or below that value is refused before anything is simulated, with an error of class dmar_argument_invalid, reason "width_not_reachable", that gives it as width_limit; a target above it is reached by some cluster size. There the search starts from the smaller of two sizes: the size the unstandardized difference needs, and the size at which the expected width (or its quantile) with the within-cluster standard deviation known falls below the target. That width is the value the width levels off at, with the variance of a cluster mean at the size in place of the between-cluster variance. The unstandardized difference uses the t quantile on K - 2 degrees of freedom, and with few clusters it needs far more: with 10 clusters and an intraclass correlation of .2 it needs 1,875 individuals per cluster for

a target of 1.46, where about 10 reach it with `true_es = 0.5` (the size the unstandardized difference needs does not depend on the effect, but the size that reaches the target grows with it). With equal cluster sizes, no covariate, and no `assurance`, the start is a lower bound on the cluster size the target needs (the second of the two sizes uses `true_R2_within` only with a covariate, since the simulated data use it only then). The likelihood-based interval contains the interval computed with the within-cluster standard deviation fixed at its maximum likelihood estimate, whose expected width is the width with the deviation known times the mean of the ratio of the deviation to its estimate, which exceeds 1; so at every smaller cluster size the expected width is above the target. With `es_type = 0` or `2`, or with a covariate at the cluster level, the search starts from the size the unstandardized difference needs, and only when that is at most 100,000 individuals per cluster. The cluster size search simulates cluster sizes up to 100 above the size the unstandardized difference needs (above its own start when that size is beyond 100,000) and never above 500, since the model has one variable per individual of a cluster and a fit takes far longer as the clusters grow; a search that would start above 500 individuals per cluster is refused before anything is simulated, with an error of class `dmar_argument_invalid`, `reason "cluster_size_beyond_reach"`, that gives the start as `start` and, as `lower_bound`, whether it is a lower bound in the sense just given; the message calls it one only then. A width still above the target there stops it with an error of class `dmar_computation_failed`, `reason "width_not_reached"`, that gives the sizes searched and the narrowest width found with its Monte Carlo interval. The message says whether a larger cluster size reaches the target (where the value the width levels off at is known) or that this was not determined, and suggests a larger `G` only when the target lies within that Monte Carlo interval.

The number of clusters the unstandardized difference needs, from which the search over the number of clusters starts, is at least `ceiling(2 / min(pr_treat, 1 - pr_treat)) + n_covariates`, which puts two clusters in each condition (see `ss_aipe_crd`), and the search steps down no further than the last design with two clusters in each condition. A start beyond 1,000,000 clusters, the most the unstandardized search considers, is refused before anything is simulated, with an error of class `dmar_argument_invalid`, `reason "n_clusters_beyond_reach"`. The planners that search over both sizes start from the plan of the unstandardized difference, which keeps the same floor, and do not simulate a number of clusters at which the budget does not buy clusters of 2.

A call simulates at most 500 designs; the two planners that search over both sizes share that bound among the searches they run, and three neighboring designs with no finite width stop `ss_aipe_crd_es_both_fixed_budget`. A call that reaches the bound stops with an error of class `dmar_computation_failed`, `reason "search_budget"`, that gives the totals. These bounds limit the computation; none of them is a statistical rule.

Refused calls

The design arguments, the costs, and the budget are checked before anything is simulated, as in `ss_aipe_crd` (see its section on refused calls): `n_covariates` a whole number of 0 or more (and here at most 1), `cluster_size` a whole number of at least 2, `n_clusters` a whole number of at least `3 + n_covariates` that puts a cluster in each condition, `pr_treat` strictly between 0 and 1, `true_R2_between` and `true_R2_within` in [0, 1), `true_R2_between` 0 when `n_covariates = 0` (`reason "R2_between_without_covariates"`), `diff_size` `NULL` or a vector of one or more finite numbers, and each cost and the budget a single finite number of 0 or more. Wherever a width is simulated (always in the four planners that take a width or search over both sizes, and with `G` in the other two), `es_type` must be 0, 1, or 2, `true_icc` as described under its argument, and, with a covariate, `true_icc_covariate` in [0, 1]. Costs that are both 0 are refused where they choose the design (`reason "costs_all_zero"`), and `individual_cost = 0` where they

choose the cluster size (`reason "individual_cost_zero"`). A budget that does not buy the smallest design a planner can return is refused with `reason "budget_infeasible"` and the cost of that design as `min_cost` (with `G`, the smallest design with a cluster in each condition and a degree of freedom), and a budget whose design leaves a condition without a cluster, when its width is asked for, with `reason "arm_without_clusters"`. A fixed `n_clusters` that leaves a condition without a cluster is refused with the same `reason` where a width is simulated: always in `ss_aipe_crd_es_n_individuals_fixed_width`, and in `ss_aipe_crd_es_n_individuals_fixed_budget` only with `G`, since without it `pr_treat` is not used. Each of these has class `dmar_argument_invalid`. `assurance` must be `NULL` or a single number strictly between 0 and 1 in all six planners (class `dmar_assurance_invalid`), and the three planners that take a width refuse one that is not a single positive number. `ss_aipe_crd_es_n_individuals_fixed_width` refuses, with `reason "cluster_size_beyond_reach"` and `start = Inf`, a search whose start, the cluster size the unstandardized difference needs, is beyond 100,000.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Boker, S. M., Neale, M. C., Maes, H. H., Wilde, M., Spiegel, M., Brick, T. R., ... Fox, J. (2011). OpenMx: An open source extended structural equation modeling framework. *Psychometrika, 76*(2), 306–317. doi:10.1007/s11336-010-9200-6

Cheung, M. W.-L. (2009). Constructing approximate confidence intervals for parameters with structural equation models. *Structural Equation Modeling, 16*(2), 267–294. doi:10.1080/10705510902751291

Pornprasertmanit, S., & Schneider, W. J. (2010). *Efficient sample size for power and desired accuracy in Cohen's d estimation in two-group cluster randomized design* (Master Thesis). Illinois State University, Normal, IL.

Pornprasertmanit, S., & Schneider, W. J. (2014). Accuracy in parameter estimation in cluster randomized designs. *Psychological Methods, 19*(3), 356–379. doi:10.1037/a0037036

See Also

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Two of these planners answer a question the budget alone settles. Given
# what it costs to open a cluster and what it costs to collect one more
# individual, the first reports how many clusters a budget buys at a fixed
# cluster size and the second reports how large each cluster can be at a
# fixed number of clusters. Clusters cost nothing to open here and each
# individual costs 1, so the budget buys 1000 individuals and the only
# question is how to arrange them.
ss_aipe_crd_es_n_clusters_fixed_budget(budget = 1000, cluster_size = 20,
  cluster_cost = 0, individual_cost = 1)

ss_aipe_crd_es_n_individuals_fixed_budget(budget = 1000, n_clusters = 200,
```

```
  cluster_cost = 0, individual_cost = 1)

# The interval width these planners can report, and every answer that
# targets a width, rests on an a priori Monte Carlo simulation: a candidate
# design is evaluated by generating G data sets and reading the
# likelihood-based confidence interval on the standardized effect size from
# OpenMx. The calls below use G = 2 so that the page runs quickly; a
# reported plan deserves the default G = 1000. Each call describes a
# population standardized effect size of 0.5, with es_type = 1 putting that
# effect size in individual-level standard deviation units and a quarter of
# the outcome variance lying between clusters.

# Supplying G and the population values to a budget planner adds the
# expected width of the interval the affordable design buys.
ss_aipe_crd_es_n_clusters_fixed_budget(budget = 1000, cluster_size = 20,
  cluster_cost = 0, individual_cost = 1, true_es = 0.5, es_type = 1, true_icc = 0.25,
  pr_treat = 0.5, G = 2, seed = 113)

# Cluster size needed for a target width, given the number of clusters.
# With 250 clusters the planner settles on the smallest cluster size it
# will consider, two individuals per cluster, and the expected width still
# comes in under the target: for a contrast between conditions that are
# assigned at the cluster level, precision is bought with clusters rather
# than with what happens inside them.
ss_aipe_crd_es_n_individuals_fixed_width(width = 0.5, n_clusters = 250,
  true_es = 0.5, es_type = 1, true_icc = 0.25, pr_treat = 0.5, G = 2,
  seed = 113)

# Once recruiting a cluster costs 5, the number of clusters and the cluster
# size trade off against each other, and this planner searches the
# combinations the budget allows for the narrowest expected interval.
ss_aipe_crd_es_both_fixed_budget(budget = 1000, cluster_cost = 5,
  individual_cost = 1, true_es = 0.5, es_type = 1, true_icc = 0.25, pr_treat = 0.5,
  G = 2, seed = 113)

# Unequal cluster sizes. Every planner accepts diff_size, which gives each
# cluster's deviation from cluster_size as an integer offset or as a
# multiplicative factor, recycled across the clusters; the planner prints
# the resulting cluster sizes and their frequencies ahead of its table.
# The budget planner shows both forms here.
ss_aipe_crd_es_n_clusters_fixed_budget(budget = 1000, cluster_size = 20,
  cluster_cost = 0, individual_cost = 1, diff_size = c(-2, 1, 0, 2, -1, 3, -3, 0, 0))

ss_aipe_crd_es_n_clusters_fixed_budget(budget = 1000, cluster_size = 20,
  cluster_cost = 0, individual_cost = 1,
  diff_size = c(0.6, 1.2, 0.8, 1.4, 1, 1, 1.1, 0.9))
```

---

ss_aipe_crossed *Sample Size Planning From the AIPE Perspective for the Mean of a Crossed Design of Units and Raters*

Description

Plans the numbers of units and raters for a fully crossed design, in which every rater scores every unit once, so that the confidence interval for the mean score is sufficiently narrow at the least cost. The units are the objects of measurement (persons, essays, recordings, or the topics of an information retrieval evaluation) and the raters are whoever or whatever produces the scores (judges, coders, assessors, or automated scoring models). When the raters are a sample from a population of raters, as the units are a sample of units, the interval for the population mean carries the variability of both samples, and no number of units can make up for too few raters. The interval planned for is the one `ci_crossed` computes from the data.

Usage

```
ss_aipe_crossed(
  true_var_unit,
  true_var_rater,
  true_var_residual,
  width,
  conf_level = 0.95,
  assurance = NULL,
  n_raters = NULL,
  unit_cost = 0,
  rater_cost = 0,
  judgment_cost = 1,
  quantile = c("t", "normal"),
  max_units = 1e+05,
  max_raters = 500
)
```

Arguments

`true_var_unit`, `true_var_rater`, `true_var_residual`
: Planning values of the three variance components: of the unit effects (σ_u^2, how much units differ in their mean score over the population of raters), of the rater effects (σ_r^2, how much raters differ in leniency), and of the residuals (σ_e^2, the unit by rater interaction together with error, which one score per cell cannot separate). The first two must be nonnegative and the third positive. The `var_unit`, `var_rater`, and `var_residual` rows of `ci_crossed` applied to a pilot array estimate them.

`width`
: Desired full width of the confidence interval for the mean.

`conf_level`
: Confidence level of the interval (default `0.95`).

`assurance`
: Optional probability, strictly between 0 and 1, that the realized interval is no wider than `width`. The default `NULL` plans for the width at the planning values. A plan with an assurance is never smaller than the plan without one (see Details), so an assurance at or below one half returns the plan without one. An assurance requires `quantile = "t"`, since it concerns the sampling variability of the estimated variance that the *t* quantile accounts for.

`n_raters` Optional fixed number of raters (a whole number of at least 2), for a study whose panel of raters is already set; only the number of units is then planned. The default `NULL` plans both.

`unit_cost`, `rater_cost`, `judgment_cost`
Costs of adding a unit, of recruiting a rater, and of one judgment (one rater's score of one unit), in any common currency. A design with n_u units and n_r raters costs $c_u n_u + c_r n_r + c_j n_u n_r$. The defaults `0`, `0`, and `1` make the cost the number of judgments, so the plan is the design that collects the fewest scores. The costs must be nonnegative and not all zero.

`quantile` `"t"` (the default) or `"normal"`: the quantile the planned interval is formed with (see Details).

`max_units`, `max_raters`
The largest numbers of units and of raters the search considers. Defaults `100000` and `500`.

Details

The model and the variance of the mean. The score of unit u by rater r is $y_{ur} = \mu + a_u + b_r + e_{ur}$, with independent unit effects, rater effects, and residuals of mean zero and variances σ_u^2, σ_r^2, and σ_e^2. Both units and raters are random, so the target is μ, the mean over the population of units and the population of raters, and the grand mean of a complete array of n_u units by n_r raters has variance

$$V = \frac{\sigma_u^2}{n_u} + \frac{\sigma_r^2}{n_r} + \frac{\sigma_e^2}{n_u n_r}.$$

In the vocabulary of generalizability theory (Brennan, 2001) the units are the objects of measurement and the raters a random facet crossed with them.

The rater floor. Whatever the number of units, V is at least σ_r^2/n_r, so with n_r raters the half-width cannot fall below $z\,\sigma_r/\sqrt{n_r}$, with z the normal quantile. A design meeting a target half-width h (half of `width`) under the normal quantile therefore needs more than $z^2\sigma_r^2/h^2$ raters; that number is the `rater_floor` row. The *t* quantile raises the requirement further. When raters vary, adding units cannot make up for a panel that is too small, and a plan with `n_raters` at or below the floor is refused with that explanation.

The interval planned for. `ci_crossed` estimates V by the ANOVA estimates of the three components, a negative estimate truncated at zero, and refers the grand mean to a *t* distribution whose degrees of freedom are Satterthwaite's (1946) for that combination of mean squares. With `quantile = "t"`, the default, the plan forms the interval the same way at the planning values: the planned half-width is $t_\nu\sqrt{V}$, where the degrees of freedom ν are those of the plug-in estimate $(MS_u + MS_r - MS_e)/(n_u n_r)$ computed from the expected mean squares. With `quantile = "normal"` the planned half-width is $z\sqrt{V}$, the half-width of an interval whose variance is known; `quantile = "normal"` reproduces planning that uses the normal quantile. Its designs are smaller, and because the analysis interval uses the *t* quantile, its realized intervals exceed the target width more often than those of a plan made with the *t* quantile. The gap is largest when the rater component is at or near zero. The normal plan then uses two raters, since raters add nothing to V, but the analysis still estimates the rater component from those two. With no rater variance, MS_r exceeds MS_e by chance in about a third of samples and the estimate is kept; the Satterthwaite degrees of freedom then fall, to one or two when the unit component is small beside the residual one, and the interval can be several times wider than the target. The *t* plan sees the few degrees of freedom two raters give at the planning

values and adds raters or units to make up for them. ss_aipe_crossed_sensitivity shows how a given plan fares.

Assurance. The plan without an assurance gives an interval no wider than the target at the planning values, and the realized interval is narrower or wider as the estimated variance falls below or above V. An assurance γ (Kelley & Maxwell, 2003) asks instead for a realized width at most `width` with probability at least γ. The plan approximates the distribution of the plug-in variance by a scaled chi square on the Satterthwaite degrees of freedom, as Satterthwaite (1946) does, and requires

$$t_\nu \sqrt{V \max(1, q_\gamma/\nu)} \le h,$$

with q_γ the γ quantile of the chi square distribution on ν degrees of freedom. Under the same approximation the plan without an assurance already meets the target with probability at least $P(\chi^2_\nu \le \nu)$, which exceeds one half because the median of a chi square lies below its mean: about .58 at $\nu = 5$, .53 at $\nu = 30$, and nearer one half as ν grows. An assurance at or below that probability asks for nothing more, and q_γ/ν is then at most 1; taking the larger of 1 and q_γ/ν keeps such an assurance from shrinking the design. The rule is that a plan with an assurance is never smaller than the plan without one: it costs at least as much, and with a fixed panel it has at least as many units. An assurance at or below one half always returns the plan without one.

The search. For every number of raters from 2 to `max_raters` (or for `n_raters` alone) the planner finds the smallest number of units that meets the target, prices each such design, and returns the cheapest; designs of equal cost are resolved toward fewer raters. With the *t* quantile the planned width is not monotone in the number of units: as units are added, V approaches its rater floor while the degrees of freedom eventually fall toward $n_r - 1$, and past some point the larger quantile outweighs the smaller variance, so more units lengthen the planned interval. The search therefore looks for the smallest feasible number of units on the stretch where the planned width is still falling, and it counts a number of raters as infeasible when the shortest interval that number allows is too wide. A warning says so when the cheapest design found uses `max_raters` raters, since a larger limit might find a cheaper one. When no design within the limits meets the target, the error names the limit that binds: if more units would do, it gives the design they would allow and points to `max_units`; otherwise more raters are needed, and it says whether the reason is the rater floor or the loss of degrees of freedom under the *t* quantile.

A closed form for the allocation. The search needs no closed form, but one special case has one, which serves as a check. When only judgments cost (`unit_cost = 0`, `rater_cost = 0`), the problem of minimizing V for a fixed number of judgments has the form Marcoulides and Goldstein (1990) solved for a two-facet generalizability study, whose error variance is $\sigma^2_1/n_1 + \sigma^2_2/n_2 + \sigma^2_{12}/(n_1 n_2)$ under a budget on $n_1 n_2$: the optimum has $n_1/n_2 = \sigma^2_1/\sigma^2_2$ (their Equations 5 and 6). Here that is $n_u/n_r = \sigma^2_u/\sigma^2_r$. The cost is flat near the optimum, so the least costly design in whole numbers can stray from that ratio while collecting barely more judgments than the continuous optimum.

An information retrieval evaluation. In a comparison of two retrieval systems over a set of topics, each judged by a set of relevance assessors (people, or automated judges such as language models), the units are the topics, the raters are the assessors, and each cell holds the difference between the two systems' effectiveness scores on the topic as computed from that assessor's judgments. The grand mean is then the mean difference between the systems over the populations of topics and assessors, and `true_var_unit` and `true_var_rater` describe how much the difference between the systems varies over topics and over assessors (the topic by system and assessor by system interactions).

When the planning values are uncertain. The plan is only as good as the planning values, and a rater component that is underestimated is costly, since it governs the floor. ss_aipe_crossed_sensitivity

shows by Monte Carlo simulation what a plan delivers when the true components differ from the planning values. When no trustworthy planning values exist, a two-stage sequential procedure, in which a pilot array determines the final numbers of units and raters, can take their place.

Value

A `data.frame` (class `dmar_ss_aipe`) with rows `necessary_n_units` and `necessary_n_raters` (the design; the second is `specified_n_raters` when `n_raters` was supplied), `total_judgments` ($n_u n_r$, the number of scores to collect), `cost`, `expected_width` (the full width of the interval at the planning values, at most `width`; a plan with an assurance guards against an unlucky sample, so its expected width is no wider, and narrower once the assurance asks for more than the plan without one delivers), `df` (the Satterthwaite degrees of freedom the interval will have at the planned design, computed from the planning values), `rater_floor` (see Details), the echoes `true_var_unit`, `true_var_rater`, `true_var_residual`, `width`, and `conf_level`, and `assurance` when one was supplied. The quantile the plan used is the `"quantile"` attribute. `tidy()` reports the number of units as the sample size, and `glance()` keeps every row.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Brennan, R. L. (2001). *Generalizability theory*. Springer. doi:10.1007/9781475734560

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Marcoulides, G. A., & Goldstein, Z. (1990). The optimization of generalizability studies with resource constraints. *Educational and Psychological Measurement, 50*(4), 761–768. doi:10.1177/0013164490504004

Satterthwaite, F. E. (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin, 2*(6), 110–114. doi:10.2307/3002019

See Also

`ci_crossed` for the interval the plan is for, `ss_aipe_crossed_sensitivity` for what a plan delivers when the planning values are wrong, and `icc` for the reliability of a single rater's scores in the same design.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`, `ss_aipe_equivalence_r_`
`ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`, `ss_aipe_icc()`, `ss_aipe_icc_sensitivi`
`ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivity()`, `ss_aipe_mean()`,
`ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared()`,
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r()`, `ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`,
`ss_aipe_two_phase_cost()`, `ss_aipe_two_phase_sensitivity()`

Examples

```
# Essays scored by raters. Planning values: essays differ with
# variance 0.50, raters differ in leniency with variance 0.10, and the
# residual variance is 0.40. The goal is a 95% interval no wider than
# 0.40 for the mean score; by default the plan collects the fewest
# scores.
ss_aipe_crossed(true_var_unit = 0.50, true_var_rater = 0.10, true_var_residual = 0.40,
                width = 0.40)

# Recruiting and training a rater costs as much as 40 scores, so the
# cheapest design uses fewer raters and more essays.
ss_aipe_crossed(true_var_unit = 0.50, true_var_rater = 0.10, true_var_residual = 0.40,
                width = 0.40, rater_cost = 40, judgment_cost = 1)

# A panel of 30 raters is already set: plan the essays only, with 80%
# assurance that the interval is no wider than 0.40.
ss_aipe_crossed(true_var_unit = 0.50, true_var_rater = 0.10, true_var_residual = 0.40,
                width = 0.40, n_raters = 30, assurance = 0.80)
```

---

ss_aipe_crossed_sensitivity

Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for the Mean of a Crossed Design of Units and Raters

---

Description

Shows by Monte Carlo simulation what a plan from ss_aipe_crossed delivers when the true variance components differ from the planning values. The design is either planned from the supposed components, as a researcher would plan it, or fixed by the user; complete arrays of that size are then drawn from the population with the true components, the interval of ci_crossed is formed from each, and the realized widths and the coverage of the true mean are summarized. A planning value of the rater component that is too small is the costly mistake in this design, since that component sets the floor below which no number of units can bring the interval.

Usage

```
ss_aipe_crossed_sensitivity(
  true_var_unit,
  true_var_rater,
  true_var_residual,
  supposed_var_unit = NULL,
  supposed_var_rater = NULL,
  supposed_var_residual = NULL,
  specified_n_units = NULL,
  specified_n_raters = NULL,
  width,
  conf_level = 0.95,
```

```
  assurance = NULL,
  n_raters = NULL,
  unit_cost = 0,
  rater_cost = 0,
  judgment_cost = 1,
  quantile = c("t", "normal"),
  max_units = 1e+05,
  max_raters = 500,
  true_grand_mean = 0,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_var_unit`, `true_var_rater`, `true_var_residual`
: The data generating variance components of the unit effects, the rater effects, and the residuals (see `ss_aipe_crossed`). The first two must be nonnegative and the third positive.

`supposed_var_unit`, `supposed_var_rater`, `supposed_var_residual`
: The supposed variance components, the planning values the design is planned from (not estimates computed from data), all three together. Supply these or `specified_n_units` and `specified_n_raters`, but not both.

`specified_n_units`, `specified_n_raters`
: A design to evaluate (whole numbers of at least 2, both together) in place of a planned one; the planner is then bypassed.

`width`
: Desired full width of the confidence interval for the mean, against which the realized widths are judged.

`conf_level`
: Confidence level of the interval (default `0.95`).

`assurance`
: Optional assurance, strictly between 0 and 1, passed to `ss_aipe_crossed` when the design is planned; echoed in the output when supplied, with a specified design as well.

`n_raters`, `unit_cost`, `rater_cost`, `judgment_cost`, `quantile`, `max_units`, `max_raters`
: Passed to `ss_aipe_crossed` when the design is planned, and unused with a specified design.

`true_grand_mean`
: The data generating mean (default `0`). The widths and the coverage do not depend on it; it sets the scale of the reported estimates.

`G`
: Number of Monte Carlo replications (default `1000`).

`print_iter`
: Logical. Print the replication index as the simulation runs; default `FALSE`.

`filename`
: Optional path of a CSV file to receive the per-replication results (the grand mean, the two limits, the width, the degrees of freedom, and the two tail misses), appended when the file already holds a table and created otherwise; the default

NULL writes nothing, and a throwaway run that wants the file should point it at tempfile(fileext = ".csv").

seed — Optional integer seed. The default NULL draws from, and advances, the current random number stream; an integer makes the call reproducible, and the caller's random number state is restored when the call returns.

Details

Each replication draws n_u unit effects, n_r rater effects, and $n_u n_r$ residuals from normal distributions with the true variances, forms the complete array, and computes the interval exactly as ci_crossed does: the ANOVA estimates of the components, negative ones truncated at zero, plugged into the variance of the grand mean, with Satterthwaite (1946) degrees of freedom. The summary separates two sources of error: the planning values (compare a well specified run with a misspecified one) and the approximations of the plan itself (a well specified run whose pct_ci_less_width is well below one half, or below the assurance, or whose coverage is far from conf_level; a plan without an assurance meets its width somewhat more than half the time, as ss_aipe_crossed explains). A plan with quantile = "normal" shows the second kind, since its target ignores the t quantile the interval uses; with a rater component at or near zero it can fall far short.

Value

A data.frame (class dmar_tbl) with rows mean_grand_mean, median_grand_mean, and sd_grand_mean (summaries of the G estimates); mean_ci_width, median_ci_width, and sd_ci_width (of the realized widths); pct_ci_less_width (the proportion of intervals no wider than width); pct_ci_miss_low (the proportion lying entirely above the true mean), pct_ci_miss_high (entirely below it), and total_type_I_error (their sum); mean_df (the mean Satterthwaite degrees of freedom); and the echoes n_units and n_raters (the design evaluated), the three true_var_* and three supposed_var_* components (the latter NA for a specified design), true_grand_mean, width, conf_level, and assurance when one was supplied. The proportions are on the 0 to 1 scale.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Satterthwaite, F. E. (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin, 2*(6), 110–114. doi:10.2307/3002019

See Also

`ss_aipe_crossed`, `ci_crossed`

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_equivalence_r()`, `ss_aipe_equivalence_r_sensitivity(`
`ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`, `ss_aipe_icc()`, `ss_aipe_icc_sensitivi`
`ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivity()`, `ss_aipe_mean()`,
`ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared()`,
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r()`, `ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`,
`ss_aipe_two_phase_cost()`, `ss_aipe_two_phase_sensitivity()`

Examples

```
# G = 200 keeps the example quick; a reported study deserves the
# default G = 1000 or more. The first plan on the help page of
# ss_aipe_crossed(), evaluated when its planning values are right:
ss_aipe_crossed_sensitivity(
  true_var_unit = 0.50, true_var_rater = 0.10, true_var_residual = 0.40,
  supposed_var_unit = 0.50, supposed_var_rater = 0.10,
  supposed_var_residual = 0.40,
  width = 0.40, G = 200, seed = 113)

# The same plan when the raters in fact differ twice as much as
# planned: nearly every interval is now wider than the target, since
# the design has too few raters for the larger rater component.
ss_aipe_crossed_sensitivity(
  true_var_unit = 0.50, true_var_rater = 0.20, true_var_residual = 0.40,
  supposed_var_unit = 0.50, supposed_var_rater = 0.10,
  supposed_var_residual = 0.40,
  width = 0.40, G = 200, seed = 113)
```

`ss_aipe_cv` *Sample Size Planning for the Coefficient of Variation Given the Goal of Accuracy in Parameter Estimation Approach to Sample Size Planning*

Description

Takes a planning value of the population coefficient of variation and a desired confidence interval width, and returns the smallest sample size at which the interval of `ci_cv` is expected to be that narrow (Kelley, 2007). With `assurance` it returns the sample size of the published procedure of that article, at which the realized two-sided interval is that narrow with about the stated probability. The interval and the plan assume a normal population (see Details).

The population coefficient of variation may be given directly as `true_cv` or through `true_mean` and `true_sd`, in which case `true_cv` is taken as `true_sd / true_mean`. It must be positive.

Usage

```
ss_aipe_cv(
  true_cv = NULL,
  width = NULL,
  conf_level = 0.95,
  assurance = NULL,
  true_mean = NULL,
  true_sd = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL
)
```

Arguments

`true_cv`	Population coefficient of variation the sample size is planned for (kappa in Kelley, 2007). The planner treats this value as the truth; `ss_aipe_cv_sensitivity` shows what the plan delivers when the truth (`true_cv` there) differs from the value planned with (`supposed_cv` there).
`width`	Desired (full) width of the confidence interval
`conf_level`	Confidence interval coverage; 1-Type I error rate. A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in `ci_nc_t`)
`assurance`	Optional. The probability that the realized interval is no wider than `width`: a single number strictly between 0 and 1 (for example .80, .90, or .99). A value such as .50 may be used; it asks that the realized interval be no wider than `width` in about half of the studies that could be run, where the plan without `assurance` asks that the width at the expected value of the sample coefficient be no larger than `width`. Near .50 the realized probability can be several points below the request and the sample size can be one smaller than without `assurance`; see Details for the procedure and the numbers. Any other value is refused with a condition of class `dmar_assurance_invalid`. `assurance` cannot be given with `alpha_lower = 0`, a one-sided upper confidence bound (see **Edge cases**).

true_mean	Population mean, a positive number (specified with true_sd when true_cv is not specified)
true_sd	Population standard deviation, a positive number (specified with true_mean when true_cv is not specified)
alpha_lower	Type I error for the lower confidence limit, given with alpha_upper when conf_level is NULL: a single number that is at least 0 and below 1
alpha_upper	Type I error for the upper confidence limit. A nonzero tail area below 1e-9 is refused, as in ci_cv

Details

Estimand and interval. The quantity planned for is the population coefficient of variation $\kappa = \sigma/\mu$ of a normal population with a positive mean, estimated from N independent observations by $k = s/M$. The interval is the one ci_cv reports, from the noncentral t distribution of $\sqrt{N}/k$, two-sided at conf_level (or with the tail areas alpha_lower and alpha_upper when conf_level is NULL).

The expected-width plan. At each candidate sample size N the width is evaluated at the value Kelley (2007, Equation 3) gives for the expected value of the sample coefficient of variation given true_cv and N,

$$E[k \mid \kappa, N] \approx \kappa \left[1 - \frac{1}{4(N-1)} + \frac{\kappa^2}{N} + \frac{1}{2(N-1)^2}\right],$$

and the smallest N whose width at that value is no larger than width is returned. The expression is the approximation of Sokal and Braumann (1980, p. 51, Expression 3) for normal data. It stops at the third-order term and is above the expected value at small N: for a coefficient of .25, by 3.1 percent at $N = 5$, 0.6 percent at $N = 10$, and 0.1 percent at $N = 20$. The function keeps it because the published tables were computed with it. The search starts at $N = 4$. The interval is computed exactly; for how this relates to the sample sizes tabled in Kelley (2007), see **The published tables** below.

The width of the interval is not a linear function of the sample coefficient of variation, so the width at the value of Equation 3 is not exactly the expected width. Over 62 conditions (coefficients of variation of .05 to .50, widths of .025 to .20, and confidence levels of .90, .95, and .99, with plans from 5 to 1,500) the returned N was the smallest N with an expected width no larger than width in 47 and one smaller in 15; where it was one smaller the expected width exceeded width by at most 0.7 percent.

The assurance step. With assurance the function runs the computation that produced the tables of Kelley (2007). At the expected-width sample size N_0 it finds κ_γ, the value that the sample coefficient exceeds with probability 1 - assurance, from the noncentral t distribution of Equation 5 of the article. It then takes the value of Equation 3 at κ_γ and N_0 as the planning value and runs the expected-width search for it, which applies Equation 3 to that value again at each N. The quantile and the intervals are computed exactly (see ci_nc_t). The article's text (p. 759) describes the search as run for κ_γ itself. Recomputed with R's own noncentral t functions, as in 2007, the computation described here gives the printed value in 524 of the 528 assurance cells of Tables 1 to 3 (the other four are in the two rows that show a misprint), where the search for κ_γ itself gives the printed value in 365. The two computations give different plans in 160 cells.

The published tables. Kelley (2007) computed Tables 1 to 3 (pp. 760–762) with R's noncentral t distribution, which is a normal approximation above a noncentrality of 37.62. Here the noncentrality

is $\sqrt{N}/k$, which narrow intervals pass, and the function computes there exactly, so it does not return every printed value. Two rows of the tables show a misprint. Table 1 (90 percent confidence), assurance .80, width .125, coefficient of variation .40 prints 127, between 69 and 122 in its row; the procedure gives 93. Table 3 (99 percent confidence), assurance .80, width .010 prints 14,279; 28,321; 50,209; 38,120; and 82,817 for coefficients of variation of .30 to .50, which do not increase; the procedure gives 14,277; 20,473; 28,320; 38,118; and 50,207, so the printed 28,321 and 50,209 are its values for .40 and .50 (to within the approximation), one and two columns to the left of where they belong. Apart from the four misprinted cells (127; 28,321; 50,209; and 82,817), the function returns the printed value in 468 of the 788 cells, one to three fewer cases in 232, and one to three more in 88. It returns every printed value with a width of .125 or more at 90 and 95 percent confidence, the misprint aside, and 9 of the 120 with a width of .075 or less at 99 percent. The worked example of the article (pp. 759–760), a coefficient of variation of .25, a width of .10, and 99 percent confidence, gives 101, and 142 with an assurance of .99, where the article gives 99 and 141: at 99 the interval at the expected value is .1006 wide, and .0999 by the approximation.

The procedure is a close approximation and not an exact search, so the probability that the realized interval is no wider than `width` at the returned sample size is near `assurance` and not exactly it. In the 488 printed cells with an assurance and at most 5,000 cases that probability was .744 to .902 where .80 was requested (below .80 in 37 of 244 cells) and .9916 to .9997 where .99 was requested. Away from the published tables, and more so as the request falls toward .50, it can fall below the request: for a coefficient of variation of .22, a width of .09, a 95 percent interval, and an assurance of .60 the function returns 57, at which the probability is .587, and one more case gives .621. In 105 plans computed at 95 percent confidence (coefficients of variation from .05 to .50, widths of a quarter, a half, and three quarters of the coefficient) the probability was below the request by at most .038 for a request of .50, .036 for .55, .058 for .60, .032 for .70, and .001 for .80, and it was not below the request in any plan for .90 and .99. A request of .50 does not reproduce the plan without `assurance`: in 12 of the 15 plans it returned a sample size one smaller than the plan without `assurance` does, and for a coefficient of variation of .05 and a width of .0375 it returns 17, at which the probability is .462, where the plan without `assurance` returns 18. In the other direction, when `width` is close to `true_cv` and `assurance` is high the returned sample size can be well above the smallest sufficient one: for a coefficient of variation of .5, a width of .5, a 95 percent interval, and an assurance of .99 the function returns 57, and the probability reaches .99 at 37. `ss_aipe_cv_sensitivity` estimates the probability for any plan by simulation.

Populations that are not normal. The interval and the plan assume a normal population. With a skewed population the coverage of the interval departs from the stated level, and a larger sample does not repair it (see `ci_cv`). In 40,000 samples at each of the three plans for a coefficient of variation of .50, a width of .20, and 95 percent confidence (79 without `assurance`, 97 with an assurance of .80, and 142 with .99), coverage was .97 for a gamma population with that coefficient of variation, and for a lognormal one it was .915, .913, and .905, lowest at the largest plan. The proportion of intervals no wider than `width` was .55, .88, and 1.00 for the gamma population and .59, .84, and .99 for the lognormal one. For a normal population that probability, computed from the noncentral *t* distribution, is .53, .84, and 1.00.

Edge cases. When `width` is already met at $N = 4$ the function returns 4. A `width` that no sample size reaches within 100,000 steps of the search stops with an error. With `alpha_upper = 0` the interval is a one-sided lower confidence bound whose upper limit is infinite at every sample size, so no plan exists, and the call is refused at once with a condition of class `dmar_argument_invalid`. With `alpha_lower = 0`, an upper bound, the lower limit is 0 and the width is the bound itself, which comes down toward the coefficient of variation as N grows, so a `width` larger than `true_cv` can be planned without `assurance`. With `assurance` the call is refused, with class `dmar_argument_invalid`:

the assurance step was written for a two-sided interval and does not give the requested probability for a bound. For a coefficient of variation of .2, a width of .25, `conf_level = NULL`, `alpha_lower = 0`, and `alpha_upper = .05`, where the plan without `assurance` is 37, the step gives 33, 91, and 219 for an assurance of .50, .80, and .90. The probability that the bound is no larger than the width is .470, .878, and .998 at those sizes, and the smallest sample sizes at which it reaches the request are 36, 74, and 98. With `alpha_upper` below .5 the bound at the value of Equation 3 stays above `true_cv` at every sample size the search can reach: over `true_cv` from .001 to 5 and N from 4 to 100,004 the smallest bound was 1.000006 times `true_cv`. (The value of Equation 3 is itself below `true_cv` when `true_cv` is below $\sqrt{N(N-3)}/(2(N-1))$, which is .33 at $N = 4$, .46 at $N = 10$, and .49 at $N = 20$, and approaches .5.) A `width` at or below `true_cv` is therefore never reached, and the call is refused at once, also with class `dmar_argument_invalid`. With a large coefficient of variation the first sample sizes of the search can be so small that the upper limit of the interval is infinite (at 99 percent confidence and a coefficient of .45, $N = 4$ is one). An infinite interval is never narrow enough, so the search goes on, and the interval at the returned sample size is finite.

Comparison with MBESS. MBESS 3.4.0 and later evaluate the width at `true_cv` multiplied by $(1 + 1/(4N))$, the nearly unbiased estimate of Kelley (2007, Equation 4) applied to the population value, which is larger than `true_cv` where the expected value is smaller, and use R's approximation to the noncentral *t* beyond a noncentrality of 37.62. Their sample sizes are one larger than the tables in Kelley (2007) in many conditions, and up to three larger with `assurance`; for the worked example they are 100 and 142, where this function gives 101 and 142.

Value

A `data.frame` with the columns `term` and `value` and one row, `necessary_N`, the planned sample size, with or without `assurance`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., & Kelley, K. (2016). Estimation of the coefficient of variation with minimum risk: A sequential method for minimizing sampling error and study cost. *Multivariate Behavioral Research, 51*(5), 627–648. doi:10.1080/00273171.2016.1203279

Kelley, K. (2007). Sample size planning for the coefficient of variation from the accuracy in parameter estimation approach. *Behavior Research Methods, 39*(4), 755–766. doi:10.3758/BF03192966

Sokal, R. R., & Braumann, C. A. (1980). Significance tests for coefficients of variation and variability profiles. *Systematic Zoology, 29*(1), 50–66. doi:10.1093/sysbio/29.1.50

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on confidence intervals for effect sizes.)

See Also

`ss_aipe_cv_sensitivity`, `cv`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Suppose one wishes to have a confidence interval with an expected width of .10
# for a 99% confidence interval when the population coefficient of variation is .10.
ss_aipe_cv(true_cv = .1, width = .1, conf_level = .99)

# The same planning problem parameterized by the population mean and standard
# deviation: true_mean = 10 and true_sd = 1 imply the same coefficient of variation, .10.
ss_aipe_cv(true_mean = 10, true_sd = 1, width = .1, conf_level = .99)

# Ensuring that the confidence interval will be sufficiently narrow with a 99%
# certainty for the situation above.
ss_aipe_cv(true_cv = .1, width = .1, conf_level = .99, assurance = .99)
```

---

ss_aipe_cv_sensitivity

Sensitivity Analysis for Sample Size Planning From the Accuracy in Parameter Estimation Perspective for the Coefficient of Variation

---

Description

Simulates a sample size plan for the coefficient of variation to show what it delivers when the population value differs from the planning value, or at a sample size chosen in advance. It takes the population coefficient of variation, either a planning value or a sample size, and a target width, and returns summaries of the estimate, the interval width, the proportion of intervals no wider than the target, and coverage over `G` simulated normal samples (Kelley, 2007; Maxwell, Delaney, & Kelley, 2027, Section 3.11 on power-based sample size planning).

Usage

```
ss_aipe_cv_sensitivity(
  true_cv = NULL,
  supposed_cv = NULL,
  width = NULL,
  assurance = NULL,
  true_mean = 100,
  specified_N = NULL,
  conf_level = 0.95,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_cv`	Population coefficient of variation (the data generating value); a single positive number

supposed_cv	The supposed coefficient of variation, the planning value from which ss_aipe_cv plans the study (a value the researcher supposes, not an estimate computed from data); must be positive. Supply this or specified_N but not both.
width	Desired (full) width of the two-sided confidence interval for the population coefficient of variation; a single positive number. With specified_N it may be left NULL, and pct_ci_less_width is then NaN.
assurance	Probability with which the realized interval should be no wider than width, passed to ss_aipe_cv: NULL (plan for the expected width) or a single number strictly between 0 and 1; see ss_aipe_cv for how the procedure behaves near .50. With specified_N it is only echoed.
true_mean	Population mean used by the simulator to generate data, a single positive number (the standard deviation is determined by true_mean and true_cv, since CV = sigma/mu). Default 100.
specified_N	Pre-specified sample size to evaluate, a whole number of at least 4 (use this when you want the sensitivity results at a fixed *N* rather than at the *N* that ss_aipe_cv would recommend); incompatible with supposed_cv.
conf_level	Desired confidence level (i.e., 1 - Type I error rate); default 0.95
G	Number of Monte Carlo replications; defaults to 1000. Increase (e.g., 5000 or 10000) for stable Type I error estimates.
print_iter	Logical. If TRUE the simulation prints the iteration index after each replication (helpful for long runs); default FALSE.
filename	An optional path for a comma separated file recording every replication (the two confidence limits, the realized coefficient of variation, a coverage indicator, and the interval width): nothing is written when filename is NULL (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at tempfile(fileext = ".csv").
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the run reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Sample size planning for the coefficient of variation under the Accuracy in Parameter Estimation framework chooses *N* so that the expected (or, with assurance, the high-probability) confidence interval width is no larger than width (Kelley, 2007). Because the procedure assumes the planning value supposed_cv matches the population value true_cv, in practice the realized width will deviate from the planned width whenever the planning value is wrong. This sensitivity analysis quantifies the deviation by Monte Carlo simulation: the planned *N* is obtained from ss_aipe_cv with supposed_cv, then samples are drawn from the *true* population (with coefficient of variation true_cv) and the realized confidence interval widths are summarized.

For planning a sample size so that a confidence interval is sufficiently narrow, see Maxwell, Delaney, & Kelley (2027, Section 3.9.1.5, and Section 4.9 on assurance).

The samples are drawn from a normal population, which the interval and the plan assume, so the function does not show what a plan delivers for a population that is not normal. With a skewed

population the coverage of the interval departs from the stated level: in 40,000 samples at each of the three plans for a coefficient of variation of .50, a width of .20, and 95 percent confidence (79 without `assurance`, 97 with an assurance of .80, and 142 with .99), coverage was .97 for a gamma population with that coefficient of variation, and for a lognormal one it was .915, .913, and .905, lowest at the largest plan. The proportion of intervals no wider than `width` was .55, .88, and 1.00 for the gamma population and .59, .84, and .99 for the lognormal one. For a normal population that probability, computed from the noncentral *t* distribution, is .53, .84, and 1.00 (see `ss_aipe_cv` and `ci_cv`).

Value

A `data.frame` with columns `term` and `value` summarizing the Monte Carlo results across the `G` replications. The `term` entries are: `"mean_cv"`, `"median_cv"`, `"sd_cv"` (mean / median / SD of the observed sample coefficients of variation); `"mean_ci_width"`, `"median_ci_width"`, `"sd_ci_width"` (corresponding summaries of the realized interval widths); `"pct_ci_less_width"` (proportion of intervals at or below the planning width `width`); `"pct_ci_miss_low"` and `"pct_ci_miss_high"` (tail-specific non-coverage); `"total_type_I_error"` (overall empirical non-coverage of `true_cv`); plus the input echoes `"total_N"` (the sample size evaluated), `"true_cv"`, `"supposed_cv"` (NA when `specified_N` was supplied instead), `"width"`, `"conf_level"`, and `"assurance"` (present only when an assurance was supplied). The proportion rows are on the 0 to 1 scale, not percentages, so `total_type_I_error` is the sum of `pct_ci_miss_low` and `pct_ci_miss_high`.

Which replications each row describes. Three rules decide it.

1. A replication whose sample mean is not positive has no confidence interval for the coefficient of variation (`ci_cv` requires a positive mean), and its sample coefficient is negative or undefined; this can happen when `true_cv` is large and the sample is small. All ten statistic rows, `mean_cv`, `median_cv`, and `sd_cv` included, describe the replications with a positive mean, so that every row is about the same samples. Such a replication is recorded with its coefficient and missing limits in the file `filename` names. One warning of class `dmar_replications_without_interval` states how many there were, and the count is `attr(, "simulation")$failed`.
2. An interval can have an infinite upper limit (see `ci_cv`). Such an interval has infinite width and counts as not meeting the width target in `pct_ci_less_width`. `mean_ci_width` and `sd_ci_width` are over the finite widths, since one infinite width would make both uninformative, and `median_ci_width` is over all widths, infinite ones included. The number of infinite widths is `attr(, "simulation")$n_infinite_width`.
3. An interval with an infinite upper limit is a valid interval of `ci_cv`, which signals nothing for it. One warning of class `dmar_replications_with_infinite_limit` states how many replications had one and how they enter the rows; the condition carries the count as `n_replications` and the number of replications as `G`.

The attribute `simulation` (see the section on the record of the simulation) has one element more here, `n_infinite_width`.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals

that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted` − `returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., & Kelley, K. (2016). Estimation of the coefficient of variation with minimum risk: A sequential method for minimizing sampling error and study cost. *Multivariate Behavioral Research, 51*(5), 627–648. doi:10.1080/00273171.2016.1203279

Kelley, K. (2007). Sample size planning for the coefficient of variation from the accuracy in parameter estimation approach. *Behavior Research Methods, 39*(4), 755–766. doi:10.3758/BF03192966

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on confidence intervals for effect sizes.)

See Also

`ss_aipe_cv`, `ci_cv`, `cv`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Textbook scenario (Kelley, 2007). A researcher plans a study to estimate the
# coefficient of variation of reaction times with a 95% confidence interval no
# wider than .10. They guess from prior work that the population coefficient of
# variation is around .25 and apply ss_aipe_cv() to obtain a planned N.
```

```
# Question 1: how does the realized interval width behave when the planning
# value is correct (well-specified case)? G = 100 keeps the example quick;
# a reported sensitivity analysis deserves the default G = 1000 or more.
set.seed(113)
ss_aipe_cv_sensitivity(
  true_cv          = .25,
  supposed_cv      = .25,
  width            = .10,
  assurance        = NULL,
  conf_level       = .95,
  G                = 100,
  print_iter       = FALSE
)

# Question 2: what happens if the planning value is materially smaller than
# the true coefficient of variation (a common direction of misspecification,
# since planning values are often optimistic)? The intervals will be wider on
# average than the target and the pct_ci_less_width will fall.
set.seed(113)
ss_aipe_cv_sensitivity(
  true_cv          = .35,
  supposed_cv      = .25,
  width            = .10,
  assurance        = NULL,
  conf_level       = .95,
  G                = 100,
  print_iter       = FALSE
)
```

`ss_aipe_c_ancova` *Sample Size Planning for a Contrast in Randomized ANCOVA From the Accuracy in Parameter Estimation (AIPE) Perspective*

Description

Plans the sample size per group so that the confidence interval for an unstandardized contrast in a one-covariate randomized ANCOVA is sufficiently narrow, following the accuracy in parameter estimation (AIPE) approach. To the extent the covariate correlates with the response, the covariate adjustment shrinks the error variance, so the desired precision is reached with a smaller sample size than the corresponding ANOVA design requires.

Usage

```
ss_aipe_c_ancova(
  true_var_error_ancova = NULL,
  true_var_error_anova = NULL,
  true_cor = NULL,
  c_weights,
```

```
  width,
  conf_level = 0.95,
  assurance = NULL
)
```

Arguments

true_var_error_ancova	The population error variance of the ANCOVA model (i.e., the mean square within of the ANCOVA model)
true_var_error_anova	The population error variance of the ANOVA model (i.e., the mean square within of the ANOVA model)
true_cor	The population correlation coefficient of the response and the covariate
c_weights	The contrast weights
width	The desired full width of the obtained confidence interval
conf_level	The desired confidence interval coverage, (i.e., 1 - Type I error rate)
assurance	Parameter to ensure that the obtained confidence interval width is narrower than the desired width with a specified degree of certainty (must be NULL or between zero and unity)

Details

Either the error variance of the ANCOVA model or of the ANOVA model can be used to plan the appropriate sample size per group. When using the error variance of the ANOVA model to plan sample size, the correlation coefficient of the response and the covariate is also needed.

Value

A 1-row `data.frame` with columns `term` and `value`:

necessary_n_per_group	The necessary sample size *per group*

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

See Also

ci_c_ancova, ci_sc_ancova, ss_aipe_c

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Suppose the population error variance of some three-group ANOVA model
# is believed to be 40, and the population correlation coefficient
# of the response and the covariate is 0.22. The researcher is
# interested in the difference between the mean of group 1 and
# the average of means of group 2 and 3. To plan the sample size so
# that, with 90 percent certainty, the obtained 95 percent full
# confidence interval width is no wider than 3:

ss_aipe_c_ancova(true_var_error_anova = 40, true_cor = .22, c_weights = c(1, -0.5, -0.5),
                 width = 3, assurance = .90)
```

---

ss_aipe_c_ancova_sensitivity

Sensitivity Analysis for Sample Size Planning for the (Unstandardized) Contrast in Randomized ANCOVA From the Accuracy in Parameter Estimation (AIPE) Perspective

---

Description

Performs a sensitivity analysis when planning sample size from the Accuracy in Parameter Estimation (AIPE) Perspective for the (unstandardized) contrast in randomized ANCOVA design.

Usage

```
ss_aipe_c_ancova_sensitivity(
  true_var_error_ancova = NULL,
  supposed_var_error_ancova = NULL,
  true_var_error_anova = NULL,
  supposed_var_error_anova = NULL,
  true_cor,
  supposed_cor = NULL,
  G = 10000,
  true_mean_Y,
  true_sd_Y,
  true_mean_X,
  true_sd_X,
  c_weights,
  width,
  conf_level = 0.95,
```

```
  assurance = NULL,
  filename = NULL,
  seed = NULL
)
```

Arguments

true_var_error_ancova	population error variance of the ANCOVA model
supposed_var_error_ancova	the supposed error variance of the ANCOVA model, the planning value given to ss_aipe_c_ancova to choose the sample size
true_var_error_anova	population error variance of the ANOVA model (i.e., excluding the covariate)
supposed_var_error_anova	the supposed error variance of the ANOVA model (i.e., excluding the covariate), a planning value used with supposed_cor
true_cor	population correlation coefficient of the response and the covariate
supposed_cor	the supposed correlation coefficient of the response and the covariate, a planning value used with supposed_var_error_anova
G	number of generations (i.e., replications) of the simulation
true_mean_Y	vector that contains the response's population mean of each group
true_sd_Y	the population standard deviation of the response
true_mean_X	the population mean of the covariate
true_sd_X	the population standard deviation of the covariate
c_weights	the contrast weights
width	the desired full width of the obtained confidence interval
conf_level	the desired confidence interval coverage, (i.e., 1 - Type I error rate)
assurance	parameter to ensure that the obtained confidence interval width is narrower than the desired width with a specified degree of certainty (must be NULL or between zero and unity)
filename	an optional path for a comma separated file recording every replication (the realized contrast, its full and covariate-ignoring standard errors and their ratio, the interval width, and the three non-coverage indicators): nothing is written when filename is NULL (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at tempfile(fileext = ".csv")
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

The arguments true_mean_Y, true_mean_X, true_sd_Y, and true_sd_X are used to generate random data in the simulations for the sensitivity analysis. The value of true_sd_Y should be the same as the square root of true_var_error_anova.

So far this function is based on one-covariate randomized ANCOVA design only. The argument true_mean_X should be a single number, because it is assumed that the population mean of the covariate is equal across groups in randomized ANCOVA.

Value

A data.frame with columns term and value summarizing the Monte Carlo sensitivity analysis across G replications. The term entries are: mean_contrast, median_contrast, sd_contrast (summaries of the realized unstandardized contrast); mean_ci_width, median_ci_width, sd_ci_width (summaries of the realized interval widths); pct_ci_less_width (proportion of intervals narrower than the planning target width); pct_ci_miss_low and pct_ci_miss_high (tail-specific empirical non-coverage of the population contrast); total_type_I_error (overall empirical non-coverage, the sum of the two tails); mean_se_ratio (mean ratio of the contrast standard error that ignores the covariate-imbalance term to the full ANCOVA standard error); and the input echoes n_per_group, total_N, true_contrast (the population contrast implied by true_mean_Y and c_weights), supposed_var_error_ancova (as supplied or as resolved from supposed_var_error_anova and supposed_cor), true_cor, width, conf_level, and assurance (present only when an assurance was supplied). The proportion rows are on the 0 to 1 scale, not percentages. The per-replication vectors (contrast, se_contrast, se_contrast_restricted, width_obs) are not returned; they are written to the comma separated file named by filename when one is supplied.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

width_target_mc_ci the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as G grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives G, and the three counts when a replication

was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

See Also

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Monte Carlo sensitivity sweep; G is small here so the example runs quickly.
# Raise G (e.g., G = 1000 or more) for a stable sensitivity analysis.
set.seed(113)
ss_aipe_c_ancova_sensitivity(true_var_error_ancova = 30,
                             supposed_var_error_ancova = 30,
                             true_cor = .2, true_mean_Y = c(10, 12, 15, 13),
                             true_mean_X = 2, true_sd_X = 1.3, true_sd_Y = 2,
                             c_weights = c(1, 0, -1, 0), width = 3, G = 25)

ss_aipe_c_ancova_sensitivity(true_var_error_anova = 36,
                             supposed_var_error_anova = 36,
                             true_cor = .2, supposed_cor = .2,
                             true_mean_Y = c(10, 12, 15, 13), true_mean_X = 2,
                             true_sd_X = 1.3, true_sd_Y = 6,
                             c_weights = c(1, 0, -1, 0), width = 3,
                             assurance = NULL, G = 25)
```

---

ss_aipe_c_sensitivity *Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for an Unstandardized Contrast*

Description

Quantifies how much misspecification of the population error variance distorts an AIPE-based sample size plan for an unstandardized contrast of means. Because the half-width of the confidence interval on $\psi = \sum_j c_j \mu_j$ depends on the error variance, the contrast weights, and the per-group sample size, but not on the value of ψ itself, this sensitivity analysis varies the planning value of the error variance. On each replication the function simulates n observations per group from a normal population with variance `true_var_error`, builds the confidence interval via `ci_c`, and summarizes the realized widths and coverage of `true_contrast`.

Usage

```
ss_aipe_c_sensitivity(
  true_var_error = NULL,
  supposed_var_error = NULL,
  c_weights,
  width,
  true_contrast = 0,
  specified_n_per_group = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_var_error`	Population error variance (the data generating value). Must be positive.
`supposed_var_error`	Error variance used to plan the study (the value passed to `ss_aipe_c`). Supply this or `specified_n_per_group` but not both.
`c_weights`	Contrast weight vector. Must sum to zero.
`width`	Desired full width of the confidence interval on the unstandardized contrast.
`true_contrast`	Population value of the contrast; the simulator places group means such that $\sum_j c_j \mu_j =$`true_contrast`. The width of the interval does not depend on this value but the realized coverage of `true_contrast` does. Default `0`.
`specified_n_per_group`	Per-group sample size to evaluate (incompatible with `supposed_var_error`); when used, the planner is bypassed.
`conf_level`	Confidence level (default `0.95`). A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in `ci_nc_t`).
`assurance`	Optional probability that the realized interval is no wider than `width`; passed to `ss_aipe_c` when resolving the planned sample size.
`G`	Number of Monte Carlo replications (default 1000).

print_iter — Logical. Print the iteration index after each replication (helpful for long runs); default FALSE.

filename — Optional path of a CSV file to receive the per-replication results (the contrast estimate, the confidence limits, the interval width, and the two tail misses), appended when the file already exists and created otherwise; the default NULL writes nothing, and a throwaway run that wants the file should point it at tempfile(fileext = ".csv").

seed — NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Value

A data.frame with rows for mean / median / SD of the realized contrast (mean_contrast, median_contrast, sd_contrast) and of the interval width, the proportion of intervals at or below width, the tail-specific and overall empirical non-coverage of true_contrast, and the input echoes (per-group sample size, total sample size, the true and supposed error variances true_var_error and supposed_var_error, true_contrast, width, confidence level, and, when one was supplied, assurance).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

width_target_mc_ci the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as G grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives G, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so failed is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons.)

See Also

ss_aipe_c, ci_c, ss_aipe_sc_sensitivity

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensitivity(), ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(), ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(), ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivit ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(), ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# G = 50 keeps the example quick; a reported analysis deserves the
# default G of 1000. First the well-specified case: the planner used an
# error variance of 4, and the truth is 4.
set.seed(113)
ss_aipe_c_sensitivity(
  true_var_error     = 4,
  supposed_var_error = 4,
  c_weights = c(-1, 0, 1),
  width = 1, G = 50, print_iter = FALSE
)

# Misspecified: planner used 4, truth is 9. Realized widths inflate.
set.seed(113)
ss_aipe_c_sensitivity(
  true_var_error     = 9,
  supposed_var_error = 4,
  c_weights = c(-1, 0, 1),
  width = 1, G = 50, print_iter = FALSE
)
```

`ss_aipe_equivalence_r` *AIPE Sample Size Planning for an Equivalence Test on the Pearson Correlation*

Description

Computes the minimum sample size needed so that the equivalence CI (the 100(1 - 2α)% CI on the Pearson correlation ρ) has expected full width $\leq \omega$ (Kelley, 2007; Lakens, 2017), where ρ is `true_cor`, ω is `width`, and α is `alpha_level`. With the standard $\alpha = 0.05$ TOST level, the equivalence CI is the 90% CI. The interval is the Fisher's Z construction that `equivalence_r` and `ci_r` use, so the plan and the analysis invert the same interval.

Usage

```
ss_aipe_equivalence_r(
  true_cor = 0,
  width,
  alpha_level = 0.05,
  assurance = NULL,
  seed = NULL
)
```

Arguments

`true_cor`	Population correlation ρ the sample size is planned for. The planner treats this value as the truth; `ss_aipe_equivalence_r_sensitivity` shows what the plan delivers when the truth (`true_cor` there) differs from the value planned with (`supposed_cor` there). Default `0`: the confidence interval for a correlation is widest at $\rho = 0$, so the default plans under the widest-interval case and is conservative for any other value.
`width`	Target full CI width on the correlation scale (e.g., `0.20` for a 90% CI of width 0.20). Must be in $(0, 2)$, the width of the correlation scale itself.
`alpha_level`	One-sided TOST significance level α. The CI used in planning is at confidence level $1 - 2\alpha$. Default `0.05` (90% CI).
`assurance`	Optional assurance probability in $(0, 1)$. When supplied, the function chooses *N* so that the probability of achieving `width` or less is at least `assurance` (Kelley, Maxwell, & Rausch, 2003). Default `NULL` (no assurance correction).
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Closed form on the Fisher's Z scale. The equivalence CI has half-width $h = z_{1-\alpha}/\sqrt{N-3}$ on the Fisher's Z scale, and its width on the correlation scale is

$$w(N) \;=\; \tanh(Z_\rho + h) - \tanh(Z_\rho - h),$$

where $Z_\rho = \tanh^{-1}(\rho)$. The function returns the smallest integer $N \geq 4$ with $w(N) \leq \omega$. At $\rho = 0$ this is available in closed form, $N = \lceil 3 + (z_{1-\alpha}/\tanh^{-1}(\omega/2))^2 \rceil$, and away from zero the back-transform shortens the interval, so the required *N* can only decrease as $|\rho|$ grows.

Choosing the width from equivalence bounds. To leave room for an equivalence verdict inside bounds $(-b, b)$, the interval must at minimum fit inside the bounds when centered at the anticipated ρ, so a width somewhat below $2b$ (for ρ near 0) is the natural target; the Monte Carlo sensitivity sibling `ss_aipe_equivalence_r_sensitivity` reports the realized proportion of equivalence verdicts at the planned *N*.

Assurance. Under `assurance = q`, the function increments N until the Monte Carlo probability that the realized width is $\leq \omega$ is at least q, drawing the sampling distribution of $\widehat{Z}$ as normal with mean Z_ρ and variance $1/(N-3)$.

Value

A 4-row `data.frame` with columns `term` and `value`: the recommended sample size `necessary_N`, the target `width`, the planning value `true_cor`, and the resulting `expected_width` at the chosen *N*.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Counsell, A., & Cribbie, R. A. (2015). Equivalence tests for comparing correlation and regression coefficients. *British Journal of Mathematical and Statistical Psychology, 68*(2), 292–309. doi:10.1111/bmsp.12045

Goertzen, J. R., & Cribbie, R. A. (2010). Detecting a lack of association: An equivalence testing approach. *British Journal of Mathematical and Statistical Psychology, 63*(3), 527–537. doi:10.1348/000711009X475853

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Lakens, D. (2017). Equivalence tests: A practical primer for *t* tests, correlations, and meta-analyses. *Social Psychological and Personality Science, 8*(4), 355–362. doi:10.1177/1948550617697177

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

See Also

`equivalence_r`, `ss_aipe_r`, `ci_r`, `ss_aipe_equivalence_smd`, `ss_aipe_equivalence_r_sensitivity`

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r_sensit`
`ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`, `ss_aipe_icc()`, `ss_aipe_icc_sensitivit`
`ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivity()`, `ss_aipe_mean()`,

ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(), ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(), ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# 1. Plan for a 90% CI on the correlation of width <= 0.20, under
#    the widest-interval planning value true_cor = 0:
ss_aipe_equivalence_r(true_cor = 0, width = 0.20)

# 2. The same width assuming a true correlation of 0.30 requires
#    fewer participants, since the interval narrows away from zero:
ss_aipe_equivalence_r(true_cor = 0.30, width = 0.20)

# 3. With 80% assurance (the assurance path is Monte Carlo, so give a
#    seed for a reproducible result):
ss_aipe_equivalence_r(true_cor = 0.30, width = 0.20, assurance = 0.80,
                      seed = 113)
```

---

ss_aipe_equivalence_r_sensitivity

Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for an Equivalence-Test Correlation

---

Description

Quantifies how much misspecification of the population correlation distorts an AIPE-based sample size plan for the two-one-sided-tests (TOST) confidence interval on the Pearson correlation. On each replication the function simulates N bivariate normal pairs with population correlation `true_cor`, computes the sample correlation and its Fisher's Z confidence interval via `ci_r`, and summarizes the realized widths and the proportion of replications in which the computed interval falls entirely inside (`equivalent`) the specified equivalence bounds.

Usage

```
ss_aipe_equivalence_r_sensitivity(
  true_cor = 0,
  supposed_cor = NULL,
  width,
  cor_lower = NULL,
  cor_upper = NULL,
  specified_N = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
```

```
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_cor`	Population correlation (the data generating value). Defaults to `0` (no association, the exact equivalence case).
`supposed_cor`	Planning value of the population correlation passed to `ss_aipe_equivalence_r`; supply this or `specified_N` but not both.
`width`	Desired full width of the two-sided CI on the correlation.
`cor_lower, cor_upper`	Equivalence bounds on the correlation, as positive magnitudes with the same meaning as in `equivalence_r`: the region is $(-\rho_L, +\rho_U)$, where ρ_L is `cor_lower` and ρ_U is `cor_upper`. `cor_upper` is required; `cor_lower` defaults to `cor_upper` (a symmetric region). The simulator records whether the realized CI falls entirely inside the region.
`specified_N`	Sample size to evaluate.
`conf_level`	Confidence level (default `0.95`). A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in `ci_nc_t`).
`assurance`	Optional assurance probability.
`G`	Number of Monte Carlo replications.
`print_iter`	Logical.
`filename`	Optional path for a comma separated file recording every replication (the sample correlation, the two confidence limits, the interval width, whether the interval fell inside the equivalence region, and two indicators of whether the interval missed `true_cor` below or above): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`.
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Value

A `data.frame` with rows for mean / median / SD of the realized correlation and CI width, the proportion of intervals at or below `width`, tail-specific and overall non-coverage of `true_cor`, the proportion of intervals classified as `equivalent` (CI fully inside the bounds), and the input echoes, including `assurance` (present only when an assurance was supplied).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals

that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted` − `returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Counsell, A., & Cribbie, R. A. (2015). Equivalence tests for comparing correlation and regression coefficients. *British Journal of Mathematical and Statistical Psychology, 68*(2), 292–309. doi:10.1111/bmsp.12045

Goertzen, J. R., & Cribbie, R. A. (2010). Detecting a lack of association: An equivalence testing approach. *British Journal of Mathematical and Statistical Psychology, 63*(3), 527–537. doi:10.1348/000711009X475853

See Also

`ss_aipe_equivalence_r`, `equivalence_r`, `ss_aipe_r_sensitivity`, `ss_aipe_equivalence_smd_sensitivity`

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`, `ss_aipe_icc()`, `ss_aipe_icc_sensitivi`
`ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivity()`, `ss_aipe_mean()`,
`ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared()`,
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r()`, `ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`,
`ss_aipe_two_phase_cost()`, `ss_aipe_two_phase_sensitivity()`

Examples

```
# Reduced Monte Carlo sweep (small G) for a fast, illustrative run.
set.seed(113)
ss_aipe_equivalence_r_sensitivity(
  true_cor     = 0.0,
  supposed_cor = 0.0,
  width        = 0.30,
  cor_upper    = 0.20,
  G = 50, print_iter = FALSE
)
```

`ss_aipe_equivalence_smd` *AIPE Sample Size Planning for an Equivalence Test on the Standardized Mean Difference*

Description

Computes the smallest per-group sample size at which the confidence interval that `equivalence_smd` reports, the 100(1 - 2α)% interval on the standardized mean difference whose position against the equivalence bounds decides the two one-sided tests (Lakens, 2017), is no wider than `width` when the sample standardized mean difference equals the planning value `true_smd` (accuracy in parameter estimation, AIPE; Kelley & Rausch, 2006). With the conventional $\alpha = .05$ the interval is the 90% interval. With `assurance` the function returns the smallest per-group sample size at which the realized interval is no wider than `width` with at least that probability.

Usage

```
ss_aipe_equivalence_smd(
  true_smd = 0,
  width,
  alpha_level = 0.05,
  assurance = NULL,
  balanced = TRUE
)
```

Arguments

`true_smd`	Planning value of the population standardized mean difference δ, a single finite number. Default `0`. The interval is narrowest when the sample standardized mean difference is 0, so for a given width the plan at 0 is the smallest over planning values.
`width`	Target full width of the interval on the standardized mean difference scale (e.g., `0.20` for a 90% interval of width 0.20).

`alpha_level` One-sided significance level of each of the two tests. The interval planned for is at confidence level $1 - 2\alpha$, as in `equivalence_smd`. Default `0.05` (90% interval).

`assurance` Optional probability in $(0, 1)$ that the realized interval is no wider than `width`. The function returns the smallest per-group sample size, at or above the one planned without it, at which that probability is at least `assurance` (Kelley, Maxwell, & Rausch, 2003; Kelley & Rausch, 2006). Default `NULL` (the plan is for the width at `true_smd` only).

`balanced` Logical; `TRUE` (default) plans equal-n groups. Unequal-n planning is not yet supported.

Details

The interval planned for. `equivalence_smd` reports the interval

$$\hat{d} \pm t_{1-\alpha,\, n_1+n_2-2} \sqrt{\frac{n_1 + n_2}{n_1 n_2} + \frac{\hat{d}^2}{2(n_1 + n_2)}},$$

a large-sample standard error of $\hat{d}$ with a critical value from Student's *t* distribution on $n_1 + n_2 - 2$ degrees of freedom, and it declares equivalence when that interval lies inside the bounds. This function plans for the same interval, because it is the interval the analysis reports; a plan for any other interval would describe a width the analysis does not show. With n in each group the width is

$$w(\hat{d}, n) \;=\; t_{1-\alpha,\, 2n-2} \sqrt{(8 + \hat{d}^2)/n}.$$

The plan is the smallest n of at least 2 with $w(\delta, n) \le \omega$, where δ is `true_smd` and ω is `width`. Because w decreases in n, the interval with $n - 1$ in each group is wider than ω. The search starts at $z^2_{1-\alpha}(8 + \delta^2)/\omega^2$, the answer with the standard normal quantile in place of the *t* quantile, which is not above the answer because the *t* quantile is the larger.

Assurance. Without `assurance` the plan makes the interval no wider than the target when the sample standardized mean difference equals `true_smd`; a sample with a larger $|\hat{d}|$ gives a wider interval. With `assurance` the function finds the sample size at which the realized interval is no wider than the target with at least that probability, from the sampling distribution of $\hat{d}$, as Kelley and Rausch (2006, pp. 372–373) do for the noncentral *t* interval:

1. At a given n the width depends on the sample only through $\hat{d}$ and increases with $|\hat{d}|$, so the interval is no wider than ω exactly when $|\hat{d}| \le c(n)$, where $c(n)^2 = n\omega^2/t^2_{1-\alpha,\, 2n-2} - 8$. When $c(n)^2 \le 0$, even the interval at $\hat{d} = 0$ is wider than ω and the probability is 0.
2. For two independent samples from normal populations with a common variance, $T = \hat{d}\sqrt{n/2}$ is the two-group *t* statistic, distributed as noncentral *t* on $\nu = 2n - 2$ degrees of freedom with noncentrality $\lambda = \delta\sqrt{n/2}$. The probability is $P_n = P(|T| \le c(n)\sqrt{n/2})$, the form of Kelley and Rausch (2006, Equation 20). With $T = (Z + \lambda)/\sqrt{V/\nu}$, where Z is standard normal and V is chi square on ν degrees of freedom, independent of Z,

$$P(|T| \le k) \;=\; E_V\Big[\Phi\big(k\sqrt{V/\nu} - \lambda\big) - \Phi\big(-k\sqrt{V/\nu} - \lambda\big)\Big],$$

which is computed by numerical integration over V. Only the normal distribution function and the chi square density enter, because R's noncentral *t* distribution function is an approximation at noncentrality values above 37.62, which plans for narrow intervals reach.

3. The function returns the smallest n, at or above the sample size planned without `assurance`, with $P_n \geq$ `assurance`, found by doubling a step upward and then bisecting. `assurance_attained` is P_n at that n.

The probability is exact for the normal model, up to numerical integration. Nothing is simulated, so the function draws no random numbers and returns the same plan on every call. The probability agreed with R's noncentral *t* distribution function to within 10^{-10} wherever that function is accurate, and with simulation from normal scores: in 24 planning conditions of 20,000 simulated studies each, the proportion of intervals no wider than the target at the planned size differed from P_n by at most 2.0 Monte Carlo standard errors.

Three properties of the step are worth knowing.

- The answer is never below the sample size planned without `assurance`, and an `assurance` at or below the probability at that size returns it unchanged. At or above that size P_n did not decrease in any of 48 planning conditions scanned over every whole n (6,639 sizes in all).
- At `true_smd = 0` the plan without `assurance` often meets a moderate request already. The probability at that size depends on how far rounding up to a whole number narrows the interval: over widths from 0.05 to 3 at $\alpha = .05$ it ranged from .11 to .97, with a median of .85, and for a width of 0.50 it is .839 with 88 per group.
- As $|\delta|$ grows, the probability at the plan without `assurance` falls toward one half, because $|\hat{d}|$ exceeds $|\delta|$ about half the time, and the step adds more cases, as it does when the target narrows: for `true_smd = 0.5`, a width of 0.30, and `assurance = 0.99` the plan moves from 249 to 257 per group, and for `true_smd = 1`, a width of 0.20, and `assurance = 0.90` from 610 to 621.

The noncentral t interval. Kelley and Rausch (2006) plan for the interval from the noncentral *t* distribution, which `ci_smd` computes (Kelley, 2007) and `ss_aipe_smd` plans for; they use the interval with the standard normal quantile only as the starting value of their search (pp. 371–372). The noncentral *t* interval has exact coverage for the normal model, where the interval of `equivalence_smd` is a large-sample approximation. At the same $\hat{d}$, n, and level the interval of `equivalence_smd()` is usually the wider of the two: for 90% intervals with $|\hat{d}|$ up to 3, by 18 to 30 percent with 3 per group and by less than 1 percent with 100 per group. At the 80 percent level it can be the narrower, by less than 1 percent, when $|\hat{d}|$ is 5 or more (`equivalence_smd` says where the comparison was computed). A study that will report `ci_smd()` at level $1 - 2\alpha$ is planned with `ss_aipe_smd(conf_level = 1 - 2 * alpha_level)`.

Value

A `data.frame` with columns `term` and `value` and one row per term:

`necessary_n_per_group` the planned sample size of each group.

`total_N` twice `necessary_n_per_group`.

`width` the target width, as given.

`true_smd` the planning value.

`expected_width` the full width of the interval `equivalence_smd` reports with `necessary_n_per_group` in each group when the sample standardized mean difference equals `true_smd`, that is, `ci_upper` minus `ci_lower` of `equivalence_smd(smd = true_smd, n_1 = n, n_2 = n, alpha_level = alpha_level)` with `n` the planned size. It is no larger than `width`. Without `assurance`, the

same interval with one fewer case in each group is wider than `width`, unless the plan is the smallest size, 2 per group.

`assurance` the requested probability; present only when `assurance` was given.

`assurance_attained` the probability, with `necessary_n_per_group` in each group and population standardized mean difference `true_smd`, that the realized interval is no wider than `width`; present only when `assurance` was given. It is computed, not simulated (see Details), and it is at least `assurance`.

The `conf_level` attribute records the confidence level of the interval, $1 - 2\alpha$.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Lakens, D. (2017). Equivalence tests: A practical primer for *t* tests, correlations, and meta-analyses. *Social Psychological and Personality Science, 8*(4), 355–362. doi:10.1177/1948550617697177

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

See Also

`equivalence_smd`, the analysis planned for; `ss_aipe_equivalence_smd_sensitivity`, which simulates the plan; `ss_aipe_smd` and `ci_smd`, for the noncentral *t* interval.

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd_sensitivity()`, `ss_aipe_icc()`,
`ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivity()`,
`ss_aipe_mean()`, `ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared`
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r()`, `ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`,
`ss_aipe_two_phase_cost()`, `ss_aipe_two_phase_sensitivity()`

Examples

```
# 1. Plan for a 90% interval on the standardized mean difference of
#    width 0.20, with a planning value of 0.
ss_aipe_equivalence_smd(true_smd = 0, width = 0.20)

# 2. The analysis at the planned size, when the sample standardized mean
#    difference equals the planning value, reports an interval no wider
#    than the target (expected_width is its width).
plan <- ss_aipe_equivalence_smd(true_smd = 0.05, width = 0.20)
n <- plan$value[plan$term == "necessary_n_per_group"]
fit <- equivalence_smd(smd = 0.05, n_1 = n, n_2 = n, smd_upper = 0.3)
fit$value[fit$term == "ci_upper"] - fit$value[fit$term == "ci_lower"]

# 3. With 80% assurance that the realized interval is no wider than 0.20.
#    The last row is the probability at the returned size.
ss_aipe_equivalence_smd(true_smd = 0.05, width = 0.20, assurance = 0.80)
```

`ss_aipe_equivalence_smd_sensitivity` *Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for an Equivalence-Test SMD*

Description

Quantifies how much misspecification of the population standardized mean difference distorts an AIPE-based sample size plan for the two-one-sided-tests (TOST) confidence interval on the standardized mean difference. On each replication the function simulates two normal groups of size n per group with population standardized mean difference `true_smd`, computes the sample standardized mean difference and the interval `equivalence_smd` reports for it, and summarizes the realized widths, the proportion of replications in which the interval falls entirely inside the equivalence bounds (`equivalent`), and the proportions that miss `true_smd`.

Usage

```
ss_aipe_equivalence_smd_sensitivity(
  true_smd = 0,
  supposed_smd = NULL,
  width,
  smd_lower = NULL,
  smd_upper = NULL,
  specified_n_per_group = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

true_smd	Population standardized mean difference (the data generating value), a single finite number. Defaults to 0 (perfect equivalence).
supposed_smd	The supposed population SMD, the planning value passed to ss_aipe_equivalence_smd (not an estimate computed from data), a single finite number; supply this or specified_n_per_group but not both.
width	Desired full width of the two-sided CI on the SMD.
smd_lower, smd_upper	Equivalence bounds on the SMD, as positive magnitudes with the same meaning as in equivalence_smd: the region is $(-\delta_L, +\delta_U)$, with smd_lower as δ_L and smd_upper as δ_U. smd_upper is required; smd_lower defaults to smd_upper (a symmetric region). The simulator records whether the realized CI falls entirely inside the region.
specified_n_per_group	Per-group sample size to evaluate, a whole number of at least 2.
conf_level	Confidence level of the interval (default 0.95); the two one-sided tests it decides are at level (1 - conf_level) / 2.
assurance	Optional assurance probability, passed to ss_aipe_equivalence_smd when the sample size is planned from supposed_smd.
G	Number of Monte Carlo replications.
print_iter	Logical.
filename	Optional path for a comma separated file recording every replication (the sample standardized mean difference, the two confidence limits, the interval width, whether the interval fell inside the equivalence region, and two indicators of whether the interval missed true_smd below or above): nothing is written when filename is NULL (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at tempfile(fileext = ".csv").
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

The interval simulated. The interval is the one equivalence_smd reports and ss_aipe_equivalence_smd plans for,

$$\hat{d} \pm t_{1-\alpha,\,2n-2}\sqrt{(8+\hat{d}^2)/(4n)},$$

a large-sample standard error of $\hat{d}$ with a Student t critical value, at confidence level conf_level, so that α is (1 - conf_level) / 2: the default 0.95 is the interval of the two one-sided tests at $\alpha = .025$, and conf_level = 0.90 is the interval of the tests at .05. A sample size from supposed_smd is planned by ss_aipe_equivalence_smd(alpha_level = (1 - conf_level) / 2), so the plan, the analysis, and the simulation describe one interval at one level: a simulated sample with standardized mean difference $\hat{d}$ gives the limits equivalence_smd(smd = d, n_1 = n, n_2 = n,alpha_level = (1 - conf_level) / 2) reports. When true_smd equals supposed_smd, pct_ci_less_width estimates the probability that the interval at the planned size is no wider than width, which the planner

computes and reports as `assurance_attained` when `assurance` is given; the `simulation` attribute (below) gives the Monte Carlo limits of the estimate.

The interval is a large-sample approximation, so its coverage of `true_smd` (the `pct_ci_miss_low`, `pct_ci_miss_high`, and `total_type_I_error` rows) can differ from 1 - `conf_level`, most in small samples; the noncentral *t* interval of `ci_smd`, which `ss_aipe_smd_sensitivity` simulates, has exact coverage for the normal model.

Value

A `data.frame` with rows for mean / median / SD of the realized SMD and CI width, the proportion of intervals at or below `width`, tail-specific and overall non-coverage of `true_smd`, the proportion of intervals classified as `equivalent` (CI fully inside the bounds), and the input echoes, including `assurance` (present only when an assurance was supplied).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted` - `returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*, 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ss_aipe_equivalence_smd, equivalence_smd, ss_aipe_smd_sensitivity

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_icc(), ss_aipe_icc_sensitivity(
ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivity(), ss_aipe_mean(),
ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(),
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(),
ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(),
ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(),
ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# Reduced Monte Carlo sweep (small G) for a fast, illustrative run.
set.seed(113)
ss_aipe_equivalence_smd_sensitivity(
  true_smd      = 0.0,
  supposed_smd  = 0.0,
  width         = 0.30,
  smd_upper     = 0.20,
  G = 50, print_iter = FALSE
)
```

---

ss_aipe_icc *Sample Size for AIPE on an Intraclass Correlation Coefficient*

Description

Determines the sample size needed for a confidence interval on a population intraclass correlation coefficient (ICC) to have a desired width, using Bonett's (2002) Fisher-style variance-stabilizing transformation. The function inverts the asymptotic variance on the transformed scale (where the CI is symmetric and approximately normal), solves for the smallest N that achieves the target half-width on the back-transformed (raw-ICC) scale, and optionally inflates the result by a chi squared assurance correction (Kelley & Maxwell, 2003).

Usage

```
ss_aipe_icc(
  true_icc,
  n_raters,
  width,
  which_width = c("Full", "Lower", "Upper"),
  conf_level = 0.95,
 type = c("ICC(1,1)", "ICC(2,1)", "ICC(3,1)", "ICC(1,k)", "ICC(2,k)", "ICC(3,k)"),
  assurance = NULL
)
```

Arguments

`true_icc`	Anticipated population ICC (ρ in the formulas) at the level matching `type`, in $[0, 1)$. This is the value the researcher expects the truth to be near, motivated by prior literature, a pilot, or substantive theory. Because the planning formula inverts the asymptotic variance *at* this value, a wrong guess inflates or deflates the realized confidence interval width relative to `width`; the function `ss_aipe_icc_sensitivity` quantifies the impact of misspecification by Monte Carlo.
`n_raters`	Number of raters (or measurements per subject), k in the formulas and in the names of the ICC forms; must be at least 2.
`width`	Desired full width of the back-transformed CI on the ICC.
`which_width`	Whether `width` is the `"Full"` width of the interval (default) or a half-width: `"Lower"` and `"Upper"` both interpret `width` as half the full width, so they plan for a full width of twice `width` and return the same sample size. Because the interval is not generally symmetric about the estimate, its realized lower and upper half-widths can differ from each other and from half the full width; the planner does not target them separately. A genuinely one-sided width target is not currently offered.
`conf_level`	Desired confidence level (default `0.95`).
`type`	Which Shrout-Fleiss (1979) ICC form is being planned. One of the single-rater forms `"ICC(1,1)"`, `"ICC(2,1)"`, `"ICC(3,1)"` (which share the planning variance) or the average-of-k forms `"ICC(1,k)"`, `"ICC(2,k)"`, `"ICC(3,k)"`; default `"ICC(1,1)"`. Both `true_icc` and `width` are interpreted on the scale of the requested form: for an average-of-k form the planning value is mapped to the single-rater scale through the inverse Spearman-Brown relation and each candidate confidence limit is mapped back, so the returned sample size targets the width of the interval on the average-of-k ICC itself (see Details).
`assurance`	Optional. Probability that the realized CI is no wider than `width`; when supplied, the sample size is inflated by the standard chi squared correction.

Details

Bonett's (2002) Fisher-style transform. Bonett (2002) showed that the transformation

$$L(\rho) \;=\; \frac{1}{2}\log\!\left(\frac{1+(k-1)\rho}{1-\rho}\right)$$

approximately variance-stabilizes the single-rater ICC, with

$$\mathrm{Var}(L(\hat{\rho})) \;\approx\; \frac{k}{2\,(k-1)\,(N-2)}.$$

A confidence interval is constructed by adding $\pm z_{1-\alpha/2}$ standard errors on the L scale and back-transforming to the raw-ICC scale via $\rho = (e^{2L}-1)/(e^{2L}-1+k)$. The minimum sample size is found by searching for the smallest N whose back-transformed CI width is below the target.

Single-rater vs.\ average-of-k ICC. The Bonett (2002) variance applies directly to the single-rater forms (`ICC(1,1)`, `ICC(2,1)`, `ICC(3,1)`). For the average-of-k forms the planning value ρ_k is first mapped to the single-rater scale through the inverse Spearman-Brown relation $\rho = \rho_k/[k-(k-1)\rho_k]$, the interval is formed on the L scale as above, and each candidate limit is mapped back to the average-of-k scale (composing the inverse L transform with the Spearman-Brown formula reduces to $\rho_k = 1 - e^{-2L}$). The width criterion therefore applies to the confidence interval on the average-of-k ICC itself, following the convention used by `var_icc`. Because the two scales differ, an average-of-k plan generally recommends a different sample size than a single-rater plan at the same numeric `true_icc` and `width`; at `true_icc = 0.7`, `n_raters = 3`, and `width = 0.20`, planning for `ICC(1,1)` recommends $N = 69$ subjects while planning for `ICC(1,k)` recommends $N = 110$.

The assurance correction can fall slightly short. Monte Carlo evaluation with `ss_aipe_icc_sensitivity` shows that the chi squared inflation tends to deliver a little less assurance than requested. At the condition of the second example below (`true_icc = 0.7`, `n_raters = 3`, `width = 0.20`, `assurance = 0.80`), the recommended $N = 79$ yields an empirical assurance of about .77 against the requested .80 (10,000 replications of the F-based interval computed by `icc`), and the smallest sample size whose empirical assurance reaches .80 is $N = 81$. The mechanism is that the realized interval widths on the raw-ICC scale have a heavier upper tail than the chi squared inflation on the transformed scale accounts for, so the buffer the correction adds is slightly too small. When meeting the assurance target matters, check the recommended sample size with `ss_aipe_icc_sensitivity` and increase N until the empirical assurance reaches the target.

Value

A `data.frame` with rows for the recommended sample size (*number of subjects*), the expected back-transformed CI width, and the inputs echoed back: the rows are `necessary_N`, `expected_width`, `true_icc`, `n_raters`, `width`, and `conf_level`. The Shrout-Fleiss form the plan targets is stored as the `"icc_type"` attribute so the `value` column stays numeric.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G. (2002). Sample size requirements for estimating intraclass correlations with desired precision. *Statistics in Medicine, 21*(9), 1331–1335. doi:10.1002/sim.1108

Donner, A. (1986). A review of inference procedures for the intraclass correlation coefficient in the one-way random effects model. *International Statistical Review, 54*(1), 67–82.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Shrout, P. E., & Fleiss, J. L. (1979). Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin, 86*(2), 420–428.

Smith, C. A. B. (1956). On the estimation of intraclass correlation. *Annals of Human Genetics, 21*(4), 363–373.

See Also

icc, var_icc

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(),
ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivity(),
ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(),
ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(),
ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(),
ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# 1. Plan N so the 95% CI on a single-rater ICC has full width <= 0.20
#        with 3 raters and an anticipated ICC of 0.7.
ss_aipe_icc(true_icc = 0.7, n_raters = 3, width = 0.20)

# 2. With 80% assurance:
ss_aipe_icc(true_icc = 0.7, n_raters = 3, width = 0.20, assurance = 0.80)
```

---

ss_aipe_icc_sensitivity

Sensitivity Analysis for Sample Size Planning From the Accuracy in Parameter Estimation Perspective for an Intraclass Correlation Coefficient

---

Description

Quantifies how much misspecification of the population ICC can distort an AIPE-based sample size plan. Given a true (population) ICC and the value used in planning, the function simulates draws of size $n \times k$ from the relevant variance-components model, computes the ICC and its F-distribution confidence interval on each replication via icc, and summarizes how often the realized interval width is below the desired target and how often the interval covers the population value. This is the standard sensitivity-analysis workflow described in Kelley (2007) and Maxwell, Delaney, and Kelley (2027, Section 3.11 on sample size planning).

Usage

```
ss_aipe_icc_sensitivity(
  true_icc = NULL,
  supposed_icc = NULL,
  n_raters,
  width,
  assurance = NULL,
  specified_N = NULL,
  conf_level = 0.95,
 type = c("ICC(1,1)", "ICC(2,1)", "ICC(3,1)", "ICC(1,k)", "ICC(2,k)", "ICC(3,k)"),
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_icc`	Population intraclass correlation coefficient (the data generating value), at the level matching `type`. Must lie in $[0, 1)$.
`supposed_icc`	The supposed ICC, the planning value from which `ss_aipe_icc` plans the study (a value the researcher supposes, not an estimate computed from data). Supply this or `specified_N` but not both.
`n_raters`	Number of raters (or repeated measurements) per subject; must be at least 2.
`width`	Desired full width of the (back-transformed) confidence interval on the ICC.
`assurance`	Probability with which the realized interval should be no wider than `width` (must be `NULL` or strictly between 0 and 1; `NULL` means plan to the *expected* width without an assurance constraint).
`specified_N`	Pre-specified number of subjects to evaluate (use this when you want the sensitivity results at a fixed n rather than at the n that `ss_aipe_icc` would recommend); incompatible with `supposed_icc`.
`conf_level`	Desired confidence level (i.e., 1 minus the Type I error rate); default 0.95.
`type`	Which Shrout-Fleiss (1979) ICC form is being planned. One of `"ICC(1,1)"` (default; one-way random model), `"ICC(2,1)"` (two-way random model, absolute agreement), `"ICC(3,1)"` (two-way mixed model, consistency), or the average-of-k versions `"ICC(1,k)"`, `"ICC(2,k)"`, `"ICC(3,k)"`. The simulation generates data from the variance- components model matching the requested type, so the realized ICC on each replication is comparable to `true_icc`.
`G`	Number of Monte Carlo replications; defaults to 1000. Increase (e.g., 5000 or 10000) for stable empirical-coverage estimates.
`print_iter`	Logical. If `TRUE` the simulation prints the iteration index after each replication (helpful for long runs); default `FALSE`.
`filename`	An optional path for a comma separated file recording every replication (the realized ICC, the two confidence limits, the interval width, and the two tail-specific non-coverage indicators): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that

path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`.

`seed` `NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Sample size planning for the intraclass correlation coefficient under the Accuracy in Parameter Estimation framework chooses *n* so that the expected (or, with assurance, the high-probability) confidence interval width is no larger than `width` (Bonett, 2002; Kelley & Maxwell, 2003). Because the procedure assumes the planning value `supposed_icc` matches the population value `true_icc`, in practice the realized width will deviate from the planned width whenever the planning value is wrong. This sensitivity analysis quantifies the deviation by Monte Carlo simulation: the planned *n* is obtained from `ss_aipe_icc` with `supposed_icc`, then samples are drawn from the *true* population (with population ICC `true_icc`) and the realized confidence interval widths are summarized.

Data generating model. For `type` starting with `"ICC(1,"` the simulation uses the one-way random model: each row (subject) gets a subject random effect, every cell adds independent Gaussian noise, and the population ICC equals $\sigma^2_{\text{subj}}/(\sigma^2_{\text{subj}}+\sigma^2_{\text{err}})$. For `type` starting with `"ICC(2,"` the simulation also adds a rater random effect, so the population `ICC(2,1)` equals $\sigma^2_{\text{subj}}/(\sigma^2_{\text{subj}}+\sigma^2_{\text{rater}}+\sigma^2_{\text{err}})$. For `type` starting with `"ICC(3,"` the rater effect is fixed (centered constants), so the population `ICC(3,1)` equals $\sigma^2_{\text{subj}}/(\sigma^2_{\text{subj}}+\sigma^2_{\text{err}})$ but the two-way decomposition is used in the estimator. Within each cell, the total variance is unity by construction. For the single-rater forms `true_icc` therefore maps directly to the subject-variance share; for the average-of-k forms `true_icc` is first mapped to the single-rater scale through the inverse Spearman-Brown relation $\rho = \rho_k/[k-(k-1)\rho_k]$, so that the population ICC at the average-of-k level equals `true_icc`. Coverage is checked against `true_icc` on its own scale, matching the scale on which `icc` reports each form.

Value

A `data.frame` with columns `term` and `value` summarizing the Monte Carlo results across the `G` replications. The `term` entries are: `"mean_icc"`, `"median_icc"`, `"sd_icc"` (mean / median / SD of the *G* observed ICC estimates); `"mean_ci_width"`, `"median_ci_width"`, `"sd_ci_width"` (corresponding summaries of the realized interval widths); `"pct_ci_less_width"` (proportion of intervals at or below the planning width `width`); `"pct_ci_miss_low"` and `"pct_ci_miss_high"` (tail-specific non-coverage of `true_icc`); `"total_type_I_error"` (overall empirical non-coverage of `true_icc`); plus the input echoes `"total_N"`, `"n_raters"`, `"true_icc"`, `"supposed_icc"`, `"width"`, `"conf_level"`, and `"assurance"` (present only when an assurance was supplied). The ICC type is not a row; it is stored as the `"icc_type"` attribute on the returned object so the `value` column stays numeric.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G. (2002). Sample size requirements for estimating intraclass correlations with desired precision. *Statistics in Medicine, 21*(9), 1331–1335. doi:10.1002/sim.1108

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapters 10 and 11 on intraclass correlation and reliability.)

Shrout, P. E., & Fleiss, J. L. (1979). Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin, 86*(2), 420–428.

See Also

`ss_aipe_icc`, `icc`, `var_icc`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivity()`, `ss_aipe_mean()`,
`ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared()`,
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,

ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# Reduced G and a wide target width keep this Monte Carlo example fast;
# raise G (e.g., 1000 or more) for stable empirical-coverage estimates.

# Well-specified case: plan with the true population ICC.
set.seed(113)
ss_aipe_icc_sensitivity(
  true_icc      = 0.70,
  supposed_icc  = 0.70,
  n_raters      = 3,
  width         = 0.40,
  conf_level    = 0.95,
  G             = 25,
  print_iter    = FALSE
)

# Misspecified case: the planner used .70 but the truth is .50.
# The realized interval widths will tend to be wider than the target.
set.seed(113)
ss_aipe_icc_sensitivity(
  true_icc      = 0.50,
  supposed_icc  = 0.70,
  n_raters      = 3,
  width         = 0.40,
  conf_level    = 0.95,
  G             = 25,
  print_iter    = FALSE
)

# Fixed-n mode: skip the planner and evaluate at a chosen sample size.
set.seed(113)
ss_aipe_icc_sensitivity(
  true_icc    = 0.50,
  specified_N = 40,
  n_raters    = 3,
  width       = 0.40,
  G           = 25,
  print_iter  = FALSE
)
```

---

ss_aipe_indirect_effect

Sample Size for AIPE on a Mediated (Indirect) Effect ab

Description

Determines the sample size needed for the confidence interval on a mediated effect ab (the product of the $X \rightarrow M$ and $M \rightarrow Y$ coefficients in a simple three-variable mediation model) to have a desired full width. Two methods are available. `"closed_form"` (the default) plans for the symmetric Wald interval built on the delta method standard error of the product (Sobel, 1982) and answers instantly. `"monte_carlo"` plans for the Monte Carlo confidence interval (MacKinnon, Lockwood, & Williams, 2004; Tofighi & MacKinnon, 2011), the interval `mediation_mbco` reports under `ci_method = "monte_carlo"`, and it plans by a priori Monte Carlo simulation: at each candidate sample size the mediation model is fit to `G` simulated data sets and the realized interval widths are recorded. Because the Monte Carlo interval respects the skewness of the sampling distribution of a product, and because the simulation measures the widths that fitted models actually deliver, `method = "monte_carlo"` is the recommended way to settle on the final sample size; the closed form is its fast first approximation.

Usage

```
ss_aipe_indirect_effect(
  true_std_reg_coef_a,
  true_std_reg_coef_b,
  width,
  method = c("closed_form", "monte_carlo"),
  conf_level = 0.95,
  max_N = 10000L,
  B = 5000L,
  G = 1000L,
  seed = NULL
)
```

Arguments

`true_std_reg_coef_a`
: Anticipated population standardized regression coefficient of path a, $X \rightarrow M$. Numeric scalar in $(-1, 1)$.

`true_std_reg_coef_b`
: Anticipated population standardized regression coefficient of path b, $M \rightarrow Y$ controlling for X. Numeric scalar in $(-1, 1)$.

`width`
: Desired full width of the confidence interval on ab.

`method`
: One of `"closed_form"` (default) or `"monte_carlo"`; see Details.

`conf_level`
: Desired confidence level (default `0.95`).

`max_N`
: Upper bound on the search over the total sample size; default `10000`.

`B`
: Number of Monte Carlo draws forming the interval within each simulated study when `method = "monte_carlo"`; default `5000`. This is the same `B` the analysis-stage interval uses (see `mediation_mbco`), and it follows the same rule: at least 10 draws beyond each limit, which is 400 at the 95 percent level (see `dmar_bootstrap`). A smaller value is refused unless `options(dmar.bootstrap_exploratory = TRUE)` is set. Not used by `method = "closed_form"`.

`G` Number of simulated studies per candidate sample size when `method = "monte_carlo"`; default `1000`. The mean simulated width carries a simulation error of about its standard deviation over $\sqrt{G}$; raise `G` for a sharper answer.

`seed` Optional integer seed for the Monte Carlo method, used locally (the caller's random number generator state is restored on exit). Default `NULL` draws the Monte Carlo replications from the current random number stream, which advances that stream.

Details

The mediation model. The simple mediator model is

$$M = \alpha_1 + aX + \varepsilon_M,$$

$$Y = \alpha_2 + c'X + bM + \varepsilon_Y,$$

with the indirect (mediated) effect of X on Y through M equal to ab (MacKinnon, Lockwood, Hoffman, West, & Sheets, 2002). Both methods plan on the standardized scale with no direct effect: the planning population takes X, M, and Y with unit variances and $c' = 0$, so `true_std_reg_coef_a` and `true_std_reg_coef_b` are the standardized paths a and b.

The closed form. Under the planning population the sampling variance of $\hat{a}$ is $(1 - a^2)/(N - 2)$. In the equation for Y the mediator is regressed alongside X, with which it is correlated at a, so the sampling variance of $\hat{b}$ carries the variance inflation factor $1/(1 - a^2)$:

$$\mathrm{Var}(\hat{b}) \;=\; \frac{1 - b^2}{(N - 3)(1 - a^2)}.$$

The estimators come from two separate equations and are uncorrelated, so the delta method (Sobel, 1982) standard error of the product is

$$\mathrm{SE}(\hat{a}\hat{b}) \;=\; \sqrt{a^2\mathrm{Var}(\hat{b}) + b^2\mathrm{Var}(\hat{a})}\,,$$

and the closed form returns the smallest N at which the Wald width $2z_{1-\alpha/2}\,\mathrm{SE}(\hat{a}\hat{b})$ is at or below `width`. Two approximations remain. The Wald interval is symmetric while the sampling distribution of a product is skewed, so the Wald interval is not the interval an indirect effect should be reported with (Tofighi & Kelley, 2020). And the closed form evaluates the standard error at the planning values, while a fitted model evaluates it at the estimates, which leaves a discrepancy of a percent or two in realized width at moderate sample sizes. Both are reasons to treat the closed form as the first approximation and to verify the final plan with `method = "monte_carlo"`, which measures the realized widths directly.

Planning for the Monte Carlo interval. With `method = "monte_carlo"`, each candidate N is evaluated by a priori Monte Carlo simulation (Muthén & Muthén, 2002; Schoemann, Boulton, & Short, 2017): `G` data sets of size N are drawn from the planning population, the two mediation regressions are fit to each, and the Monte Carlo interval is formed by drawing `B` pairs $(\tilde{a}, \tilde{b})$ from normal distributions centered at the estimates with the estimated standard errors, multiplying, and reading off the empirical $(\alpha/2, 1 - \alpha/2)$ quantiles (MacKinnon, Lockwood, & Williams, 2004). Since $\hat{a}$ and $\hat{b}$ are uncorrelated here, the independent draws realize the joint normal approximation of the estimates, the same construction `mediation_mbco` uses for its Monte Carlo interval. The necessary sample size is the smallest N whose mean simulated width is at or below `width`; the

search starts from the closed-form answer, brackets the crossing geometrically, and bisects. A planning call at the default G and B fits the mediation model several thousand times and takes a few seconds; the Monte Carlo example below lowers both to keep the page quick. The necessary sample size inherits the simulation error of the mean widths; raising G narrows it, and supplying seed makes a plan reproducible.

Relation to the MBCO procedure. The model-based constrained optimization (MBCO) likelihood ratio test of Tofighi and Kelley (2020), implemented in mediation_mbco, is the test they advocate for a hypothesis about an indirect effect, alongside a confidence interval, and the intervals that suit an indirect effect are the profile likelihood interval and the Monte Carlo interval, both of which accommodate the skewness of the product. This planner targets the Monte Carlo interval. Planning for the profile likelihood interval would require inverting a pair of constrained optimizations in every simulated study (two constrained **OpenMx** fits per interval, times G, times every candidate sample size), while the Monte Carlo interval costs B products of normal draws per study and is the inexpensive interval that also accommodates the skewness, the one Tofighi and Kelley (2020) report for their memory example. A study planned with method = "monte_carlo" and analyzed with mediation_mbco(ci_method = "monte_carlo") is therefore planned and analyzed on the same interval.

Beyond the simple model. The planning population assumes standardized observed variables, one mediator, no covariates, and no direct effect. With a nonzero direct effect the residual variance of Y is $1 - b^2 - c'^2 - 2abc'$ rather than $1 - b^2$, so assuming $c' = 0$ errs toward a larger sample whenever $c'(c' + 2ab) > 0$ (consistent mediation) and toward a smaller one otherwise. When the direct effect, covariates, several mediators, or latent variables matter to the design, plan by simulation from the full model with ss_aipe_composite_sem, labeling the paths and defining the indirect effect via ab := a*b; its intervals are the Wald intervals of the fitted model, the same target as the closed form here. ss_aipe_indirect_effect_sensitivity quantifies what a plan from this page delivers when the population paths differ from the planning values.

Value

A data.frame with rows for the recommended sample size (necessary_N), the expected CI width at that size (expected_width), and the inputs echoed back: the planning values true_std_reg_coef_a and true_std_reg_coef_b, their product true_std_indirect_effect (the standardized indirect effect ab), width, and conf_level. Under method = "closed_form" the expected width is the delta method width evaluated at the returned sample size; under method = "monte_carlo" it is the mean simulated width there. The method is carried on the returned object as the ci_method attribute. Under method = "monte_carlo" the result also carries the attributes B, B_used, n_dropped, B_required, and bootstrap_status of dmar_bootstrap; they record the request each simulated study ran (B_used equals B), and bootstrap_status is "exploratory" when B is below B_required.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Fritz, M. S., & MacKinnon, D. P. (2007). Required sample size to detect the mediated effect. *Psychological Science, 18*(3), 233–239. doi:10.1111/j.14679280.2007.01882.x

Lachowicz, M. J., Preacher, K. J., & Kelley, K. (2018). A novel measure of effect size for mediation analysis. *Psychological Methods, 23*(2), 244–261. doi:10.1037/met0000165

MacKinnon, D. P., Lockwood, C. M., Hoffman, J. M., West, S. G., & Sheets, V. (2002). A comparison of methods to test mediation and other intervening variable effects. *Psychological Methods, 7*(1), 83–104. doi:10.1037/1082989X.7.1.83

MacKinnon, D. P., Lockwood, C. M., & Williams, J. (2004). Confidence limits for the indirect effect: Distribution of the product and resampling methods. *Multivariate Behavioral Research, 39*(1), 99–128. doi:10.1207/s15327906mbr3901_4

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Muthén, L. K., & Muthén, B. O. (2002). How to use a Monte Carlo study to decide on sample size and determine power. *Structural Equation Modeling, 9*(4), 599–620. doi:10.1207/S15328007SEM0904_8

Preacher, K. J., & Kelley, K. (2011). Effect size measures for mediation models: Quantitative strategies for communicating indirect effects. *Psychological Methods, 16*(2), 93–115. doi:10.1037/a0022658

Schoemann, A. M., Boulton, A. J., & Short, S. D. (2017). Determining power and sample size for simple and complex mediation models. *Social Psychological and Personality Science, 8*(4), 379–386. doi:10.1177/1948550617715068

Sobel, M. E. (1982). Asymptotic confidence intervals for indirect effects in structural equation models. *Sociological Methodology, 13*, 290–312.

Tofighi, D., & Kelley, K. (2020). Improved inference in mediation analysis: Introducing the model-based constrained optimization procedure. *Psychological Methods, 25*(4), 496–515. doi:10.1037/met0000259

Tofighi, D., & MacKinnon, D. P. (2011). RMediation: An R package for mediation analysis confidence intervals. *Behavior Research Methods, 43*(3), 692–700. doi:10.3758/s134280110076x

See Also

`mediation_mbco` for the analysis the plan feeds; `ss_aipe_composite_sem` for AIPE planning of an indirect effect in an arbitrary lavaan model; `ss_aipe_indirect_effect_sensitivity`; `ss_aipe_partial_r`, `ss_aipe_semipartial_r`, `ss_aipe_rc`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect_sensitivity()`, `ss_aipe_mean()`,
`ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared()`,
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r()`, `ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`,
`ss_aipe_two_phase_cost()`, `ss_aipe_two_phase_sensitivity()`

Examples

```
# 1. Plan N so the 95% CI on ab has full width <= 0.20, with
#        anticipated standardized a = 0.40 and b = 0.40. The closed
#        form answers instantly:
ss_aipe_indirect_effect(true_std_reg_coef_a = 0.40,
                        true_std_reg_coef_b = 0.40, width = 0.20)

# 2. The recommended plan targets the Monte Carlo interval directly:
#        every candidate sample size fits the mediation model to G
#        simulated data sets and measures the realized widths. G = 100
#        and B = 1000 keep the example quick; a reported plan deserves
#        the defaults G = 1000 and B = 5000. The answer sits a little
#        above the closed form because the interval it plans for is a
#        little wider than the Wald interval:
ss_aipe_indirect_effect(true_std_reg_coef_a = 0.40,
                        true_std_reg_coef_b = 0.40, width = 0.20,
                        method = "monte_carlo", G = 100, B = 1000,
                        seed = 113)
```

`ss_aipe_indirect_effect_sensitivity` *Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for an Indirect Effect*

Description

Quantifies how much misspecification of the population mediation path coefficients a and b distorts an AIPE-based sample size plan for the indirect effect ab. On each replication the function simulates a three-variable mediation system $X \rightarrow M \rightarrow Y$ of size n with population standardized path coefficients `true_std_reg_coef_a` and `true_std_reg_coef_b`, fits the two regressions of M on X and Y on M and X, computes the sample indirect effect $\hat{a}\hat{b}$, and forms the interval the plan targeted: the symmetric Wald interval from the delta method standard error (`method = "closed_form"`) or the Monte Carlo interval (`method = "monte_carlo"`), matching `ss_aipe_indirect_effect`.

Usage

```
ss_aipe_indirect_effect_sensitivity(
  true_std_reg_coef_a = NULL,
  true_std_reg_coef_b = NULL,
  supposed_std_reg_coef_a = NULL,
  supposed_std_reg_coef_b = NULL,
  width,
  specified_N = NULL,
  method = c("closed_form", "monte_carlo"),
  conf_level = 0.95,
  B = 5000L,
  G = 1000,
```

```
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_std_reg_coef_a`	Population standardized regression coefficient of path a (from X to M); the data generating value.
`true_std_reg_coef_b`	Population standardized regression coefficient of path b (from M to Y after controlling for X); the data generating value.
`supposed_std_reg_coef_a`, `supposed_std_reg_coef_b`	The standardized path coefficients supposed when the study is planned (passed to `ss_aipe_indirect_effect` as its `true_std_reg_coef_a` and `true_std_reg_coef_b`). They are planning assumptions, not estimates from data, and may differ from the data generating values. Supply both or neither (if neither, supply `specified_N`).
`width`	Desired full width of the CI on ab.
`specified_N`	Sample size to evaluate (incompatible with `supposed_std_reg_coef_a` and `supposed_std_reg_coef_b`).
`method`	One of `"closed_form"` (default) or `"monte_carlo"`; the interval computed on each replication, also forwarded to the planner when the sample size is planned from `supposed_std_reg_coef_a` and `supposed_std_reg_coef_b`. A planning call with `method = "monte_carlo"` runs the planner's a priori Monte Carlo search at its default `G`, so it takes a few seconds.
`conf_level`	Confidence level (default `0.95`).
`B`	Number of Monte Carlo draws used for the indirect-effect CI when `method = "monte_carlo"` (default 5000). The interval is read from the quantiles of the draws, so `B` follows the rule of `dmar_bootstrap`: at least 400 at the 95 percent level, unless `options(dmar.bootstrap_exploratory = TRUE)` is set.
`G`	Number of outer simulation replications (default 1000).
`print_iter`	Logical.
`filename`	Optional path for a comma separated file recording every replication (the sample indirect effect $\hat{a}\hat{b}$ in the column `std_indirect_effect`, the two confidence limits, the interval width, and two indicators of whether the interval missed `true_std_reg_coef_a * true_std_reg_coef_b` below or above): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`.
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Value

A `data.frame` with rows for the mean, median, and SD of the sample indirect effect $\hat{a}\hat{b}$ (`mean_std_indirect_effect`, `median_std_indirect_effect`, `sd_std_indirect_effect`) and of the CI width, the proportion of intervals at or below `width`, tail-specific and overall non-coverage of the population value `true_std_indirect_effect` (`true_std_reg_coef_a * true_std_reg_coef_b`), and the input echoes `total_N`, `true_std_reg_coef_a`, `true_std_reg_coef_b`, `supposed_std_reg_coef_a`, `supposed_std_reg_coef_b` (`NA` when `specified_N` was given), `width`, and `conf_level`. Under `method = "monte_carlo"` the result carries the attributes `B`, `B_used`, `n_dropped`, `B_required`, and `bootstrap_status` of `dmar_bootstrap`; they record the request each replication ran (`B_used` equals `B`), and `bootstrap_status` is `"exploratory"` when `B` is below `B_required`.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Preacher, K. J., & Kelley, K. (2011). Effect size measures for mediation models: Quantitative strategies for communicating indirect effects. *Psychological Methods, 16*(2), 93–115. doi:10.1037/a0022658

Tofighi, D., & Kelley, K. (2020). Improved inference in mediation analysis: Introducing the model-based constrained optimization procedure. *Psychological Methods, 25*(4), 496–515. doi:10.1037/met0000259

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ss_aipe_indirect_effect, var_indirect_effect

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(),
ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_mean(),
ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(),
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(),
ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(),
ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(),
ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# Reduced replications and a wide target width keep this fast.
set.seed(113)
ss_aipe_indirect_effect_sensitivity(
  true_std_reg_coef_a = 0.4, true_std_reg_coef_b = 0.3,
  supposed_std_reg_coef_a = 0.4, supposed_std_reg_coef_b = 0.3,
  width = 0.40, method = "closed_form",
  G = 50, print_iter = FALSE
)
```

---

ss_aipe_mean — *Sample Size Planning for a Mean From the Accuracy in Parameter Estimation (AIPE) Perspective*

Description

Plans the sample size *N* needed for the confidence interval for a population mean to be sufficiently narrow, from the accuracy in parameter estimation (AIPE) perspective. The mean may be that of a single sample or the mean of paired differences: in a pretest-posttest or matched pairs design each unit contributes one difference score, the interval for the mean difference is the single sample interval computed on those scores, and the plan is the same with `true_sd` the standard deviation of the differences. Without an `assurance` the plan makes the expected width of the interval no wider than `width`; with one, it makes the realized width no wider than `width` with that probability.

Usage

```
ss_aipe_mean(
  true_sd,
  width,
  conf_level = 0.95,
  assurance = NULL,
  true_kurtosis = 0
)
```

Arguments

`true_sd`	The planning value of the population standard deviation of the scores whose mean is estimated. For a paired design it is the standard deviation of the differences, which for two measures with standard deviations σ_1 and σ_2 and correlation ρ is $\sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\,\sigma_1\sigma_2}$. Must be positive.
`width`	The desired full width of the confidence interval (twice its half-width), in the units of the scores.
`conf_level`	The confidence level of the interval (default `0.95`).
`assurance`	`NULL` (the default) to plan the expected width, or the desired probability, strictly between 0.5 and 1, that the realized width will be no wider than `width`.
`true_kurtosis`	The planning value of the population excess kurtosis of the scores, on the scale `kurtosis` reports (0 for a normal population). It enters only the assurance adjustment (see Details). The default 0 is normal theory. The value must be greater than -2: no distribution has an excess kurtosis below -2, and only a population of two equally likely values reaches it. A pilot's `kurtosis()` can usually be passed as it is, but `kurtosis()` corrects its estimate for small-sample bias, and for a pilot whose scores take nearly two values it can return -2 or less (`kurtosis(rep(c(0, 1), 123))` is about -2.016), which is refused. Such a pilot says the population is close to that two-valued limit. Pass instead a value a little above -2, such as -1.9, or 0, the normal theory plan, which for an assurance of .80 or more is never the smaller of the two.

Details

The interval. The $100(1-\alpha)\%$ interval for the mean is $\bar{y} \pm t_{1-\alpha/2,\,N-1}\,S/\sqrt{N}$, the interval `t.test()` reports for a single sample and for paired data. Its full width $W = 2\,t_{1-\alpha/2,\,N-1}\,S/\sqrt{N}$ is random because the sample standard deviation S is, so a plan is a statement either about the expected width or about the probability that W is no larger than the desired width ω. These are the two criteria Kelley and Rausch (2006) developed for the standardized mean difference and that Kelley, Maxwell, and Rausch (2003) developed for the unstandardized mean difference.

Expected width. Without an assurance, `necessary_N` is the smallest N at which

$$E(W) = 2\,t_{1-\alpha/2,\,N-1}\,c_4(N)\,\sigma/\sqrt{N} \le \omega,$$

where, under normality,

$$c_4(N) = \frac{E(S)}{\sigma} = \sqrt{\frac{2}{N-1}}\,\frac{\Gamma(N/2)}{\Gamma((N-1)/2)}.$$

The expected width is therefore exact under normality. It falls as *N* rises, so the criterion, once met, holds at every larger *N*. The `expected_width` row reports it at `necessary_N`.

How the plan differs from `ss_aipe_c()`. `ss_aipe_c()`, which plans a contrast among independent groups, sets S equal to σ in the width. Because $E(S) = c_4(N)\,\sigma < \sigma$, the width it plans is larger than the expected width, and the same rule applied to a single mean gives either the *N* returned here or one more. With an assurance the difference disappears: the criterion is then the probability that S is small enough, which does not involve $E(S)$, and under normality it is the criterion `ss_aipe_c()` uses for a contrast, here applied to a single mean. (`ss_aipe_c()` cannot plan a single mean, since it requires contrast weights that sum to zero.)

Assurance. With assurance γ, `necessary_N` is the smallest *N* from which $P(W \le \omega) \ge \gamma$ holds at every larger *N*. Because $W \le \omega$ exactly when

$$\frac{S^2}{\sigma^2} \le \frac{N\omega^2}{4\,t^2_{1-\alpha/2,\,N-1}\,\sigma^2},$$

the probability is that of S^2 falling below a bound. Under normality $(N-1)S^2/\sigma^2$ is chi square on $N-1$ degrees of freedom, so the probability is exact, and the criterion, once met, holds at every larger *N*; `necessary_N` is then simply the smallest *N* at which $P(W \le \omega) \ge \gamma$. The two readings part only under the kurtosis adjustment (see below).

Kurtosis. For *N* independent scores whose excess kurtosis is κ, the variance of the sample variance is

$$\mathrm{Var}(S^2) = \sigma^4 \left\{ \frac{2}{N-1} + \frac{\kappa}{N} \right\},$$

against $2\sigma^4/(N-1)$ under normality. With a nonzero `true_kurtosis` the chi square law is replaced by a scaled chi square, $S^2/\sigma^2 \approx \chi^2_\nu/\nu$, with

$$\nu = \frac{2}{2/(N-1) + \kappa/N}$$

chosen so that the approximation has the exact mean and variance of S^2/σ^2; $\nu = N-1$ when $\kappa = 0$. Tails heavier than the normal's ($\kappa > 0$) make S more variable, and for an assurance of .80 or more they make it harder to reach, while lighter tails ($\kappa < 0$, as for scores on a bounded scale) make it easier. Precisely, with an assurance of .80 or more, when the plan made with a given κ has at least $\kappa + 3$ observations, no smaller planning value of κ gives a larger plan, so an overstated κ errs on the side of a larger *N*. With an assurance close to one half the criterion turns instead on the median of S^2, which heavier tails pull further below σ^2, and a larger κ can give a slightly smaller plan. The argument is on the excess scale of `kurtosis`, so `kurtosis(pilot)` can usually be passed as it is (see the `true_kurtosis` argument for a pilot that gives -2 or less). A pilot's kurtosis is itself imprecise. The kurtosis does not enter the expected width plan: there it would lower $E(S)$ by a relative amount of about $\kappa/(8N)$ and *N* by about $\kappa/4$, less than one when $|\kappa| < 4$, so the plan keeps normal theory's $c_4(N)$ and the function warns when a nonzero `true_kurtosis` is supplied without an `assurance`.

The two-moment approximation is coarse when κ is large and *N* is small. Once *N* is below about $\kappa/2$, ν is below one, and the smaller ν is, the more of the scaled chi square's probability sits near zero, much as most small samples from a population with heavy tails miss the tails. The probability can then meet the assurance at a very small *N*, fall below it over a stretch of larger *N*, and meet it again. This is where the definition above matters: `necessary_N` is the start of the stretch over which the criterion holds at every larger *N*, not the first *N* at which it holds, and in this region a

larger κ can give a smaller plan. The function finds that N exactly: it bounds the criterion over whole ranges of N, using only that ν and the bound on S^2/σ^2 both rise with N and that a chi square quantile rises with its degrees of freedom, and evaluates it directly where the bounds cannot decide. `ss_aipe_mean_sensitivity()`, with a pilot's scores as the population, checks a plan without the approximation, and it also shows the coverage of the t interval under the shape of those scores, which the plan does not address.

When no planning value is trustworthy. A fixed N is only as good as the planning value of σ. Sequential fixed-width confidence interval procedures for a mean keep sampling until the interval reaches the target width and need no planning value of σ.

Value

A `data.frame` of class `dmar_ss_aipe` with columns `term` and `value` and the rows `necessary_N` (the planned sample size), `expected_width` (the expected width of the interval at that size under normality), the input echoes `true_sd`, `width`, and `conf_level`, and, when an assurance was supplied, `assurance` and `true_kurtosis`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

See Also

`ss_aipe_mean_sensitivity` for a Monte Carlo check of a plan; `ss_aipe_c` for a contrast among independent groups; `ss_aipe_sm` for the standardized mean; `kurtosis`; `design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivi`
`ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared()`,
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r()`, `ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`,
`ss_aipe_two_phase_cost()`, `ss_aipe_two_phase_sensitivity()`

Examples

```
# A pilot: the IQ gains (end of year minus pretest) of the 246 children
# in the control group of the pygmalion data. A gain is a paired
# difference, so its mean is planned as a single mean.
gain <- pygmalion$iq_gain[pygmalion$treatment == "Control"]
sd(gain)
kurtosis(gain)

# The N at which the expected full width of the 95 percent interval for
# the mean gain is at most 4 IQ points.
ss_aipe_mean(true_sd = sd(gain), width = 4)

# With 90 percent assurance that the realized width is at most 4 points,
# first under normal theory and then with the heavier than normal tails
# the pilot shows.
ss_aipe_mean(true_sd = sd(gain), width = 4, assurance = 0.90)
ss_aipe_mean(true_sd = sd(gain), width = 4, assurance = 0.90,
             true_kurtosis = kurtosis(gain))
```

---

ss_aipe_mean_sensitivity

Sensitivity Analysis for Sample Size Planning for a Mean From the Accuracy in Parameter Estimation (AIPE) Perspective

---

Description

Evaluates a sample size plan for the confidence interval for a mean by Monte Carlo simulation. Each of `G` replications draws a sample of the planned (or a specified) size from the data generating population, forms the *t* interval, and records its width and whether it missed the true mean. The summary shows what a plan from `ss_aipe_mean` delivers when the planning value of the standard deviation is wrong, when the population is not normal, or both. The population is either normal, with mean `true_mean` and standard deviation `true_sd`, or the empirical distribution of a vector of observed scores, resampled with replacement, which is how a plan is checked against the skewness, tails, and discreteness of real data such as a pilot's difference scores.

Usage

```
ss_aipe_mean_sensitivity(
  true_sd = NULL,
  supposed_sd = NULL,
  specified_N = NULL,
  width,
  true_mean = 0,
  population = NULL,
  conf_level = 0.95,
  assurance = NULL,
  true_kurtosis = 0,
```

```
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

true_sd	The data generating standard deviation of a normal population. Supply it, or population, but not both.
supposed_sd	The supposed standard deviation, the planning value passed to ss_aipe_mean to plan the sample size (not an estimate computed from data). Supply this or specified_N, but not both.
specified_N	A sample size to evaluate in place of a planned one; the planner is then bypassed. A single whole number of at least 2.
width	The desired full width of the confidence interval: the planning target, and the bound against which pct_ci_less_width counts the realized widths.
true_mean	The data generating mean of a normal population (default 0). The width does not depend on it; the misses are counted against it. Not supplied with population.
population	NULL (the default) for a normal population, or a numeric vector of scores whose empirical distribution is the population: each replication draws its sample from these scores with replacement. The true mean is then their mean and the true standard deviation their standard deviation with divisor equal to their number, the standard deviation of the distribution resampled. Missing values are refused rather than dropped.
conf_level	The confidence level of the interval (default 0.95).
assurance	NULL (the default) or the assurance passed to ss_aipe_mean when it plans the sample size; when one is supplied it is also echoed, as the value pct_ci_less_width is read against.
true_kurtosis	The planning value of the excess kurtosis passed to ss_aipe_mean (default 0), on the scale kurtosis reports. It changes the plan only together with supposed_sd and an assurance; the function warns when a nonzero value would go unused. As there, it must be greater than -2; the true_kurtosis argument of ss_aipe_mean says what to pass for a pilot whose kurtosis() is -2 or less.
G	The number of Monte Carlo replications (default 1000).
print_iter	Logical. Print the index of each replication as it runs, which is helpful in a long run; default FALSE.
filename	Optional path of a CSV file to receive the per-replication results (the sample mean and standard deviation, the interval limits and width, and the two tail misses), appended when the file already exists and created otherwise. The default NULL writes nothing; a throwaway run that wants the file should point it at tempfile(fileext = ".csv").
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the run reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

What each replication does. A sample of `total_N` scores is drawn, and the interval $\bar{y} \pm t_{1-\alpha/2,\, N-1}\, S/\sqrt{N}$ is formed, the one `t.test()` reports and the one the plan was made for. Every replication returns an interval, so none is dropped. A resample whose scores are all equal, which a discrete population allows at a small *N*, gives an interval of zero width, and it is counted as it falls.

What to read. The width rows against `width` show the effect of a planning value of the standard deviation that differs from the true one: the width is proportional to S, so a population standard deviation larger than the planning value widens every interval. `pct_ci_less_width` against the plan's assurance shows whether the plan delivered the probability it promised; with a resampled population it is the check, free of the two-moment approximation, on the planner's `true_kurtosis` adjustment. The two miss rows, each against $\alpha/2$, show the coverage of the *t* interval in the population at hand: under normality both tails miss at the nominal rate, and a skewed population misses more often in one tail than in the other, which the plan does not address.

A pilot as the population. Passing a pilot sample's scores as `population` and its standard deviation as `supposed_sd` evaluates the plan in a population with exactly the pilot's shape, so the planning value is correct (up to the divisor) and what the simulation isolates is the effect of the shape. The pilot is itself a sample, so its shape, and above all its tails, carries sampling error of its own.

Sensitivity analysis by simulation, in which the population value that generates the data and the value the plan assumed are set separately, was introduced for the standardized mean difference by Kelley and Rausch (2006); the `ss_aipe_*_sensitivity()` family carries it to the other AIPE planners, with a common set of rows.

Value

A `data.frame` with columns `term` and `value`. The rows `mean_mean`, `median_mean`, and `sd_mean` summarize the `G` sample means; `mean_ci_width`, `median_ci_width`, and `sd_ci_width` summarize the interval widths; `pct_ci_less_width` is the proportion of intervals no wider than `width`, to be read against the assurance of the plan (or against a value near one half for an expected width plan); `pct_ci_miss_low` is the proportion of intervals lying entirely above the true mean (the true mean below the lower limit) and `pct_ci_miss_high` the proportion lying entirely below it, with `total_type_I_error` their sum. These proportions are on the 0 to 1 scale. The input echoes are `total_N` (the sample size evaluated), `true_sd` and `true_mean` (computed from `population` when one is supplied), `supposed_sd` (NA when `specified_N` was supplied instead), `width`, `conf_level`, and, when an assurance was supplied, `assurance`. The attribute `population` records whether the population was `"normal"` or `"resampled"`.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted` - `returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

See Also

`ss_aipe_mean`, `kurtosis`, `ss_aipe_c_sensitivity`

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivi`
`ss_aipe_mean()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared()`, `ss_aipe_omega_squared_se`
`ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`, `ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`,
`ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`, `ss_aipe_semipartial_r()`,
`ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`, `ss_aipe_two_phase_cost()`,
`ss_aipe_two_phase_sensitivity()`

Examples

```
# The pilot of the ss_aipe_mean() page: the IQ gains of the 246 children
# in the control group of the pygmalion data.
gain <- pygmalion$iq_gain[pygmalion$treatment == "Control"]

# A plan for 90 percent assurance made under normal theory, evaluated by
# resampling the pilot's own gains, whose tails are heavier than the
# normal's: the share of intervals no wider than 4 points (the row
# pct_ci_less_width) falls short of .90.
ss_aipe_mean_sensitivity(supposed_sd = sd(gain), width = 4,
                         population = gain, assurance = 0.90, seed = 113)
```

```
# The plan made with the pilot's excess kurtosis is six children larger,
# and the share comes close to .90.
ss_aipe_mean_sensitivity(supposed_sd = sd(gain), width = 4,
                         population = gain, assurance = 0.90,
                         true_kurtosis = kurtosis(gain), seed = 113)

# A normal population whose standard deviation is 15 when the plan
# assumed 13.3: the realized intervals run wider than the target.
ss_aipe_mean_sensitivity(true_sd = 15, supposed_sd = 13.3, width = 4,
                         seed = 113)
```

`ss_aipe_mixed_effects` *AIPE Sample Size Planning for a Fixed Effect in a Two-Level Mixed-Effects Model*

Description

Computes the smallest number of clusters (i.e., Level 2 units) for which the confidence interval for the fixed-effect slope of one predictor has an expected full width no larger than `width`, in a balanced two-level model with a random intercept. The predictor may be measured on the individuals (i.e., Level 1) or on the clusters (i.e., Level 2), and may be centered at its cluster mean, centered at its grand mean, or left as it is. `ss_aipe_mixed_effects_sensitivity` simulates the same design, analysis, and interval, so the two can be used together: plan here, then check the plan there, including under values other than the ones planned with.

Usage

```
ss_aipe_mixed_effects(
  true_total_var,
  true_var_X,
  true_icc,
  width,
  cluster_size = 20L,
  conf_level = 0.95,
  predictor_level = 1,
  centering = c("cluster_mean", "grand_mean", "none"),
  true_icc_X = 0
)
```

Arguments

`true_total_var` Population variance of the outcome given the predictor (a planning value): the variance of the random intercept (between clusters) plus the residual variance (within clusters). It is the name the total variance has in the cluster randomized planners (`ss_aipe_crd`).

`true_var_X` Population variance of the predictor (a planning value). For an individual-level (i.e., Level 1) predictor it is the total variance over individuals, between and within clusters together (see `true_icc_X`); for a cluster-level (i.e., Level 2) predictor it is the variance over clusters.

`true_icc` Population intraclass correlation of the outcome given the predictor (a planning value): the share of `true_total_var` that lies between clusters. In [0, 1).

`width` Desired full width of the confidence interval for the slope.

`cluster_size` Number of individuals (i.e., Level 1 units) in each cluster (i.e., Level 2 unit), the same in every cluster. Default `20L`.

`conf_level` Confidence level. Default `0.95`.

`predictor_level`
`1` (the default) for an individual-level predictor, which varies within a cluster, or `2` for a cluster-level predictor, which is the same for every individual of a cluster.

`centering` How the predictor enters the model. `"cluster_mean"` (the default) subtracts each cluster's mean from its individuals' values, so the slope is the within-cluster slope; it applies to an individual-level predictor only. `"grand_mean"` subtracts the overall mean, and `"none"` leaves the predictor as it is. These two give the same slope and the same interval (only the intercept differs), and both are offered so that the call can say what the analysis will do.

`true_icc_X` Population intraclass correlation of an individual-level predictor (a planning value): the share of `true_var_X` that lies between clusters. Default `0`, a predictor whose cluster means differ by sampling alone. It is not used with `predictor_level = 2`.

Details

The model. For individual i of cluster k,

$$y_{ki} = \gamma_0 + \beta x_{ki} + u_k + e_{ki},$$

with $u_k \sim N(0, \tau^2)$ the random intercept of the cluster and $e_{ki} \sim N(0, \sigma^2)$ the residual of the individual, in K clusters of m individuals each. With V for `true_total_var` and ρ for `true_icc`, $\tau^2 = \rho V$ and $\sigma^2 = (1-\rho)V$. The slope has the same value within and between clusters. Writing σ_X^2 for `true_var_X` and ρ_X for `true_icc_X`, an individual-level predictor has variance $\rho_X \sigma_X^2$ between clusters and $\sigma_W^2 = (1-\rho_X)\sigma_X^2$ within them.

The interval planned for. The interval is $\hat{\beta} \pm t_{1-\alpha/2,\,\nu}\, \widehat{SE}(\hat{\beta})$, with $\nu = K(m-1) - 1$ for an individual-level predictor and $\nu = K - 2$ for a cluster-level predictor (the regression of K cluster means). Its expected width is $2\, t_{1-\alpha/2,\,\nu}\, E[\widehat{SE}(\hat{\beta})]$, and the function returns the smallest K at which that is no larger than `width`. No value of the slope is needed: β does not appear in any of the expressions below.

The expected standard error. Each expression follows from the sums of squares of a balanced design.

- *Individual-level predictor centered at its cluster mean.* The estimate is the within-cluster slope, and for a normally distributed predictor the expected standard error is exactly

$$E[\widehat{SE}(\hat{\beta})] = \frac{\sigma}{\sigma_W \sqrt{K(m-1) - 1}}.$$

Clustering of the outcome enters only through σ: the design effect $1+(m-1)\rho$ that inflates the variance of a cluster-level quantity does not appear, because centering removes the differences among clusters from both variables.

- *Cluster-level predictor.* The estimate is the slope of the regression of the cluster means of the outcome on the predictor, and for a normally distributed predictor the expected standard error is exactly

$$E[\widehat{SE}(\hat{\beta})] = \frac{\sqrt{\tau^2 + \sigma^2/m}}{\sigma_X\sqrt{K-2}}.$$

  Here the number of clusters is what buys precision: as m grows the numerator approaches τ and not 0.

- *Individual-level predictor centered at its grand mean, or not centered.* The estimate pools the within-cluster and the between-cluster information about the one slope, each weighted by its precision, and to first order

$$E[\widehat{SE}(\hat{\beta})]^2 \approx \left[\frac{\{K(m-1)-1\}\,\sigma_W^2}{\sigma^2} + \frac{(K-2)\,(\rho_X\sigma_X^2 + \sigma_W^2/m)}{\tau^2 + \sigma^2/m}\right]^{-1},$$

  which reduces to the two expressions above when either part is absent. It is never larger than the value for the within-cluster slope, and the gain is what the assumption of one slope within and between clusters buys; when that assumption is in doubt, plan for the cluster-mean-centered analysis.

In 4,000 simulated data sets for each of six designs (clusters of 5 to 20, 9 to 44 clusters, intraclass correlations of .05 to .30), the mean width of the interval was within 0.3 percent of the expected width in the four designs with an exact expression, and 0.5 and 1.3 percent narrower than it in the two with a pooled slope, so the first-order expression errs toward a slightly larger number of clusters. `ss_aipe_mixed_effects_sensitivity` reports the mean width and the coverage for any design.

Scope. One predictor, a random intercept, a fixed slope, and clusters of equal size. For random slopes, cross-level interactions, or unequal cluster sizes the variance changes and a Monte Carlo planner should be used (Schoemann, Boulton, & Short, 2017). The two-level random-intercept model is treated at length by Snijders and Bosker (2012).

Value

A `data.frame` with rows for the necessary number of clusters `necessary_n_clusters`, the implied total sample size `total_N` (`necessary_n_clusters * cluster_size`), the desired `width`, the `expected_width` at the necessary number of clusters, the degrees of freedom `df` of the interval's *t* quantile, and the inputs `true_icc`, `true_icc_X`, `cluster_size`, and `predictor_level`. The centering is recorded in `attr(x, "centering")`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Schoemann, A. M., Boulton, A. J., & Short, S. D. (2017). Determining power and sample size for simple and complex mediation models. *Social Psychological and Personality Science, 8*(4), 379–386. doi:10.1177/1948550617715068

Snijders, T. A. B., & Bosker, R. J. (2012). *Multilevel analysis: An introduction to basic and advanced multilevel modeling* (2nd ed.). Sage.

See Also

ss_aipe_mixed_effects_sensitivity to check a plan by simulation; ss_power_mixed_effects, var_icc, ss_aipe_icc

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_s ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other mixed models: R2_mixed_effects(), R2_mixed_effects_decomposition(), icc_lmer(), manova_split_plot(), mixed_anova(), ss_aipe_mixed_effects_sensitivity(), ss_power_mixed_effects(), ss_power_split_plot_anova()

Examples

```
# 1. An individual-level predictor centered at its cluster mean (the
#    default): clusters of 20, intraclass correlation .10, unit
#    variances, and a desired width of .20.
ss_aipe_mixed_effects(true_total_var = 1, true_var_X = 1, true_icc = 0.10,
                      width = 0.20, cluster_size = 20)

# 2. The same predictor not centered, when 30 percent of its variance
#    lies between clusters.
ss_aipe_mixed_effects(true_total_var = 1, true_var_X = 1, true_icc = 0.10,
                      width = 0.20, cluster_size = 20,
                      centering = "none", true_icc_X = 0.30)

# 3. A cluster-level predictor: far more clusters are needed for the
#    same width, because each cluster supplies one value of it.
ss_aipe_mixed_effects(true_total_var = 1, true_var_X = 1, true_icc = 0.10,
                      width = 0.20, cluster_size = 20,
                      predictor_level = 2, centering = "grand_mean")
```

`ss_aipe_mixed_effects_sensitivity`
Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for a Mixed-Effects Fixed Effect

Description

Simulates the design that `ss_aipe_mixed_effects` plans and reports what its confidence interval for the fixed-effect slope delivers: the mean width, the share of intervals no wider than desired, and the coverage. The two functions take the same description of the predictor (`predictor_level`, `centering`, the intraclass correlation of the predictor), use the same expressions, and form the same interval, so a plan made by the first can be checked by the second, at the planning values or at population values that differ from them.

On each replication the function draws `n_clusters` clusters (i.e., Level 2 units) of `cluster_size` individuals (i.e., Level 1 units) from

$$y_{ki} = \beta x_{ki} + u_k + e_{ki},$$

with $u_k \sim N(0, \rho V)$ and $e_{ki} \sim N(0, (1-\rho)V)$, where β is `true_reg_coef`, V is `true_total_var`, and ρ is `true_icc`. An individual-level (i.e., Level 1) predictor is $x_{ki} = b_k + w_{ki}$, with variance $\rho_X \sigma_X^2$ between clusters and $(1 - \rho_X)\sigma_X^2$ within them (`true_var_X`, `true_icc_X`); a cluster-level (i.e., Level 2) predictor is $x_k \sim N(0, \sigma_X^2)$, the same for every individual of a cluster. The predictor is then centered as `centering` says, the model is fit, and the interval $\hat{\beta} \pm t_{1-\alpha/2,\,\nu}\,\widehat{SE}(\hat{\beta})$ is recorded, with the degrees of freedom ν of `ss_aipe_mixed_effects`.

Usage

```
ss_aipe_mixed_effects_sensitivity(
  true_total_var = NULL,
  true_var_X = NULL,
  true_icc = NULL,
  true_reg_coef = 0,
  supposed_total_var = NULL,
  supposed_var_X = NULL,
  supposed_icc = NULL,
  width,
  cluster_size = 20L,
  specified_n_clusters = NULL,
  conf_level = 0.95,
  predictor_level = 1,
  centering = c("cluster_mean", "grand_mean", "none"),
  true_icc_X = 0,
  supposed_icc_X = true_icc_X,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_total_var`	Population variance of the outcome given the predictor, the variance of the random intercept plus the residual variance: the value the data are generated from.
`true_var_X`	Population variance of the predictor the data are generated from: over individuals for an individual-level predictor (between and within clusters together), over clusters for a cluster-level predictor.
`true_icc`	Population intraclass correlation of the outcome given the predictor (the share of `true_total_var` between clusters).
`true_reg_coef`	Population value of the fixed-effect slope, the regression coefficient of the outcome on the predictor (β in the model above; default `0`). The width of the interval does not depend on it; the coverage is evaluated against it.
`supposed_total_var`, `supposed_var_X`, `supposed_icc`	Supposed values, the planning values passed to `ss_aipe_mixed_effects` (not estimates computed from data), which may differ from the population values above; supply all three, or `specified_n_clusters`, but not both.
`width`	Desired full width of the confidence interval for the slope.
`cluster_size`	Number of individuals in each cluster (the same in every cluster).
`specified_n_clusters`	Number of clusters to evaluate, in place of the number the planning values give.
`conf_level`	Confidence level (default `0.95`).
`predictor_level`, `centering`	The level of the predictor (`1`, the default, or `2`) and how it is centered (`"cluster_mean"`, `"grand_mean"`, or `"none"`), as in `ss_aipe_mixed_effects`. The same values are used to plan and to analyze each simulated data set.
`true_icc_X`	Population intraclass correlation of an individual-level predictor: the share of `true_var_X` between clusters in the data generated. Default `0`. Not used with `predictor_level = 2`.
`supposed_icc_X`	The supposed value of `true_icc_X`, the planning value passed to `ss_aipe_mixed_effects`. Default: `true_icc_X`.
`G`	Number of Monte Carlo replications.
`print_iter`	Logical.
`filename`	Optional path for a comma separated file recording every replication (the fixed effect estimate, the two confidence limits, the interval width, and two indicators of whether the interval missed `true_reg_coef` below or above): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`.
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

How each case is fit. Two of the three analyses have closed forms in a balanced design and are computed directly. The slope of an individual-level predictor centered at its cluster mean is the within-cluster regression slope, with the residual mean square on $K(m-1)-1$ degrees of freedom. The slope of a cluster-level predictor is the slope of the regression of the K cluster means of the outcome on the predictor, on $K-2$ degrees of freedom. In both, restricted maximum likelihood for the random-intercept model gives the same estimate and standard error when its estimate of the intercept variance is positive. An individual-level predictor that is centered at its grand mean or not centered has no closed form, because its weights depend on the estimated variance components; it is fit by restricted maximum likelihood with **lme4**, which must then be installed, and its interval uses $K(m-1)-1$ degrees of freedom.

What to look at. `mean_ci_width` against `width` says whether the plan delivers the width on average, and `pct_ci_less_width` how often. `total_type_I_error` against $1-$ `conf_level` says whether the interval covers as it should; with few clusters and a cluster-level predictor this is where a problem would show. Replications in which no interval could be formed are counted in the record of the simulation.

Value

A `data.frame` with rows for the mean, median, and standard deviation of the estimated slope (`mean_reg_coef`, `median_reg_coef`, `sd_reg_coef`) and of the interval width, the share of intervals no wider than `width` (`pct_ci_less_width`), the shares that missed `true_reg_coef` below and above and their sum, the design simulated (`n_clusters`, `cluster_size`, `total_N`, `predictor_level`), the `expected_width` that `ss_aipe_mixed_effects` gives for that design at the population values, and the inputs. The centering is recorded in `attr(x, "centering")`.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

McNeish, D., & Kelley, K. (2019). Fixed effects versus mixed effects models for clustered data: Reviewing the approaches, disentangling the differences, and making recommendations. *Psychological Methods, 24*, 20–35. doi:10.1037/met0000182

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapters 15 and 16 on mixed-effects models.)

See Also

`ss_aipe_mixed_effects`, `icc_lmer`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivit`
`ss_aipe_mean()`, `ss_aipe_mean_sensitivity()`, `ss_aipe_omega_squared()`, `ss_aipe_omega_squared_sensitivity(`
`ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`, `ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`,
`ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`, `ss_aipe_semipartial_r()`,
`ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`, `ss_aipe_two_phase_cost()`,
`ss_aipe_two_phase_sensitivity()`

Other mixed models: `R2_mixed_effects()`, `R2_mixed_effects_decomposition()`, `icc_lmer()`,
`manova_split_plot()`, `mixed_anova()`, `ss_aipe_mixed_effects()`, `ss_power_mixed_effects()`,
`ss_power_split_plot_anova()`

Examples

```
# Plan, then check the plan (few replications, for a fast example).
ss_aipe_mixed_effects(true_total_var = 1, true_var_X = 1, true_icc = 0.10,
                      width = 0.40, cluster_size = 10)
ss_aipe_mixed_effects_sensitivity(
  true_total_var = 1, true_var_X = 1, true_icc = 0.10,
  true_reg_coef = 0.30,
  supposed_total_var = 1, supposed_var_X = 1, supposed_icc = 0.10,
  width = 0.40, cluster_size = 10,
  G = 50, print_iter = FALSE, seed = 113
)
```

```
# A cluster-level predictor, at a number of clusters chosen by hand.
ss_aipe_mixed_effects_sensitivity(
  true_total_var = 1, true_var_X = 1, true_icc = 0.10,
  true_reg_coef = 0.30,
  specified_n_clusters = 25, cluster_size = 10, width = 0.40,
  predictor_level = 2, centering = "grand_mean",
  G = 50, print_iter = FALSE, seed = 113
)
```

`ss_aipe_omega_squared` *Sample Size for AIPE on Omega Squared (ANOVA Effect Size)*

Description

Determines the sample size needed for the noncentral *F* confidence interval on the population omega squared (ω^2) to have a desired width (Accuracy in Parameter Estimation; Kelley, 2008; Steiger, 2004). The function uses the same noncentral *F* machinery as `ci_omega_squared`: at each candidate N it computes the expected CI width by inverting the noncentral *F* distribution and stops at the smallest N that achieves the target width.

Usage

```
ss_aipe_omega_squared(
  true_omega_squared,
  df_effect,
  width,
  which_width = c("Full", "Lower", "Upper"),
  conf_level = 0.95,
  assurance = NULL
)
```

Arguments

`true_omega_squared`	Anticipated population ω^2. Must lie in $[0, 1)$.
`df_effect`	Numerator degrees of freedom for the effect (e.g., $a - 1$ for an a-group one-way ANOVA, or the appropriate per-effect numerator df in a factorial design).
`width`	Desired full width of the CI on ω^2.
`which_width`	Whether `width` is the `"Full"` width of the interval (default) or a half-width: `"Lower"` and `"Upper"` both interpret `width` as half the full width, so they plan for a full width of twice `width` and return the same sample size. Because the interval is not generally symmetric about the estimate, its realized lower and upper half-widths can differ from each other and from half the full width; the planner does not target them separately. A genuinely one-sided width target is not currently offered.
`conf_level`	Desired confidence level (default `0.95`).

`assurance` Optional. Probability that the realized CI is no wider than `width`; when supplied, the sample size is inflated by the standard chi squared correction (Kelley, 2008).

Details

Connection to noncentral *F* machinery. The CI on ω^2 is built by inverting the noncentral F sampling distribution of the observed F statistic, following Steiger (2004) and Kelley (2007); see `ci_omega_squared`. To plan a sample size, we iterate: for each candidate N, compute the F the analyst would observe *at the population effect size*, build its CI on ω^2, and stop at the smallest N whose CI width is below the target.

Population-effect-to-*F* mapping. Given a target ω^2, the expected sample F that yields exactly that ω^2 as the point estimate from $\hat{\omega}^2 = df_{\text{eff}}(F-1)/[df_{\text{eff}}(F-1)+N]$ is $F = 1+\omega^2 N/[df_{\text{eff}}(1-\omega^2)]$. This is the F value used at each iteration of the search.

Tolerance behavior at small *N*. For small candidate N the noncentral F lower limit is often clamped to zero (see `?ci_nc_F`). The search ignores these clamps in the iteration and reports the final clamp count, if any, as an informational message; this matches the convention in `ss_aipe_R2`.

Value

A `data.frame` with rows for the recommended *total* sample size `necessary_N`, the expected CI width at that sample size, and the inputs echoed back.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J., Moulder, B. C., & Moser, B. K. (2002). Sample size requirements for accurate estimation of squared semi-partial correlation coefficients. *Multivariate Behavioral Research, 37*(1), 37–57. doi:10.1207/s15327906mbr3701_02

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*(4), 524–555. doi:10.1080/00273170802490632

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*, 137–152. doi:10.1037/a0028086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on η^2, Chapter 7 on factorial designs, and Chapter 11 on generalized η^2 for within-subjects designs.)

Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

See Also

ci_omega_squared, omega_squared, ss_aipe_R2

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(),
ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivi
ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared_
ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(), ss_aipe_pcm_sensitivity(), ss_aipe_r(),
ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(),
ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(),
ss_aipe_two_phase_sensitivity()

Examples

```
# 1. Plan total N so the 95% CI on omega^2 has full width <= 0.10
#        in a 3-group one-way ANOVA (df_effect = 2), anticipated
#        omega^2 = 0.10.
ss_aipe_omega_squared(true_omega_squared = 0.10,
                      df_effect = 2,
                      width = 0.10)

# 2. Same problem with 80% assurance:
ss_aipe_omega_squared(true_omega_squared = 0.10,
                      df_effect = 2,
                      width = 0.10,
                      assurance = 0.80)
```

`ss_aipe_omega_squared_sensitivity`
Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for Omega Squared

---

Description

Quantifies how much misspecification of the population ω^2 distorts an AIPE-based sample size plan. The planner ss_aipe_omega_squared solves for the smallest N that yields an expected CI width below the target at the planning value. Here we generate `G` datasets from a balanced one-way ANOVA with population ω^2 =`true_omega_squared` and `df_effect` + 1 groups at the planner-recommended N, compute the noncentral F confidence interval on each replication via ci_omega_squared, and summarize the realized widths and coverage of `true_omega_squared`.

Usage

```
ss_aipe_omega_squared_sensitivity(
  true_omega_squared = NULL,
  supposed_omega_squared = NULL,
  df_effect,
  width,
  specified_N = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_omega_squared`	Population ω^2 (the data generating value); in $[0, 1)$.
`supposed_omega_squared`	The supposed ω^2, the planning value from which `ss_aipe_omega_squared` plans the study (not an estimate computed from data); supply this or `specified_N` but not both.
`df_effect`	Numerator degrees of freedom for the omnibus *F*, equal to the number of groups minus 1.
`width`	Desired full width of the confidence interval on ω^2.
`specified_N`	Total sample size to evaluate (incompatible with `supposed_omega_squared`).
`conf_level`	Confidence level (default `0.95`).
`assurance`	Optional assurance probability passed to `ss_aipe_omega_squared` when resolving the planned sample size.
`G`	Number of Monte Carlo replications (default 1000).
`print_iter`	Logical. Print iteration index per replication.
`filename`	Optional path for a comma separated file recording every replication (the sample $\hat{\omega}^2$, the two confidence limits, the interval width, and two indicators of whether the interval missed `true_omega_squared` below or above): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`.
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Value

A `data.frame` with rows for mean / median / SD of the realized $\hat{\omega}^2$ and interval width, the proportion of intervals at or below `width`, tail-specific and overall empirical non-coverage of `true_omega_squared`, and the input echoes, including `assurance` (present only when an assurance was supplied).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on effect size measures.)

See Also

`ss_aipe_omega_squared`, `ci_omega_squared`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivi`
`ss_aipe_mean()`, `ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared`
`ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`, `ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`,
`ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`, `ss_aipe_semipartial_r()`,

ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# Well-specified: planner used omega^2 = 0.10, truth is 0.10.
# G is kept small here so the example runs quickly; raise it for a
# stable sensitivity estimate.
set.seed(113)
ss_aipe_omega_squared_sensitivity(
  true_omega_squared     = 0.10,
  supposed_omega_squared = 0.10,
  df_effect = 2, width = 0.10,
  G = 25, print_iter = FALSE
)
```

---

ss_aipe_partial_r *Sample Size for AIPE on a Partial Correlation*

Description

Takes a planning value of the population partial correlation $\rho_{XY \cdot Z_1 \cdots Z_{p-1}}$, the number of predictors (X and the variables partialed out), and a desired confidence interval width, and returns the smallest sample size at which the interval is expected to be that narrow (accuracy in parameter estimation, AIPE; Kelley, 2008). With `assurance` it returns the smallest sample size at which the realized interval is that narrow with at least the stated probability.

Usage

```
ss_aipe_partial_r(
  true_cor,
  p,
  width,
  which_width = c("Full", "Lower", "Upper"),
  conf_level = 0.95,
  fisher_z = FALSE,
  assurance = NULL
)
```

Arguments

`true_cor`	Population partial correlation $\rho_{XY \cdot Z_1 \cdots Z_{p-1}}$ (written ρ in Details) the sample size is planned for, in $(-1, 1)$. The planner treats this value as the truth; ss_aipe_partial_r_sensitivit shows what the plan delivers when the truth (`true_cor` there) differs from the value planned with (`supposed_cor` there).

`p` Number of predictors in the regression of Y on X and the variables partialed out, so `p - 1` variables are partialed out and `p` is at least 2. A categorical variable with a levels counts as a - 1 predictors. It is the `p` of `ss_power_R2`, `ci_R2`, and `ci_reg_coef`: the correlation concerns one predictor of a regression with `p` predictors. In the notation of a model comparison, q is the number of predictors removed from the full model to give the restricted model ($q = p_{full} - p_{restricted}$, the predictors being tested), and the number of variables partialed out is $p - q$; this correlation concerns one predictor, so $q = 1$ and $p - 1$ variables are partialed out.

`width` Desired full width of the confidence interval on the partial correlation.

`which_width` Whether `width` is the `"Full"` width of the interval (default) or a half-width: `"Lower"` and `"Upper"` both interpret `width` as half the full width, so they plan for a full width of twice `width` and return the same sample size. Because the interval is not generally symmetric about the estimate (with `fisher_z = TRUE` in particular), its realized lower and upper half-widths can differ from each other and from half the full width; the planner does not target them separately. A genuinely one-sided width target is not currently offered.

`conf_level` Desired confidence level (default `0.95`).

`fisher_z` Logical. If `TRUE`, the plan is for the interval formed on Fisher's Z scale (variance $1/(N - p - 2)$) and back-transformed with $\tanh$. If `FALSE` (default), the plan is for the interval formed on the raw scale from the first-order variance.

`assurance` Optional. The probability that the realized interval is no wider than `width`. The function returns the smallest sample size, at or above the sample size planned without it, at which that probability is at least `assurance`. A single number above 0.5 and below 1; when `NULL` (the default) the plan is for the expected width only.

Details

Estimand and interval. The quantity planned for is the population partial correlation between X and Y given $Z_1, \ldots, Z_{p-1}$, for independent observations from a multivariate normal population. The interval is two-sided at `conf_level` and is one of two Wald intervals, chosen by `fisher_z`; the plan is for the one chosen, and a study that reports the other is not the study that was planned. Below, ρ is that partial correlation (`true_cor`) and $1 - \alpha$ is `conf_level`.

Raw scale, first-order variance (`fisher_z = FALSE`). The first-order variance of a simple correlation is $(1 - \rho^2)^2/N$ (Olkin & Finn, 1995, Equation 3). The sample partial correlation given $p - 1$ variables is distributed as a simple correlation from $N - (p - 1)$ observations (Fisher, 1924, p. 330, for normally distributed variables), and the functions here use $(1 - \rho^2)^2/(N - p)$, which differs from that at order $1/N^2$ (see `var_partial_r`). The interval is $r \pm z_{1-\alpha/2}(1 - r^2)/\sqrt{N - p}$, with half-width

$$w_{1/2}(r, N) \;=\; z_{1-\alpha/2}\,\frac{1 - r^2}{\sqrt{N - p}}.$$

Setting $r = \rho$ and solving for the smallest whole N with $w_{1/2} \le$ the target half-width gives

$$N \;=\; p + \left\lceil z^2_{1-\alpha/2}\,(1 - \rho^2)^2/w^2_{1/2} \right\rceil.$$

Fisher's *Z* scale (`fisher_z = TRUE`). Bonett and Wright (2000, Section 3, pp. 24–25) plan the sample size for a simple correlation so that the interval formed on Fisher's *Z* scale (Fisher, 1921, 1924) and back-transformed has the desired width. The same plan is applied here to the partial correlation, with the standard error $1/\sqrt{N-p-2}$ in place of $1/\sqrt{N-3}$. With $Z = \operatorname{arctanh}(r)$ and standard error $1/\sqrt{N-p-2}$, the interval is $\tanh(Z \pm z_{1-\alpha/2}/\sqrt{N-p-2})$, and half its width is

$$w_{1/2}(r, N) \;=\; \tfrac{1}{2}\Big\{ \tanh\Big(\operatorname{arctanh}(r) + \frac{z_{1-\alpha/2}}{\sqrt{N-p-2}}\Big) - \tanh\Big(\operatorname{arctanh}(r) - \frac{z_{1-\alpha/2}}{\sqrt{N-p-2}}\Big)\Big\}.$$

The planner finds the smallest N of at least $p+3$ at which this is no larger than the target at $r = \rho$, by a one-dimensional search (the half-width decreases in N). This interval stays inside $(-1, 1)$ and is generally the more accurate of the two when $N - p$ is small or $|\rho|$ is large.

The assurance step. A plan without `assurance` makes the interval as narrow as the target when the sample partial correlation equals `true_cor`; in about half of all samples it is wider. With `assurance` the function finds the sample size at which the realized interval is no wider than the target with at least that probability, from the sampling distribution of the sample partial correlation:

1. At a given N the width of either interval depends on the sample only through r, and both half-widths above decrease as $|r|$ grows. The realized interval is therefore no wider than the target exactly when $|r| \geq r^*(N)$, where $r^*(N)$ is the $|r|$ at which the half-width equals the target half-width h: on the raw scale $r^*(N) = \sqrt{1 - h\sqrt{N-p}/z_{1-\alpha/2}}$, and on Fisher's scale the root of $w_{1/2}(r, N) = h$. When the half-width at $r = 0$, the widest the interval can be, already meets the target, every sample does and the probability is 1.
2. Under multivariate normality the sample partial correlation given $p-1$ variables from N observations is distributed as a simple correlation from $m = N - (p-1)$ observations. Its distribution function is computed as

$$F(c) \;=\; \int_0^1 T_{2m-3}\Big(\sqrt{2m-3}\,\Big\{\frac{c\sqrt{1-b}}{\sqrt{1-c^2}} - \frac{\rho\sqrt{b}}{\sqrt{1-\rho^2}}\Big\}\Big)\, g(b)\, db,$$

   where T_{2m-3} is the distribution function of Student's *t* on $2m-3$ degrees of freedom and g is the density of the beta distribution with parameters $(m-1)/2$ and $(m-2)/2$. The form follows from writing $r/\sqrt{1-r^2}$ as $(Z + \theta\sqrt{U})/\sqrt{V}$, with Z standard normal, U and V chi square on $m-1$ and $m-2$ degrees of freedom, all independent, and $\theta = \rho/\sqrt{1-\rho^2}$, and then conditioning on $U/(U+V)$. It uses the central *t* only, because R's noncentral *t* is an approximation at the noncentrality values that plans for narrow intervals reach.
3. The probability at N is $P_N = 1 - F(r^*(N)) + F(-r^*(N))$.
4. The function returns the smallest N, at or above the sample size planned without `assurance`, with $P_N \geq$ `assurance`, found by doubling a step upward and then bisecting. `assurance_attained` is P_N at that N.

The probability is exact, up to numerical integration, under multivariate normality for the interval that is planned. It was checked by simulation in two ways: in 112 planning sets of 200,000 simulated correlations each, the proportion of intervals no wider than the target at the planned sample size was within .0023 of P_N; and in 16 further planning sets simulated from regression data (20,000 data sets each, with the variables partialed out by least squares) it was within .004 of P_N, about two Monte Carlo standard errors, and between .002 and .012 above the request.

Four properties of the step are worth knowing.

- The answer is never below the sample size planned without `assurance`. Below that sample size P_N is not monotone in N (at a very small N the sample correlation is variable enough to exceed $r^*(N)$ by chance, and does so less often as N grows, before the narrowing of the interval takes over), so a smaller N that happens to meet the request is not a plan. At or above it P_N did not decrease in any of 72 planning sets scanned over every whole N.
- When `true_cor` is 0 the plan without `assurance` already makes the interval as narrow as the target in every sample, since the interval is widest at $r = 0$; `assurance` then adds nothing and `assurance_attained` is 1. Near 0 the same holds approximately and the step adds little.
- As $|\rho|$ grows the width becomes more variable from sample to sample and the step adds more: for a width of .10 with one variable partialed out and an assurance of .80, the plan moves from 867 to 915 at $\rho = .50$ and from 121 to 151 at $\rho = .85$.
- `assurance` must be above 0.5 and below 1; other values are refused.

Why the step is not a chi square factor. Multiplying the sample size planned without `assurance` by a chi square quantile over its degrees of freedom is the assurance step for an interval whose width is proportional to a sample standard deviation (Kelley & Maxwell, 2003); the width of an interval for a correlation depends on the sample through r instead, and the factor gives too few cases when $|\rho|$ is large and too many when it is small. For one variable partialed out ($p = 2$), a width of .10, and a requested .80 it gives 426 at $\rho = .70$, where the probability is .675 (448 are needed), and 1318 at $\rho = .30$, where 1310 suffice; in the 16 planning sets simulated from regression data its sample sizes delivered from .14 below the request to .15 above it.

When to use partial or simple correlation planning. Use this function when the target is the population correlation between X and Y after statistically controlling for $Z_1, \ldots, Z_{p-1}$. For the simple Pearson correlation, see `ss_aipe_r`. `ss_aipe_partial_r_sensitivity` simulates the plan, including under a population value other than the planning value.

Value

A `data.frame` with the columns `term` and `value` and one row per term:

`necessary_N` the planned total sample size.

`expected_width` the full width of the interval at `necessary_N` when the sample partial correlation equals `true_cor`.

`true_cor`, `p`, `width`, `conf_level` the inputs, with `width` the `width` as given.

`assurance` the requested probability; present only when `assurance` was given.

`assurance_attained` the probability, at `necessary_N` and `true_cor`, that the realized interval is no wider than the target; present only when `assurance` was given. It is at least `assurance` and exceeds it because the sample size is a whole number.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J., & Olejnik, S. (2003). Sample size tables for correlation analysis with applications in partial correlation and multiple regression analysis. *Multivariate Behavioral Research, 38*(3), 309–323. doi:10.1207/s15327906mbr3803_02

Bonett, D. G., & Wright, T. A. (2000). Sample size requirements for estimating Pearson, Kendall and Spearman correlations. *Psychometrika, 65*(1), 23–28. doi:10.1007/BF02294183

Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron, 1*, 3–32.

Fisher, R. A. (1924). The distribution of the partial correlation coefficient. *Metron, 3*, 329–332.

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*(4), 524–555. doi:10.1080/00273170802490632

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on the one-way ANOVA and Chapter 4 on contrasts.)

Olkin, I., & Finn, J. D. (1995). Correlations redux. *Psychological Bulletin, 118*(1), 155–164. doi:10.1037/00332909.118.1.155

See Also

var_partial_r, expected_partial_r, ss_aipe_partial_r_sensitivity, ss_aipe_semipartial_r, ss_aipe_r

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(),
ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivi
ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r_sensitivity(), ss_aipe_pcm_sensitivity(),
ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(),
ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(),
ss_aipe_two_phase_sensitivity()

Examples

```
# 1. Plan N so the 95% CI on rho_XY.Z (two variables partialed out, so
#        p = 3 predictors) has full width <= 0.20, when the planning
#        value of the partial correlation is 0.30.
ss_aipe_partial_r(true_cor = 0.30, p = 3, width = 0.20)

# 2. The same problem for the interval formed on Fisher's Z scale
#        (Bonett & Wright, 2000).
ss_aipe_partial_r(true_cor = 0.30, p = 3, width = 0.20, fisher_z = TRUE)

# 3. With an assurance of .80: the smallest N at which the realized
#        interval is no wider than 0.20 with probability at least .80.
#        The last row is the probability at the returned N.
```

```
ss_aipe_partial_r(true_cor = 0.30, p = 3, width = 0.20, assurance = 0.80)

# 4. A larger correlation needs fewer cases for the same width, and
#        the assurance step adds proportionally more of them.
ss_aipe_partial_r(true_cor = 0.70, p = 3, width = 0.20)
ss_aipe_partial_r(true_cor = 0.70, p = 3, width = 0.20, assurance = 0.80)
```

`ss_aipe_partial_r_sensitivity` *Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for a Partial Correlation*

Description

Simulates a sample size plan for a partial correlation to show what it delivers when the population value differs from the planning value, or at a sample size chosen in advance. It takes the population partial correlation, either a planning value or a sample size, the number of predictors (X and the variables partialed out), and a target width, and returns summaries of the estimate, the interval width, the proportion of intervals no wider than the target, and coverage over `G` simulated samples.

Usage

```
ss_aipe_partial_r_sensitivity(
  true_cor = NULL,
  supposed_cor = NULL,
  p,
  width,
  specified_N = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  fisher_z = FALSE,
  seed = NULL
)
```

Arguments

`true_cor`	Population partial correlation ρ between Y and X controlling for $Z_1, \ldots, Z_{p-1}$; must lie in $(-1, 1)$.
`supposed_cor`	Planning value of the partial correlation passed to `ss_aipe_partial_r`; supply this or `specified_N` but not both.
`p`	Number of predictors in the regression of Y on X and the variables partialed out, so `p - 1` variables are partialed out and `p` is at least 2. A categorical variable with a levels counts as a - 1 predictors. It is the `p` of `ss_power_R2`, `ci_R2`,

	and ci_reg_coef: the correlation concerns one predictor of a regression with p predictors. In the notation of a model comparison, q is the number of predictors removed from the full model to give the restricted model ($q = p_{full} - p_{restricted}$, the predictors being tested), and the number of variables partialed out is $p - q$; this correlation concerns one predictor, so $q = 1$ and $p - 1$ variables are partialed out. The simulated data have $p + 1$ columns.
width	Desired full width of the CI on the partial correlation.
specified_N	Sample size to evaluate (incompatible with supposed_cor); a whole number of at least $p + 2$, or $p + 3$ with fisher_z = TRUE.
conf_level	Confidence level (default 0.95).
assurance	Optional assurance probability passed to ss_aipe_partial_r; used only with supposed_cor.
G	Number of Monte Carlo replications (default 1000).
print_iter	Logical. Print iteration index per replication.
filename	Optional path for a comma separated file recording every replication (the sample partial correlation, the two confidence limits, the interval width, and two indicators of whether the interval missed true_cor below or above): nothing is written when filename is NULL (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at tempfile(fileext = ".csv").
fisher_z	Logical. The interval that is planned and simulated. FALSE (the default) is the raw-scale interval, $r \pm z_{1-\alpha/2}(1 - r^2)/\sqrt{N - p}$; TRUE is the interval formed on Fisher's Z scale with standard error $1/\sqrt{N - p - 2}$ and back-transformed. The value is passed to ss_aipe_partial_r, so the interval simulated is always the one planned.
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the run reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Estimand and interval. The population partial correlation between Y and X given $Z_1, \ldots, Z_{p-1}$, estimated from N independent multivariate normal observations by the correlation of the two sets of least squares residuals. The interval is two-sided at conf_level and is the one fisher_z names. The raw-scale limits are not truncated at -1 and 1, so that the widths are those the planner plans; with a large $|\rho|$ and a small sample a limit can pass 1 in absolute value.

What is simulated. The function constructs a $(p + 1) \times (p + 1)$ population covariance matrix for Y, X, and $Z_1, \ldots, Z_{p-1}$, all standardized and uncorrelated but for a correlation of true_cor between Y and X, so that the partial correlation is true_cor. Under normality the distribution of the sample partial correlation depends on the population only through that value, p, and N, so this population stands for every normal population with the same partial correlation. With supposed_cor, N is the plan of ss_aipe_partial_r for that value (with assurance and fisher_z passed on); with specified_N it is that number. Each of G replications draws N rows, computes the sample partial correlation, and forms the interval. The $p - 1$ variables partialed out are the count ss_aipe_partial_r plans for and var_partial_r uses. A sample with nothing partialed out is the case of ss_aipe_r_sensitivity.

Reading the result. pct_ci_less_width is the proportion of the G intervals no wider than width: about one half for a plan without assurance whose planning value is correct, and at least assurance, within Monte Carlo error, for a plan with it. The Monte Carlo standard error of a proportion A is $\sqrt{A(1-A)/G}$. pct_ci_miss_low counts intervals that lay entirely above true_cor and pct_ci_miss_high those entirely below; their sum is total_type_I_error. Every replication has an interval, so all rows are over G replications.

Value

A data.frame with the columns term and value: mean_partial_r, median_partial_r, and sd_partial_r (the sample partial correlations); mean_ci_width, median_ci_width, and sd_ci_width (the full widths of the intervals); pct_ci_less_width (the proportion of intervals no wider than width); pct_ci_miss_low, pct_ci_miss_high, and total_type_I_error (non-coverage of true_cor, by side and in total); and the echoes total_N (the sample size simulated), p, true_cor, supposed_cor (NA when specified_N was given), width, conf_level, and assurance (present only when an assurance was supplied). The proportions are on the 0 to 1 scale. The attribute interval is "raw" or "fisher_z", and the printed table names the interval beneath it.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

width_target_mc_ci the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as G grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives G, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so failed is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ss_aipe_partial_r, ss_aipe_semipartial_r_sensitivity

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(),
ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivi
ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_pcm_sensitivity(),
ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(),
ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(),
ss_aipe_two_phase_sensitivity()

Examples

```
# A plan for a full width of .40 at a planning value of .40 that is
# correct. G = 50 keeps the example quick; a reported check deserves
# the default G = 1000 or more.
ss_aipe_partial_r_sensitivity(
  true_cor = 0.40, supposed_cor = 0.40, p = 4, width = 0.40,
  G = 50, seed = 113
)

# The same plan when the population value is smaller than planned: the
# intervals are wider and fewer meet the target.
ss_aipe_partial_r_sensitivity(
  true_cor = 0.20, supposed_cor = 0.40, p = 4, width = 0.40,
  G = 50, seed = 113
)

# Fisher's Z interval, planned and simulated.
ss_aipe_partial_r_sensitivity(
  true_cor = 0.40, supposed_cor = 0.40, p = 4, width = 0.40,
  G = 50, fisher_z = TRUE, seed = 113
)
```

`ss_aipe_pcm` *Sample Size Planning for Polynomial Change Models in Longitudinal Study*

Description

This function plans sample size with respect to the group-by-time interaction in the context of a longitudinal design with two groups. It plans sample size from the accuracy in parameter estimation (AIPE) perspective, where the goal is to obtain a sufficiently narrow confidence interval for the fixed effect polynomial change coefficient parameter (e.g., linear, quadratic, etc.). The sample size returned can be one such that (a) the expected confidence interval width is sufficiently narrow, or (b) the observed confidence interval will be sufficiently narrow with a specified high degree of assurance (e.g., .99, .95, .90, etc.). This function accompanies Kelley and Rausch (2011).

Usage

```
ss_aipe_pcm(
  true_var_trend,
  true_var_error = NULL,
  true_var_trend_error = NULL,
  duration,
  frequency,
  width,
  conf_level = 0.95,
  trend = "linear",
  assurance = NULL
)
```

Arguments

`true_var_trend` The variance of the individuals' true change coefficients (i.e., $\sigma^2_{\upsilon_m}$ in Kelley & Rausch, 2011) for the polynomial trend (e.g., linear, quadratic, etc.) of interest

`true_var_error` The true within-subject (i.e., Level 1) error variance (σ^2_ϵ in Kelley & Rausch, 2011). Either `true_var_error` or `true_var_trend_error` must be supplied; if `true_var_trend_error` is given directly, `true_var_error` may be omitted.

`true_var_trend_error`
The sampling-error variance of one individual's estimated change coefficient: the variance, across individuals, of the difference between the mth estimated and the mth true change coefficient of an individual, $\sigma^2_{\hat{\pi}_m - \pi_m}$ (Kelley & Rausch, 2011, Equation 19). It is the part of the variance of the estimated change coefficients that measurement error contributes, beside `true_var_trend`, the variance of the true ones. Kelley and Rausch (2011, Equation 19) give it as $\sigma^2_\epsilon / \sum_t c^2_{mt}$, where $\sum_t c^2_{mt}$ is the sum of the squared orthogonal polynomial contrast weights over the measurement occasions; with the change coefficient expressed per unit of time it is $\sigma^2_\epsilon f^{2p} / \sum_t c^2_{mt}$, where f is the frequency and p the polynomial order, which is how the function computes it from `true_var_error` (the two

	agree at $f = 1$). A user who already has this variance may supply it directly and omit `true_var_error`.
`duration`	The duration of the study
`frequency`	The number of times measurement occurs within each unit of time
`width`	Width of the confidence interval
`conf_level`	The desired level of confidence for the confidence interval that will be computed at the completion of the study
`trend`	The polynomial trend (1st-3rd) of interest specified as "linear", "quadratic", or "cubic"
`assurance`	Value with which confidence can be placed that describes the likelihood of obtaining a confidence interval less than the value specified (e.g, .80, .90, .95)

Value

A `data.frame` (class `dmar_tbl`) with a single row, `necessary_n_per_group`, giving the necessary number of subjects *per group* (the total study size is twice this value) for the combination of the desired confidence interval width, confidence level, optional assurance, and the population parameters at the specified design.

Note

Like in all formal sample size planning methods that require the value of one or more population parameter(s), if the population parameters are incorrectly specified, there is no guarantee that the sample size this function returns will be accurate. Of course, the further away from the true values, the further away the true sample size will tend to be.

The number of timepoints in a study (say M) is defined by $f \times D + 1$, where f is the frequency and D is the duration.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Kelley, K., & Rausch, J. R. (2011). Sample size planning for longitudinal models: Accuracy in parameter estimation for polynomial change parameters. *Psychological Methods, 16*(4), 391–405. doi:10.1037/a0023352

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapters 11, 15.)

Raudenbush, S. W., & Liu, X.-F. (2001). Effects of study duration, frequency of observation, and sample size on power in studies of group differences in polynomial change. *Psychological Methods, 6*(4), 387–401. doi:10.1037/1082989X.6.4.387

See Also

ss_power_pcm for the power analytic analog (planning to detect the group-by-time change difference rather than to estimate it precisely) on the same model, and ss_aipe_pcm_sensitivity for a Monte Carlo check of how parameter misspecification affects the plan.

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# The examples reproduce the tolerance-of-antisocial-thinking illustration
# of Kelley and Rausch (2011, Tables 1 and 2), which draws on the National
# Youth Survey data also used by Raudenbush and Liu (2001). The
# within-subject (i.e., Level 1) error variance is 0.0262 and the
# between-subject slope variance is 0.003.
# The planner finds the sample size needed for a confidence interval on the
# group-by-time slope difference that is no wider than `width`. The returned
# necessary_n_per_group is per group, so the total study size is twice that. Unlike
# power analysis, the value of the slope is not needed here: the confidence
# interval width does not depend on it.

# (1) Expected-width planning. With five measurement occasions
#     (M = frequency * duration + 1 = 1 * 4 + 1) and a target width of
#     0.025, the expected 95% confidence interval is sufficiently narrow at
#     278 subjects per group (Kelley & Rausch, 2011, Table 1, T = 5).
ss_aipe_pcm(true_var_trend = 0.003, true_var_error = 0.0262,
            duration = 4, frequency = 1, width = 0.025, conf_level = .95)

# (2) More measurement occasions sharpen the estimate. Extending the study
#     so that M = 10 (duration = 9, frequency = 1) cuts the expected-width
#     requirement from 278 to 165 per group (Kelley & Rausch, 2011, Table 1,
#     T = 10).
ss_aipe_pcm(true_var_trend = 0.003, true_var_error = 0.0262,
            duration = 9, frequency = 1, width = 0.025, conf_level = .95)

# (3) A wider tolerated interval costs less. Relaxing the target width from
#     0.025 to 0.05 at M = 5 drops the requirement from 278 to 71 per group
#     (Kelley & Rausch, 2011, Table 1, T = 5).
ss_aipe_pcm(true_var_trend = 0.003, true_var_error = 0.0262,
            duration = 4, frequency = 1, width = 0.05, conf_level = .95)

# (4) Adding an assurance parameter. Requiring 85% assurance that the
#     realized confidence interval will be no wider than 0.025 raises the
#     M = 5 requirement from 278 to 295 per group (Kelley & Rausch, 2011,
#     Table 2, T = 5). Assurance guards against the expected-width plan being
#     too small for the particular sample obtained.
ss_aipe_pcm(true_var_trend = 0.003, true_var_error = 0.0262,
            duration = 4, frequency = 1, width = 0.025, conf_level = .95,
            assurance = .85)

# (5) A higher assurance costs more. Demanding 99% assurance rather than 85%
#     raises the per-group requirement further, from 295 to 316.
```

```
ss_aipe_pcm(true_var_trend = 0.003, true_var_error = 0.0262,
            duration = 4, frequency = 1, width = 0.025, conf_level = .95,
            assurance = .99)
```

`ss_aipe_pcm_sensitivity` *Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for a Polynomial Change Parameter*

Description

Quantifies how much misspecification of the population between-subject slope variance and within-subject error variance distorts an AIPE-based sample size plan for the group-by-time polynomial change parameter. On each replication the function simulates two independent groups of *n* subjects each, measured at $M = f \times D + 1$ timepoints, where every subject has a true linear slope drawn from a normal distribution with mean 0 and variance `true_var_trend`, and within-subject observations have residual variance `true_var_error`. Subject-level OLS slopes are computed in each group, the between-group difference in mean slopes (the change parameter β_{m1} that `ss_aipe_pcm` plans for) is estimated, and a two-group *t*-confidence interval on that difference (pooled standard error, $2n - 2$ degrees of freedom) is recorded. The function only handles `trend = "linear"` in the simulator; for quadratic / cubic trends, the planner's closed-form solution is still available via `ss_aipe_pcm`.

Usage

```
ss_aipe_pcm_sensitivity(
  true_var_trend = NULL,
  true_var_error = NULL,
  supposed_var_trend = NULL,
  supposed_var_error = NULL,
  duration,
  frequency,
  width,
  specified_n_per_group = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_var_trend`	Population between-subject variance of the polynomial change coefficient (the data generating $\sigma^2_{v_m}$ of Kelley & Rausch, 2011).
`true_var_error`	Population within-subject error variance (σ^2_ϵ).

`supposed_var_trend`
: Planning value of `true_var_trend` passed to `ss_aipe_pcm`; supply this and `supposed_var_error`, or supply `specified_n_per_group`.

`supposed_var_error`
: Planning value of `true_var_error` passed to `ss_aipe_pcm`.

`duration`
: Study duration (in time units).

`frequency`
: Number of measurements per unit time. The number of measurement occasions is $f \times D + 1$ (the row `n_occasions`).

`width`
: Desired full width of the CI on the between-group difference in change parameters (β_{m1}).

`specified_n_per_group`
: Number of subjects per group to evaluate (incompatible with the supposed-variance arguments).

`conf_level`
: Confidence level (default `0.95`).

`assurance`
: Optional assurance probability passed to the planner.

`G`
: Number of Monte Carlo replications.

`print_iter`
: Logical.

`filename`
: Optional path to a CSV file; when supplied, the per-replication results (the slope difference, its interval limits and width, and the two tail misses) are written there, appended when the file already exists, and a throwaway run should point it at `tempfile(fileext = ".csv")`; the default `NULL` writes nothing.

`seed`
: `NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Value

A `data.frame` with rows for mean / median / SD of the realized estimated slope difference and CI width, the proportion of intervals at or below `width`, tail-specific and overall non-coverage of the population slope difference (0 by construction in this simulator), and the input echoes, including `assurance` (present only when an assurance was supplied).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Rausch, J. R. (2011). Sample size planning for longitudinal models: Accuracy in parameter estimation for polynomial change parameters. *Psychological Methods, 16*(4), 391–405. doi:10.1037/a0023352

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapters 11 and 15.)

See Also

`ss_aipe_pcm`, `ss_aipe_mixed_effects_sensitivity`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivi`
`ss_aipe_mean()`, `ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared(`
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`, `ss_aipe_semipartial_r()`,
`ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`, `ss_aipe_two_phase_cost()`,
`ss_aipe_two_phase_sensitivity()`

Examples

```
# Every replication simulates two full groups of subjects, fits a slope
# for each subject, and forms a confidence interval on the difference in
# mean slopes. G = 10 keeps the example quick; a reported sensitivity
# study deserves the default G = 1000. With the planning values equal to
# the population values, the realized mean width should sit close to
# the target.
set.seed(113)
ss_aipe_pcm_sensitivity(
```

```
  true_var_trend = 0.003, true_var_error = 0.0262,
  supposed_var_trend = 0.003, supposed_var_error = 0.0262,
  duration = 4, frequency = 1, width = 0.05,
  G = 10, print_iter = FALSE
)
```

`ss_aipe_r` *Sample Size for AIPE on a Pearson Correlation*

Description

Determines the sample size needed for a confidence interval on a population Pearson correlation ρ to have a desired width (accuracy in parameter estimation; Kelley & Maxwell, 2003). The interval planned for is the Fisher's Z interval that `correlations_test` reports for `method = "pearson"`: the correlation is transformed as $Z(r) = \operatorname{atanh}(r)$, an interval with standard error $1/\sqrt{N-3}$ is formed on the Z scale, and the limits are back-transformed through $\tanh(\cdot)$ (Fisher, 1921; Bonett & Wright, 2000, Equation 2). Because the plan targets the same interval the analysis will report, the planned width and the analyzed width agree.

Usage

```
ss_aipe_r(true_cor, width, conf_level = 0.95, assurance = NULL)
```

Arguments

`true_cor`	Population Pearson correlation ρ the sample size is planned for, in $(-1, 1)$. The planner treats this value as the truth; `ss_aipe_r_sensitivity` shows what the plan delivers when the truth (`true_cor` there) differs from the value planned with (`supposed_cor` there).
`width`	Desired full width of the confidence interval on the correlation.
`conf_level`	Desired confidence level (default `0.95`). A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in `ci_nc_t`).
`assurance`	Optional. The probability that the realized interval is no wider than `width`. The function returns the smallest sample size, at or above the sample size planned without it, at which that probability is at least `assurance`. A single number above 0.5 and below 1; when `NULL` (the default) the plan is for the width at $r = \rho$ only.

Details

Closed-form first pass. On the Fisher's Z scale the interval has half-width $z_{1-\alpha/2}/\sqrt{N-3}$, and the delta method maps it back to the correlation scale as approximately

$$w \;\approx\; 2\, z_{1-\alpha/2}\, \frac{1-\rho^2}{\sqrt{N-3}},$$

which solves to the first-stage approximation of Bonett and Wright (2000),

$$n_0 \;=\; 3 + \left\lceil 4\,(z_{1-\alpha/2})^2(1-\rho^2)^2/w^2 \right\rceil .$$

Here and below ρ is `true_cor`, w is `width`, and $1-\alpha$ is `conf_level`.

Exact iteration. The back-transformed width depends on ρ through $\tanh(\cdot)$, so the delta method approximation can land a few observations off in either direction. Starting from n_0, the function evaluates the exact back-transformed width $\tanh(z_\rho + z_{1-\alpha/2}/\sqrt{N-3}) - \tanh(z_\rho - z_{1-\alpha/2}/\sqrt{N-3})$ and steps the integer N until it is the smallest sample size whose width is at or below `width`. Where Bonett and Wright (2000) stop after a single second-stage adjustment, this search is exact.

The assurance step. A plan without `assurance` makes the interval as narrow as the target when the sample correlation equals `true_cor`; in many samples it is wider. With `assurance` the function finds the sample size at which the realized interval is no wider than the target with at least that probability, from the sampling distribution of the sample correlation *r* of *N* independent observations from a bivariate normal population:

1. At a given *N* the width of the interval depends on the sample only through *r*, and it decreases as $|r|$ grows. The realized interval is therefore no wider than the target exactly when $|r| \geq r^*(N)$, where $r^*(N)$ is the root of $\tanh(\operatorname{atanh}(r) + z_{1-\alpha/2}/\sqrt{N-3}) - \tanh(\operatorname{atanh}(r) - z_{1-\alpha/2}/\sqrt{N-3}) = w$. When the width at $r = 0$, the widest the interval can be, already meets the target, every sample does and the probability is 1.
2. The distribution function of *r* is computed as

$$F(c) \;=\; \int_0^1 T_{2N-3}\!\left(\sqrt{2N-3}\left\{\frac{c\sqrt{1-b}}{\sqrt{1-c^2}} - \frac{\rho\sqrt{b}}{\sqrt{1-\rho^2}}\right\}\right) g(b)\,db,$$

   where T_{2N-3} is the distribution function of Student's *t* on $2N-3$ degrees of freedom and g is the density of the beta distribution with parameters $(N-1)/2$ and $(N-2)/2$. The form follows from writing $r/\sqrt{1-r^2}$ as $(Z + \theta\sqrt{U})/\sqrt{V}$, with Z standard normal, U and V chi square on $N-1$ and $N-2$ degrees of freedom, all independent, and $\theta = \rho/\sqrt{1-\rho^2}$, and then conditioning on $U/(U+V)$. It is the computation `ss_aipe_partial_r` uses with no variables partialed out.
3. The probability at *N* is $P_N = 1 - F(r^*(N)) + F(-r^*(N))$, and the function returns the smallest *N*, at or above the sample size planned without `assurance`, with $P_N \geq$ `assurance`, found by doubling a step upward and then bisecting. `assurance_attained` is P_N at that *N*.

The probability is exact, up to numerical integration, under bivariate normality. It was checked against the density of *r* computed from its hypergeometric series, and against the proportion of intervals no wider than the target among samples of bivariate normal data. The search does not go below the plan without `assurance`, so a plan with `assurance` is never smaller than the plan without it. Near $\rho = 0$ a slightly smaller *N* can attain the probability, because there the interval at $r = 0$ is barely wider than the target (at $\rho = 0$ and a width of .10 the probability is .917 at 1537, one below the plan without `assurance`). At and above the plan without `assurance`, P_N did not decrease from one *N* to the next in any of the 15 planning sets scanned ($|\rho|$ of 0, .1, .5, .9, and .99 with widths of .05, .1, and .2). The result depends on `true_cor` only through $|\rho|$. At `true_cor = 0` the plan without `assurance` already makes every interval as narrow as the target, since the interval is widest at $r = 0$; `assurance` then adds nothing and `assurance_attained` is 1. As $|\rho|$ grows the width becomes more variable from sample to sample and the step adds more: for a width of .10 and an assurance of .80 the plan moves from 62 to 83 at $\rho = .90$.

Why the step is not a chi square factor. Multiplying the sample size planned without `assurance` by $\chi^2_{\gamma,N-2}/(N-2)$, the γ quantile of a chi square on $N-2$ degrees of freedom over its degrees of freedom, with γ the `assurance`, does not serve here. That factor is the γ quantile of the ratio of a sample variance to its population value, so it applies to an interval whose width is proportional to a sample standard deviation; the width of the interval for a correlation depends on the sample through *r* instead. At `true_cor = .90`, a width of .10, and an assurance of .80 the factor gives 72, where the probability is .672 (83 are needed); with an assurance of .99 it gives 92, where the probability is .887 (122 are needed). At `true_cor = 0` it gives more than the plan without `assurance`, which already attains a probability of 1.

The planning value matters least near zero. At a fixed sample size the back-transformed width is largest at $\rho = 0$ and shrinks as $|\rho|$ grows, so a planning value closer to zero yields a larger, more conservative sample size. When little is known about the population correlation, `true_cor = 0` gives the sample size that suffices for any population value.

When to use simple vs. partial correlation planning. Use this function when the inferential target is the correlation between two variables with nothing partialed out. When the target is the correlation after statistically controlling for other variables, see `ss_aipe_partial_r`.

The Monte Carlo companion `ss_aipe_r_sensitivity` evaluates how the plan behaves when the population correlation differs from the planning value.

Value

A `data.frame` with the columns `term` and `value` and one row per term:

`necessary_N` the planned total sample size.

`expected_width` the full width of the interval at `necessary_N` when the sample correlation equals `true_cor`.

`true_cor`, `width`, `conf_level` the inputs.

`assurance` the requested probability; present only when `assurance` was given.

`assurance_attained` the probability, at `necessary_N` and `true_cor`, that the realized interval is no wider than `width`; present only when `assurance` was given. It is at least `assurance`, and above it because the sample size is a whole number or because the plan without `assurance` already exceeds the request.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G., & Wright, T. A. (2000). Sample size requirements for estimating Pearson, Kendall and Spearman correlations. *Psychometrika, 65*(1), 23–28. doi:10.1007/BF02294183

Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron, 1*, 3–32.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on correlations.)

See Also

ss_aipe_r_sensitivity, correlations_test, ss_aipe_partial_r, ss_power_r, convert_r_Z

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(),
ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivit
ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(),
ss_aipe_pcm_sensitivity(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(),
ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(),
ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# Plan N so the 95% CI on the Pearson correlation has full width
# at most 0.20, when the anticipated correlation is 0.30.
ss_aipe_r(true_cor = 0.30, width = 0.20)

# A narrower target width requires a larger sample size.
ss_aipe_r(true_cor = 0.30, width = 0.10)

# The smallest N at which the realized interval is no wider than
# 0.20 with probability at least .80. The last row is the probability
# at the returned N.
ss_aipe_r(true_cor = 0.30, width = 0.20, assurance = 0.80)

# At a large correlation the width varies more from sample to sample,
# and the assurance step adds more cases.
ss_aipe_r(true_cor = 0.90, width = 0.10)
ss_aipe_r(true_cor = 0.90, width = 0.10, assurance = 0.80)

# Planning at true_cor = 0 gives the sample size that suffices for any
# population correlation, since the interval is widest there.
ss_aipe_r(true_cor = 0, width = 0.20)
```

---

ss_aipe_R2 *Sample Size Planning for Accuracy in Parameter Estimation for the Multiple Correlation Coefficient*

Description

Determines necessary sample size for the multiple correlation coefficient so that the confidence interval for the population multiple correlation coefficient is sufficiently narrow. Optionally, there is a certainty parameter that allows one to be a specified percent certain that the observed interval will be no wider than desired.

Usage

```
ss_aipe_R2(
  true_R2 = NULL,
  conf_level = 0.95,
  width = NULL,
  random_predictors = TRUE,
  which_width = "Full",
  p = NULL,
  assurance = NULL,
  verify_ss = FALSE,
  tol = 1e-09,
  G_initial = 500,
  G = 10000
)
```

Arguments

true_R2	Population squared multiple correlation coefficient the sample size is planned for. The planner treats this value as the truth; ss_aipe_R2_sensitivity shows what the plan delivers when the truth (true_R2 there) differs from the value planned with (supposed_R2 there).
conf_level	Confidence interval level (e.g., .95, .99, .90); 1-Type I error rate
width	Width of the confidence interval (see which_width)
random_predictors	Whether or not the predictor variables are random (set to TRUE) or are fixed (set to FALSE)
which_width	Which width width refers to: "Full" (the default), the full width of the confidence interval; "Lower", the distance from the expected sample R^2 to the lower limit; or "Upper", the distance from the upper limit to the expected sample R^2. An abbreviation such as "L" stands for the value it begins. With "Lower" or "Upper" the plan is for that distance at the expected sample R^2: the function returns the smallest N such that the distance is no larger than width at that N and at every larger sample size. (The search begins at $N = 2p + 3$, so no such plan is smaller.) The distance to the upper limit does not fall steadily as N grows. With several predictors and a small N the expected sample R^2 is well above the population value and the upper limit is close to that expected value or below it, so the distance is small or negative; it rises to a maximum as N grows and then falls. With true_R2 = .70, width = .05, p = 10, and which_width = "Upper" the distance is .043 at N = 23, is larger than .05 at every N from 27 to 201, and the function returns 202. expected_R2 and ci_R2 give the distance at any N. A very small width (.0025 or less in the calls examined) can be narrower than can be planned with R's distribution functions, and such a half-width call is refused with an error that names width. To confirm that the distance stays within width the search needs the limits at sample sizes several times the plan, and there the noncentrality parameter is in the millions, beyond the range in which R's noncentral F and beta distribution functions (pf and pbeta) are accurate (their help says the code is meant for moderate values of the noncentrality parameter), or

	the sample sizes pass 10,000,000, the largest the search tries. The refusal does not mean that no sample size meets `width`. `assurance` can be used only with `"Full"`: given with `"Lower"` or `"Upper"` it is refused with an error, because the assurance step does not deliver the requested probability for a half-width. `verify_ss = TRUE` is refused with a half-width as well.
`p`	The number of predictor variables
`assurance`	Value with which confidence can be placed that describes the likelihood of obtaining a confidence interval less than the value specified (e.g, .80, .90, .95). It can be used only with `which_width = "Full"`.
`verify_ss`	Evaluates numerically via an internal Monte Carlo simulation the exact sample size given the specifications
`tol`	The tolerance of the iterative function `ci_nc_t` for convergence
`G_initial`	Used only with `verify_ss = TRUE`: the number of Monte Carlo replications for each candidate *N* in the coarse search that brackets the sample size (default 500).
`G`	Used only with `verify_ss = TRUE`: the number of Monte Carlo replications in the final pass that confirms the sample size near that bracket (default 10000).

Details

This function determines a necessary sample size so that the expected confidence interval width for the squared multiple correlation coefficient is sufficiently narrow (when `assurance=NULL`) so that the obtained confidence interval is no larger than the value specified with some desired degree of certainty (i.e., a probability that the obtained width is less than the specified width). The method depends on whether or not the regressors are regarded as fixed or random. This is the case because the distribution theory for the two cases is different and thus the confidence interval procedure is conditional on the type of regressors. The default methods are approximate but can be made exact with the specification of `verify_ss=TRUE`, which performs an a priori Monte Carlo simulation study. Kelley (2008) and Kelley & Maxwell (2008) detail the methods used in the function, with the former focusing on random regressors and the latter on fixed regressors.

What the assurance plan delivers. The assurance procedure is not exact. In simulation with a population R^2 of .10, 4 predictors, and width .20 the realized probability was .86 for a requested .90 and .73 for a requested .80; with a population R^2 of .40 and 2 predictors it was 1.00 for a requested .90. The largest shortfall found was at a population R^2 of .10 with 5 predictors and width .30, where the realized probability was .67 for a requested .80. (Each value is the proportion of 2,000 simulated samples, at the planned sample sizes of 172, 156, 227, and 75, whose interval was no wider than the target; its Monte Carlo standard error is .008 to .011 for the three that fell short.) The shortfall appears where the population R^2 is small and there are several predictors. These are the plans that were simulated, not a limit on the shortfall, so a plan that was not simulated can fall further short. Check a plan with `ss_aipe_R2_sensitivity`, whose `pct_ci_less_width` row is the realized probability at the planned sample size, or use `verify_ss = TRUE`.

Using `verify_ss = TRUE` is recommended. It takes the method implied sample size as a starting value and then finds, by Monte Carlo simulation, the sample size that meets the goals specified. When `verify_ss=TRUE`, the default number of iterations is 10,000 but this can be changed by specifying G=5000 (or some other value; 10000 is the recommended). When `verify_ss=TRUE` is specified, an internal function `verify_ss_aipe_r2` calls upon the `ss_aipe_R2_sensitivity` function for purposes of the internal Monte Carlo simulation study, with `G_initial` replications

for each candidate N in the coarse search that brackets the answer and `G` replications in the final pass that confirms the sample size near that bracket.

Value

A 1-row `data.frame` with columns `term` and `value`. The `term` value is `"necessary_N"` and `value` is the necessary total sample size N given the input specifications.

Note

With `verify_ss = TRUE` the function can take some time to converge (e.g., several minutes to a quarter hour) because the closed form approximation is followed by an a priori Monte Carlo simulation. The default `verify_ss = FALSE` returns the closed form approximation only and is essentially instantaneous.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J. & Olejnik, S. (2000). Determining sample size for accurate estimation of the squared multiple correlation coefficient. *Multivariate Behavioral Research, 35*, 119–137. doi:10.1207/s15327906mbr3501_5

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*, 524–555. doi:10.1080/00273170802490632

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Steiger, J. H., & Fouladi, R. T. (1992). R2: A computer program for interval estimation, power calculations, sample size estimation, and hypothesis testing in multiple regression. *Behavior Research Methods, Instruments, & Computers, 24*(4), 581–582. doi:10.3758/BF03203611

See Also

`ci_R2`, `ci_nc_t`, `ss_aipe_R2_sensitivity`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# 1. Closed form planner under random predictors (the typical case).
#    Sample size sufficient for the expected CI width on rho^2 to be .10.
ss_aipe_R2(true_R2 = .50, conf_level = .95, width = .10,
           which_width = "Full", p = 5, random_predictors = TRUE)

# 2. The same target under fixed predictors (planned dosing levels,
#    factorial covariates, and the like) needs a smaller N, since fixed
#    predictors contribute no sampling variability of their own.
ss_aipe_R2(true_R2 = .50, conf_level = .95, width = .10,
           which_width = "Full", p = 5, random_predictors = FALSE)

# 3. An assurance of .85, so that the realized width is no larger than the
#    target in 85 percent of replications rather than only on average,
#    needs a larger N than the expected width plan in (1).
ss_aipe_R2(true_R2 = .50, conf_level = .95, width = .10,
           which_width = "Full", p = 5, assurance = .85,
           random_predictors = TRUE)

# 4. verify_ss = TRUE follows the closed form approximation with an a
#    priori Monte Carlo simulation of the realized width, starting from
#    the closed form answer and returning the sample size the simulation
#    settles on, which is what a plan meant to be defended deserves. The
#    coarse search runs G_initial replications per candidate N and the final pass
#    runs G; the small counts here keep the example quick, and a reported
#    plan deserves the defaults of G_initial = 500 and G = 10000.
set.seed(113)
ss_aipe_R2(true_R2 = .50, conf_level = .95, width = .10,
           which_width = "Full", p = 5, random_predictors = TRUE,
           verify_ss = TRUE, G_initial = 10, G = 30)
```

`ss_aipe_R2_sensitivity`

Sensitivity Analysis for Sample Size Planning With the Goal of Accuracy in Parameter Estimation (I.e., a Narrow Observed Confidence Interval)

Description

Given `supposed_R2` and `true_R2`, one can perform a sensitivity analysis to determine the effect of a misspecified population squared multiple correlation coefficient using the Accuracy in Parameter Estimation (AIPE) approach to sample size planning. The function evaluates the effect of a misspecified `true_R2` on the width of obtained confidence intervals.

Usage

```
ss_aipe_R2_sensitivity(
  true_R2 = NULL,
```

```
  supposed_R2 = NULL,
  width = NULL,
  p = NULL,
  random_predictors = TRUE,
  specified_N = NULL,
  assurance = NULL,
  conf_level = 0.95,
  generate_random_predictors = TRUE,
  true_cor_YX = 0.3,
  true_cor_XX = 0.3,
  G = 10000,
  print_iter = TRUE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_R2`	Value of the population squared multiple correlation coefficient
`supposed_R2`	Supposed value of the squared multiple correlation coefficient, the planning value from which the sample size is planned
`width`	Full confidence interval width of interest
`p`	Number of predictors
`random_predictors`	Whether or not the sample size procedure and the simulation itself should be based on random (set to `TRUE`) or fixed predictors (set to `FALSE`)
`specified_N`	Selected sample size to use in order to determine distributional properties at a given value of sample size. Give `supposed_R2` or `specified_N`, not both; a call that gives both is refused with a condition of class `dmar_argument_invalid`.
`assurance`	Parameter to ensure confidence interval width with a specified degree of certainty
`conf_level`	Confidence interval coverage (symmetric coverage)
`generate_random_predictors`	Specify whether the simulation should be based on random (default) or fixed regressors.
`true_cor_YX`	Value of the correlation between *y* (dependent variable) and each of the *x* variables (independent variables)
`true_cor_XX`	Value of the correlation among the *x* variables (independent variables)
`G`	Number of generations (i.e., replications) of the simulation
`print_iter`	Should the iteration number (between 1 and `G`) during the run of the function
`filename`	Optional path of a CSV file to receive the per-replication results (the confidence limits `ci_lower` and `ci_upper`, the observed R^2 `R2`, the one-sided widths `ci_width_lower` and `ci_width_upper`, measured from the observed R^2 to each limit, and the full width `ci_width`, the quantities the `*_ci_width_lower`,

*_ci_width_upper, and *_ci_width rows summarize), overwriting any file already at that path; the default NULL writes nothing, and a throwaway run that wants the file should point it at tempfile(fileext = ".csv").

seed — NULL (the default) to draw from the current random number stream, or a single number that makes the run reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

When supposed_R2=true_R2, the results are that of a simulation study when all assumptions are satisfied. Rather than specifying supposed_R2, one can specify specified_N to determine the results of a particular sample size (when doing this supposed_R2 cannot be specified).

The sample size estimation procedure technically assumes multivariate normal variables (p+1) with fixed predictors (x/independent variables), yet the function assumes random multivariate normal predictors (having a p+1 multivariate distribution). As Gatsonis and Sampson (1989) note in the context of statistical power analysis (recall this function is used in the context of precision), there is little difference in the outcome.

In the behavioral, educational, and social sciences, predictor variables are almost always random, and thus random_predictors should generally be used. random_predictors=TRUE specifies how both the sample size planning procedure and the confidence intervals are calculated based on the random predictors/regressors. The internal simulation generates random or fixed predictors/regressors based on whether variables predictor variables are random or fixed. However, when random_predictors=FALSE, only the sample size planning procedure and the confidence intervals are calculated based on the parameter. The parameter generate_random_predictors (where the default is TRUE so that random predictors/regressors are generated) allows random or fixed predictor variables to be generated. Because the sample size planning procedure and the internal simulation are both specified, for purposes of sensitivity analysis random/fixed can be crossed to examine the effects of specifying sample size based on one but using it on data based on the other.

Value

A data.frame with columns term and value summarizing the Monte Carlo sensitivity analysis across G replications. The term entries are: mean_ci_lower, median_ci_lower, sd_ci_lower, mean_ci_upper, median_ci_upper, sd_ci_upper (summaries of the realized confidence limits); mean_R2, median_R2, sd_R2 (summaries of the observed R^2); mean_ci_width_lower, median_ci_width_lower, sd_ci_width_lower, mean_ci_width_upper, median_ci_width_upper, sd_ci_width_upper (summaries of the one-sided widths, measured from the observed R^2 to each limit); mean_ci_width, median_ci_width, sd_ci_width (summaries of the full interval widths); pct_ci_less_width (proportion of intervals with width at or below the planning target width); pct_ci_miss_low and pct_ci_miss_high (tail-specific empirical non-coverage of true_R2); total_type_I_error (overall empirical non-coverage, the sum of the two tails); and the input echoes total_N (the sample size evaluated), p, true_R2, supposed_R2 (NA when specified_N was supplied instead), width, conf_level, and assurance (present only when an assurance was supplied). The proportion rows are on the 0 to 1 scale, not percentages. The number of replications on which a confidence interval could not be obtained is attr(x, "simulation")$failed (see the section on the record of the simulation).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J. & Olejnik, S. (2000). Determining sample size for accurate estimation of the squared multiple correlation coefficient. *Multivariate Behavioral Research, 35*, 119–137. doi:10.1207/s15327906mbr3501_5

Gatsonis, C. & Sampson, A. R. (1989). Multiple Correlation: Exact power and sample size calculations. *Psychological Bulletin, 106*(3), 516–524.

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals, *Multivariate Behavioral Research, 43*(4), 524–555. doi:10.1080/00273170802490632

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Steiger, J. H., & Fouladi, R. T. (1992). R2: A computer program for interval estimation, power calculations, sample size estimation, and hypothesis testing in multiple regression. *Behavior Research Methods, Instruments, & Computers, 24*(4), 581–582. doi:10.3758/BF03203611

See Also

ci_R2, ci_nc_t, ss_aipe_R2

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# The planner used .4 for a population squared multiple correlation
# coefficient that is really .5. G = 25 keeps the example quick; a
# reported sensitivity analysis deserves the default G of 10000.
ss_aipe_R2_sensitivity(true_R2 = .5, supposed_R2 = .4, width = .10, p = 5,
                       conf_level = 0.95, G = 25, print_iter = FALSE,
                       seed = 113)
```

---

ss_aipe_rc *Sample Size Necessary for the Accuracy in Parameter Estimation Approach for an Unstandardized Regression Coefficient of Interest*

Description

A function used to plan sample size from the accuracy in parameter estimation perspective for an unstandardized regression coefficient of interest given the input specification.

Usage

```
ss_aipe_rc(
  true_R2_Y_X = NULL,
  true_R2_j_X_without_j = NULL,
  p = NULL,
  true_reg_coef_j = NULL,
  width,
  which_width = "Full",
  true_sd_Y = 1,
  true_sd_X = 1,
  true_cor_XX = NULL,
  true_cor_YX = NULL,
  which_predictor = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  conf_level = 0.95,
  assurance = NULL,
  assurance_method = "analytic"
)
```

Arguments

true_R2_Y_X	Population value of the squared multiple correlation coefficient
true_R2_j_X_without_j	Population value of the squared multiple correlation coefficient predicting the *j*th predictor variable from the remaining *p*-1 predictor variables
p	The number of predictor variables
true_reg_coef_j	The population regression coefficient for the *j*th predictor variable (i.e., the predictor of interest), in the units true_sd_Y and true_sd_X describe: the standardized coefficient times true_sd_Y / true_sd_X. On the covariance structure path it may be left out; when it is given it must equal that value (see ss_aipe_reg_coef)
width	The desired width of the confidence interval
which_width	Which Width ("Full", "Lower", or "Upper") the width refers to (at present, only "Full" can be specified)
true_sd_Y	The population standard deviation of *Y* (i.e., the dependent variable)
true_sd_X	The population standard deviation of the *j*th *X* variable (i.e., the predictor variable of interest)
true_cor_XX	Population correlation matrix for the *p* predictor variables
true_cor_YX	Population *p* length vector of correlation between the dependent variable (*Y*) and the *p* independent variables
which_predictor	Identifies which of the *p* predictors is of interest
alpha_lower	Type I error rate for the lower confidence interval limit, at least 1e-9 and at most .5; a tail area below 1e-9 is refused, as in ss_aipe_reg_coef
alpha_upper	Type I error rate for the upper confidence interval limit, at least 1e-9 and at most .5
conf_level	Desired level of confidence for the computed interval (i.e., 1 - the Type I error rate); a level above .999999998, whose tail areas are below 1e-9, is refused
assurance	Optional probability with which the realized confidence interval is to be no wider than width; when NULL (the default) the plan is for the width of the interval at the expected values of the sample squared multiple correlations (see ss_aipe_reg_coef), not for the mean width over samples
assurance_method	How the probability that the realized interval is no wider than width is found, used only with assurance. Only "analytic" (the default) is offered here: the probability is computed exactly (an *F* probability) and reported as assurance_attained, and the plan is the smallest sample size, at or above the plan without assurance, at which it is at least assurance. "simulation", which ss_aipe_sc_ancova, ss_aipe_reg_coef with noncentral = TRUE, and ss_aipe_src offer, where the probability has no exact form, is refused with a condition of class dmar_argument_invalid. See "The assurance step" in ss_aipe_reg_coef.

Details

Not all of the arguments need to be specified, only those that provide all of the necessary information so that the sample size can be determined for the conditions specified.

The interval planned for is the central *t* interval for the coefficient in the units of the data, the interval `ci_rc` reports, for random predictors from a multivariate normal population, with the *t* quantiles of its two tails. The plan without `assurance` is the smallest sample size, of at least $p + 2$, at which that interval, at the expected squared multiple correlations, is no wider than `width`; it can be a case or more smaller than that of **MBESS** (see `ss_aipe_reg_coef`). With `assurance` the plan is the smallest sample size at which that interval is no wider than `width` with at least that probability, computed exactly (`assurance_method = "analytic"`). The published chi square approximation of Kelley and Maxwell (2003) can give a smaller or a larger sample size than the exact plan.

Value

A `data.frame` with columns `term` and `value`: `necessary_N`, the necessary total sample size; with `assurance`, the rows `assurance` and `assurance_attained`, the exact probability at `necessary_N` that the realized interval is no wider than `width`. With `assurance` the printed footer names the assurance method, which the attribute `assurance_method` records, with `assurance_probability` `"exact"`.

Note

This function calls upon `ss_aipe_reg_coef` in DMAR but has a different naming scheme. See `ss_aipe_reg_coef` for more details.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

See Also

`ss_aipe_reg_coef_sensitivity`, `ci_nc_t`, `ss_aipe_reg_coef`, `ss_aipe_src`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Exchangeable correlation structure
true_cor_YX <- c(.3, .3, .3, .3, .3)
true_cor_XX <- rbind(c(1, .5, .5, .5, .5), c(.5, 1, .5, .5, .5), c(.5, .5, 1, .5, .5),
c(.5, .5, .5, 1, .5), c(.5, .5, .5, .5, 1))

ss_aipe_rc(width = .1, which_width = "Full", true_sd_Y = 1, true_sd_X = 1,
           true_cor_XX = true_cor_XX,
           true_cor_YX = true_cor_YX, which_predictor = 1, conf_level = 1 - .05)

# The smallest N at which the interval is no wider than .1 with
# probability at least .85; the last row is that probability.
ss_aipe_rc(width = .1, which_width = "Full", true_sd_Y = 1, true_sd_X = 1,
           true_cor_XX = true_cor_XX,
       true_cor_YX = true_cor_YX, which_predictor = 1, conf_level = 1 - .05, assurance = .85)

# In the units of the data (Y with standard deviation 10, the predictor
# with standard deviation 2), a width of .1 * 10 / 2 gives the same plan.
ss_aipe_rc(width = .5, true_sd_Y = 10, true_sd_X = 2, true_cor_XX = true_cor_XX,
           true_cor_YX = true_cor_YX, which_predictor = 1, assurance = .85)
```

---

ss_aipe_rc_sensitivity

Sensitivity Analysis for Sample Size Planning From the Accuracy in Parameter Estimation Perspective for the Unstandardized Regression Coefficient

---

Description

Performs a sensitivity analysis when planning sample size from the Accuracy in Parameter Estimation Perspective for the unstandardized regression coefficient. It is ss_aipe_reg_coef_sensitivity under the name that pairs with ss_aipe_rc; ss_aipe_src_sensitivity is the counterpart for the standardized coefficient.

Usage

```
ss_aipe_rc_sensitivity(
  true_var_Y = NULL,
  true_cov_YX = NULL,
  true_cov_XX = NULL,
  supposed_var_Y = NULL,
  supposed_cov_YX = NULL,
  supposed_cov_XX = NULL,
  specified_N = NULL,
  which_predictor = 1,
  width = NULL,
  noncentral = FALSE,
```

```
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = TRUE,
  filename = NULL,
  assurance_method = c("analytic", "simulation"),
  seed = NULL
)
```

Arguments

true_var_Y	Population variance of the dependent variable (Y)
true_cov_YX	Population covariances vector between the p predictor variables and the dependent variable (Y)
true_cov_XX	Population covariance matrix of the p predictor variables
supposed_var_Y	Supposed variance of the dependent variable (Y), the planning value from which the sample size is planned
supposed_cov_YX	Supposed covariances vector (planning values) between the p predictor variables and the dependent variable (Y)
supposed_cov_XX	Supposed covariance matrix (planning values) of the p predictor variables
specified_N	Directly specified sample size (instead of planning one from the supposed covariance structure)
which_predictor	identifies which of the p predictors is of interest
width	Desired confidence interval width for the regression coefficient of interest, in the units of the data
noncentral	specify with a TRUE or FALSE statement whether or not the noncentral approach to sample size planning should be used; the noncentral plan is for a standardized coefficient, so a plan from the supposed values with noncentral = TRUE needs supposed variances of 1 for Y and the targeted predictor and is refused otherwise (see ss_aipe_reg_coef_sensitivity)
conf_level	desired level of confidence for the computed interval (i.e., 1 - the Type I error rate)
assurance	degree of certainty that the obtained confidence interval will be sufficiently narrow (i.e., the probability that the observed interval will be no larger than desired)
G	the number of generations (i.e., replications) of the simulation within the function
print_iter	specify with a TRUE/FALSE statement if the iteration number should be printed as the simulation within the function runs
filename	Optional path of a CSV file to receive the per-replication results (the unstandardized coefficient estimate reg_coef_j, its confidence limits ci_lower and ci_upper, the observed R^2 R2, the standard error se_reg_coef_j, and the t statistic t_value), appended when the file already exists and created otherwise;

the default NULL writes nothing, and a throwaway run that wants the file should point it at tempfile(fileext = ".csv").

assurance_method

Passed to ss_aipe_reg_coef, which plans the sample size from the supposed values with assurance: "analytic" (the default) for the smallest sample size at which the exact probability of an interval no wider than width is at least assurance with the central interval, and for the published approximation of **MBESS**, which does not compute that probability, with noncentral = TRUE; "simulation", offered with noncentral = TRUE only, for the smallest sample size at which a simulated estimate of the probability (made by the planner from its default G of 20000 samples with the random number stream that seed sets) is at least assurance. With noncentral = FALSE "simulation" is refused with a condition of class dmar_argument_invalid. Not used with specified_N or without assurance.

seed

NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Direct specification of true_cov_YX and true_cov_XX is necessary, even if one is interested in a single regression coefficient, so that the covariance/correlation structure can be specified when the simulation within the function runs.

Value

A data.frame with columns term and value summarizing the Monte Carlo sensitivity analysis, the table of ss_aipe_reg_coef_sensitivity: mean / median / SD summaries of the realized unstandardized regression coefficient (mean_reg_coef_j, median_reg_coef_j, sd_reg_coef_j), the realized interval widths, and the realized squared multiple correlation coefficient; the proportion of intervals at or below the planning target (pct_ci_less_width); the tail-specific and overall empirical non-coverage rates (pct_ci_miss_low, pct_ci_miss_high, total_type_I_error), all proportions on the 0 to 1 scale; and the input echoes (total_N, p, which_predictor, true_reg_coef_j, supposed_reg_coef_j, width, conf_level, and, when one was supplied, assurance). See ss_aipe_reg_coef_sensitivi for the full row list and the scale of each value.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Note

Note that when the true and supposed covariance structures agree (`true_cov_YX` equals `supposed_cov_YX` and `true_cov_XX` equals `supposed_cov_XX`), the results are not literally from a sensitivity analysis, rather the function performs a standard simulation study. A simulation study can be helpful in order to determine if the sample size procedure under or overestimates necessary sample size. See `ss_aipe_reg_coef_sensitivity` in DMAR for more details.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

See Also

`ss_aipe_reg_coef_sensitivity`, `ss_aipe_src_sensitivity`, `ss_aipe_reg_coef`, `ci_reg_coef`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Sensitivity analysis for an unstandardized regression coefficient
# with two correlated predictors. G is kept small here so the example
# runs quickly; raise G for a stable Monte Carlo summary.
set.seed(113)
cov_XX <- matrix(c(1, 0.3, 0.3, 1), nrow = 2)
cor_YX <- c(0.4, 0.3)
cov_YX <- cor_YX
```

```
ss_aipe_rc_sensitivity(
  true_var_Y = 1, true_cov_YX = cov_YX, true_cov_XX = cov_XX,
  supposed_var_Y = 1, supposed_cov_YX = cov_YX, supposed_cov_XX = cov_XX,
  which_predictor = 1, width = 0.20, conf_level = 0.95,
  G = 20, print_iter = FALSE
)
```

`ss_aipe_reg_coef` *Sample Size Planning for a Single Regression Coefficient (AIPE)*

Description

Computes the necessary sample size for the confidence interval on a targeted regression coefficient β_j, the quantity `true_reg_coef_j` stands for, in a multiple regression with p predictors to be no wider than a user-specified value. The coefficient is on the scale `true_sd_Y` and `true_sd_X` describe: with both 1, the default, its population value is the standardized coefficient. This is the accuracy in parameter estimation (AIPE) framework of Kelley and Maxwell (2003), targeted at a specific coefficient rather than at the omnibus R^2. The default (`noncentral = FALSE`) plans for the central *t* interval for the coefficient in the units of the data, the interval `ci_rc` reports; `noncentral = TRUE` plans for the noncentral *t* interval for the standardized coefficient (Kelley, 2007a), the interval `ci_src` reports, with unit standard deviations. The central *t* interval for the standardized coefficient, the interval of Kelley and Maxwell (2003, Equation 1), is a third interval, and `ss_aipe_src` with `noncentral = FALSE` plans for it; see "The interval planned for" in Details. Supplying `assurance` returns the smallest *N*, not below the plan without it (the plan for the width at the expected values of the sample squared multiple correlations), at which the realized interval is no wider than `width` with at least that probability, computed exactly for the central interval; for the noncentral interval it returns by default the published approximation that **MBESS** uses, which does not compute the probability, and with `assurance_method = "simulation"` the smallest *N* at which a simulated estimate of the probability reaches the request.

Usage

```
ss_aipe_reg_coef(
  true_R2_Y_X = NULL,
  true_R2_j_X_without_j = NULL,
  p = NULL,
  true_reg_coef_j = NULL,
  width,
  which_width = "Full",
  true_sd_Y = 1,
  true_sd_X = 1,
  true_cor_XX = NULL,
  true_cor_YX = NULL,
  which_predictor = NULL,
  noncentral = FALSE,
  alpha_lower = NULL,
```

```
  alpha_upper = NULL,
  conf_level = 0.95,
  assurance = NULL,
  assurance_method = c("analytic", "simulation"),
  G = 20000,
  seed = NULL
)
```

Arguments

true_R2_Y_X	Population value of $\rho^2_{Y \cdot X_1,\ldots,X_p}$, the squared multiple correlation of the outcome *Y* with all *p* predictors.
true_R2_j_X_without_j	Population value of $\rho^2_{X_j \cdot X_{-j}}$, the squared multiple correlation when the *j*th predictor is regressed on the remaining $p - 1$ predictors. Quantifies the multicollinearity faced by the targeted coefficient.
p	The number of predictor variables.
true_reg_coef_j	Population value β_j of the regression coefficient for the *j*th predictor, the predictor of interest, on the scale true_sd_Y and true_sd_X describe: with both 1 it is the standardized coefficient, and otherwise the unstandardized one, the standardized coefficient times true_sd_Y / true_sd_X (Kelley & Maxwell, 2003, p. 316); b_j is its sample estimate. On the covariance structure path it may be left out; when it is given it must equal the coefficient the correlations imply on that scale.
width	Desired (full) width of the two-sided confidence interval on β_j.
which_width	Which portion of the confidence interval width refers to. Only "Full" (or an abbreviation of it, such as "F") is currently implemented; "Lower", "Upper", and their abbreviations are refused.
true_sd_Y	Population standard deviation of *Y*, which with true_sd_X sets the scale of the coefficient and of width.
true_sd_X	Population standard deviation of the *j*th predictor.
true_cor_XX	Population correlation matrix for the *p* predictor variables. If supplied with true_cor_YX, true_R2_Y_X and true_R2_j_X_without_j are derived from the covariance structure. With one predictor it is matrix(1) (or the number 1), and p and which_predictor are then 1 when not given.
true_cor_YX	Length-*p* vector of population correlations between *Y* and the *p* predictors.
which_predictor	Which of the *p* predictors is the targeted coefficient.
noncentral	If TRUE, plans for the noncentral *t* interval for the standardized coefficient (Kelley, 2007a), the interval of ci_src. If FALSE (the default), plans for the central *t* interval for the coefficient in the units of the data, the interval of ci_rc, even when both standard deviations are 1; the central interval for the standardized coefficient is planned by ss_aipe_src with noncentral = FALSE. The noncentral path requires true_sd_Y = true_sd_X = 1 (*i.e.*, a standardized solution).

alpha_lower — Type I error rate for the lower confidence limit, at least 1e-9 and at most .5. A tail area below 1e-9 is refused: the noncentral *t* limits lose accuracy there, and the central plan keeps the same floor.

alpha_upper — Type I error rate for the upper confidence limit, at least 1e-9 and at most .5.

conf_level — Confidence level (i.e., $1-\alpha$, where α is the Type I error rate). Default `0.95`. Mutually exclusive with `alpha_lower` and `alpha_upper`. A level above .999999998, whose tail areas are below 1e-9, is refused.

assurance — Optional probability with which the realized confidence interval is to be no wider than `width`, a single number above .5 and below 1. When `NULL` (the default), the plan is the smallest *N* at which the interval is no wider than `width` when the two sample squared multiple correlations take their expected values (see "The interval planned for" in Details): the width at those expected values, not the mean of the width over samples.

assurance_method — How the probability that the realized interval is no wider than `width` is found, used only with `assurance` (see "The assurance step" in Details). `"analytic"` (the default) computes without simulation: for the central interval (`noncentral = FALSE`) the probability is exact, and the function returns the smallest sample size, at or above the plan without `assurance`, at which it is at least `assurance`, reported as `assurance_attained`; for the noncentral interval it returns the published approximation that **MBESS** uses, which does not compute the probability (`assurance_attained` is `NA`) and can attain less than `assurance`. `"simulation"`, offered for the noncentral interval only, returns the smallest sample size, at or above the plan without `assurance`, at which the probability estimated from `G` simulated samples is at least `assurance`, and reports that estimate. With `noncentral = FALSE` `"simulation"` is refused with a condition of class `dmar_argument_invalid`, since a simulated estimate of an exact probability would only add Monte Carlo error.

G — Number of simulated samples at each sample size the noncentral assurance step tries, and in its final estimate of the attained probability; a whole number of at least 100 (default 20000). Used and checked only with `noncentral = TRUE`, `assurance`, and `assurance_method = "simulation"`.

seed — Optional single number. When given, the noncentral assurance step is reproducible, and the random number stream of the calling session is left as it was. With `NULL` (the default) the draws come from the session's current stream. Used only where `G` is.

Details

Calling conventions. The function offers several mutually exclusive ways to supply the population information needed to plan N; the user picks the one most aligned with their available planning values. Specify exactly one of:

- *Covariance structure path.* Supply `true_cor_XX` and `true_cor_YX` along with `which_predictor`. The function derives $\rho^2_{Y \cdot X}$ and $\rho^2_{X_j \cdot X_{-j}}$ (the values of `true_R2_Y_X` and `true_R2_j_X_without_j`) from the covariance structure, and the population coefficient β_j, the *j*th element of $R_{XX}^{-1} r_{XY}$ (R_{XX} is `true_cor_XX` and r_{XY} is `true_cor_YX`) times `true_sd_Y / true_sd_X`, against which a supplied `true_reg_coef_j` is checked.

- *Squared multiple correlations path.* Supply `true_R2_Y_X`, `true_R2_j_X_without_j`, `p`, and `true_reg_coef_j` directly when the user already has these planning values from prior research and does not need to specify the full covariance structure.
- *Standardized solution.* For the `noncentral = TRUE` path, set `true_sd_Y = true_sd_X = 1`; the result returns the standardized regression coefficient sample size.

The interval planned for. With `noncentral = FALSE` the interval is the central *t* interval for the coefficient in the units of the data, b_j plus the *t* quantiles (on $N - p - 1$ degrees of freedom) at `alpha_lower` and `1 - alpha_upper` times its standard error, the interval `ci_rc` and `ci_reg_coef` report from unstandardized data, and this is so whatever `true_sd_Y` and `true_sd_X` are: with both 1 the coefficient is on the standardized scale, but the interval planned for is still the one computed in the units of the data, whose standard error comes from the residual sums of squares of *Y* and of X_j. The plan without `assurance` is the smallest *N* at which that interval, computed at the expected values of the two sample squared multiple correlations, is no wider than `width` (Kelley & Maxwell, 2003, Equations 1 and 2, with the *t* quantile); the two tails exchanged give the same plan. The plan is the smallest such *N* of at least $p + 2$, the smallest sample with a residual degree of freedom, and the expected squared multiple correlation of X_j with the other predictors is that of a regression on those $p-1$ predictors (0 with one predictor; see "Comparison with MBESS"). Kelley and Maxwell's Equation 1 is the central interval for the standardized coefficient, whose standard error is formed from the two sample squared multiple correlations; its plan without `assurance` is this one at unit standard deviations, but the probability that it is no wider than `width` is not the probability the assurance step below computes, and `ss_aipe_src` with `noncentral = FALSE` plans its assurance. With `noncentral = TRUE` the interval is the noncentral *t* interval for the standardized coefficient of `ci_src`: the limits of the noncentrality parameter at the observed *t* statistic, times the standard error $\sqrt{(1 - R^2)/[(1 - R^2_{X_j \cdot X_{-j}})(N - p - 1)]}$, and the plan without `assurance` is the smallest *N*, of at least $p + 2$, at which it is no wider than `width` at the expected squared multiple correlations and β_j. That standard error treats the sample standard deviations as constants, which for random predictors is an approximation (Kelley & Maxwell, 2003, pp. 315–316). The central path is fast and adequate at moderate to large *N*; the noncentral path is the one to use when the analysis will report the noncentral interval for the standardized coefficient. A tail of 0, or two of .5, is refused, and so is a tail area below 1e-9 or a `conf_level` above .999999998.

The assurance step. A plan without `assurance` makes the interval as narrow as the target when the sample values equal their expectations; in roughly half of all samples it is wider. With `assurance` the function plans so that the realized interval is no wider than `width` with at least that probability, for random predictors from a multivariate normal population.

- *The central interval* (`noncentral = FALSE`). The only method is `"analytic"`: the function computes P_N, the probability that the realized interval is no wider than `width` at sample size *N*, exactly, and returns the smallest *N*, at or above the plan without `assurance`, at which P_N is at least `assurance`, found by doubling a step upward and then bisecting. `assurance_attained` is P_N at the returned *N*. The squared standard error of b_j is the residual sum of squares of *Y* on all *p* predictors over $N - p - 1$, divided by the residual sum of squares of X_j on the other $p - 1$. Under multivariate normality the first, over $\sigma^2_Y(1 - \rho^2_{Y \cdot X})$, is chi square on $N - p - 1$ degrees of freedom whatever the predictors are, so it is independent of the second, which over $\sigma^2_{X_j}(1 - \rho^2_{X_j \cdot X_{-j}})$ is chi square on $N - p$ degrees of freedom. The squared standard error is then $KF/(N - p)$, with $K = \sigma^2_Y(1 - \rho^2_{Y \cdot X})/[\sigma^2_{X_j}(1 - \rho^2_{X_j \cdot X_{-j}})]$ and *F* an *F* variate on $N - p - 1$ and $N - p$ degrees of freedom, and with q the sum of the two *t* quantiles of the

interval,

$$P_N = \Pr(F \leq w^2(N-p)/(q^2K)),$$

where w is `width`. (With fixed predictors the second sum of squares is a constant and the same event is a chi square probability; the plan is for random predictors, the model `ss_aipe_reg_coef_sensitivity` simulates.) The units of *Y* and X_j cancel, so changing them, with `width` changed to match, leaves the plan as it was, and the sign of β_j does not enter.

- *The noncentral interval* (`noncentral = TRUE`). The width depends on the observed *t* statistic and on both sample squared multiple correlations, and no closed form for its distribution is used. With `assurance_method = "analytic"`, the default, the plan is the approximation that **MBESS** uses (Kelley, 2007a, 2007b): the plan without `assurance` made again with a lower confidence limit (at level `assurance`) of the squared multiple correlation in place of its population value, and not below the first. It does not compute the probability it attains, so `assurance_attained` is `NA`, and the printed footer says so. Kelley and Maxwell (2008, pp. 186–187) wrote that no satisfactory method then existed for the standardized coefficient and described finding, by simulation, the smallest sample size that attains the requested degree of certainty, which is what `assurance_method = "simulation"` does: P_N is estimated as the proportion of `G` samples, drawn from the exact sampling distribution of the sums of squares and products (Wishart, through the entries of its triangular factor that the statistics depend on), whose interval is no wider than `width`. The same `G` samples serve every sample size the search tries (common random numbers: each chi square entry is the quantile, at the degrees of freedom of that *N*, of one uniform draw), so the estimate changes smoothly with *N*. The search assumes that it does not decrease as *N* grows, which held in every set checked although a single simulated interval can be wider at $N + 1$ than at *N* (see below), and under that assumption the step returns the smallest *N* at which the estimate is at least `assurance`. The distribution of the width depends on the population through *p*, β_j, the two squared multiple correlations, and the correlation of *Y* with X_j. On the covariance structure path that correlation is given; on the squared multiple correlations path it is not, and the step uses the smallest estimated probability over 21 equally spaced values of it, from the smallest to the largest that the three values allow. That is an approximate minimum over the range: between two of the values the probability can be a little lower than at either, so for a correlation there the plan can fall a little short of the request. (With $p = 2$ only the two ends of the range are possible, and the step uses both.) `assurance_attained` is the estimate at the returned *N*, so it is at least `assurance`, and its Monte Carlo standard error is the attribute `assurance_mc_se`. The exact P_N differs from the estimate by that error, so the returned *N* can be a case or two from the one the exact probability would give, and P_N there can be a little below `assurance`; give `seed` for a plan that can be reproduced, and a larger `G` for a more precise one. Reversing the sign of β_j (with `alpha_lower` and `alpha_upper` exchanged when they differ) gives the same plan.

For both intervals P_N did not decrease from one *N* to the next at or above the plan without `assurance`: for the central interval in 1,530 planning sets (*p* from 1 to 15, squared multiple correlations from 0 to .9, widths from .05 to 2, and tails of .025 and .025, .005 and .005, and .01 and .04) over every *N* from that plan to three times it, and for the noncentral interval in five sets by simulation. With the samples held fixed across *N*, the estimate did not decrease from one *N* to the next in seven sets (*p* from 1 to 10), and of 95 million comparisons of a simulated interval at *N* and at $N + 1$, 3208, all at *N* below 60, were wider at $N + 1$. Below the plan without `assurance` P_N can decrease, which is why the search starts there.

The approximations can attain more or less than the request. For two predictors correlated .3 with correlations of .4 and .3 with *Y*, a width of .3, and an assurance of .90, the chi square approximation

of Kelley and Maxwell (2003, Equation 3; 2008, Equation 37) for the central interval, the plan without `assurance` with the variance of b_j multiplied by the `assurance` quantile of a chi square on $N-1$ degrees of freedom over $N-p-1$, is 181, at which P_N is .852, and the exact plan is 187 (P_N = .902); the noncentral approximation, the default plan for that interval, is 179, at which the simulated probability is .948, and `assurance_method = "simulation"` returns 176 (.908 with `seed = 1`). For the five exchangeable predictors of Kelley and Maxwell (2003, pp. 312–313), correlated .4 with one another and .3 with *Y*, at a width of .2 and an assurance of .80, the chi square approximation is 486 (P_N = .765) and the exact plan 491 (.800). Each was checked against the proportion of intervals no wider than the target among samples of raw multivariate normal data. These probabilities, and those of the first example, are for the central interval in the units of the data, the interval this function plans for. Kelley and Maxwell's example is of the standardized coefficient, with the central interval for it (their Equation 1), which is computed from standardized data with the sample standard deviations treated as constants; its width varies less, and it is no wider than .2 with probability .877 at 486 and .911 at 491 (their own simulation, p. 317, reported .816 at their plan of 485, where raw multivariate normal data give .871 with the *t* quantile of their Equation 1, or .887 with the normal quantile of their plan). For that interval `ss_aipe_src` with `noncentral = FALSE`, `assurance_method = "simulation"`, and `seed = 1` plans 478 (.807, and .797 at 477), and in the first example 171 (.914, and .895 at 170), where the plans above reach .986 at 181 and .997 at 187; its plan is simulated, and another seed can give one a case away (170 for one of seeds 1 to 30 in the first example).

Comparison with MBESS. The plans without `assurance` of `ss.aipe.reg.coef()` in **MBESS** differ from these in two ways. They take the expected squared multiple correlation of X_j with the other predictors to be that of a regression on *p* predictors, where that regression has $p-1$, which overstates it and so the width. And their search goes up one case at a time from a start, the larger of $2p+1$ and ten below the normal approximation for the central interval and six below the central plan for the noncentral one, so it returns the start when a smaller sample size already meets the width: for `true_R2_Y_X = .8`, `true_R2_j_X_without_j = 0`, `p = 10`, and `width = 2`, 21 where 14 does. In 1,530 planning sets the two differences make the central plan here one case smaller in 1,329 and 2 to 13 cases smaller in 104, and in 810 sets the noncentral plan one smaller in 596 and 2 to 10 smaller in 177. The approximations start from these plans (Kelley and Maxwell's example above is 487 in **MBESS** and 486 here). With unequal tails `ss.aipe.reg.coef()` also takes both *t* quantiles of its central plans at the average of the two tails, so it plans for an interval narrower than the one `ci_rc` reports; the central plans here take the quantile at each tail.

Value

A `data.frame` with columns `term` and `value`:

`necessary_N` the necessary total sample size.

`assurance` the requested probability; present only when `assurance` was given.

`assurance_attained` the probability, at `necessary_N`, that the realized interval is no wider than `width`; present only when `assurance` was given. Exact for the central interval; for the noncentral interval `NA` with `assurance_method = "analytic"`, whose approximation does not compute it, and with `"simulation"` a Monte Carlo estimate, whose standard error is the attribute `assurance_mc_se`.

`G` the number of simulated samples; present only when the noncentral assurance step simulated.

With `assurance` the printed footer names the assurance method and says how `assurance_attained` was found, or that the approximation does not compute it; the attributes `assurance_method` (`"analytic"`

or "simulation") and assurance_probability ("exact", "simulation", or "not computed") record the same.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

See Also

ss_aipe_reg_coef_sensitivity, ci_nc_t

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# 1. Covariance structure path: supply the population correlation
#    matrix and the population YX cross-correlations. Five predictors
#    in an exchangeable structure (all pairwise correlations 0.5,
#    all Y-X correlations 0.3).
true_cor_YX <- c(.3, .3, .3, .3, .3)
true_cor_XX <- rbind(c(1, .5, .5, .5, .5), c(.5, 1, .5, .5, .5),
                c(.5, .5, 1, .5, .5), c(.5, .5, .5, 1, .5),
                c(.5, .5, .5, .5, 1))

# Closed-form (central t) planning, targeting the first predictor's
# standardized coefficient.
ss_aipe_reg_coef(width = .10, which_width = "Full",
                 true_sd_Y = 1, true_sd_X = 1,
                 true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
                 which_predictor = 1, noncentral = FALSE,
```

```
                  conf_level = .95)

# With an assurance of .85: the smallest N at which the realized
# interval is no wider than 0.10 with probability at least .85. The
# last row is that probability at the returned N, computed exactly.
ss_aipe_reg_coef(width = .10, which_width = "Full",
                  true_sd_Y = 1, true_sd_X = 1,
                  true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
                  which_predictor = 1, noncentral = FALSE,
                  conf_level = .95, assurance = .85)

# The noncentral t interval for the standardized coefficient. Required
# N differs at small to moderate samples.
ss_aipe_reg_coef(width = .10, which_width = "Full",
                  true_sd_Y = 1, true_sd_X = 1,
                  true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
                  which_predictor = 1, noncentral = TRUE,
                  conf_level = .95)

# With assurance its default plan is the approximation that MBESS uses,
# which does not compute the probability it attains; by simulation, the
# smallest N at which the estimated probability is at least .85. G = 2000
# keeps the example quick; a reported plan deserves the default G = 20000.
ss_aipe_reg_coef(width = .10, true_cor_XX = true_cor_XX,
                  true_cor_YX = true_cor_YX, which_predictor = 1,
                  noncentral = TRUE, assurance = .85)
plan <- ss_aipe_reg_coef(width = .10, true_cor_XX = true_cor_XX,
                         true_cor_YX = true_cor_YX, which_predictor = 1,
                         noncentral = TRUE, assurance = .85,
                         assurance_method = "simulation",
                         G = 2000, seed = 113)
plan
attr(plan, "assurance_mc_se")

# 2. Squared multiple correlations path: when the user has planning
#    values for true_R2_Y_X and true_R2_j_X_without_j directly (e.g.,
#    from a prior power analysis), without specifying the full
#    covariance structure. true_reg_coef_j is required on this path.
ss_aipe_reg_coef(true_R2_Y_X = 0.30, true_R2_j_X_without_j = 0.20,
                  p = 5, true_reg_coef_j = 0.25,
                  width = .15, which_width = "Full",
                  true_sd_Y = 1, true_sd_X = 1,
                  noncentral = FALSE, conf_level = .95)

# 3. One predictor: its correlation matrix is matrix(1), and p and
#    which_predictor follow from it.
ss_aipe_reg_coef(width = .20, true_cor_XX = matrix(1), true_cor_YX = .4)
```

`ss_aipe_reg_coef_sensitivity`
Sensitivity Analysis for Sample Size Planning From the Accuracy in Parameter Estimation Perspective for the Unstandardized Regression Coefficient

---

Description

This function performs a sensitivity analysis when planning sample size from the Accuracy in Parameter Estimation Perspective for the unstandardized regression coefficient. `ss_aipe_rc_sensitivity` is the same analysis under the name that pairs with `ss_aipe_rc`, and `ss_aipe_src_sensitivity` is its counterpart for the standardized regression coefficient.

Usage

```
ss_aipe_reg_coef_sensitivity(
  true_var_Y = NULL,
  true_cov_YX = NULL,
  true_cov_XX = NULL,
  supposed_var_Y = NULL,
  supposed_cov_YX = NULL,
  supposed_cov_XX = NULL,
  specified_N = NULL,
  which_predictor = 1,
  width = NULL,
  noncentral = FALSE,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = TRUE,
  filename = NULL,
  assurance_method = c("analytic", "simulation"),
  seed = NULL
)
```

Arguments

`true_var_Y`	Population variance of the dependent variable (Y)
`true_cov_YX`	Population covariances vector between the p predictor variables and the dependent variable (Y)
`true_cov_XX`	Population covariance matrix of the p predictor variables
`supposed_var_Y`	Supposed variance of the dependent variable (Y), the planning value from which the sample size is planned
`supposed_cov_YX`	Supposed covariances vector (planning values) between the p predictor variables and the dependent variable (Y)

supposed_cov_XX
: Supposed covariance matrix (planning values) of the p predictor variables

specified_N
: Directly specified sample size (instead of planning one from the supposed covariance structure)

which_predictor
: Identifies which of the *p* predictors is of interest

width
: Desired confidence interval width for the regression coefficient of interest, in the units of the data

noncentral
: Specify with a TRUE or FALSE statement whether or not the noncentral approach to sample size planning should be used. The noncentral plan is for a standardized coefficient, so when the sample size is planned from the supposed values, noncentral = TRUE needs supposed variances of 1 for *Y* and the targeted predictor, where the two scales coincide; otherwise the call is refused (with a condition of class dmar_argument_invalid), and ss_aipe_src_sensitivity is the function for the standardized coefficient.

conf_level
: Desired level of confidence for the computed interval (i.e., 1 - the Type I error rate)

assurance
: Degree of certainty that the obtained confidence interval will be sufficiently narrow

G
: The number of generations (i.e., replications) of the simulation within the function

print_iter
: Specify with a TRUE/FALSE statement if the iteration number should be printed as the simulation within the function runs

filename
: Optional path of a CSV file to receive the per-replication results (the coefficient estimate reg_coef_j, its confidence limits ci_lower and ci_upper, the observed R^2 R2, the standard error se_reg_coef_j, and the *t* statistic t_value), appended when the file already exists and created otherwise; the default NULL writes nothing, and a throwaway run that wants the file should point it at tempfile(fileext = ".csv").

assurance_method
: Passed to ss_aipe_reg_coef when the sample size is planned from the supposed values with assurance. "analytic" (the default) plans, for the central interval, the smallest sample size at which the exact probability of an interval no wider than width is at least assurance, and, for the noncentral interval, the approximation that **MBESS** uses, which does not compute that probability. "simulation", offered with noncentral = TRUE only, plans the smallest sample size at which a simulated estimate of the probability (made by the planner from its default G of 20000 samples with the random number stream that seed sets) is at least assurance; with noncentral = FALSE it is refused with a condition of class dmar_argument_invalid, as ss_aipe_reg_coef refuses it. Not used with specified_N or without assurance.

seed
: NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Direct specification of `true_cov_YX` and `true_cov_XX` is necessary, even if one is interested in a single regression coefficient, so that the covariance/correlation structure can be specified when the simulation within the function runs.

The coefficient. The coefficient is the unstandardized one, the *j*th element of $\Sigma_{XX}^{-1}\sigma_{XY}$, where Σ_{XX} is `true_cov_XX` and σ_{XY} is `true_cov_YX`; call it b_j. It is the true value the intervals are judged against, and the supposed coefficient echoed beside it is the same element computed from `supposed_cov_XX` and `supposed_cov_YX`. Every replication is fit in the units of the data, and a sample size planned from the supposed values is planned by `ss_aipe_reg_coef` with the supposed standard deviations of *Y* and of the targeted predictor (the square roots of `supposed_var_Y` and of the *j*th diagonal element of `supposed_cov_XX`), so that `width` is a width in the units of the data. The standardized coefficient, $\beta_j = b_j\sigma_{X_j}/\sigma_Y$ (Kelley & Maxwell, 2003, p. 316), is the subject of `ss_aipe_src_sensitivity`.

The interval of each replication is that of `ci_reg_coef` with the standard error of the fitted regression, central or, with `noncentral = TRUE`, noncentral.

Value

A `data.frame` with columns `term` and `value` summarizing the Monte Carlo sensitivity analysis across `G` replications. The `term` entries are: `mean_reg_coef_j`, `median_reg_coef_j`, `sd_reg_coef_j` (summaries of the realized regression coefficient point estimates); `mean_ci_width`, `median_ci_width`, `sd_ci_width` (summaries of the realized interval widths); `pct_ci_less_width` (proportion of intervals at or below the planning target `width`); `pct_ci_miss_low` and `pct_ci_miss_high` (tail-specific empirical non-coverage of the population coefficient); `total_type_I_error` (overall empirical non-coverage, the sum of the two tails); `mean_R2`, `median_R2`, `sd_R2` (summaries of the realized squared multiple correlation coefficient); and the input echoes `total_N` (the sample size evaluated), `p`, `which_predictor`, `true_reg_coef_j` and `supposed_reg_coef_j` (the population and planning values of the targeted coefficient implied by the supplied covariance structures), `width`, `conf_level`, and `assurance` (present only when an assurance was supplied). The proportion and Type I error rows are proportions on the 0 to 1 scale, not percentages, so `total_type_I_error` is the sum of `pct_ci_miss_low` and `pct_ci_miss_high`.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Note

Note that when the true and supposed covariance structures agree (`true_cov_YX` equals `supposed_cov_YX` and `true_cov_XX` equals `supposed_cov_XX`), the results are not literally from a sensitivity analysis, rather the function performs a standard simulation study. A simulation study can be helpful in order to determine if the sample size procedure under or overestimates necessary sample size.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

See Also

`ss_aipe_reg_coef`, `ci_reg_coef`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Sensitivity analysis for an unstandardized regression coefficient
# with two predictors at a modest R squared. The Monte Carlo loop is
# run with a small number of generations (G) here so the example is
# fast; use a larger G (for example G = 1000) in real applications.
set.seed(113)
cov_XX <- matrix(c(1, 0.3, 0.3, 1), nrow = 2)
cov_YX <- c(0.4, 0.3)
ss_aipe_reg_coef_sensitivity(
  true_var_Y = 1, true_cov_YX = cov_YX, true_cov_XX = cov_XX,
  supposed_var_Y = 1, supposed_cov_YX = cov_YX, supposed_cov_XX = cov_XX,
  which_predictor = 1, width = 0.20, conf_level = 0.95,
  G = 100, print_iter = FALSE
)
```

```
# The outcome in units whose variance is 4. The unstandardized
# coefficient of the first predictor is about .68 units of Y per unit of
# the predictor, and the 0.55 width is in those units. For the same
# structure ss_aipe_src_sensitivity() reports the standardized
# coefficient, .68 times 1 / 2, about .34, with a width in standardized
# units.
ss_aipe_reg_coef_sensitivity(
  true_var_Y = 4, true_cov_YX = c(0.8, 0.6), true_cov_XX = cov_XX,
  specified_N = 200, which_predictor = 1, width = 0.55,
  G = 100, print_iter = FALSE, seed = 113
)
ss_aipe_src_sensitivity(
  true_var_Y = 4, true_cov_YX = c(0.8, 0.6), true_cov_XX = cov_XX,
  specified_N = 200, which_predictor = 1, width = 0.28,
  G = 100, print_iter = FALSE, seed = 113
)
```

`ss_aipe_reliability` *Sample Size Planning for Accuracy in Parameter Estimation for Reliability Coefficients*

Description

Computes the necessary sample size for the confidence interval on a population reliability coefficient (coefficient alpha or coefficient omega, depending on the assumed measurement model) to have expected width no larger than `width`, or (when `assurance` is supplied) to be no wider than `width` with the specified probability. This is the accuracy in parameter estimation (AIPE) counterpart to power based planning for reliability and is the companion of `reliability`. Without an assurance the sample size comes from a closed form for the expected width of the interval `type` names: the normal theory variance of coefficient alpha of van Zyl, Neudecker, and Nel (2000) for the parallel model and for the normal theory interval of the tau equivalent model (Terry & Kelley, 2012, Eq. 19), and the delta method variance of the model implied coefficient for the factor analytic interval of the tau equivalent model (equal loadings) and for the congeneric model (coefficient omega). When `assurance` is supplied, a Monte Carlo search starting from the closed form finds the smallest *N* delivering the requested assurance (Terry & Kelley, 2012).

Coefficient alpha (Guttman, 1945; subsequently popularized by Cronbach, 1951) is planned for under the parallel and tau equivalent (*i.e.*, the so called True Score) measurement models; coefficient omega (McDonald, 1999) is planned for under the congeneric model. The coefficient is the one `reliability_alpha` and `reliability_omega` estimate from the raw item scores: the population, however it is given, is used as a covariance matrix, so items of unequal variance are planned for as they are, and a correlation matrix plans for standardized items.

When the coefficient will be estimated from incomplete data by full information maximum likelihood, the interval at a given *N* is wider than it would be with complete data. The `fmi` argument plans for that: it is the fraction of missing information anticipated for the coefficient, and the plan grows to carry the information that complete data would have carried (see the section on planning under missing data).

Usage

```
ss_aipe_reliability(
  model = NULL,
  type = NULL,
  width = NULL,
  true_cov_matrix = NULL,
  conf_level = 0.95,
  assurance = NULL,
  data = NULL,
  n_items = NULL,
  true_item_covariance = NULL,
  true_factor_loading = NULL,
  true_residual_var = NULL,
  G_initial = 500,
  G = 5000,
  start_N = NULL,
  verbose = FALSE,
  fmi = 0,
  seed = NULL
)
```

Arguments

model	The measurement model assumed for the population. Accepts (case-sensitive aliases shown in parentheses): "Parallel" ("parallel", "SB", "Spearman Brown", "Spearman-Brown", "sb") for the strictly parallel items model; "True Score" ("True Score Equivalent", "True-Score Equivalent", "Equivalent", "Tau Equivalent", "tau-equivalent", "Tau-Equivalent", "True-Score", "true-score", "true score", "Cronbach", "cronbach", "Chronbach", "alpha") for the tau equivalent model (in which case the function plans for coefficient alpha); or "Congeneric" ("congeneric", "omega", "Omega") for the congeneric model (in which case the function plans for coefficient omega).
type	The method used to construct the confidence interval on the reliability coefficient: either "Factor Analytic" (McDonald, 1999), available for all three measurement models, or "Normal Theory" (van Zyl, Neudecker, & Nel, 2000), available for the parallel and tau equivalent models only (with the congeneric model it is an error).
width	The desired full width of the two-sided confidence interval.
true_cov_matrix	A symmetric population covariance matrix among the items (a planning value, not a sample covariance matrix), used as it is to imply the population reliability and its sampling distribution; a matrix with a negative eigenvalue is refused. A correlation matrix is a covariance matrix of standardized items and plans for them.
conf_level	Confidence level (i.e., $1 - \alpha$, where α is the Type I error rate). Default 0.95.
assurance	Optional probability with which the realized interval is to be no wider than width (a sample that returns no interval counts as one that is wider). When NULL

(the default), the planner targets the *expected* width; when supplied (e.g., 0.80, 0.85, 0.95), the function follows the closed form with a Monte Carlo search.

`data` A data set from which the population covariance matrix should be inferred, from its complete cases; the number of items is the number of its columns.

`n_items` Number of items (*J* in the formulas of the Details). Needed when the population is given by `true_item_covariance` or `true_factor_loading` with `true_residual_var`; with `true_cov_matrix` or `data` it is the number of columns and need not be given.

`true_item_covariance`
The presumed common covariance *c* of every pair of items, which with `true_residual_var` gives the population: the common loading is the square root of *c*, so two items covary c and an item's variance is c plus its `true_residual_var`. It is the inter-item correlation of Terry and Kelley (2012) when the items are standardized, that is, when `true_residual_var` is $1 - c$; otherwise items *j* and *k* correlate $c/\sqrt{(c+\psi_j^2)(c+\psi_k^2)}$, with ψ_j^2 the residual variance of item *j*. With `true_item_covariance = .3` and `true_residual_var = .2`, for example, the items correlate .6, not .3. One value.

`true_factor_loading`
Population factor loading or loadings: one number for the Parallel and True Score models, whose items share one loading, and a vector of `n_items` loadings for the Congeneric model. A vector for the Parallel or True Score model is an error that names the Congeneric model, since unequal loadings are what distinguish it.

`true_residual_var`
Population residual (unique, or error) variance or variances: one number for the Parallel model, whose items share one, and a vector of `n_items` variances for the True Score and Congeneric models, each greater than zero; with the normal theory interval of the True Score model, any values that leave the covariance matrix they imply a covariance matrix (see Details).

`G_initial` Number of Monte Carlo replications in each pass of the initial assurance search (default 500): a single whole number of at least 1.

`G` Number of Monte Carlo replications in each pass of the final assurance verification (default 5000), the count the reported `assurance_attained` rests on: a single whole number of at least 1. Anything else given for either count is refused with an error of class `dmar_count_invalid`. Neither is used without `assurance`.

`start_N` Optional starting total sample size for the iterative assurance search.

`verbose` If `TRUE`, prints the current sample size and empirical assurance at each step of the Monte Carlo search.

`fmi` The fraction of missing information anticipated for the reliability coefficient when it is estimated by full information maximum likelihood from incomplete data: a single number in $[0, 1)$, with the default `0` for complete data, which returns exactly the complete data plan. It is not the proportion of missing values. Writing γ for `fmi`, the plan divides the complete data $N - 1$ by $1 - \gamma$; see the section on planning under missing data for the derivation and for where a planning value comes from.

`seed` Optional seed for the Monte Carlo assurance search. The default `NULL` draws from the current state of the random number generator; a number makes the search reproducible, and the caller's generator state is restored when the function returns. The closed form (no `assurance`) draws nothing.

Details

The closed forms. Each closed form takes the sampling variance of the coefficient to be $V/(N-1)$ and solves $2z\sqrt{V/(N-1)} = w$ for *N*, so that $N = 4z^2V/w^2 + 1$, where *w* is `width` and *z* the standard normal quantile for `conf_level`. For the parallel model *V* is the compound symmetry form $2(1-\alpha)^2 J/(J-1)$ of van Zyl et al. (2000), with *J* items, which at a population of parallel items is also the variance of the factor analytic interval, so the parallel model has one closed form for both types. For the normal theory interval of the tau equivalent model *V* is the general variance of alpha of van Zyl et al. (Terry & Kelley, 2012, Eqs. 7, 8, and 19), evaluated on the population covariance matrix as the formula defines it. For the congeneric model *V* is the delta method variance of the model implied omega, $\omega = (\sum \lambda_j)^2/((\sum \lambda_j)^2 + \sum \psi_j^2)$, with λ_j the loadings (`true_factor_loading`) and ψ_j^2 the error variances (`true_residual_var`): the gradient of omega taken through the inverse of the expected information of the single-factor model at the population (Raykov, 2002; Terry & Kelley, 2012, Eqs. 13 to 16), the variance behind `reliability_omega(denominator = "model_implied", ci_method = "ml")`, whose squared standard error is V/N; taking it as $V/(N-1)$, as for alpha, adds one case and keeps the planning under missing data below the same for every model. When the population is given by `true_cov_matrix` or `data`, the single-factor model is first fit to it by maximum likelihood. Terry and Kelley (2012, p. 390) had no closed form for omega: they used the alpha formula as if true score equivalence held only as a starting value, and found the size whose mean interval width met the target by an a priori Monte Carlo simulation. That starting value, taken as the plan for omega, is too large when the loadings differ among standardized items and can be far too small when the item variances differ. The delta method form gives the expected width to first order. The mean width of the factor analytic interval at the planned *N* meets the target where the samples at that *N* stay away from an improper solution and from empirical underidentification, and exceeds it where some do not, whose intervals are far wider, while the median width stays near the target (within 4 percent of it in the simulations below, above it in some). Which samples come near depends on the loadings, the number of items, and *N* together, so no bound on the loadings alone marks the populations the closed form serves. With three items the single-factor model is just identified, each squared loading is a ratio of the covariances ($\lambda_1^2 = \sigma_{12}\sigma_{13}/\sigma_{23}$), and a sample covariance near zero makes the solution run away: three loadings of .3 planned for a width of .20 take 687 cases, at which the median width is .202 and the mean .264, and 3 percent of samples give no interval (1,000 samples), while at a width of .10 (2743 cases) the mean is on target; loadings of .5, .4, and .3 at .20 (462 cases) average .242, with a median of .206, and three loadings of .4 (468 cases) .211, with a median of .199 (500 samples each). With four loadings of .3 the mean width was on target at .20 (527 cases) but averaged .323 at .30 (235 cases), where a tenth of the samples give no interval. Where the factor rests on weak loadings the same happens with more items: loadings of .7, .7, .1, and .1 (omega .46) planned for a width of .10 take 974 cases, at which the median width is .104 and the mean .181, and four loadings of .15 take 3437 cases, with a median of .098 and a mean of .148. In the other populations simulated for this planner, with four or more items whose loadings share a sign, the mean was within a few percent of the target (for loadings of opposite sign, see below). For a population like those above, plan with an assurance above .5, whose search counts the widths themselves.

For the factor analytic interval of the tau equivalent model *V* is the same delta method variance with

the loadings held equal, the variance behind `reliability_alpha(estimator = "model_implied",ci_method = "ml")`: the single loading's information is the sum of the loading block of the information above, and its gradient the sum of the loadings' gradients. That coefficient is the maximum likelihood estimator under the model, so its variance is no larger than the normal theory variance of alpha, and it is smaller whenever the error variances differ: with loadings .7 and error variances .1, .2, 1, and 2, a width of .10 takes 242 cases with the factor analytic interval and 367 with the normal theory one. With equal error variances the two coincide.

Populations that are refused. The planning population has to be one the model describes. A population given as `true_cov_matrix` must be a covariance matrix (no negative eigenvalue), and when the single-factor model is fit to it, the fit must be proper. Where that model is fit, for the congeneric model and for the tau equivalent model with the factor analytic interval, every error variance in `true_residual_var` must be above zero, as the parallel model requires: a negative one is not a variance, and a zero one makes an item free of error, the boundary at which the single-factor solution is improper. The same population given as `true_cov_matrix` is refused through its fit. With the normal theory interval of the tau equivalent model nothing is fit: the loading and the error variances only build the covariance matrix the plan is made from, which must be a covariance matrix, as `true_cov_matrix` must, and the plan is the one the same matrix given as `true_cov_matrix` gives. The congeneric model needs at least three items with a nonzero loading, without which it is not identified at the population and omega has no sampling variance. With the factor analytic interval of the parallel and tau equivalent models, items to which the single-factor model with equal loadings gives a loading of zero share no common factor in that model, and the coefficient that interval estimates is zero, on the boundary of its range, where the closed form does not describe the interval. For a population given by its loading (`true_item_covariance` or `true_factor_loading`) that is a loading of zero, and for one given as `true_cov_matrix` or `data` the loading of the maximum likelihood fit. For items of equal variance it happens exactly when their covariances average zero or below (alpha of the population matrix at or below zero); with unequal variances the fit weighs each covariance by the inverse of its two items' variances, and the two can disagree either way (three items of variance 1 that correlate .3 and a fourth of variance 4 that correlates -.25 with them have alpha -0.276 and a fitted coefficient of .315, which the tau equivalent model plans for). The parallel model's closed form is a function of alpha of the population matrix, and it is refused when alpha is at or below zero; a population whose fitted coefficient is nonetheless positive is far from parallel, and the message names the tau equivalent model. A supplied population under `model = "Parallel"` must have equal diagonal entries and equal off-diagonal entries, each within $\sqrt{\epsilon}\max|S|$, where ϵ is machine precision. This check uses the original item units. A sample covariance will generally depart from an exact parallel population; specify the intended population with `true_factor_loading` and `true_residual_var` when parallelism is a planning assumption rather than a property of `true_cov_matrix`.

Where the congeneric closed form is refused. The delta method takes omega, a square of the sum of the loadings L over the variance of the composite, to be normal with its first-order variance, which needs L to be large relative to its standard error. When it is not, as when loadings of opposite sign nearly cancel in the unweighted sum, omega behaves near zero as the square of a normal variable, and its first-order variance understates the width of its interval; where the loadings cancel exactly, the gradient of omega vanishes and the closed form gives one case. The closed form is therefore checked at the *N* it gives, or at `n_items + 1` cases when it gives fewer, the fewest at which the factor analytic interval can be computed (with fewer, the sample covariance matrix is singular). Write σ_L for the standard error of $\hat{L}$ there, $r = \sigma_L/|L|$, and P for the sum of the error variances. Near zero $\hat{\omega}$ is about $\hat{L}^2/P$ and its delta method standard error about $2|\hat{L}|\sigma_L/P$, so with $\hat{L}$ normal,

the interval the estimation functions report, clamped at zero, averages

$$\frac{E[\min(4z|X|, X^2 + 2z|X|)]}{4z/r}, \qquad X \sim N(1/r, 1),$$

times the width planned for. That ratio stays below one, the clamped interval making the plan conservative, until r is about 1.5 at 95 percent confidence (1.36 at 90 percent and 1.66 at 99), and it exceeds one beyond; the closed form is refused once it does. In simulations of the factor analytic interval at the planned *N* for loadings .7, -.7, .5, and a fourth loading of -.2, -.32, -.35, or -.37 (r of .41, 1.07, 1.52, and 2.00 there, the last two refused), the mean width was .082, .082, .089, and .121 for a target of .10. With a fourth loading of -.4 (omega .0038, $r = 3.35$) the closed form's 49 cases give a mean width of .22, where the interval needs about 95, and with -.5 the loadings cancel and the delta method gives one case. An assurance plan is not refused: its search counts the widths of the intervals themselves, starting at no fewer than `n_items + 1` cases. A congeneric closed form below `n_items + 1` is floored at that size before any missing-information inflation, with a warning. This is only a feasibility minimum. It does not establish that the expected width target is met at such a small sample size; an assurance plan is recommended.

Whenever the loadings of a congeneric population differ in sign, the planner warns. An item that loads opposite to the others lowers omega of the unweighted sum, as an item keyed in the opposite direction does when it is not reverse scored. The plan is for omega of the sum of the items as they are given; reverse score such items before planning, as before the coefficient is estimated.

The parallel and tau equivalent models need no such check. Their one loading enters the covariance matrix only through its square, the common covariance, of which their coefficient is a smooth function, so its variance does not vanish near zero, and there the clamped interval only makes the plan conservative: four parallel items that covary .03 (alpha .11) planned for a width of .30 take 362 cases, at which the factor analytic interval averages .236 wide.

The assurance search. The Monte Carlo search draws samples from the population covariance matrix and, at each candidate sample size, computes the realized confidence interval width with the same machinery the estimation side of the package uses. For `type = "Factor Analytic"` the single-factor model is fit by maximum likelihood (equal loadings for the parallel and tau equivalent models, free loadings for the congeneric model) and the interval is the delta method Wald interval on the model implied reliability, the interval of `reliability_omega(denominator = "model_implied", ci_method = "ml")` and of `reliability_alpha(estimator = "model_implied", ci_method = "ml")`. For `type = "Normal Theory"` the interval uses the van Zyl, Neudecker, and Nel (2000) closed form standard error for the tau equivalent model and its compound symmetry simplification for the parallel model, the same closed form behind `reliability_alpha(ci_method = "ml")`. The congeneric model has no normal theory form, so `type = "Normal Theory"` is an error there.

The search starts from the closed form size, and with the factor analytic interval at no fewer than `n_items + 1` cases, the fewest whose sample covariance matrix the single-factor model can be fit to (a highly reliable population planned for a wide interval can have a closed form below that). When that size falls short of the assurance, the size is raised one case at a time until a pass of `G_initial` samples meets it, and a final pass of `G` samples confirms the size, again raising it if needed (Terry & Kelley, 2012). When the closed form size already meets the assurance, which happens when the assurance is near one half or below, or when the interval of `type` is narrower than the one the closed form describes (as the factor analytic interval of the parallel model can be when the population given as `true_cov_matrix` or `data` is not parallel), the search moves down: to the size at which the widths it has drawn would just meet the target (the width of a Wald interval shrinks as $1/\sqrt{N-1}$), and from there up one case at a time to the first size that meets it.

Each sample is treated as the estimation functions treat it, so the search counts the intervals a user would be given. A sample for which reliability_omega or the model implied reliability_alpha would return no interval (a single-factor fit that does not converge, three items with no single-factor solution, or a solution that has run away, whose error variances sum below zero so that the model implied coefficient is above one; McDonald, 2004) gave no interval, and so no narrow one: it is counted as wider than width. So is a sample whose likelihood has no maximum, whatever the sum, one error variance running off below zero while the likelihood is flat or still rising, which the estimation functions refuse with an error of class dmar_no_maximum (see reliability_omega). The assurance is the probability that the study yields an interval no wider than width (Terry & Kelley, 2012, p. 373), not that probability among the studies that yield an interval. When some samples of the final pass returned no interval, the result says how many (the no_interval attribute, printed under the table), since at the planned N that share of studies would end without a coefficient. Each confidence limit is clamped into [0, 1], as the estimation functions clamp it, so a negative alpha whose interval lies wholly below zero has the interval [0, 0]. A size at which no sample returns an interval stops the search with the first error.

The cost of the assurance search depends on the interval. The normal theory intervals are closed forms, so a search at the default G_initial = 500 and G = 5000 finishes in seconds. With the factor analytic interval a one factor model is fit at every Monte Carlo iteration and at every candidate sample size the search visits, so a congeneric plan with an assurance runs for tens of seconds even at G_initial = 50 and G = 200 and for many minutes at the defaults. The examples on this page therefore stop at the closed form for the congeneric plan; the assurance version is the same call with assurance supplied (for example, assurance = .80). A seed makes the search reproducible and leaves the caller's random number stream as it was; without one the search draws from, and advances, the current stream.

Value

A data.frame with columns term and value. Without assurance the data frame has a single row, "necessary_N", giving the necessary N. With assurance supplied, the data frame has five rows: "necessary_N" (necessary N), "width" (echo of the target width), "assurance" (echo of the requested probability), "assurance_attained" (the assurance achieved at the returned N in the Monte Carlo verification), and "G" (the number of Monte Carlo replications of the final pass). When fmi is above 0, necessary_N is the size under missing data, the rows are otherwise unchanged, and two attributes record the plan: fmi, the value planned for, and necessary_N_complete, the size complete data would need (for the closed form, its solution rounded up, while the plan maps the unrounded solution; for an assurance plan, the size at which the assurance was simulated, so assurance_attained refers to it). With fmi = 0 neither attribute is added. An assurance plan carries the number of samples of its final Monte Carlo pass that returned no interval as the no_interval attribute (zero when every sample returned one), printed under the table when above zero; they are counted as wide in assurance_attained. Both forms carry the class dmar_ss_aipe of the AIPE planners: generics::tidy() gives the planned N, beside the target width for an assurance plan (the closed form's table holds only necessary_N, so its width is NA), and generics::glance() adds the remaining rows as columns, and the fmi, necessary_N_complete, and no_interval attributes when the plan has them (see dmar_tidiers).

Planning Under Missing Data

The fraction of missing information (FMI) of a parameter is

$$\gamma = 1 - \frac{SE_C^2}{SE_O^2},$$

where SE_O is the standard error of the estimate from the data as observed and SE_C the standard error the same sample would have given had it been complete (Savalei & Rhemtulla, 2012, Eq. 4). It is the proportion of the information about the parameter that the missing values take away, one minus the asymptotic efficiency of the incomplete data estimator relative to the complete data one. At a fixed *N*, missing values therefore multiply the sampling variance of the estimate by $1/(1-\gamma)$ and the width of a Wald interval by $1/\sqrt{1-\gamma}$, the width inflation factor of Savalei and Rhemtulla (2012); an FMI of .56 makes the interval 1.5 times as wide.

Every closed form on this page has a sampling variance of the form $V/(N-1)$, so its information about the coefficient is $(N-1)/V$. With missing data that information becomes $(N-1)(1-\gamma)/V$, and the width target is met when $(N-1)(1-\gamma)$ equals $N_c - 1$, where N_c is the complete data solution of the closed form before it is rounded up to a whole number:

$$N = \frac{N_c - 1}{1-\gamma} + 1.$$

The factor scales $N-1$ rather than *N* because $N-1$ is what the planner's information is proportional to; the target sample size of Savalei and Rhemtulla (2012), $N/(1-\gamma)$, is the same idea for an information proportional to *N*, and the two differ by $\gamma/(1-\gamma)$ cases (one case at an FMI of .5). An FMI of .5 doubles $N-1$, not *N*. The plan is the smallest whole *N* at or above that value. The complete data sample size, the whole number N_c rounds up to, travels with the result as the `necessary_N_complete` attribute and is printed under the table. Because the plan maps the unrounded N_c, the formula applied to the printed whole number can give a larger *N* than the plan, by up to about $1/(1-\gamma)$ cases (the population of the first example below needs 38 cases with complete data and 42 at an FMI of .1, while the formula applied to 38 gives 43).

With an `assurance`, the Monte Carlo search runs exactly as it does for complete data, starting from the complete data closed form, and the size N_e it finds is mapped back with the same formula. The search therefore takes the distribution of the interval width at *N* under missing data to be the complete data width distribution at the effective size $(N-1)(1-\gamma)+1$, which keeps the larger relative variability of the widths of a smaller sample (scaling the complete data widths at *N* by the width inflation factor would not). This is a first-order approximation, and `ss_aipe_reliability_sensitivity` checks it by simulating incomplete data and analyzing them by full information maximum likelihood (its `fmi` and `missing_prop` arguments). The `assurance_attained` row is the assurance found at N_e, the `necessary_N_complete` attribute is N_e itself (a whole number, so the formula applied to it reproduces the plan), and a `start_N` is read as a size under missing data and converted to its effective size before the search.

Where the FMI comes from. The FMI belongs to the coefficient, not to the items, and it is not the proportion of missing values: it depends on the missing data mechanism, the model, and how strongly the items are related. Information lost on one item is partly recovered from the items correlated with it, so the FMI can be lower than the rate of missing data; under data missing completely at random it is bounded by the largest rates of univariate and pairwise missingness, and under data missing at random it can be well above the rate of missing data (Savalei & Rhemtulla, 2012). At the design stage it comes from a pilot study or a comparable data set analyzed the way the planned

study will be: fit the single-factor model by full information maximum likelihood (for example `fa_confirmatory(..., missing = "fiml", output = "fit")`, with `equal_loading = TRUE` for the tau equivalent coefficient that the Parallel and True Score plans target) and take the FMI of its `omega_` defined parameter from `fraction_missing_information`. That FMI refers to the cases the fit used: a case with no observed item carries no information, and the fit sets it aside (the `N_excluded` attribute counts such cases). When the planned *N* counts every case recruited, those who will answer no item included, fold them in as $1 - (N_{used}/N)(1 - \gamma)$, with N_{used} the cases the fit used and *N* those recruited; `ss_aipe_reliability_sensitivity` reports the FMI its simulated data realized in the same way. A sample FMI is imprecise. Chen and Savalei (2021) found it close to unbiased even at $N = 50$ but variable from sample to sample: precise estimates took several hundred cases at overall missing rates of 10 to 20 percent and more than 1,000 at 30 percent. Planning with the upper limit of an interval for the pilot's FMI, rather than with its point estimate, is the conservative choice. Von Hippel (2020) recommends the same use of a pilot's upper limit when deciding how many imputations to run; carrying it over to sample size planning is our extension.

Two things `fmi` leaves alone. The `data` argument still estimates the planning covariance matrix from the complete cases of the pilot data (`na.omit()`); `fmi` is what carries the missing data of the planned study. And the planned interval is the same Wald interval with a normal critical value; the FMI changes how much information the sample carries, not the reference distribution.

Warning

In some conditions the factor analytic model fit by `cfa_1` (via **lavaan**) may fail to converge, and you may see a non-convergence message from **lavaan**. The Monte Carlo assurance search counts a non-converged iteration as an interval wider than `width` and continues, and the number of such iterations in the final pass is printed under the result; a sample size at which no iteration converges stops the search with the first error. Frequent non-convergence usually means the model is poorly determined by the data, for example because of a small sample size or a poorly behaved covariance matrix, and it raises the planned *N*, since a study that yields no interval has not yielded a narrow one.

Note

Not all of the items can be entered into the function to represent the population values. For example, either `data` can be used, or `true_cov_matrix`, or `n_items`, `true_item_covariance`, and `true_residual_var`, or `n_items`, `true_factor_loading`, and `true_residual_var`. With a large number of replications (`G`) this function may take considerable time.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chen, L., & Savalei, V. (2021). Three sample estimates of fraction of missing information from full information maximum likelihood. *Frontiers in Psychology, 12*, 667802. doi:10.3389/fpsyg.2021.667802

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334.

Guttman, L. (1945). A basis for analyzing test-retest reliability. *Psychometrika, 10*(4), 255–282. doi:10.1007/BF02288892

Kelley, K., & Cheng, Y. (2012). Estimation of and confidence interval formation for reliability coefficients of homogeneous measurement instruments. *Methodology, 8*, 39–50. doi:10.1027/1614-2241/a000036

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

McDonald, R. P. (1999). *Test theory: A unified treatment*. Mahwah, NJ: Lawrence Erlbaum Associates.

McDonald, R. P. (2004). Respecifying improper structures. *Structural Equation Modeling, 11*(2), 194–209.

Raykov, T. (2002). Analytic estimation of standard error and confidence interval for scale reliability. *Multivariate Behavioral Research, 37*, 89–103. doi:10.1207/S15327906MBR3701_04

Savalei, V., & Rhemtulla, M. (2012). On obtaining estimates of the fraction of missing information from full information maximum likelihood. *Structural Equation Modeling, 19*(3), 477–494. doi:10.1080/10705511.2012.687669

Terry, L. J., & Kelley, K. (2012). Sample size planning for composite reliability coefficients: Accuracy in parameter estimation via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 371–401. doi:10.1111/j.20448317.2011.02030.x

van Zyl, J. M., Neudecker, H., & Nel, D. G. (2000). On the distribution of the maximum likelihood estimator of Cronbach's alpha. *Psychometrika, 65*(3), 271–280. doi:10.1007/BF02296146

von Hippel, P. T. (2020). How many imputations do you need? A two-stage calculation using a quadratic rule. *Sociological Methods & Research, 49*(3), 699–718. doi:10.1177/0049124117747303

See Also

`reliability`, `cfa_1`, `ss_aipe_reliability_sensitivity` to check a plan, under complete or missing data, by simulation.

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Expected confidence interval width (closed form, no Monte Carlo search)
# for six parallel items that correlate .6: standardized items, so
# true_residual_var is 1 - true_item_covariance.
ss_aipe_reliability(model = "Parallel", type = "Normal Theory", width = .1,
  n_items = 6, true_item_covariance = .6, true_residual_var = .4,
  conf_level = .95, assurance = NULL)

# The same plan when the coefficient will be estimated by full information
# maximum likelihood and is expected to lose 30 percent of its information
# to missing values (an FMI of .30 from a pilot study). The complete data
# N minus 1 is divided by 1 minus .30, and the complete data answer prints
# under the table.
ss_aipe_reliability(model = "Parallel", type = "Normal Theory", width = .1,
```

```
  n_items = 6, true_item_covariance = .6, true_residual_var = .4,
  conf_level = .95, fmi = .30)

# The assurance cases run a Monte Carlo search. G_initial = 20 and
# G = 100 keep the examples quick; a reported plan deserves the
# defaults of 500 and 5000. The seed makes each search reproducible and
# leaves the random number stream of the session where it was.

# Same population, now targeting an assurance.
ss_aipe_reliability(model = "Parallel", type = "Normal Theory", width = .1,
  n_items = 6, true_item_covariance = .6, true_residual_var = .4,
  conf_level = .95, assurance = .85, G_initial = 20, G = 100, seed = 113)

# The assurance plan under the same FMI. The search runs at the complete
# data size, the necessary N of the call above (kept as the attribute
# necessary_N_complete), and maps it back.
ss_aipe_reliability(model = "Parallel", type = "Normal Theory", width = .1,
  n_items = 6, true_item_covariance = .6, true_residual_var = .4,
  conf_level = .95, assurance = .85, G_initial = 20, G = 100, seed = 113,
  fmi = .30)

# The true score (tau equivalent) model takes true_residual_var as a
# vector of length n_items (number of items) while true_item_covariance
# stays a single value. Error
# variances around 1 - .5 give item variances from .9 to 1.1 and
# correlations near .5, and the plan is for alpha of these items as they
# are, unequal variances and all.
ss_aipe_reliability(model = "True Score", type = "Normal Theory",
  width = .1, n_items = 5, true_item_covariance = .5,
  true_residual_var = c(.4, .5, .6, .4, .6),
  conf_level = .95, assurance = .85, G_initial = 20, G = 100,
  seed = 113)

# Congeneric model, planned from the item loadings and error variances rather
# than from a single covariance. With assurance = NULL the necessary N comes
# from the delta method variance of the model implied omega at the
# population, the variance behind reliability_omega(denominator =
# "model_implied", ci_method = "ml"). Adding an assurance to this plan fits
# a one factor model at every Monte Carlo iteration and is the slow case,
# so the page stops at the closed form (see Details).
ss_aipe_reliability(model = "Congeneric", type = "Factor Analytic", width = .15,
  n_items = 4, true_factor_loading = c(.8, .7, .7, .8),
  true_residual_var = c(.4, .5, .5, .4),
  conf_level = .95, assurance = NULL)

# Planning from a presumed population correlation matrix among the items,
# which plans for standardized items.
pop_mat <- rbind(
  c(1.0000000, 0.3813850, 0.4216370, 0.3651484, 0.4472136),
  c(0.3813850, 1.0000000, 0.4020151, 0.3481553, 0.4264014),
  c(0.4216370, 0.4020151, 1.0000000, 0.3849002, 0.4714045),
  c(0.3651484, 0.3481553, 0.3849002, 1.0000000, 0.4082483),
  c(0.4472136, 0.4264014, 0.4714045, 0.4082483, 1.0000000))
```

```
ss_aipe_reliability(model = "True Score", type = "Normal Theory", width = .15,
  true_cov_matrix = pop_mat, conf_level = .95, assurance = .85,
  G_initial = 20, G = 100, seed = 113)
```

`ss_aipe_reliability_sensitivity`
Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for a Reliability Coefficient

---

Description

Quantifies how much misspecification of the population reliability coefficient distorts an AIPE-based sample size plan for the reliability of a composite score. On each replication the function simulates an $n \times J$ data matrix of subjects by items from a single-factor model of parallel tests whose population reliability of the sum score equals `true_reliability`, fits the requested estimator (alpha or omega) via the corresponding `reliability_*` function with the supplied `ci_method`, and records the realized reliability estimate and its confidence interval.

Population model. Each item has a single common-factor loading and uncorrelated unique error. With per-item variance normalized to 1, the loading and unique variance are chosen so the reliability of the sum score equals `true_reliability`:

$$\lambda^2 \;=\; \frac{\rho}{J(1-\rho)+\rho}, \qquad \psi^2 \;=\; 1-\lambda^2,$$

where ρ =`true_reliability` and J is the number of items, `n_items`. Item scores are $y_{ij} = \lambda T_i + e_{ij}$, with $T_i \sim N(0,1)$ and $e_{ij} \sim N(0,\psi^2)$.

Missing data. Two arguments carry a plan into incomplete data, and they stand to each other as `supposed_reliability` stands to `true_reliability`. `fmi` is the planning value: the fraction of missing information the plan assumes for the coefficient, passed to `ss_aipe_reliability`, which enlarges the sample size accordingly. `missing_prop` is the data generating mechanism: each simulated item score is deleted independently with that probability (missing completely at random), and every replication is analyzed by full information maximum likelihood. A plan made with `fmi` is then judged by the widths the incomplete data actually deliver, and a plan that ignored the missing data (`fmi = 0`) is seen to fall short. Planning with `fmi` while simulating complete data shows the opposite misspecification, widths narrower than planned.

Usage

```
ss_aipe_reliability_sensitivity(
  true_reliability = NULL,
  supposed_reliability = NULL,
  n_items,
  width,
  specified_N = NULL,
  estimator = c("alpha", "omega"),
  ci_method = NULL,
```

```
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  fmi = 0,
  missing_prop = 0,
  B = NULL,
  seed = NULL
)
```

Arguments

`true_reliability`	Population reliability coefficient (in $[0, 1)$).
`supposed_reliability`	Reliability supposed when the study is planned, a planning assumption rather than an estimate from data; the function passes the factor loading and residual variance it implies to `ss_aipe_reliability` as `true_factor_loading` and `true_residual_var`.
`n_items`	Number of items in the composite (J in the formulas): a whole number of at least 2, and at least 3 for `estimator = "omega"`.
`width`	Desired full width of the CI on reliability.
`specified_N`	Sample size to evaluate (incompatible with `supposed_reliability`).
`estimator`	One of `"alpha"` (default; coefficient alpha via `reliability_alpha`) or `"omega"` (composite reliability via `reliability_omega`). For a population of parallel tests the two coincide; differences in sample estimates reflect estimator-specific finite-sample bias and CI behavior.
`ci_method`	CI method passed to the estimator. The default `NULL` is `"ml"` for alpha and omega, with complete or incomplete data: the normal theory Wald interval the plan is made for (see *Details*). Any other interval the estimator offers can be named.
`conf_level`	Confidence level (default `0.95`).
`assurance`	Optional assurance probability passed to the planner, whose Monte Carlo search then runs at its defaults (`G_initial = 500`, `G = 5000`) and draws from the current random number stream, so `set.seed()` before the call makes the plan and the simulation reproducible together. The plan is the one the planner reports for the parallel model with the normal theory interval, which the default `ci_method` evaluates; `specified_N` evaluates any other size.
`G`	Number of Monte Carlo replications.
`print_iter`	Logical.
`filename`	Optional path for a comma separated file recording every replication (the sample reliability estimate, the two confidence limits, the interval width, and two indicators of whether the interval missed `true_reliability` below or above, followed, when `missing_prop` is above 0, by the fraction of missing information the replication realized for the coefficient): nothing is written when `filename`

is NULL (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at tempfile(fileext = ".csv").

fmi — The fraction of missing information the plan assumes for the coefficient, passed to ss_aipe_reliability when supposed_reliability resolves the sample size: a single number in $[0, 1)$, with the default 0 for a complete data plan. With specified_N there is no plan for it to act on, and a value above 0 is an error.

missing_prop — The probability with which each simulated item score is missing, completely at random: a single number in $[0, 1)$. The default 0 simulates complete data; above 0, every replication is analyzed by full information maximum likelihood (see *Details*).

B — Number of bootstrap replications inside each simulated study when ci_method is a bootstrap method. The default, NULL, is the larger of 200 and one and a quarter times the number the interval method and the level require (see dmar_bootstrap): 500 for "percentile", 2500 for "bca", and 200 for the two bootstrap standard error methods at the 95 percent level. The quarter is room for replications that return no value (a resample without a one-factor solution): the required number (400, 2000, 100) is held to the request, so a simulated study that loses a few replications still reports its interval, and with the quarter its limits still rest on about the required number. A simulated study in which more than half of the replications return no value reports no interval. Every simulated study runs B replications, so the run time is proportional to G times B. Not used by the other interval methods.

seed — NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

The interval evaluated. An AIPE plan is made for a particular confidence interval procedure, so a check of the plan evaluates that procedure. The plan here is the one ss_aipe_reliability makes for the parallel model with the normal theory interval: its closed form takes the sampling variance of alpha to be the compound symmetry form $2(1 - \rho)^2 J/((J - 1)(N - 1))$ of van Zyl, Neudecker, and Nel (2000), and its assurance search forms, on each simulated sample, the Wald interval with that variance at the sample alpha. The default ci_method = "ml" evaluates the interval of reliability_alpha(ci_method = "ml"), the Wald interval with the general variance of van Zyl et al. at the sample covariance matrix, which at a population of parallel items is the same variance; for omega it evaluates the interval of reliability_omega(denominator = "model_implied", ci_method = "ml"), the delta method Wald interval of the model implied omega, whose variance at a population of parallel items is that same compound symmetry form (it is the variance ss_aipe_reliability() plans with for the congeneric model, and the two agree to machine precision there). The plan does not describe other intervals, among them Bonett's (2002) for alpha and "mlr" for omega. With four items, a true and planned reliability of .80, a width of .15, and an assurance of .80, the plan of 99 cases gave alpha intervals no wider than .15 in 81.3 percent of 2,000 replications with "ml" and in 76.75 percent with Bonett's interval (set.seed(113) before each call); for omega the two intervals came closer (83.3 percent of 1,000 with "ml" and 81.9 with "mlr"). Any other interval the estimator offers can still be named in ci_method, which asks how the planned *N* serves an interval other than the one it was planned for.

The analysis under missing data. With `missing_prop` above 0, coefficient alpha is `reliability_alpha(missing = "fiml")`, the classical coefficient computed from the full information maximum likelihood estimate of the item covariance matrix, whose `"ml"` interval, the default here as with complete data, takes its standard error from the FIML information matrix of the saturated model. Coefficient omega is `reliability_omega(missing = "fiml")`, whose `"ml"` interval takes its standard error from the FIML information matrix of the single-factor model. These are the Wald intervals the planner's `fmi` adjustment plans for, with the standard error the missing data inflate. The complete data closed forms (Bonett, Feldt, Fisher, Hakstian-Whalen) and the asymptotic distribution-free intervals do not apply to partially observed rows, so, with `missing_prop` above 0, `ci_method` must be `"ml"` or `"ml_logistic"` for alpha and one of `"ml"`, `"mlr"`, `"ml_logistic"`, or `"mlr_logistic"` for omega. A row that loses every item score carries no information and is dropped, as lavaan drops it.

Choosing `fmi` for a given `missing_prop`. The two are not the same number. The FMI of the coefficient depends on the number of items and how strongly they are correlated, because information lost on one item is partly recovered from the others; under data missing completely at random it is bounded by the largest rates of univariate and pairwise missingness (Savalei & Rhemtulla, 2012). For the population of the second example (four parallel items, reliability .80, width .15), `missing_prop = .2` gives coefficient alpha and coefficient omega an FMI of .184 relative to every case, the footing of the planner's `fmi` (from one FIML fit to 200,000 simulated cases; it is .182 relative to the cases the fit keeps, since a share $.2^4 = .0016$ of them loses every item). In 1,000 replications with the default `ci_method = "ml"` (`set.seed(113)` before each call), the complete data plan ($N = 74$) gave alpha intervals that averaged .153 wide with complete data and .170 with `missing_prop = .2`, wider by close to the width inflation factor $1/\sqrt{1 - .184} = 1.107$, while the plan made with `fmi = .184` ($N = 91$) averaged .152 with `missing_prop = .2`. The help page of `ss_aipe_reliability` describes where a planning FMI comes from in practice.

The fraction of missing information the data realized. With `missing_prop` above 0, every replication that returns an interval also estimates the FMI of its coefficient from the same simulated data, as `fraction_missing_information` does with its default unstructured estimator: the FIML fit behind the coefficient is refit at its estimates with the information of the observed data and with the information complete data would have carried, and the FMI is one minus the ratio of the coefficient's complete data sampling variance to its sampling variance with the data as observed. For alpha the fit is the saturated model of the items whose covariance matrix the classical coefficient is computed from, and the three estimators of Chen and Savalei (2021) coincide for a saturated model (p. 9); for omega it is the single-factor model, and the FMI is that of its model implied coefficient. The FMI is taken relative to all *n* simulated cases, the *N* of the plan. A case that loses every item score is set aside by the fit, and the FMI relative to the n_{used} cases the fit used, γ_{used} (the FMI that `fraction_missing_information` reports), leaves out the information those cases would have carried; it is folded in as $1 - (n_{used}/n)(1 - \gamma_{used})$. The difference matters with few items and much missing data: a share p^J of the cases loses every item, with p the value of `missing_prop`, which is .125 for three items at `missing_prop = .5`. The mean over the replications is the `mean_realized_fmi` attribute and is printed under the table, so the `fmi` a plan assumed can be read beside the FMI that the simulated missing data cost the coefficient. A single sample FMI is a poor guide: in the simulations of Chen and Savalei (2021, p. 9) the sample FMI of a regression coefficient was largely unbiased but varied from sample to sample, with root mean squared errors of about .10 or more at *N* = 50 and 100. The mean over many replications is the stable figure, and it is the value to carry into `fmi` for a study whose missing data the simulation mimics. A replication whose FMI cannot be computed (a fit that fails, or an improper solution that **lavaan** will not refit at its estimates; see `fraction_missing_information`) is left out of the mean, and the `n_realized_fmi` attribute counts the replications that entered it. The computation adds a model fit

and two refits to every replication, about doubling the time a replication takes.

Bootstrap intervals. When `ci_method` is a bootstrap method (`"bootstrap_se"`, `"bootstrap_se_logistic"`, `"percentile"`, or `"bca"`), every one of the `G` replications runs its own bootstrap of `B` resamples, so the simulation computes the coefficient $G \times B$ times. The default `B` is the larger of 200 and one and a quarter times the number of replications the interval method and the level require (see `dmar_bootstrap`): at the 95 percent level, 500 for `"percentile"`, 2500 for `"bca"`, and 200 for the two bootstrap standard error methods. The quarter leaves room for replications that return no value: the number required is held to the request, so a simulated study that loses a few replications still reports its interval, and with the quarter its limits still rest on about the number required. That default is far below the `B = 10000` of `reliability_alpha` and `reliability_omega` and keeps the product manageable. It also makes each replication's limits less precise than the reported interval's would be: a percentile limit at `B = 500` rests on about 12 resamples in each tail, and the Monte Carlo error of the limits adds to the spread of the realized widths. A check of a bootstrap interval that will be reported with a large `B` is closer to that interval with a larger `B` here, at a cost that grows in proportion. A `B` below the required number is refused unless `options(dmar.bootstrap_exploratory = TRUE)` is set; the run is then marked by one warning of class `dmar_bootstrap_exploratory` for the call, not one for each simulated study.

The omega coefficient. Omega is computed with `denominator = "model_implied"`, the coefficient whose delta method interval the planner evaluates and the one the Wald intervals are defined for. For the parallel population simulated here the model implied and observed denominators estimate the same value.

Coverage at a true reliability of zero. Coverage and the misses in each tail are counted on the interval the estimation function reports, whose limits are each clamped into [0, 1] (see `reliability_alpha`), since that is the interval a plan is made for and a study reports. Clamping can only raise the coverage of a value in [0, 1], and at the boundary it does: with `true_reliability = 0`, an alpha interval lying wholly below zero is reported as [0, 0], which contains zero, so `pct_ci_miss_high` is zero by construction and `total_type_I_error` is the rate of misses low. A miss high there would place the reliability below zero, outside the range a reliability coefficient can take. With four items, $N = 40$, and Bonett's interval (400 replications, `set.seed(1)`) the intervals miss .0075 of the time, all of it low. For any true reliability above zero, an interval lying wholly below zero misses high.

Replications that fail. A replication that returns no interval (a fit that did not converge, which is more common under full information maximum likelihood with small samples and much missing data, a fit whose likelihood has no maximum, which the estimation functions refuse with an error of class `dmar_no_maximum` when an error variance runs off below zero while the likelihood is flat or still rising, or an interval the method could not form) is set aside: it counts in no summary, and one warning states how many were set aside and the first reason. When no replication returns an interval, the function stops with that reason rather than return a table of missing values.

Value

A `data.frame` with rows for mean / median / SD of the realized reliability and CI width, the proportion of intervals at or below `width`, tail-specific and overall non-coverage of `true_reliability`, and the input echoes `total_N`, `n_items`, `true_reliability`, `supposed_reliability`, `width`, `conf_level`, and `assurance` (present only when an assurance was supplied). The proportion of intervals at or below `width`, the widths, and the noncoverage proportions are over the replications that returned an interval. The `simulation` attribute records the numbers of replications attempted, with an interval, and without one (see the section on the record of the simulation),

and, here, the message of the first failure (`first_failure`). When `fmi` or `missing_prop` is above 0, the rows are unchanged and the missing data inputs travel as attributes: `fmi` (`NA` with `specified_N`), `missing_prop`, and, when `fmi` is above 0, `necessary_N_complete`, the planner's complete data sample size. When `missing_prop` is above 0, two more attributes record the fraction of missing information the simulated data realized for the coefficient: `mean_realized_fmi`, its mean over the replications in which it was computed, and `n_realized_fmi`, the number of those replications. With `fmi` and `missing_prop` both at 0 no attribute is added. With a bootstrap `ci_method` the result also carries the attributes `B`, `B_used`, `n_dropped`, `B_required`, and `bootstrap_status` of `dmar_bootstrap`; they record the request each simulated study ran (`B_used` equals `B` and `n_dropped` is 0, because replications dropped inside a simulated study are not totaled), and `bootstrap_status` is `"exploratory"` when `B` is below `B_required`.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G. (2002). Sample size requirements for testing and estimating coefficient alpha. *Journal of Educational and Behavioral Statistics, 27*(4), 335–340. doi:10.3102/10769986027004335

Chen, L., & Savalei, V. (2021). Three sample estimates of fraction of missing information from full information maximum likelihood. *Frontiers in Psychology, 12*, 667802. doi:10.3389/fpsyg.2021.667802

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Savalei, V., & Rhemtulla, M. (2012). On obtaining estimates of the fraction of missing information from full information maximum likelihood. *Structural Equation Modeling, 19*(3), 477–494. doi:10.1080/10705511.2012.687669

Terry, L. J., & Kelley, K. (2012). Sample size planning for composite reliability coefficients: Accuracy in parameter estimation via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 371–401. doi:10.1111/j.20448317.2011.02030.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

van Zyl, J. M., Neudecker, H., & Nel, D. G. (2000). On the distribution of the maximum likelihood estimator of Cronbach's alpha. *Psychometrika, 65*(3), 271–280. doi:10.1007/BF02296146

See Also

ss_aipe_reliability, reliability_alpha, reliability_omega

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(),
ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivit
ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(),
ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_semipartial_r(),
ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(),
ss_aipe_two_phase_sensitivity()

Examples

```
# Reduced Monte Carlo sweep (small G) so the example runs quickly;
# raise G for a production sensitivity analysis.
set.seed(113)
ss_aipe_reliability_sensitivity(
  true_reliability     = 0.80,
  supposed_reliability = 0.80,
  n_items = 4, width = 0.15,
  estimator = "alpha",
  G = 20, print_iter = FALSE
)

# The same plan made for an FMI of .18 and checked against data with each
# item score missing completely at random with probability .2, analyzed
# by full information maximum likelihood. The second line under the table
# gives the FMI those data realized for the coefficient, to read beside
# the .18 the plan assumed. Every replication fits a model, so G = 3 only
# shows the call; a reported check deserves G = 1000.
```

```
set.seed(113)
ss_aipe_reliability_sensitivity(
  true_reliability     = 0.80,
  supposed_reliability = 0.80,
  n_items = 4, width = 0.15,
  estimator = "alpha",
  G = 3, fmi = 0.18, missing_prop = 0.2
)
```

`ss_aipe_rmsea` *Sample Size Planning for RMSEA in SEM*

Description

Sample size planning for the population root mean square error of approximation (RMSEA) from the accuracy in parameter estimation (AIPE) perspective. The sample size is planned so that the expected width of a confidence interval for the population RMSEA is no larger than desired.

Usage

```
ss_aipe_rmsea(true_rmsea, df, width, conf_level = 0.9)
```

Arguments

`true_rmsea`	The population RMSEA planned for
`df`	Degrees of freedom of the model
`width`	Desired confidence interval width
`conf_level`	Desired confidence level of the planned interval. The default, 0.90, plans the conventional interval for the RMSEA, the one `ci_rmsea` forms by default (see Details).

Details

The width planned is that of the interval `ci_rmsea` forms when the sample RMSEA equals `true_rmsea`, with the sample size multiplier $N - 1$ of Kelley and Lai (2011). The RMSEA is conventionally reported with a 90 percent interval, because a $100(1 - 2\alpha)$ percent interval decides the one-sided tests of close and not close fit at level α (MacCallum, Browne, & Sugawara, 1996, p. 136), and every RMSEA interval in the package defaults to it, the interval planned here included. Kelley and Lai (2011) planned 95 percent intervals in their examples; a higher level needs a larger sample for the same width. For an RMSEA of .045 on 5 degrees of freedom and a width of .060, the plan is 802 at the default and 1068 with `conf_level = 0.95`. The printed table states the level beneath it.

Value

Returns the necessary total sample size in order to achieve the desired degree of accuracy (i.e., the sufficiently narrow confidence interval).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Lai, K. (2011). Accuracy in parameter estimation for the root mean square error of approximation: Sample size planning for narrow confidence intervals. *Multivariate Behavioral Research, 46*, 1–32. doi:10.1080/00273171.2011.543027

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130–149. doi:10.1037/1082989X.1.2.130

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ci_rmsea

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# The expected width of the 90 percent interval, the default, is .05.
ss_aipe_rmsea(true_rmsea = .035, df = 50, width = .05)

# The same width for a 95 percent interval needs a larger sample.
ss_aipe_rmsea(true_rmsea = .035, df = 50, width = .05, conf_level = .95)
```

---

ss_aipe_rmsea_sensitivity

A Priori Monte Carlo Simulation for Sample Size Planning for RMSEA in SEM

Description

Conduct a priori Monte Carlo simulation to empirically study the effects of (mis)specifications of input information on the calculated sample size. The sample size is planned so that the expected width of a confidence interval for the population RMSEA is no larger than desired. Random data are generated from the true covariance matrix but fit to the proposed model, whereas the sample size is calculated based on the input covariance matrix and proposed model.

Usage

```
ss_aipe_rmsea_sensitivity(
  width,
  model,
  true_cov_matrix,
  specified_N = NULL,
  conf_level = 0.9,
  G = 200,
  filename = NULL,
  ...,
  seed = NULL
)
```

Arguments

`width`	desired confidence interval width for the population RMSEA.
`model`	the model the researcher proposes, which may or may not be the true model, written in **lavaan** model syntax (see `model.syntax`). The observed variable names in the model must match the row and column names of `true_cov_matrix`.
`true_cov_matrix`	the true population covariance matrix, which is used to generate random data for the simulation study. The row and column names of `true_cov_matrix` must match the observed variables in `model`.
`specified_N`	the total sample size *N*. If `specified_N` is given, random samples of that size are generated. Otherwise the sample size is calculated with the sample size planning method so that the expected width of a confidence interval for the population RMSEA is no larger than `width`, for the interval formed as lavaan forms it (see Details).
`conf_level`	confidence level of the planned and the simulated intervals (i.e., 1 - the Type I error rate). The default, 0.90, is the conventional level of the RMSEA interval and the default of `ss_aipe_rmsea` and `ci_rmsea`.
`G`	number of replications in the Monte Carlo simulation.
`filename`	an optional path for a comma separated file recording every converged replication (its index, the RMSEA estimate, the two confidence limits, and the interval width): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`.
`...`	additional arguments passed to `sem` when fitting the model (for example `estimator` or `missing`). Every name given here must be an argument or an option of **lavaan** (see `lavOptions`); any other name, and any argument without a name, is refused before a model is fitted, with a condition of class `dmar_argument_invalid` that gives the name as it was typed
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

This function implements the sample size planning method proposed in Kelley and Lai (2011). It uses sem to fit the proposed model to the population covariance matrix, which recovers the population RMSEA (the model misspecification) and the model degrees of freedom, and to fit the model to each simulated sample, and it forms the confidence interval for the population RMSEA in each replication from that sample's chi square statistic, inverted with ci_nc_chisq as sem_fit_measures inverts it. The model is specified in **lavaan** syntax, so **lavaan** must be installed.

The population RMSEA is read from `lavaan::fitMeasures()`, which is computed reliably for the large-sample population fit.

Each replication's interval is formed from the statistic T of that replication's fit and the sample size multiplier a lavaan formed T with (`sample_size_convention` in ci_rmsea): $a = N$ for maximum likelihood under lavaan's default normal likelihood, and $a = N-1$ under `likelihood = "wishart"` and for the least squares estimators, either of which . . . can request. That is the interval lavaan itself reports for the sample, whatever the size of T. When T is at or below its degrees of freedom, lavaan reports an RMSEA of 0, from which ci_rmsea could rebuild only $T = df$; the interval formed from T itself is narrower the smaller T is, and its upper limit is 0 when T lies at or below the `(1 - conf_level) / 2` quantile of the central chi square, the convention lavaan and sem_fit_measures share. The replications of a model that fits well often have such statistics, so the distinction matters most when the proposed model is close to the true one. The RMSEA summaries are those of the RMSEA lavaan reports.

The width of the interval depends on the sample size only through a, and ss_aipe_rmsea plans with $a = N - 1$, the convention of Kelley and Lai (2011). The simulated samples have the planner's `necessary_N` cases, the size a study would collect, so under lavaan's default $a = N$ each realized interval is the one lavaan reports for a sample of that size, whose multiplier is one larger than the plan's, and the realized widths sit below the planned width by an amount of order $1/N$.

The RMSEA is conventionally reported with a 90 percent interval, because a $100(1 - 2\alpha)$ percent interval decides the one-sided tests of close and not close fit at level α (MacCallum, Browne, & Sugawara, 1996, p. 136), and that is the default here, as in ss_aipe_rmsea.

Value

A `data.frame` with columns `term` and `value` summarizing the a priori Monte Carlo study. The `term` entries are: `"mean_rmsea"`, `"median_rmsea"`, `"sd_rmsea"` (summaries of the realized RMSEA estimates across the converged replications); `"mean_ci_width"`, `"median_ci_width"`, `"sd_ci_width"` (summaries of the realized interval widths); `"pct_ci_less_width"` (proportion of intervals narrower than the target width); `"pct_ci_miss_low"` and `"pct_ci_miss_high"` (tail-specific empirical non-coverage of the population RMSEA); `"total_type_I_error"` (overall empirical non-coverage, the sum of the two tails); and the echoes `"total_N"` (the *N* evaluated), `"df"` (model degrees of freedom), `"true_rmsea"` (the population RMSEA recovered from fitting `model` to `true_cov_matrix`), `"width"`, and `"conf_level"`. The proportion rows are on the 0 to 1 scale, not percentages. The number of converged replications is `attr(x, "simulation")$returned` (see the section on the record of the simulation).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The pro-

portion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Note

Replications in which **lavaan** fails to converge, or for which the RMSEA is undefined, are skipped; the number of converged replications is reported as `attr(x, "simulation")$returned`. A fit whose likelihood has no maximum, an error variance running off below zero while the likelihood is flat or still rising (see `fa_bifactor`), is used as it is, at `true_cov_matrix` and in a replication: its estimates are wherever **lavaan**'s optimizer stopped on that ridge, but the RMSEA is formed from the chi square alone, which is not, since the likelihood levels off along the ridge and the optimizer stops only where its steps no longer gain. Increase `G` if many replications fail to converge.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cudeck, R., & Browne, M. W. (1992). Constructing a covariance matrix that yields a specified minimizer and a specified minimum discrepancy function value. *Psychometrika, 57*, 357–369. doi:10.1007/BF02295424

Kelley, K., & Lai, K. (2011). Accuracy in parameter estimation for the root mean square error of approximation: Sample size planning for narrow confidence intervals. *Multivariate Behavioral Research, 46*, 1–32. doi:10.1080/00273171.2011.543027

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130–149. doi:10.1037/1082989X.1.2.130

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

See Also

sem, ss_aipe_rmsea, ci_rmsea, sem_fit_measures

Examples

```
# True data generating model: two correlated factors, each measured by
# three standardized indicators. The factor correlation is 0.5 and every
# loading is 0.7. The implied population covariance matrix is assembled
# from the loading matrix, the factor correlation matrix, and the
# residual variances.
Lambda <- matrix(0, 6, 2)
Lambda[1:3, 1] <- 0.7
Lambda[4:6, 2] <- 0.7
Phi    <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
true_cov_matrix <- Lambda %*% Phi %*% t(Lambda) + diag(1 - 0.7^2, 6)
dimnames(true_cov_matrix) <- list(paste0("x", 1:6), paste0("x", 1:6))

# Proposed (misspecified) model: a single common factor.
proposed <- "g =~ x1 + x2 + x3 + x4 + x5 + x6"

# The proposed model is fit once at a very large N to recover the
# population RMSEA, the sample size is planned so that the expected width
# of the 90 percent interval is 0.05, and a fresh sample of that size is
# drawn and fit on every replication. Notice that true_rmsea is about
# 0.20, since a single factor is a poor description of two-factor data,
# that the realized widths sit close to the target, and that
# pct_ci_less_width is well away from 0 and 1: planning for the expected
# width aims the average width at the target, not the share of intervals
# narrower than it, which is what assurance controls. G = 10 keeps
# the example quick and estimates that share only roughly; a reported
# sensitivity study deserves the default G = 200 or more.
set.seed(113)
ss_aipe_rmsea_sensitivity(width = 0.05, model = proposed,
                          true_cov_matrix = true_cov_matrix, G = 10)
```

---

ss_aipe_r_sensitivity *Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for a Pearson Correlation*

Description

Quantifies how much misspecification of the population Pearson correlation distorts an AIPE-based sample size plan. On each replication the function draws an n-row sample from a bivariate normal distribution with correlation `true_cor` and computes the sample correlation and its Fisher's Z CI, the interval ss_aipe_r plans for and correlations_test reports. Because the back-transformed

width is largest at $\rho = 0$ and shrinks as $|\rho|$ grows, a planning value whose magnitude overstates the population correlation yields realized intervals wider than planned, and the summary rows report by how much.

Usage

```
ss_aipe_r_sensitivity(
  true_cor = NULL,
  supposed_cor = NULL,
  width,
  specified_N = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_cor`	Population Pearson correlation ρ; must lie in $(-1, 1)$.
`supposed_cor`	Planning value of the correlation passed to `ss_aipe_r`; supply this or `specified_N` but not both.
`width`	Desired full width of the CI on the correlation.
`specified_N`	Sample size to evaluate (incompatible with `supposed_cor`).
`conf_level`	Confidence level (default `0.95`). A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in `ci_nc_t`).
`assurance`	Optional assurance probability passed to `ss_aipe_r`.
`G`	Number of Monte Carlo replications (default 1000).
`print_iter`	Logical. Print iteration index per replication.
`filename`	Optional path for a comma separated file recording every replication (the sample correlation, the two confidence limits, the interval width, and two indicators of whether the interval missed `true_cor` below or above): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`.
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Value

A `data.frame` with rows for the realized correlation, the interval width, the proportion of intervals at or below `width`, tail-specific and overall non-coverage of `true_cor`, and the input echoes, including `assurance` (present only when an assurance was supplied).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G., & Wright, T. A. (2000). Sample size requirements for estimating Pearson, Kendall and Spearman correlations. *Psychometrika, 65*(1), 23–28. doi:10.1007/BF02294183

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

`ss_aipe_r`, `ss_aipe_partial_r_sensitivity`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,

ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivit
ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(),
ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(),
ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost(),
ss_aipe_two_phase_sensitivity()

Examples

```
# Reduced replications and a wide target interval keep this fast.
set.seed(113)
ss_aipe_r_sensitivity(
  true_cor = 0.30, supposed_cor = 0.30, width = 0.40,
  G = 50, print_iter = FALSE
)
```

`ss_aipe_sc` *Sample Size Planning for Accuracy in Parameter Estimation (AIPE) of the Standardized Contrast in ANOVA*

Description

Plans the sample size per group so that the confidence interval for a standardized contrast of means in a fixed effects analysis of variance, the interval computed by `ci_sc`, is sufficiently narrow, an application of the accuracy in parameter estimation (AIPE) approach to the standardized contrast.

Usage

```
ss_aipe_sc(
  true_std_contrast,
  c_weights,
  width,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  assurance = NULL,
  assurance_method = "analytic"
)
```

Arguments

`true_std_contrast`
: Population standardized contrast

`c_weights`
: The contrast weights, as `ci_sc` takes them: fractions whose positive values sum to 1 and whose sum is zero (for example `c(1/2, 1/2, -1)`, not `c(1, 1, -2)`), so that the standardized contrast is a difference of averages of means in units of the error standard deviation. Weights that break the rule are refused with the

message of ci_sc and a condition of class dmar_argument_invalid. The rule is applied to the values, so integer weights such as -1:1 are accepted, as they are by ci_sc.

width — The desired full width of the obtained confidence interval

conf_level — The desired confidence interval coverage (i.e., 1 - Type I error rate). Default is .95, which gives a symmetric two-sided interval. Specify either conf_level or both of alpha_lower and alpha_upper, not both. A level above .999999998, whose tail areas are below 1e-9, is refused.

alpha_lower — Lower-tail Type I error rate, used to plan an asymmetric confidence interval. When supplied together with alpha_upper, the planned interval has lower-tail probability alpha_lower and upper-tail probability alpha_upper. Set conf_level = NULL when supplying these.

alpha_upper — Upper-tail Type I error rate, used together with alpha_lower to plan an asymmetric confidence interval. Each tail area must be at least 1e-9: below it the noncentral *t* limits lose accuracy (see "Small tail areas" in ci_nc_t).

assurance — Parameter to ensure that the obtained confidence interval width is narrower than the desired width with a specified degree of certainty (must be NULL or greater than .5 and less than 1). An assurance above .999999998 is refused: the floor of 1e-9 on a tail area (see "Small tail areas" in ci_nc_t) applies to (1 - assurance) / 2.

assurance_method — How the probability that the ci_sc interval is no wider than width is found, used only with assurance. Only "analytic" (the default) is offered here: the probability is computed exactly, and the plan is the smallest sample size per group, not below the plan without assurance, at which it reaches assurance. "simulation", which ss_aipe_sc_ancova, ss_aipe_reg_coef with noncentral = TRUE, and ss_aipe_src offer, where the probability has no exact form, is refused with a condition of class dmar_argument_invalid, whether or not assurance is given, since a simulated estimate of an exact probability would only add Monte Carlo error.

Details

The plan is for the interval of ci_sc, the limits for the noncentrality parameter of a noncentral *t* on $J(n-1)$ degrees of freedom at the observed *t* times $\sqrt{\sum_j c_j^2/n}$. Without assurance it is the smallest *n* per group, and at least 4, at which that interval, evaluated at the planning value, is no wider than width; a width that would need more than 2^{53} (about 9.0×10^{15}, the largest number up to which R holds every whole number exactly) per group is refused with a condition of class dmar_argument_invalid.

With assurance the plan is the smallest sample size per group, not below the plan without assurance, at which the probability that the interval is no wider than width is at least assurance (assurance_method = "analytic"), and that probability, computed exactly, is assurance_attained. The observed *t* is noncentral *t* on $J(n-1)$ degrees of freedom with noncentrality $\delta/\sqrt{\sum_j c_j^2/n}$, the width is smallest at $t = 0$ and rises on either side, so the event is $t_L \le T \le t_U$ for the two values at which the width equals the target, and its probability is that of the noncentral *t*. The plan is found by doubling a step upward from the plan without assurance and then bisecting, which assumes that

the probability does not decrease as *n* grows from there. In 96 planning sets (contrasts of 0, .5, -1, and 2; weights for two, three, and five groups; widths of .5 and 1; assurances of .80 and .99; and tail areas of .025 and .025 or .01 and .04) it did not decrease at any *n* from the plan without assurance to ten past the plan, and the plan was the smallest *n* found by raising *n* one at a time.

The step of Kelley and Rausch (2006), as Lai and Kelley (2012, pp. 358–359 and 361) apply it to a standardized contrast, plans without assurance for the value t^* that $|T|$ does not exceed with probability `assurance` at the plan without assurance, carried back to the scale of the standardized contrast; it does not compute the probability. That step can fall short of the request: at `true_std_contrast = 0`, `c_weights = c(1, -1)`, `width = .8`, and `assurance = .95` it gives 49 per group, which attains .9499, where the plan here is 50, which attains .9946. It can also be larger than the smallest plan that attains the request: in the 96 sets above it attained the request in every set and was one case or more larger in 17.

Past about 10^{11} per group the plan and `assurance_attained` are limited by the precision of doubles. The width changes by about $1/(2n)$ of itself from one *n* to the next, and for a nonzero contrast it is the difference of two limits near the noncentrality $\delta/\sqrt{\sum_j c_j^2/n}$, which a double holds to about 1.1×10^{-16} of its size. Against a 512-bit computation of the interval, with `c_weights = c(1, -1)` and `conf_level = .95`, and with `width` and the tail areas as R holds them (each tail area, (1 - .95) / 2, is 0.025000000000000022 in doubles): the plans checked up to 4.6e9 per group were the smallest *n*; at `true_std_contrast = 2` and `width` of 1e-5, 1e-6, 3e-7, and 1e-7 the plan was 2, 1,850, 487,182, and 6,485,787 below the smallest *n* (about 4.6e11, 4.6e13, 5.1e14, and 4.6e15), and at .5 it was 4 above, 1,956 below, 129 above, and 4,854,160 above it (about 3.2e11, 3.2e13, 3.5e14, and 3.2e15). At a contrast of 0 the width is computed from the normal quantiles, and the plan is below the smallest *n* only when the *n* at which the width equals `width` is within about $10^{-15}n$ above a whole number, and then by at most the larger of 1 and about $10^{-15}n$. At `width = 1e-6` that *n* is 30,731,670,565,552.9985, and the plan, 30,731,670,565,553 per group, is the smallest *n*; at 3e-7 that *n* is 341,463,006,283,922.21, and the plan, 341,463,006,283,922, is 1 below the smallest *n*; and the plan was 2 below at 1e-7 (about 3.1e15 per group), 4 below at 7e-8 (6.3e15), and 8 below at 6e-8 (8.5e15). With `assurance = .9`, `assurance_attained` was within 1e-6 of the exact probability up to 4.6e9 per group.

With unequal tails the width at an observed *t* is not the width at $-t$: the interval at $-t$ with tail areas (`alpha_lower`, `alpha_upper`) is the reflection of the interval at *t* with the two exchanged. A negative `true_std_contrast` therefore gets the plan of its magnitude with `alpha_lower` and `alpha_upper` exchanged, so that reversing the contrast and the tails together leaves the plan unchanged. Without `assurance` the signed planning value already gives that plan. With equal tails the plan does not depend on the sign.

Value

`necessary_n_per_group`
: Necessary sample size *per group*

`assurance`, `assurance_attained`
: With `assurance` only: the request, and the exact probability at `necessary_n_per_group` that the `ci_sc` interval is no wider than `width` (at least `assurance`). The printed footer names the assurance method, which the attribute `assurance_method` records, with `assurance_probability` `"exact"`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K. (2005). The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals, *Educational and Psychological Measurement, 65*, 51–69. doi:10.1177/0013164404264850

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`ci_sc`, `ci_nc_t`, `ss_aipe_c`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Suppose the population standardized contrast is believed to be .6
# in some 5-group ANOVA model. The researcher is interested in comparing
# the average of means of group 1 and 2 with the average of group 3 and 4.

# To calculate the necessary sample size per group such that the width
# of 95 percent confidence interval of the standardized
# contrast is, with 90 percent assurance, no wider than .4 (the last row
# is the exact probability at that sample size):

ss_aipe_sc(true_std_contrast=.6, c_weights=c(.5, .5, -.5, -.5, 0), width=.4, assurance=.90)

# Asymmetric confidence interval: most of the alpha goes in the upper tail
# (e.g., when a one-sided concern dominates). Pass alpha_lower and
# alpha_upper instead of conf_level.
ss_aipe_sc(true_std_contrast = .6, c_weights = c(.5, .5, -.5, -.5, 0), width = .4,
```

```
conf_level = NULL, alpha_lower = .01, alpha_upper = .04)
```

`ss_aipe_sc_ancova` *Sample Size Planning From the AIPE Perspective for Standardized ANCOVA Contrasts*

Description

Sample size planning from the accuracy in parameter estimation (AIPE) perspective for standardized ANCOVA contrasts.

Usage

```
ss_aipe_sc_ancova(
  true_contrast = NULL,
  true_sd_anova = NULL,
  true_sd_ancova = NULL,
  true_std_contrast = NULL,
  true_sd_ratio = NULL,
  true_cor = NULL,
  divisor = "s_ancova",
  c_weights,
  width,
  conf_level = 0.95,
  alpha_lower = NULL,
  alpha_upper = NULL,
  assurance = NULL,
  assurance_method = c("analytic", "simulation"),
  G = 10000,
  seed = NULL
)
```

Arguments

`true_contrast`	The population unstandardized ANCOVA (adjusted) contrast
`true_sd_anova`	The population error standard deviation of the ANOVA model
`true_sd_ancova`	The population error standard deviation of the ANCOVA model
`true_std_contrast`	The population standardized ANCOVA (adjusted) contrast
`true_sd_ratio`	The ratio of `true_sd_ancova` over `true_sd_anova`, greater than 0 and at most 1
`true_cor`	The population correlation coefficient between the response and the covariate
`divisor`	Which error standard deviation to be used in standardizing the contrast; the value can be either `"s_ancova"` or `"s_anova"`

`c_weights` Contrast weights

`width` The desired full width of the obtained confidence interval

`conf_level` The desired confidence interval coverage (i.e., 1 - Type I error rate). Default is `.95`, which gives a symmetric two-sided interval, the interval `ci_sc_ancova` reports. A level above .999999998, whose tail areas are below 1e-9, is refused.

`alpha_lower`, `alpha_upper`
The lower-tail and upper-tail Type I error rates, an alternative to `conf_level` (give `conf_level = NULL` with them). `ci_sc_ancova` reports intervals with equal tail areas only, so the two must be equal, and `alpha_lower = alpha_upper = a` plans the interval of `conf_level = 1 - 2a`. Unequal values are refused with a condition of class `dmar_argument_invalid`: they would plan for an interval no DMAR function reports. Each must be at least 1e-9: below it the noncentral *t* limits lose accuracy (see "Small tail areas" in `ci_nc_t`).

`assurance` Parameter to ensure that the obtained confidence interval width is narrower than the desired width with a specified degree of certainty (must be `NULL` or greater than .5 and less than 1). An assurance above .999999998 is refused, with either `assurance_method`: the published assurance step finds noncentral *t* limits with tail areas of `(1 - assurance) / 2`, which must be at least 1e-9.

`assurance_method`
How the plan with `assurance` is made, used only with `assurance`. `"analytic"` (the default) is the published plan of Lai and Kelley (2012), computed without simulation, which does not compute the probability it attains and can attain less than `assurance`; `assurance_attained` is then `NA`. `"simulation"` returns the smallest sample size per group, at or above the plan without `assurance`, at which the probability that the `ci_sc_ancova` interval is no wider than `width` is at least `assurance`, with that probability estimated from `G` simulated samples and reported as `assurance_attained` (see Details).

`G` Number of simulated samples at each sample size the assurance step tries, and in its estimate of the attained probability; a whole number of at least 100 (default 10000, which gives a Monte Carlo standard error of .004 for a probability of .80). Used and checked only with `assurance` and `assurance_method = "simulation"`.

`seed` Optional single number. When given, the assurance step is reproducible, and the random number stream of the calling session is left as it was. With `NULL` (the default) the draws come from the session's current stream. Used only where `G` is.

Details

The sample size planning method this function is based on is developed in the context of simple (i.e., one-response-one-covariate) ANCOVA model and randomized design (i.e., same population covariate mean across groups).

An ANCOVA contrast can be standardized in at least two ways: (a) divided by the error standard deviation of the ANOVA model, (b) divided by the error standard deviation of the ANCOVA model. This function can be used to analyze both types of standardized ANCOVA contrasts.

Not all of the effect size arguments need to be specified. When `divisor="s_ancova"` the input is either (a) `true_std_contrast`, or (b) `true_contrast` (the unstandardized ANCOVA contrast) and `true_sd_ancova`. When `divisor="s_anova"`, the valid input combinations are (a) `true_std_contrast` and `true_sd_ratio`; (b) `true_std_contrast` and `true_cor`; or (c) `true_contrast`, `true_sd_anova`, and `true_sd_ancova`. With `divisor = "s_anova"` the ratio of the two standard deviations, $\sqrt{1-\rho^2}$, must be greater than 0 and at most 1, as it is in every population (a covariate cannot raise the error variance), so `true_cor` of 1 or -1 is refused there.

The interval planned for. The plan is for the interval of `ci_sc_ancova`: the limits for the noncentrality parameter of a noncentral t on $N - J - 1$ degrees of freedom at the observed t of the adjusted contrast, carried back to the scale of the standardized contrast by the standard error $\sqrt{C/n + (\sum_j c_j \bar{X}_j)^2 / SS_{W,X}}$, with $C = \sum_j c_j^2$, $\bar{X}_j$ the covariate means, and $SS_{W,X}$ the within-group sum of squares of the covariate (and, for the divisor `"s_anova"`, by the ratio s_{ANCOVA}/s_{ANOVA}). The plan without `assurance` is for the criterion that this interval, evaluated at the planning value, is no wider than `width` (Lai & Kelley, 2012, Equations 21 and 28); it uses $\sigma_{ANCOVA}\sqrt{C/n}$ as the standard error of the contrast, leaving out the covariate means term, which in a randomized design is small (Lai & Kelley, 2012, pp. 354–355). The plan is the smallest *n* per group that meets the criterion, searched from 2 per group, the smallest sample size with positive error degrees of freedom. The search doubles a step and then bisects, which finds the smallest *n* because the interval at the planning value narrows as *n* grows from 2 (checked at every *n* up to 80, and in steps of 10 up to 600, for 420 combinations of the contrast, the ratio $\sigma_{ANCOVA}/\sigma_{ANOVA}$, weights for two to five groups, and the confidence level). At `true_std_contrast = 0`, `c_weights = c(1, -1)`, and `width = .5` the plan is 123 per group, where the interval is .49985 wide (.50190 at 122), and at `true_std_contrast = .5`, `true_cor = .5`, `divisor = "s_anova"`, `c_weights = c(1, -1)`, and `width = 3.6` it is 2 per group, where the interval is 3.596 wide. A `width` that would need more than 2^{53} (about 9.0×10^{15}, the largest number up to which R holds every whole number exactly) per group is refused with a condition of class `dmar_argument_invalid`. The intervals have equal tail areas, as those of `ci_sc_ancova` do and as Lai and Kelley (2012, p. 354) assume.

Past about 10^{11} per group the plan and `assurance_attained` are limited by the precision of doubles, as for `ss_aipe_sc`: the width changes by about $1/(2n)$ of itself from one *n* to the next, and for a nonzero contrast it is the difference of two limits near the noncentrality. Against a 512-bit computation of the interval, with `c_weights = c(1, -1)`, `conf_level = .95`, and the divisor `"s_ancova"`, and with `width` and the tail areas as R holds them (each tail area, (1 - .95) / 2, is 0.025000000000000022 in doubles): at `true_std_contrast = .5` and `width` of 1e-5, 1e-6, and 1e-7 the plan was 4, 1,035, and 525,379 above the smallest *n* that meets the criterion (about 3.2e11, 3.2e13, and 3.2e15 per group). At a contrast of 0, where the width is computed from the normal quantiles, the plan was the smallest *n* at `width` of 1e-3, 1e-5, and 1e-6 (about 3.1e7, 3.1e11, and 3.1e13 per group) and 2 below it at 1e-7 (about 3.1e15). Past about 10^{15} per group the plan at a contrast of 0 can be below the smallest *n*, by up to about $10^{-15}n$: at `width = 6e-8` it is 8,536,575,157,098,048 per group, 8 below the smallest *n*, 8,536,575,157,098,056. With `assurance = .9`, `assurance_method = "simulation"`, `G = 200`, and `seed = 1`, `assurance_attained` was the share of the same simulated statistics whose interval, computed in 512 bits, is no wider than `width`, up to 3.1e11 per group; at about 3.1e13 per group it was .950 where that share is .940 (contrast 0) and .900 where it is .895 (contrast .5), and at about 3.1e15 it was 1.000 where it is 0 (the plan is below the smallest *n*) and .905 where it is .930.

The assurance step. With `assurance` the plan is by default (`assurance_method = "analytic"`) the published one (Lai & Kelley, 2012, pp. 358–359, a generalization of Kelley & Rausch, 2006): the value t^* that $|T|$ does not exceed with probability `assurance` under the planner's model, carried

back to the scale of the standardized contrast, is planned for without assurance. It is computed without simulation and does not compute the probability it attains, so `assurance_attained` is `NA`, and the printed footer says so. That model leaves out the covariate means term and treats s_{ANCOVA}/s_{ANOVA} as known, and the widths vary little, so the plan attains less than the request: at `true_std_contrast = .5`, `true_cor = .5`, `c_weights = c(1, -1)`, `width = .5`, and `assurance = .80`, the published plans (129 per group with `divisor = "s_ancova"`, 98 with `"s_anova"`) give an interval no wider than .5 with probability .77 and .55.

With `assurance_method = "simulation"` the plan is the smallest sample size per group, not below the plan without `assurance`, at which the `ci_sc_ancova` interval is no wider than `width` with at least that probability, for normal within-group errors and a normal covariate with the same distribution in every group (the randomized design). The width depends on the sample through five independent statistics: the contrast of the covariate means, the within-group sum of squares of the covariate, the contrast of the group means of the errors, the error of the pooled within-group slope, and the ANCOVA error sum of squares. The probability is estimated from `G` draws of these statistics from their exact sampling distributions, the same draws at every sample size the search tries, so that the estimate changes smoothly as the sample size grows, and both the covariate means term and, with `divisor = "s_anova"`, the sampling of s_{ANCOVA}/s_{ANOVA} enter it. `assurance_attained` is the estimate at the returned sample size, at least `assurance`, with its Monte Carlo standard error in the attribute `assurance_mc_se`. Give `seed` for a plan that can be reproduced. For the divisor `"s_anova"`, Lai and Kelley (2012, pp. 360–361) wrote that their assurance method is not appropriate, because the width varies with s_{ANCOVA}/s_{ANOVA} as well as with t, and gave as a practical solution an a priori Monte Carlo simulation that raises the sample size until the proportion of intervals no wider than the target reaches the assurance; this step is that simulation, made from the sufficient statistics rather than from raw data, and applied to both divisors. In the example above the simulated plans (130 and 104 per group, `seed = 1`) give an interval no wider than .5 with probability .87 and .84. Because the probability is estimated, a plan can differ by one from the smallest sample size at which the probability itself reaches `assurance`: with `"s_anova"` above, 103 gives .803. `ss_aipe_sc_ancova_sensitivity` simulates raw data at a plan, and the attained probability of any plan can be checked there.

Value

A `data.frame` with columns `term` and `value`: `necessary_n_per_group`, the per-group sample size needed for the planned ANCOVA contrast; with `assurance`, the rows `assurance` (the request) and `assurance_attained`: `NA` for the published plan of `assurance_method = "analytic"`, which does not compute it, and for `assurance_method = "simulation"` the simulated probability at `necessary_n_per_group` that the `ci_sc_ancova` interval is no wider than `width`, with its Monte Carlo standard error in the attribute `assurance_mc_se` and the row `G`. With `assurance` the printed footer names the assurance method and says how `assurance_attained` was found, or that the published plan does not compute it; the attributes `assurance_method` and `assurance_probability` (`"simulation"` or `"not computed"`) record the same.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`ss_aipe_sc`, `ss_aipe_sc_ancova_sensitivity`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
ss_aipe_sc_ancova(true_std_contrast = .8, width = .5, c_weights = c(.5, .5, 0, -1))

ss_aipe_sc_ancova(true_std_contrast = .8, true_sd_ratio = .6, width = .5,
                  c_weights = c(.5, .5, 0, -1), divisor = "s_anova")

ss_aipe_sc_ancova(true_std_contrast = .5, true_cor = .4, width = .3,
                c_weights = c(.5, .5, 0, -1), divisor = "s_anova")

# With assurance, the published plan of Lai and Kelley (2012), which
# does not compute the probability it attains.
ss_aipe_sc_ancova(true_std_contrast = .5, true_cor = .5, width = .5,
                  c_weights = c(1, -1), assurance = .90)

# By simulation: the smallest sample size per group at which the
# interval is no wider than .5 with probability .90. G = 2000 keeps the
# example quick; a reported plan deserves the default G = 10000.
plan <- ss_aipe_sc_ancova(true_std_contrast = .5, true_cor = .5, width = .5,
                          c_weights = c(1, -1), assurance = .90,
                          assurance_method = "simulation",
                          G = 2000, seed = 113)
plan
attr(plan, "assurance_mc_se")
```

`ss_aipe_sc_ancova_sensitivity` *Sensitivity Analysis for the Sample Size Planning Method for Standardized ANCOVA Contrast*

Description

Sensitivity analysis for the sample size planning method with the goal to obtain sufficiently narrow confidence intervals for standardized ANCOVA complex contrasts.

Usage

```
ss_aipe_sc_ancova_sensitivity(
  true_std_contrast = NULL,
  supposed_std_contrast = NULL,
  c_weights,
  width = NULL,
  specified_n_per_group = NULL,
  true_mean_X = 0,
  true_sd_X = 1,
  true_cor,
  divisor = "s_ancova",
  assurance = NULL,
  conf_level = 0.95,
  G = 10000,
  print_iter = TRUE,
  filename = NULL,
  assurance_method = c("analytic", "simulation"),
  seed = NULL
)
```

Arguments

`true_std_contrast`	the population standardized ANCOVA contrast
`supposed_std_contrast`	the supposed standardized ANCOVA contrast, the planning value given to `ss_aipe_sc_ancova` to choose the sample size
`c_weights`	the contrast weights
`width`	the desired full width of the obtained confidence interval. Required when `supposed_std_contrast` is given, because the sample size is planned from it; a call without it is refused with a condition of class `dmar_argument_invalid`.
`specified_n_per_group`	selected sample size per group to use in order to determine distributional properties of a given value of sample size
`true_mean_X`	the population mean for the covariate

true_sd_X	the population standard deviation of the covariate
true_cor	the population correlation coefficient between the response and the covariate
divisor	which error standard deviation to be used in standardizing the contrast; the value can be either "s_ancova" or "s_anova"
assurance	parameter to ensure that the obtained confidence interval width is narrower than the desired width with a specified degree of certainty. When the sample size is planned from supposed_std_contrast, it is passed to ss_aipe_sc_ancova and must be NULL or greater than .5 and less than 1; with specified_n_per_group it is only reported in the result.
conf_level	the desired confidence interval coverage, (i.e., 1 - Type I error rate)
G	number of generations (i.e., replications) of the simulation
print_iter	to print the current value of the iterations
filename	Optional path of a CSV file to receive the per-replication results (the observed standardized contrast, std_contrast, the full and one-sided interval widths, the tail and overall misses, and the confidence limits), appended when the file already exists and created otherwise; the default NULL writes nothing, and a throw-away run that wants the file should point it at tempfile(fileext = ".csv").
assurance_method	Passed to ss_aipe_sc_ancova, which plans the sample size from supposed_std_contrast with assurance: "analytic" (the default) for the published plan of Lai and Kelley (2012), which does not compute the probability it attains and can attain less than assurance, and "simulation" for the smallest sample size per group at which the probability of an interval no wider than width, estimated by the planner from its default G of 10000 simulated samples with the random number stream that seed sets, is at least assurance. Not used with specified_n_per_group or without assurance.
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

The sample size planning method this function is based on is developed in the context of simple (i.e., one-response-one-covariate) ANCOVA model and randomized design (i.e., same population covariate mean across groups).

An ANCOVA contrast can be standardized in at least two ways: (a) divided by the error standard deviation of the ANOVA model, (b) divided by the error standard deviation of the ANCOVA model. This function can be used to analyze both types of standardized ANCOVA contrasts.

The population mean and standard deviation of the covariate does not affect the sample size planning procedure; they can be specified as any values that are considered as reasonable by the user.

Value

A data.frame with columns term and value summarizing the Monte Carlo sensitivity analysis across G replications. The term entries are: mean_std_contrast, median_std_contrast, sd_std_contrast (summaries of the realized standardized ANCOVA contrast); mean_ci_width,

median_ci_width, sd_ci_width (summaries of the full interval widths); mean_ci_width_lower and mean_ci_width_upper (mean one-sided widths, measured from the observed contrast to each limit); pct_ci_less_width (proportion of intervals at or below the target width); pct_ci_miss_low and pct_ci_miss_high (tail-specific empirical non-coverage of true_std_contrast); total_type_I_error (overall empirical non-coverage, the sum of the two tails); and the input echoes n_per_group (the size simulated at, planned or specified), total_N, true_std_contrast, supposed_std_contrast (NA when specified_n_per_group was supplied instead), true_cor, width, conf_level, and assurance (present only when an assurance was supplied). The proportion and Type I error rows are proportions on the 0 to 1 scale, not percentages.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

width_target_mc_ci the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as G grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives G, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so failed is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`ss_aipe_sc_ancova`, `ss_aipe_sc_sensitivity`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Sensitivity analysis for a standardized ANCOVA contrast across
# three groups, contrast (-1, 0, 1), a covariate-outcome correlation
# of 0.4, and a planning target width of 0.5. G = 50 keeps the
# example quick; a reported sensitivity analysis deserves the default of
# G = 10000 replications.
set.seed(113)
ss_aipe_sc_ancova_sensitivity(
  true_std_contrast = 0.5, supposed_std_contrast = 0.5,
  c_weights = c(-1, 0, 1),
  width = 0.5, true_cor = 0.4,
  conf_level = 0.95, G = 50, print_iter = FALSE
)
```

---

ss_aipe_sc_sensitivity

Sensitivity Analysis for Sample Size Planning for the Standardized ANOVA Contrast From the Accuracy in Parameter Estimation (AIPE) Perspective

---

Description

Performs a sensitivity analysis when planning sample size from the Accuracy in Parameter Estimation (AIPE) Perspective for the standardized ANOVA contrast.

Usage

```
ss_aipe_sc_sensitivity(
  true_std_contrast = NULL,
  supposed_std_contrast = NULL,
```

```
  c_weights,
  width = NULL,
  specified_n_per_group = NULL,
  assurance = NULL,
  conf_level = 0.95,
  G = 10000,
  print_iter = TRUE,
  filename = NULL,
  assurance_method = "analytic",
  seed = NULL
)
```

Arguments

`true_std_contrast`	population standardized contrast
`supposed_std_contrast`	supposed standardized contrast, the planning value given to `ss_aipe_sc` to choose the sample size
`c_weights`	the contrast weights, fractions whose positive values sum to 1 and whose sum is zero, the rule of `ci_sc` applied to the values as `ss_aipe_sc` applies it: weights `ss_aipe_sc` refuses are refused here, with a condition of class `dmar_argument_invalid`, and integer weights such as `-1:1` are accepted
`width`	the desired full width of the obtained confidence interval
`specified_n_per_group`	selected sample size to use in order to determine distributional properties of at a given value of sample size
`assurance`	parameter to ensure that the obtained confidence interval width is narrower than the desired width with a specified degree of certainty (must be NULL or between zero and unity)
`conf_level`	the desired confidence interval coverage, (i.e., 1 - Type I error rate)
`G`	number of generations (i.e., replications) of the simulation
`print_iter`	to print the current value of the iterations
`filename`	an optional path for a comma separated file recording every replication (the realized standardized contrast, `std_contrast`, the full and the two one-sided interval widths, the three non-coverage indicators, and the two confidence limits): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`
`assurance_method`	passed to `ss_aipe_sc` when the sample size is planned from `supposed_std_contrast` with `assurance`. Only `"analytic"` (the default) is offered: the plan is the smallest sample size per group, at or above the plan without `assurance`, at which the exact probability of an interval no wider than `width` is at least `assurance`. `"simulation"` is refused with a condition of class `dmar_argument_invalid`, as it is by `ss_aipe_sc`, since that probability is exact, whether or not `assurance` or `specified_n_per_group` is given.

seed — NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Each replication draws J groups of n normal observations with standard deviation 1 and forms the ci_sc interval from the group means and the pooled standard deviation, the square root of the mean square error that ci_sc takes as s_anova.

Value

A data.frame with columns term and value summarizing the Monte Carlo sensitivity analysis across the G replications. The term entries are: mean_std_contrast, median_std_contrast, sd_std_contrast (summaries of the realized standardized contrast); mean_ci_width, median_ci_width, sd_ci_width (summaries of the full interval widths); mean_ci_width_lower and mean_ci_width_upper (mean one-sided widths, measured from the observed contrast to each limit); pct_ci_less_width (proportion of intervals at or below the target width); pct_ci_miss_low and pct_ci_miss_high (tail-specific empirical non-coverage of true_std_contrast); total_type_I_error (overall empirical non-coverage, the sum of the two tails); and the input echoes n_per_group (the size simulated at, planned or specified), total_N, true_std_contrast, supposed_std_contrast (NA when specified_n_per_group was supplied instead), width, conf_level, and assurance (present only when an assurance was supplied). The proportion and Type I error rows are proportions on the 0 to 1 scale, not percentages.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

width_target_mc_ci the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as G grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives G, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so failed is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in Parameter Estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`ss_aipe_sc`, `ss_aipe_c`, `ci_nc_t`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Sensitivity analysis for a standardized three-group ANOVA contrast
# (-1, 0, 1) with a standardized contrast of 0.5 and target full width 0.40. G is kept small
# here so the example runs quickly; raise it for a stable sweep.
set.seed(113)
ss_aipe_sc_sensitivity(
  true_std_contrast = 0.5, supposed_std_contrast = 0.5,
  c_weights = c(-1, 0, 1),
  width = 0.40,
  conf_level = 0.95, G = 50, print_iter = FALSE
)
```

`ss_aipe_semipartial_r` *Sample Size for AIPE on a Semipartial (Part) Correlation*

Description

Takes a planning value of the population semipartial correlation $\rho_{Y(X \cdot Z_1 \cdots Z_{p-1})}$ (the correlation of Y with the part of X that $Z_1, \ldots, Z_{p-1}$ do not explain, with Y not residualized), the number of predictors (X and the variables partialed out), and a desired confidence interval width, and returns the smallest sample size at which the interval is expected to be that narrow (accuracy in parameter estimation, AIPE; Kelley & Maxwell, 2003). With `assurance` it returns, by simulation, the smallest sample size at which the realized interval is that narrow with about the stated probability (the search and the probability reported are both Monte Carlo estimates; see Details).

Usage

```
ss_aipe_semipartial_r(
  true_r_sp,
  p,
  width,
  which_width = c("Full", "Lower", "Upper"),
  conf_level = 0.95,
  assurance = NULL,
  true_R2_full = NULL,
  G = 20000,
  seed = NULL
)
```

Arguments

`true_r_sp`	Population semipartial correlation the sample size is planned for, in $(-1, 1)$. The planner treats this value as the truth; `ss_aipe_semipartial_r_sensitivity` shows what the plan delivers when the truth (`true_r_sp` there) differs from the value planned with (`supposed_r_sp` there).
`p`	Number of predictors in the regression of Y on X and the variables partialed out, so `p - 1` variables are partialed out and `p` is at least 2. A categorical variable with a levels counts as a - 1 predictors. It is the `p` of `ss_power_R2`, `ci_R2`, and `ci_reg_coef`: the correlation concerns one predictor of a regression with `p` predictors. In the notation of a model comparison, q is the number of predictors removed from the full model to give the restricted model ($q = p_{full} - p_{restricted}$, the predictors being tested), and the number of variables partialed out is $p - q$; this correlation concerns one predictor, so $q = 1$ and $p - 1$ variables are partialed out.
`width`	Desired full width of the confidence interval on the semipartial correlation.
`which_width`	Whether `width` refers to the `"Full"` width (default) or the `"Lower"`/`"Upper"` half-width.
`conf_level`	Desired confidence level (default `0.95`).

`assurance` Optional. The probability that the realized interval is no wider than `width`. The function returns the smallest sample size, at or above the sample size planned without it, at which the simulated probability is at least `assurance` (see "The assurance step" in Details). A single number above 0.5 and below 1; when `NULL` (the default) the plan is for the expected width only and nothing is simulated.

`true_R2_full` Population squared multiple correlation of the full model the sample size is planned for, the model with X and every variable partialed out (`true_R2_Y_X` in `ss_aipe_reg_coef`), treated as the truth as `true_r_sp` is, in $[0, 1]$ and not smaller than `true_r_sp` squared (a smaller value is refused with an error of class `dmar_semipartial_r_incoherent`, unless the pair could have come from rounding to two decimals, when the plan proceeds as if `true_R2_full` were `true_r_sp` squared and gives a warning of class `dmar_semipartial_r_rounded`; see `var_semipartial_r`). When `NULL` (the default), the plan uses the largest variance over every full-model R^2 that is possible with `true_r_sp`, so the sample size is sufficient whatever that R^2 is.

`G` Number of simulated samples at each sample size the assurance step tries, and in the final estimate of the attained probability; a whole number of at least 100 (default 20000). A smaller value is refused, because the step estimates a probability from these samples and with fewer than 100 its standard error can exceed .05. Used and checked only when `assurance` is given.

`seed` Optional single number. When given, the assurance step is reproducible, and the random number stream of the calling session is left as it was. With `NULL` (the default) the draws come from the session's current stream. Used only when `assurance` is given.

Details

Estimand and interval. The quantity planned for is the population semipartial correlation of Y with X given $Z_1, \ldots, Z_{p-1}$, for independent observations from a multivariate normal population. The interval is the two-sided Wald interval at `conf_level`, the estimate plus and minus $z_{1-\alpha/2}\sqrt{\hat{V}/(N-p)}$, where $\hat{V}$ is the first-order variance below evaluated at the sample semipartial correlation and the sample full-model R^2 (`var_semipartial_r` with `under_null = FALSE`). Its limits are not truncated at -1 and 1.

Asymptotic variance of the semipartial. Under multivariate normality, to first order, the sample semipartial correlation has variance V/N, with

$$V \;=\; (1-\rho_{sp}^2)^2 \;-\; P_r^2\,(1-2\rho_{sp}^2),$$

where ρ_{sp} is the population semipartial correlation (`true_r_sp`) and P_r^2 is the full-model R^2 (`true_R2_full`) minus ρ_{sp}^2, that is, the squared multiple correlation of Y with the variables partialed out. This is the first-order variance of Aloe and Becker (2012, Equation 4), which they obtain from Alf and Graf (1999), with $N-p$ in place of their n: in their notation the numerator is $P^4 - 2P^2 + P_{(f)}^2 + 1 - P_{(f)}^4$, with P^2 the full-model and $P_{(f)}^2$ the restricted-model squared multiple correlation, which equals V with $P_r^2 = P_{(f)}^2$. It follows from the delta method variance of the difference between two nested squared multiple correlations (Model A of Olkin & Finn, 1995, for one variable partialed out; Alf & Graf, 1999, for any number; see `var_semipartial_r`). The planner uses the divisor $N-p$, as `var_semipartial_r` does. Inverting for the sample size needed

to achieve a target half-width $w_{1/2}$ at confidence level $1 - \alpha$:

$$N \;=\; p + \left\lceil z^2_{1-\alpha/2} \cdot V / w^2_{1/2} \right\rceil.$$

Planning without `true_R2_full`**.** V decreases in P_r^2 when $\rho_{sp}^2 < 1/2$ and increases in P_r^2 when $\rho_{sp}^2 > 1/2$, so its largest value over every possible full-model R^2 is

$$\max\{(1-\rho_{sp}^2)^2,\ \rho_{sp}^2(1-\rho_{sp}^2)\},$$

and the planner uses that value when `true_R2_full` is `NULL`. For $\rho_{sp}^2 \le 1/2$ it is $(1-\rho_{sp}^2)^2$, the variance when the variables partialed out explain none of the outcome; Algina and Olejnik (2003, p. 317) note, for the squared semipartial correlation, that the required sample size is largest when the squared multiple correlation of the restricted model is zero. A planning value of `true_R2_full` then gives a smaller sample size. For $\rho_{sp}^2 > 1/2$ the largest value is $\rho_{sp}^2(1-\rho_{sp}^2)$, approached as the full-model R^2 approaches 1.

Comparison with partial-r planning. The partial correlation $r_{XY\cdot Z}$ divides the covariance after residualizing both X and Y on Z; the semipartial divides after residualizing only X. The semipartial is the natural effect size companion to a standardized regression coefficient: its square equals the ΔR^2 contributed by X above and beyond the variables partialed out. See `var_semipartial_r` for the asymptotic variance, and `ss_aipe_partial_r` for the partial-correlation analog of this function.

Limits of the plan. The plan is a first-order one. When the returned sample size is small, which happens when $|\rho_{sp}|$ is large and `true_R2_full` is near `true_r_sp` squared, the first-order variance understates the variance of the estimate and the realized intervals are wider than planned; check such a plan with `ss_aipe_semipartial_r_sensitivity`. The planned sample size is also small when `true_R2_full` is near 1, for any `true_r_sp`, and the same check applies: the plan for `true_r_sp = .1`, `p = 4`, a width of .20, and `true_R2_full = .99` is 12, at which the intervals missed the population value in .13 of 200,000 simulated samples. When $|\rho_{sp}|$ is above about .9 and the variables partialed out explain some of the outcome, the interval misses more often on one side than the other even with several hundred observations (in one simulated condition, $\rho_{sp} = .95$, a full-model R^2 of .97, $p = 2$, and $N = 395$, it missed in .064 of samples, .051 of them on one side). A plan with `true_r_sp = 0` and `true_R2_full = 1` is refused, because the first-order variance is zero there and no sample size follows from it.

The assurance step. A plan without `assurance` makes the interval as narrow as the target when the two sample values equal their planning values; in roughly half of all samples it is wider. With `assurance` the function looks for the sample size at which the realized interval is no wider than the target with that probability. The width of the interval depends on two statistics, the sample semipartial correlation and the sample full-model R^2, and no closed form for their joint distribution is used here, so the probability is simulated:

1. *What is simulated.* Samples of size N from a multivariate normal population with semipartial correlation `true_r_sp` and full-model R^2 `true_R2_full`: X, $Z_1, \ldots, Z_{p-1}$, and an error E are independent standard normal, and $Y = \rho_{sp}X + P_r\, Z_1 + \sqrt{1-\rho_{sp}^2-P_r^2}\, E$. Under normality the distribution of the two statistics depends on the population only through ρ_{sp}, the full-model R^2, p, and N (see `ss_aipe_semipartial_r_sensitivity`), so this population stands for all of them. The sums of squares and products of the $p+1$ independent variables are Wishart with $N-1$ degrees of freedom, and the function draws the few entries of that matrix's triangular factor that the two statistics depend on (the Bartlett decomposition) in place of the data.

2. *The interval.* From each draw, the sample semipartial correlation and sample full-model R^2 give the half-width $z_{1-\alpha/2}\sqrt{\hat{V}/(N-p)}$, the interval that is planned and that ss_aipe_semipartial_r_sensitivity forms from data.
3. *The search.* The simulated probability at N is the share of G half-widths no larger than the target half-width. Starting from the sample size planned without assurance, the step upward doubles until that share reaches assurance, and a bisection finds the first sample size at which it does, with fresh draws at every sample size tried. The answer is never below the plan without assurance.
4. *The attained probability.* The share that chose N is biased upward by the choice, so assurance_attained is estimated again at the returned N from G draws that had no part in choosing it. Its standard error, $\sqrt{A(1-A)/G}$ with A that proportion, is the attribute assurance_mc_se.

Plans made this way were checked from data with ss_aipe_semipartial_r_sensitivity in 11 conditions (5,000 simulated data sets each; a Monte Carlo standard error of .003 to .006): the proportion of intervals no wider than the target was .798 to .819 for a requested .80, .854 to .858 for .85, .906 to .908 for .90, and .953 for .95.

The step has properties a closed-form plan does not.

- The answer has Monte Carlo error. Without seed, two calls can return sample sizes that differ by a few cases (eight unseeded plans at the default G for one condition gave 252 to 254, and 40 seeded plans of each of two other conditions gave 470 to 473 and 306 to 310). The probability at the smallest of those sizes was below the request (.897 and .895 for a requested .90), so a plan is the size at which the probability is about, and not at least, the request. Give seed for a plan that can be reproduced, and report it with G.
- assurance_attained can fall below assurance by a few of its standard errors, since it is estimated from draws that did not choose N. Its standard error is .003 at the default G and an assurance of .80; in 200 seeded plans the lowest value was .009 below a request of .90. It is an unbiased estimate of the probability at the sample size returned. A larger G narrows both this and the spread of the sample sizes.
- Without true_R2_full the simulated population is the one the plan without assurance is made for: variables partialed out that explain none of the outcome ($P_r^2 = 0$) when true_r_sp squared is at most 1/2, and a full-model R^2 of 1 ($P_r^2 = 1 - \rho_{sp}^2$, no error term) when it is above 1/2. The second is the limit that gives the largest first-order variance; ss_aipe_semipartial_r_sensitivity refuses a population R^2 of exactly 1, so check such a plan there with a value just below 1 or give true_R2_full.
- which_width = "Lower" or "Upper" treats width as a half-width, in the assurance step as in the plan.
- G and seed are checked and used only when assurance is given; without it nothing is drawn and the three rows are absent.

Value

A data.frame with the columns term and value and one row per term:

necessary_N the planned total sample size.

expected_width the full width of the interval at necessary_N when the two sample values equal the planning values.

`true_r_sp`, `p`, `width`, `conf_level` the inputs, with `width` the `width` as given.

`true_R2_full` the planning value of the full-model R^2; `NA` when none was supplied.

`assurance` the requested probability; present only when `assurance` was given, as are the next two rows.

`assurance_attained` the proportion of `G` fresh simulated samples of size `necessary_N` whose interval was no wider than the target. Its Monte Carlo standard error is the attribute `assurance_mc_se`.

`G` the number of simulated samples.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Alf, E. F., Jr., & Graf, R. G. (1999). Asymptotic confidence limits for the difference between two squared multiple correlations: A simplified approach. *Psychological Methods, 4*(1), 70–75. doi:10.1037/1082989X.4.1.70

Algina, J., & Olejnik, S. (2003). Sample size tables for correlation analysis with applications in partial correlation and multiple regression analysis. *Multivariate Behavioral Research, 38*(3), 309–323. doi:10.1207/s15327906mbr3803_02

Aloe, A. M., & Becker, B. J. (2012). An effect size for regression predictors in meta-analysis. *Journal of Educational and Behavioral Statistics, 37*(2), 278–297. doi:10.3102/1076998610396901

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Lawrence Erlbaum.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on the one-way ANOVA and Chapter 4 on contrasts.)

Olkin, I., & Finn, J. D. (1995). Correlations redux. *Psychological Bulletin, 118*(1), 155–164. doi:10.1037/00332909.118.1.155

See Also

`var_semipartial_r`, `ss_aipe_partial_r`, `ss_aipe_R2`, `ss_power_R2_change`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivit`
`ss_aipe_mean()`, `ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared`
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r_sensitivity()`, `ss_aipe_two_phase()`, `ss_aipe_two_phase_cost()`,
`ss_aipe_two_phase_sensitivity()`

Examples

```
# 1. Plan N so the 95% CI on r_sp (three variables partialed out, so
#        p = 4 predictors) has full width <= 0.15 when the planning
#        value of the semipartial is 0.25, whatever the full-model R^2.
ss_aipe_semipartial_r(true_r_sp = 0.25, p = 4, width = 0.15)

# 2. With a planning value of 0.42 for the full-model R^2, fewer are
#        needed.
ss_aipe_semipartial_r(true_r_sp = 0.25, p = 4, width = 0.15,
                      true_R2_full = 0.42)

# 3. With an assurance of .80, found by simulation. G = 2000 keeps the
#        example quick; a reported plan deserves the default G = 20000.
#        The last three rows are the request, the proportion of fresh
#        simulated samples that met the width at the returned N, and G.
plan <- ss_aipe_semipartial_r(true_r_sp = 0.25, p = 4, width = 0.15,
                              assurance = 0.80, G = 2000, seed = 113)
plan
attr(plan, "assurance_mc_se")
```

---

ss_aipe_semipartial_r_sensitivity

Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for a Semipartial Correlation

---

Description

Quantifies how much misspecification of the population semipartial correlation, or of the full-model squared multiple correlation, distorts an AIPE-based sample size plan. The function constructs a $(p + 1) \times (p + 1)$ population covariance matrix for *Y*, *X*, and $Z_1, \ldots, Z_{p-1}$ whose semipartial correlation between *Y* and *X* (partialing $Z_1, \ldots, Z_{p-1}$ out of *X* only, not out of *Y*) equals `true_r_sp` and whose full-model R^2 equals `true_R2_full`. On each replication it draws an *N*-row sample, computes the sample semipartial correlation and the sample full-model R^2, and forms the Wald interval that `ss_aipe_semipartial_r` plans for: the estimate plus and minus the normal critical value times the square root of the first-order variance (`var_semipartial_r` with `under_null = FALSE`) at the two sample values.

Usage

```
ss_aipe_semipartial_r_sensitivity(
  true_r_sp = NULL,
  supposed_r_sp = NULL,
  p,
  width,
  specified_N = NULL,
  conf_level = 0.95,
  assurance = NULL,
```

```
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  true_R2_full = NULL,
  supposed_R2_full = NULL,
  seed = NULL
)
```

Arguments

true_r_sp	Population semipartial correlation ρ_{sp}; must lie in $(-1, 1)$.
supposed_r_sp	Planning value passed to ss_aipe_semipartial_r; supply this or specified_N but not both.
p	Number of predictors in the regression of Y on X and the variables partialed out, so p - 1 variables are partialed out and p is at least 2. A categorical variable with a levels counts as a - 1 predictors. It is the p of ss_power_R2, ci_R2, and ci_reg_coef: the correlation concerns one predictor of a regression with p predictors. In the notation of a model comparison, q is the number of predictors removed from the full model to give the restricted model ($q = p_{full} - p_{restricted}$, the predictors being tested), and the number of variables partialed out is $p - q$; this correlation concerns one predictor, so $q = 1$ and $p-1$ variables are partialed out. The simulated data have $p + 1$ columns.
width	Desired full width of the CI on the semipartial correlation.
specified_N	Sample size to evaluate; a whole number of at least $p + 2$.
conf_level	Confidence level (default 0.95).
assurance	Optional assurance probability passed to ss_aipe_semipartial_r; used only with supposed_r_sp.
G	Number of Monte Carlo replications (default 1000).
print_iter	Logical. Print the iteration index of each replication.
filename	Optional path for a comma separated file recording every replication (the sample semipartial correlation, the two confidence limits, the interval width, and two indicators of whether the interval missed true_r_sp below or above): nothing is written when filename is NULL (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at tempfile(fileext = ".csv").
true_R2_full	Population squared multiple correlation of the full model (Y on X and $Z_1, \ldots, Z_{p-1}$); at least true_r_sp squared and below 1. When NULL (the default) it is true_r_sp squared, a population in which the variables partialed out explain none of the outcome.
supposed_R2_full	Optional planning value of the full-model R^2, passed to ss_aipe_semipartial_r as true_R2_full; when NULL that function plans for the largest variance over every possible value. It cannot be given with specified_N.
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the run reproducible; a supplied seed is set for this call only,

and the caller's random number state is restored when the function returns. With `assurance`, the seed also makes the simulated plan of `ss_aipe_semipartial_r` reproducible.

Details

Under multivariate normality the sampling distribution of the sample semipartial correlation depends on the population only through the semipartial correlation, the full-model R^2, p, and the sample size, because the statistic is unchanged by any nonsingular linear transformation of $Z_1, \ldots, Z_{p-1}$, by adding any linear function of them to X, and by positive rescaling of X and Y. The population simulated here (uncorrelated standardized X and $Z_1, \ldots, Z_{p-1}$, with Y related to X and to Z_1) therefore stands for every normal population with the same two values.

The confidence limits are not truncated at -1 and 1, so that the widths are those the planner plans; with a large $|\rho_{sp}|$ and a small sample a limit can pass 1 in absolute value.

The function simulates $p - 1$ variables partialed out, the count `ss_aipe_semipartial_r` plans for.

With `assurance`, the sample size comes from the simulated assurance step of `ss_aipe_semipartial_r` (at its default number of draws), so two unseeded calls can simulate sample sizes that differ by a case or two.

A value of `true_R2_full` or `supposed_R2_full` that falls below the squared semipartial correlation by no more than rounding to two decimals allows is taken to mean that the variables partialed out explain none of the outcome, with a warning of class `dmar_semipartial_r_rounded`; a smaller value is refused.

Value

A `data.frame` with rows for the realized semipartial correlation, the interval width, the proportion of intervals at or below `width`, tail-specific and overall non-coverage of `true_r_sp`, and the input echoes, including `true_R2_full` (the value simulated), `supposed_R2_full` (`NA` when none was supplied), and `assurance` (present only when an assurance was supplied).

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

`ss_aipe_semipartial_r`, `ss_aipe_partial_r_sensitivity`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,
`ss_aipe_equivalence_r_sensitivity()`, `ss_aipe_equivalence_smd()`, `ss_aipe_equivalence_smd_sensitivity()`,
`ss_aipe_icc()`, `ss_aipe_icc_sensitivity()`, `ss_aipe_indirect_effect()`, `ss_aipe_indirect_effect_sensitivi`
`ss_aipe_mean()`, `ss_aipe_mean_sensitivity()`, `ss_aipe_mixed_effects_sensitivity()`, `ss_aipe_omega_squared`
`ss_aipe_omega_squared_sensitivity()`, `ss_aipe_partial_r()`, `ss_aipe_partial_r_sensitivity()`,
`ss_aipe_pcm_sensitivity()`, `ss_aipe_r()`, `ss_aipe_r_sensitivity()`, `ss_aipe_reliability_sensitivity()`,
`ss_aipe_semipartial_r()`, `ss_aipe_two_phase()`, `ss_aipe_two_phase_cost()`, `ss_aipe_two_phase_sensitivity()`

Examples

```
# G = 50 keeps the example quick; a reported check deserves the
# default G = 1000 or more.
ss_aipe_semipartial_r_sensitivity(
  true_r_sp = 0.30, supposed_r_sp = 0.30, p = 4, width = 0.20,
  G = 50, print_iter = FALSE,
  true_R2_full = 0.50, supposed_R2_full = 0.50, seed = 113
)
```

ss_aipe_sem_path *Sample Size Planning for SEM Targeted Effects*

Description

Plan sample size for structural equation models so that the confidence interval for the targeted model parameter is sufficiently narrow

Usage

```
ss_aipe_sem_path(
  model,
  true_cov_matrix,
  width,
  which_path,
  conf_level = 0.95,
  assurance = NULL,
  detail = FALSE,
  internal = FALSE,
  ...
)
```

Arguments

model	A single character string giving the free analysis model in lavaan model syntax (see model.syntax). The target path must carry a parameter label so it can be referred to by name, for example "f2 ~ b*f1" labels the structural path b. This is the model that would be fit to the data; its parameters are free, not fixed to population values
true_cov_matrix	Population covariance matrix of the observed variables planned for, with row and column names matching the observed variables in model. It is typically obtained from a fully fixed population model via cov_sem
width	Desired confidence interval width for the model parameter of interest
which_path	The parameter label of the targeted path, given as a character string, for example "b" for the path labeled f2 ~ b*f1 in model
conf_level	Confidence level (i.e., 1 - Type I error rate)
assurance	The assurance that the confidence interval obtained in a particular study will be no wider than desired (must be NULL or a value between 0.50 and 1)
detail	if TRUE, additionally print the model parameter names and the observed variable names (the returned table is unchanged)
internal	option to output a list for internal use (for ss_aipe_sem_path_sensitivity)
...	Further arguments passed to sem for the fit to true_cov_matrix, from which the plan, with or without assurance, is read (for example std.lv = TRUE). Every

name given here must be an argument or an option of **lavaan** (see lavOptions); any other name, and any argument without a name, is refused before a model is fitted, with a condition of class dmar_argument_invalid that gives the name as it was typed

Details

This function implements the sample size planning methods proposed in Lai and Kelley (2011). It requires **lavaan** to be installed and uses sem to obtain the expected information, that is the asymptotic covariance matrix of the parameter estimates, by fitting the free analysis model to the population covariance matrix true_cov_matrix at a very large sample size. The analysis model is written in lavaan model syntax with the targeted path given a parameter label; see model.syntax for the syntax and sem for the fitting machinery. The population covariance matrix true_cov_matrix is most naturally produced by cov_sem from a fully fixed population model. An analysis model whose likelihood has no maximum at true_cov_matrix, an error variance running off below zero while the likelihood is flat or still rising (see fa_bifactor), stops the function with an error of class dmar_no_maximum that prints the **lavaan** code of the fit, since the asymptotic covariance matrix would describe where the optimizer stopped rather than the model.

When assurance is supplied, the assurance adjustment is based on a chi square approximation to the sampling variability of the confidence interval width and can undershoot the nominal assurance in finite samples; use ss_aipe_sem_path_sensitivity to check the realized width and coverage at the planned sample size.

Value

A data.frame (a dmar_tbl) with term and value columns whose rows are necessary_N (the planned sample size), path_index (the position of the target path among the model parameters), and var_theta_j (the population sampling variance of the target path at the planned sample size). The returned table is the same whether or not detail = TRUE. When internal = TRUE a list is returned for use by ss_aipe_sem_path_sensitivity.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2011). Accuracy in parameter estimation for targeted effects in structural equation modeling: Sample size planning for narrow confidence intervals. *Psychological Methods, 16*(2), 127–148. doi:10.1037/a0021764

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

See Also

sem, model.syntax, cov_sem, ss_aipe_sem_path_sensitivity

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Population covariance from a fully fixed model (see cov_sem()).
pop_model <- "
  f1 =~ 1*y1 + 0.8*y2 + 0.8*y3
  f2 =~ 1*y4 + 0.8*y5 + 0.8*y6
  f2 ~ 0.5*f1
  f1 ~~ 1*f1
  f2 ~~ 0.75*f2
  y1 ~~ 0.5*y1; y2 ~~ 0.5*y2; y3 ~~ 0.5*y3
  y4 ~~ 0.5*y4; y5 ~~ 0.5*y5; y6 ~~ 0.5*y6
"
Sigma <- cov_sem(pop_model)$sigma_theta

# Free analysis model with the target structural path labeled "b".
analysis_model <- "
  f1 =~ y1 + y2 + y3
  f2 =~ y4 + y5 + y6
  f2 ~ b*f1
"
ss_aipe_sem_path(model = analysis_model, true_cov_matrix = Sigma,
                 width = 0.30, which_path = "b")

# That sample size holds the interval to the desired width on average, so
# about half of the studies it plans return a wider one. Adding assurance
# plans for the width to be met in 90 percent of studies instead, at the
# cost of a larger sample size.
ss_aipe_sem_path(model = analysis_model, true_cov_matrix = Sigma,
                 width = 0.30, which_path = "b",
                 assurance = 0.90)
```

---

ss_aipe_sem_path_sensitivity

A Priori Monte Carlo Simulation for Sample Size Planning for SEM Targeted Effects

---

Description

Conduct a priori Monte Carlo simulation to empirically study the effects of (mis)specifications of input information on the calculated sample size. Random data are generated from the true covariance matrix but fit to the proposed model, whereas sample size is calculated based on the input covariance matrix and proposed model.

Usage

```
ss_aipe_sem_path_sensitivity(
  model,
  supposed_cov_matrix,
  true_cov_matrix = supposed_cov_matrix,
  which_path,
  width,
  specified_N = NULL,
  conf_level = 0.95,
  assurance = NULL,
  G = 100,
  filename = NULL,
  ...,
  seed = NULL
)
```

Arguments

model	A single character string giving the free analysis model in lavaan model syntax (see model.syntax), the model that would be fit to the data. The target path must carry a parameter label so it can be referred to by name, for example "f2 ~ b*f1" labels the structural path b. The model may or may not be the true data generating model
supposed_cov_matrix	the population covariance matrix supposed when the sample size is planned (passed to ss_aipe_sem_path as its true_cov_matrix). It is a planning assumption, not a matrix estimated from data, and it may or may not be the true covariance matrix. The row names and column names of supposed_cov_matrix should be the same as the observed variables in model
true_cov_matrix	the true population covariance matrix, which will be used to generate random data for the simulation study. The row names and column names of true_cov_matrix should be the same as the observed variables in model. Defaults to supposed_cov_matrix
which_path	the parameter label of the targeted path, given as a character string, for example "b" for the path labeled f2 ~ b*f1 in model
width	desired confidence interval width for the model parameter of interest
specified_N	the total sample size N of each set of random data. If it is NULL, it is the sample size ss_aipe_sem_path plans from supposed_cov_matrix
conf_level	confidence level (i.e., 1- Type I error rate)
assurance	the assurance that the confidence interval obtained in a particular study will be no wider than desired (must be NULL or a value between 0.50 and 1)
G	number of replications in the Monte Carlo simulation
filename	an optional path for a comma separated file recording every replication (the estimate of the targeted path, its standard error, the two confidence limits, and the interval width): nothing is written when filename is NULL (the default), a new file with a header row is created otherwise, an existing file at that path

	is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`
`...`	further arguments passed to `sem` for every fit: the planning fit to `supposed_cov_matrix` (through `ss_aipe_sem_path`), the fit to `true_cov_matrix`, and the fit to each simulated sample (for example `std.lv = TRUE`). Every name given here must be an argument or an option of **lavaan** (see `lavOptions`); any other name, and any argument without a name, is refused before a model is fitted, with a condition of class `dmar_argument_invalid` that gives the name as it was typed
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

This function implements the sample size planning methods proposed in Lai and Kelley (2011). It calls `ss_aipe_sem_path` to plan the sample size and to identify the targeted path, then fits the analysis model to data simulated from `true_cov_matrix` with `sem`. The analysis model is written in lavaan model syntax with the targeted path given a parameter label; see `model.syntax` for the syntax and `sem` for the fitting machinery. The population covariance matrices are most naturally produced by `cov_sem` from a fully fixed population model. This function requires **lavaan** and **MASS** to be installed.

Value

A `data.frame` with columns `term` and `value` summarizing the a priori Monte Carlo study. The `term` entries are: `"mean_path"`, `"median_path"`, `"sd_path"` (summaries of the realized estimates of the targeted path across the converged replications); `"mean_ci_width"`, `"median_ci_width"`, `"sd_ci_width"` (summaries of the realized interval widths); `"pct_ci_less_width"` (proportion of realized widths at or below `width`); `"pct_ci_miss_low"` and `"pct_ci_miss_high"` (tail-specific empirical non-coverage of the population path); `"total_type_I_error"` (overall empirical non-coverage, the sum of the two tails); and the echoes `"total_N"` (the N evaluated), `"true_path"` (the population value of the targeted path under `true_cov_matrix`), `"width"`, `"conf_level"`, and `"assurance"` (present only when an assurance was supplied). The proportion rows are on the 0 to 1 scale, not percentages. The number of replications that gave an interval is `attr(x, "simulation")$returned`, and the number of data sets drawn again is `attr(x, "simulation")$n_redrawn` (see the sections below).

Replications drawn again

A simulated data set whose model fit does not converge, or whose estimate is not a maximum of the likelihood, is replaced by a new one, so such a data set is not among the replications without an interval and is in no summary row. At most `20 * G` data sets are drawn in all; if that many do not give `G` converged fits, the function warns and summarizes the fits it has. The number of data sets drawn and set aside is the element `n_redrawn` of the `simulation` attribute, an integer read with `attr(x, "simulation")$n_redrawn`: 0 when every fit converged, and the total number of data sets drawn is `n_redrawn` plus the number of converged fits. The other elements of the record keep their meaning: `attempted` is `G`, `returned` the replications with an interval, and `failed` those without one (a converged fit whose variance for the path was negative or missing, or a replication

never obtained). A large `n_redrawn` beside `G` says the model is hard to fit at this sample size, and the summaries then describe only the samples in which it could be fitted.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Note

Occasionally a replication fails to converge when the analysis model is fit to a simulated data set. Such replications are not counted toward the `G` converged replications; the simulation draws fresh data and continues. A replication whose likelihood has no maximum, an error variance running off below zero while the likelihood is flat or still rising (see `fa_bifactor`), is treated the same way, and an analysis model with no maximum at `supposed_cov_matrix` or `true_cov_matrix` stops the function, before any replication is drawn, with an error of class `dmar_no_maximum` that prints the **lavaan** code of that fit. A safety cap stops the loop after `20 * G` attempts, and a single warning is issued if fewer than `G` replications converged.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2011). Accuracy in parameter estimation for targeted effects in structural equation modeling: Sample size planning for narrow confidence intervals. *Psychological Methods, 16*(2), 127–148. doi:10.1037/a0021764

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

See Also

sem, cov_sem, ss_aipe_sem_path

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# The planning values a researcher would bring to ss_aipe_sem_path(): two
# factors, each measured by three indicators, joined by a structural path
# of 0.5, with every residual variance at 0.5.
planning_model <- "
  f1 =~ 1*y1 + 0.8*y2 + 0.8*y3
  f2 =~ 1*y4 + 0.8*y5 + 0.8*y6
  f2 ~ 0.5*f1
  f1 ~~ 1*f1
  f2 ~~ 0.75*f2
  y1 ~~ 0.5*y1; y2 ~~ 0.5*y2; y3 ~~ 0.5*y3
  y4 ~~ 0.5*y4; y5 ~~ 0.5*y5; y6 ~~ 0.5*y6
"

# The population the study will actually sample from: the same structural
# path, but noisier indicators than the planning values assumed, with
# every residual variance at 0.8.
true_model <- "
  f1 =~ 1*y1 + 0.8*y2 + 0.8*y3
  f2 =~ 1*y4 + 0.8*y5 + 0.8*y6
  f2 ~ 0.5*f1
  f1 ~~ 1*f1
  f2 ~~ 0.75*f2
  y1 ~~ 0.8*y1; y2 ~~ 0.8*y2; y3 ~~ 0.8*y3
  y4 ~~ 0.8*y4; y5 ~~ 0.8*y5; y6 ~~ 0.8*y6
"

# The free analysis model, with the targeted path labeled "b".
analysis_model <- "
  f1 =~ y1 + y2 + y3
  f2 =~ y4 + y5 + y6
  f2 ~ b*f1
"

supposed_cov_matrix <- cov_sem(planning_model)$sigma_theta
true_cov_matrix <- cov_sem(true_model)$sigma_theta

# The sample size planned from the optimistic measurement quality is
# evaluated against the population that actually holds. Notice that the
# mean realized width (mean_ci_width) exceeds the desired 0.30, and that
```

```
# pct_ci_less_width, the proportion of intervals no wider than desired, falls
# well below the one half that the planned sample size would deliver if
# the planning values were right. G = 10 keeps the example quick; a
# reported sensitivity study deserves the default G = 100 or more.
set.seed(113)
ss_aipe_sem_path_sensitivity(model = analysis_model,
                             supposed_cov_matrix = supposed_cov_matrix,
                             true_cov_matrix = true_cov_matrix,
                             which_path = "b", width = 0.30, G = 10)
```

`ss_aipe_sm` *Sample Size Planning for Accuracy in Parameter Estimation (AIPE) of the Standardized Mean*

Description

Plans the sample size needed for a sufficiently narrow confidence interval for the population standardized mean, the mean divided by the standard deviation, from the accuracy in parameter estimation (AIPE) perspective.

Usage

```
ss_aipe_sm(true_sm, width, conf_level = 0.95, assurance = NULL)
```

Arguments

`true_sm`	The planning value of the population standardized mean (the population mean divided by the population standard deviation)
`width`	The desired full width of the obtained confidence interval
`conf_level`	The desired confidence interval coverage, (i.e., 1 - Type I error rate). A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in `ci_nc_t`)
`assurance`	Optional. The probability that the confidence interval of the planned study is no wider than `width`: `NULL` (the default), which plans for the expected width, or a single number strictly between 0 and 1. Any such value may be used. Values such as `.80`, `.90`, and `.99` are the usual ones, and a value at or below `.50` is accepted as well (see *Details*). Anything else is refused with a condition of class `dmar_assurance_invalid`. An assurance above .999999998 is refused with a condition of class `dmar_argument_invalid`: the assurance step finds noncentral t limits with tail areas of `(1 - assurance) / 2`, which must be at least 1e-9.

Details

The plan for the expected width. Without `assurance` the function finds the smallest sample size at which the noncentral *t* confidence interval for the standardized mean (`ci_sm`), computed at `true_sm`, is no wider than `width`. The interval of the study itself is computed at the sample standardized mean, which varies from sample to sample, and the interval is wider the larger that value is in magnitude. A plan for the expected width therefore leaves a probability near one half that the interval obtained is wider than `width`.

The assurance step. With `assurance` the function follows the one-sample analog of the procedure Kelley and Rausch (2006, Equations 18 to 20 and 25 to 31) give for the standardized mean difference. At the sample size of the plan for the expected width it finds the value that the sample standardized mean exceeds in magnitude with probability `1 - assurance`, by a search between the one-sided and the two-sided confidence limits their Equations 25 and 28 name as its bounds, and then plans for the expected width with that value in place of `true_sm`. That value is defined for every probability strictly between 0 and 1, so every such `assurance` is accepted. A value above about one half gives a larger sample size than the plan for the expected width, and a value below it gives a smaller one, which is the sample size at which the interval is no wider than `width` with only that probability: for `true_sm = 1`, `width = 0.5`, and 95 percent confidence the function returns 93 without `assurance`, 93 at `.50`, 87 at `.20`, 100 at `.80`, and 105 at `.90`. Because the value planned for is found at the first sample size and the returned sample size differs from it, the probability at the returned sample size is near `assurance` and not exactly it; in that example it is .519, .228, .812, and .925 at the four values of `assurance`, computed from the noncentral *t* distribution.

Value

A 1-row `data.frame` with columns `term` and `value`:

`necessary_N` The necessary total sample size in order to achieve the desired degree of accuracy (i.e., the sufficiently narrow confidence interval)

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K. (2005). The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals, *Educational and Psychological Measurement, 65*, 51–69. doi:10.1177/0013164404264850

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

ci_nc_t, ci_sm

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Suppose the population mean is believed to be 20, and the population
# standard deviation is believed to be 2; thus the population standardized
# mean is believed to be 10. To determine the necessary sample size for a
# study so that the full width of the 95 percent confidence interval
# obtained in the study will be, with 90% assurance, no wider than 2.5,
# the function should be specified as follows.

ss_aipe_sm(true_sm = 10, width = 2.5, conf_level = .95, assurance = .90)

# The plan for the expected width, and the sample size at which the
# interval is no wider than 0.5 with probability .50, for a standardized
# mean of 1.
ss_aipe_sm(true_sm = 1, width = 0.5)
ss_aipe_sm(true_sm = 1, width = 0.5, assurance = .50)
```

---

ss_aipe_smd *Sample Size Planning for the Standardized Mean Difference (AIPE)*

Description

Determines the per-group sample size needed for a two-independent-groups design so that the confidence interval for the population standardized mean difference, denoted δ, is no wider than a user-specified value when the sample standardized mean difference (Cohen's d) equals δ. This is the accuracy in parameter estimation (AIPE) approach of Kelley and Rausch (2006), the standardized mean difference companion to power-based planning via ss_power_smd. Optionally, supplying assurance returns the smallest sample size, not below the plan without it, at which the realized interval is no wider than the target with at least that probability, computed exactly.

Usage

```
ss_aipe_smd(
  true_smd,
  conf_level = 0.95,
  width,
  assurance = NULL,
  assurance_method = "analytic"
)
```

Arguments

`true_smd`	The supposed value of the population standardized mean difference δ the sample size is planned against: a value the researcher posits, either a minimally important effect or a value believed to be true in the population, never a sample estimate. Echoed in the returned table as the `true_smd` row.
`conf_level`	Desired confidence level (i.e., $1 - \alpha$, where α is the Type I error rate). Default `0.95`. A level above .999999998, whose tail areas are below 1e-9, is refused (see "Small tail areas" in `ci_nc_t`).
`width`	Desired (full) width of the two-sided confidence interval on δ.
`assurance`	Optional probability with which the realized confidence interval is to be no wider than `width`. When `NULL` (the default), the plan is for the width of the interval at $d = \delta$; when supplied (e.g., 0.80, 0.90, 0.99), the function returns the smallest per-group sample size, not below that plan, at which the probability that the realized interval is no wider than `width` is at least `assurance` (see Details). Must be `NULL` or strictly between 0.50 and 1. An assurance above .999999998 is refused with a condition of class `dmar_argument_invalid`: the floor of 1e-9 on a tail area (see "Small tail areas" in `ci_nc_t`) applies to `(1 - assurance) / 2`.
`assurance_method`	How the probability that the realized interval is no wider than `width` is found, used only with `assurance`. Only `"analytic"` (the default) is offered here: the probability is computed exactly, and the plan is the smallest sample size, not below the plan without `assurance`, at which it reaches `assurance`. `"simulation"`, which `ss_aipe_sc_ancova`, `ss_aipe_reg_coef` with `noncentral = TRUE`, and `ss_aipe_src` offer, where the probability has no exact form, is refused with a condition of class `dmar_argument_invalid`, whether or not `assurance` is given, since a simulated estimate of an exact probability would only add Monte Carlo error.

Details

The plan without assurance. The interval is the one `ci_smd` reports: the confidence interval for the noncentrality parameter of the noncentral *t* distribution, with $2n - 2$ degrees of freedom, at the observed *t* statistic, rescaled to δ by $\sqrt{2/n}$, with *n* the per-group sample size. The function uses δ as if it were *d* and returns the smallest *n* at which that interval is no wider than `width` (Kelley & Rausch, 2006, pp. 371–372). The width depends on δ only through $|\delta|$.

The assurance step. At a given *n* the width of the interval depends on the sample only through *d* and grows as $|d|$ grows, so the realized interval is no wider than the target exactly when $|d| \le d^*(n)$, the value of $|d|$ at which the width equals the target. With $\lambda = |\delta|\sqrt{n/2}$, $\nu = 2n - 2$, and $t^* = d^*(n)\sqrt{n/2}$, the probability of that event under normality with equal variances is

$$P_n = \int_0^\infty \left\{ \Phi\big(t^*\sqrt{v/\nu} - \lambda\big) - \Phi\big(-t^*\sqrt{v/\nu} - \lambda\big) \right\} f_\nu(v)\, dv,$$

where Φ is the standard normal distribution function and f_ν the density of a chi square on ν degrees of freedom: the *t* statistic is $(Z+\lambda)/\sqrt{V/\nu}$ with Z standard normal and V chi square, independent. The integral involves only the normal distribution and the chi square, so it does not depend on R's

noncentral *t*, which is a normal approximation above a noncentrality of 37.62. It depends on δ only through $|\delta|$, and the plan is the same for `true_smd` and `-true_smd`.

The returned sample size is the smallest *n* at or above the plan without assurance, n_0, at which P_n is at least `assurance` (`assurance_method = "analytic"`), found by doubling a step upward from n_0 and then bisecting, and `assurance_attained` is P_n there. The search assumes that P_n does not decrease as *n* grows from n_0. In 45 planning sets ($|\delta|$ of 0, .2, .5, 1, and 2, widths of .25, .5, and 1, and assurances of .80, .90, and .99) it did not decrease at any *n* from n_0 to ten past the plan, and the plan was the smallest *n* found by raising *n* one at a time from n_0. Because the width changes little with *d*, P_n rises steeply with *n*: at `true_smd = .5` and a width of .5 it is .838 at 129 and .916 at 130.

The probability was checked against the noncentral *t* distribution computed by R's `pt()` where it is exact, against a second integral that conditions on the normal variate instead of the chi square, and against the proportion of intervals no wider than the target among samples of two groups of normal data.

Large noncentrality. R's `pt()` is a normal approximation to the noncentral *t* distribution above a noncentrality of 37.62, and its series loses accuracy above 33 with more than 2,000 degrees of freedom, ranges that narrow targets with a large $|\delta|$ reach (`true_smd = 2` with a width of .25, or `true_smd = 1` with a width of .1). The interval `ci_smd` reports is computed exactly there (see `ci_nc_t`), so the width of the interval increases with $|d|$ at every sample size, the event above is $|d| \leq d^*(n)$, and P_n is exact.

The published procedure. Kelley and Rausch (2006, pp. 372–374) plan with `assurance` as follows: at n_0, find the value δ_γ that $|d|$ exceeds with probability $1 - \gamma$, where γ is `assurance`, by searching between the upper limit of a one-sided interval and the upper limit of a two-sided interval computed at $d = \delta$ (their Equations 25 to 31), and plan for δ_γ as for δ. The procedure does not compute P_n. Since δ_γ is found at n_0 and the plan for it is larger, that plan can be larger than the smallest sample size that attains the request: in the 45 sets above it attained the request in every set and was one case larger in 7, all but one at an assurance of .99; at `true_smd = 1`, a width of .5, and an assurance of .99 it is 150, where P_n is .9943, and this function's plan is 149, where it is .9903. It can also fall short: at `true_smd = 0`, a width of .8, and an assurance of .95 it is 49 per group, where P_n is .9499, and this function's plan is 50, where it is .9946.

Value

A `data.frame` with columns `term` and `value`:

`necessary_n_per_group` the necessary per-group sample size.

`true_smd`, `width` the planning inputs, so the assumptions the sample size was planned under travel with the result. The `true_smd` row is the supposed effect the plan is built on, as given (with its sign): a value the researcher posits, either a minimally important effect or a value believed to be true in the population, never a sample estimate.

`assurance` the requested probability; present only when `assurance` was given.

`assurance_attained` the exact probability, at `necessary_n_per_group` and `true_smd`, that the realized interval is no wider than `width`, at least `assurance`; present only when `assurance` was given.

The confidence level is reported in the printed footer, and with `assurance` the footer names the assurance method; the attribute `assurance_method` records it, and `assurance_probability` records that `assurance_attained` is `"exact"`.

Warning

The returned value is the sample size *per group*.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Anderson, S. F., & Kelley, K. (2024). Sample size planning for replication studies: The devil is in the design. *Psychological Methods, 29*(5), 844–867. doi:10.1037/met0000520

Anderson, S. F., Kelley, K., & Maxwell, S. E. (2017). Sample-size planning for more accurate statistical power: A method adjusting sample effect sizes for publication bias and uncertainty. *Psychological Science, 28*(11), 1547–1562. doi:10.1177/0956797617723724

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K. (2005). The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals, *Educational and Psychological Measurement, 65*, 51–69. doi:10.1177/0013164404264850

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`smd`, `smd_c`, `ci_smd`, `ci_smd_c`, `ci_nc_t`, `stats::power.t.test()`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
ss_aipe_smd(true_smd = .5, conf_level = .95, width = .30)

# With assurance the last row is the exact probability, at the planned
# sample size, that the interval is no wider than the target.
ss_aipe_smd(true_smd = .5, conf_level = .95, width = .30, assurance = .8)
ss_aipe_smd(true_smd = .5, conf_level = .95, width = .30, assurance = .95)

# Reversing the groups reverses the sign of the effect and leaves the
# plan unchanged.
ss_aipe_smd(true_smd = -.5, conf_level = .95, width = .30, assurance = .95)
```

`ss_aipe_smd_sensitivity` *Sensitivity Analysis for Sample Size Given the Accuracy in Parameter Estimation Approach for the Standardized Mean Difference*

Description

Performs sensitivity analysis for sample size determination for the standardized mean difference given a population and a standardized mean difference. Allows one to determine the effect of being wrong when estimating the population standardized mean difference in terms of the width of the obtained (two-sided) confidence intervals.

Usage

```
ss_aipe_smd_sensitivity(
  true_smd = NULL,
  supposed_smd = NULL,
  width = NULL,
  specified_n_per_group = NULL,
  assurance = NULL,
  assurance_method = "analytic",
  conf_level = 0.95,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_smd`	population standardized mean difference
`supposed_smd`	the supposed standardized mean difference: the planning value from which `ss_aipe_smd` plans the per-group sample size (not an estimate computed from data); it can equal `true_smd` to simulate a correctly specified plan

`width` describe full width for the confidence interval around the population standardized mean difference

`specified_n_per_group` selected sample size to use in order to determine distributional properties of at a given value of sample size

`assurance` parameter to ensure confidence interval width with a specified degree of certainty (must be `NULL` or between zero and unity). When the sample size is planned from `supposed_smd`, an assurance above .999999998 is refused, as in `ss_aipe_smd`.

`assurance_method` passed to `ss_aipe_smd` with `assurance` when the sample size is planned from `supposed_smd`. Only `"analytic"` (the default) is offered: the plan is the smallest per-group sample size, at or above the plan without `assurance`, at which the exact probability that the interval is no wider than `width` is at least `assurance`. `"simulation"` is refused with a condition of class `dmar_argument_invalid`, as it is by `ss_aipe_smd`, since that probability is exact, whether or not `assurance` or `specified_n_per_group` is given.

`conf_level` the desired degree of confidence (i.e., 1-Type I error rate)

`G` number of generations (i.e., replications) of the simulation

`print_iter` to print the current value of the iterations

`filename` an optional path for a comma separated file recording every replication (the realized standardized mean difference, the full and the two one-sided interval widths, the three non-coverage indicators, and the two confidence limits): nothing is written when `filename` is `NULL` (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at `tempfile(fileext = ".csv")`

`seed` `NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

For sensitivity analysis when planning sample size given the desire to obtain narrow confidence intervals for the population standardized mean difference. Given a population value and a supposed (planning) value, one can determine the effects of incorrectly specifying the population standardized mean difference (`true_smd`) on the obtained widths of the confidence intervals. Also, one can evaluate the percent of the confidence intervals that are less than the desired width (especially when modifying the `assurance` parameter); see `ss_aipe_smd`) Alternatively, one can specify `specified_n_per_group` to determine the results at a particular sample size (when doing this `supposed_smd` cannot be specified).

Value

A `data.frame` with columns `term` and `value` summarizing the Monte Carlo sensitivity analysis across the `G` replications. The `term` entries are: `mean_smd`, `median_smd`, `sd_smd` (summaries of the realized standardized mean difference); `mean_ci_width`, `median_ci_width`, `sd_ci_width` (summaries of the full interval widths); `mean_ci_width_lower` and `mean_ci_width_upper` (mean one-sided widths, measured from the observed standardized mean difference to each limit); `pct_ci_less_width`

(proportion of intervals at or below the target width); `pct_ci_miss_low` and `pct_ci_miss_high` (tail-specific empirical non-coverage of `true_smd`); `total_type_I_error` (overall empirical non-coverage, the sum of the two tails); and the input echoes `n_per_group` (the size simulated at, planned or specified), `total_N`, `true_smd`, `supposed_smd` (NA when `specified_n_per_group` was supplied instead), `width`, `conf_level`, and `assurance` (present only when an assurance was supplied). The proportion and Type I error rows are proportions on the 0 to 1 scale, not percentages.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K. (2005). The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals, *Educational and Psychological Measurement, 65*, 51–69. doi:10.1177/0013164404264850

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`ss_aipe_smd`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Since 'true_smd' equals 'supposed_smd', this usage
# returns the results of a correctly specified situation.
# Note that 'G' should be large (50 is used to make the example run easily)
set.seed(113)
Res.1 <- ss_aipe_smd_sensitivity(true_smd=.5, supposed_smd=.5, width=.30,
                                 assurance=NULL, conf_level=.95, G=50, print_iter=FALSE)

# Objects contained in the 'summary'.
Res.1$term

# True standardized mean difference is .4, but specified at .5.
# Change 'G' to some large number (e.g., G=5,000)
Res.2 <- ss_aipe_smd_sensitivity(true_smd=.4, supposed_smd=.5, width=.30,
                                 assurance=NULL, conf_level=.95, G=50, print_iter=FALSE)

# The effect of the misspecification on mean confidence intervals is:
Res.2[1,]

# True standardized mean difference is .5, but specified at .4.
Res.3 <- ss_aipe_smd_sensitivity(true_smd=.5, supposed_smd=.4, width=.30,
                                 assurance=NULL, conf_level=.95, G=50, print_iter=FALSE)

# The effect of the misspecification on mean confidence intervals is:
Res.3[1,]
```

`ss_aipe_sm_sensitivity` *Sensitivity Analysis for Sample Size Planning for the Standardized Mean From the Accuracy in Parameter Estimation (AIPE) Perspective*

Description

Performs a sensitivity analysis when planning sample size from the Accuracy in Parameter Estimation (AIPE) Perspective for the standardized mean.

Usage

```
ss_aipe_sm_sensitivity(
  true_sm = NULL,
  supposed_sm = NULL,
  width = NULL,
  specified_N = NULL,
  assurance = NULL,
  conf_level = 0.95,
  G = 10000,
  print_iter = TRUE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_sm`	population standardized mean
`supposed_sm`	supposed standardized mean, the planning value from which `ss_aipe_sm` plans the sample size (not an estimate computed from data)
`width`	desired full width of the confidence interval for the population standardized mean
`specified_N`	selected sample size to use in order to determine distributional properties of a given value of sample size
`assurance`	parameter to ensure that the obtained confidence interval width is narrower than the desired width with a specified degree of certainty (must be `NULL` or between zero and unity)
`conf_level`	the desired confidence interval coverage, (i.e., 1 - Type I error rate)
`G`	number of generations (i.e., replications) of the simulation
`print_iter`	to print the current value of the iterations
`filename`	Optional path to a CSV file; when supplied, the per-replication results (the observed standardized mean, the full and one-sided widths, the tail misses, and the interval limits) are written there, appended when the file already exists, and a throwaway run should point it at `tempfile(fileext = ".csv")`; the default `NULL` writes nothing.
`seed`	`NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Value

A data.frame with columns term and value summarizing the Monte Carlo sensitivity analysis across the G replications. The term entries are: mean_sm, median_sm, sd_sm (summaries of the realized standardized mean); mean_ci_width, median_ci_width, sd_ci_width (summaries of the full interval widths); mean_ci_width_lower and mean_ci_width_upper (mean one-sided widths, measured from the observed standardized mean to each limit); pct_ci_less_width (proportion of intervals at or below the target width); pct_ci_miss_low and pct_ci_miss_high (tail-specific empirical non-coverage of true_sm); total_type_I_error (overall empirical non-coverage, the sum of the two tails); and the input echoes total_N, true_sm, supposed_sm (NA when specified_N was supplied instead), width, conf_level, and assurance (present only when an assurance was supplied). The proportion and Type I error rows are proportions on the 0 to 1 scale, not percentages.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

width_target_mc_ci the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as G grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives G, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so failed is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement, 61*(4), 532–574. doi:10.1177/0013164401614002

Hedges, L. V. (1981). Distribution theory for Glass's Estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Kelley, K. (2005). The effects of nonnormal distributions on confidence intervals around the standardized mean difference: Bootstrap and parametric confidence intervals, *Educational and Psychological Measurement, 65*, 51–69. doi:10.1177/0013164404264850

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

See Also

`ss_aipe_sm`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# With true_sm equal to supposed_sm the planning value is correct, and
# the sweep reports what a correctly specified plan delivers: the share of
# intervals no wider than the target should sit near the assurance. G = 10
# keeps the example quick; a reported sensitivity study deserves the
# default G = 10000.
set.seed(113)
res_correct <- ss_aipe_sm_sensitivity(true_sm = 1.5, supposed_sm = 1.5,
  width = 0.5, assurance = 0.95, conf_level = 0.95, G = 10,
  print_iter = FALSE)
res_correct

# The terms the summary reports.
res_correct$term

# The proportion of realized full widths no wider than the target.
res_correct$value[res_correct$term == "pct_ci_less_width"]

# The population standardized mean is 1.5 but the plan assumed 2, so the
# planner sizes the study for a wider sampling distribution than the data
# will show, and the realized intervals come in narrower than the target.
set.seed(113)
res_misspecified <- ss_aipe_sm_sensitivity(true_sm = 1.5, supposed_sm = 2,
  width = 0.5, G = 20, print_iter = FALSE)

# The effect of the misspecification on the mean interval width.
res_misspecified$value[res_misspecified$term == "mean_ci_width"]
```

ss_aipe_src *Sample Size Necessary for the Accuracy in Parameter Estimation Approach for a Standardized Regression Coefficient of Interest*

Description

Plans the sample size from the accuracy in parameter estimation (AIPE) approach for a standardized regression coefficient of interest: the smallest N at which the confidence interval for the coefficient, computed at the expected values of the sample squared multiple correlations, is no wider than `width`, or, with `assurance`, the published approximation for an interval no wider than `width` with at least that probability or, by simulation, the smallest N at which it is. Two intervals can be planned for. The default (`noncentral = TRUE`) is the noncentral t interval for the standardized coefficient, the interval `ci_src` reports; `noncentral = FALSE` is the central t interval of Kelley and Maxwell (2003, Equation 1), the interval of `ci_reg_coef` with `noncentral = FALSE` for standardized data. For the coefficient in the units of the data use `ss_aipe_rc`.

Usage

```
ss_aipe_src(
  true_R2_Y_X = NULL,
  true_R2_j_X_without_j = NULL,
  p = NULL,
  true_std_reg_coef_j = NULL,
  width,
  which_width = "Full",
  true_sd_Y = 1,
  true_sd_X = 1,
  true_cor_XX = NULL,
  true_cor_YX = NULL,
  which_predictor = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  conf_level = 0.95,
  assurance = NULL,
  noncentral = TRUE,
  assurance_method = c("analytic", "simulation"),
  G = 20000,
  seed = NULL
)
```

Arguments

`true_R2_Y_X`	Population value of the squared multiple correlation coefficient
`true_R2_j_X_without_j`	Population value of the squared multiple correlation coefficient predicting the jth predictor variable from the remaining p-1 predictor variables
`p`	The number of predictor variables

`true_std_reg_coef_j`
: Population value of the standardized regression coefficient for the *j*th predictor variable (the predictor of interest)

`width`
: The desired width of the confidence interval

`which_width`
: Which width (`"Full"`, `"Lower"`, or `"Upper"`) the width refers to (at present, only `"Full"` can be specified)

`true_sd_Y`
: The population standard deviation of *Y* (i.e., the dependent variable). It must be 1 (the default): the plan is for the standardized coefficient. Use `ss_aipe_rc` for the coefficient in the units of the data.

`true_sd_X`
: The population standard deviation of the *j*th *X* variable (i.e., the predictor variable of interest). It must be 1 (the default), as `true_sd_Y` must.

`true_cor_XX`
: Population correlation matrix for the *p* predictor variables

`true_cor_YX`
: Population *p* length vector of correlation between the dependent variable (*Y*) and the *p* independent variables

`which_predictor`
: Identifies which of the *p* predictors is of interest

`alpha_lower`
: Type I error rate for the lower confidence interval limit, at least 1e-9 and at most .5; a tail area below 1e-9 is refused, as in `ss_aipe_reg_coef`

`alpha_upper`
: Type I error rate for the upper confidence interval limit, at least 1e-9 and at most .5

`conf_level`
: Desired level of confidence for the computed interval (i.e., 1 - the Type I error rate); a level above .999999998, whose tail areas are below 1e-9, is refused

`assurance`
: Optional probability with which the realized confidence interval is to be no wider than `width`; when `NULL` (the default) the plan is for the width of the interval at the expected values of the sample squared multiple correlations (see `ss_aipe_reg_coef`), not for the mean width over samples

`noncentral`
: `TRUE` (the default) plans for the noncentral *t* interval for the standardized coefficient, the interval of `ci_src`; `FALSE` plans for the central *t* interval of Kelley and Maxwell (2003, Equation 1), the estimate plus and minus the *t* quantiles of its two tails times the standard error formed from the two sample squared multiple correlations.

`assurance_method`
: How the plan with `assurance` is made, used only with `assurance`. Neither interval has an exact form for the probability that it is no wider than `width`. `"analytic"` (the default), computed without simulation, is the published approximation: that of **MBESS** for the noncentral interval, and the chi square approximation of Kelley and Maxwell (2003, Equation 3) for the central one; neither computes that probability (`assurance_attained` is `NA`), and either can attain more or less than `assurance`. `"simulation"` is the smallest sample size, at or above the plan without `assurance`, at which the probability estimated from `G` simulated samples is at least `assurance`, and reports that estimate as `assurance_attained` (see Details).

`G`
: Number of simulated samples at each sample size the assurance step tries, and in its final estimate of the attained probability; a whole number of at least 100 (default 20000). Used and checked only with `assurance` and `assurance_method = "simulation"`

seed — Optional single number. When given, the assurance step is reproducible, and the random number stream of the calling session is left as it was. With NULL (the default) the draws come from the session's current stream. Used only where G is

Details

Not all of the arguments need to be specified, only those that provide all of the necessary information so that the sample size can be determined for the conditions specified.

The coefficient planned for is the population standardized regression coefficient of the *j*th predictor, β_j, the quantity true_std_reg_coef_j stands for; on the covariance structure path it is the *j*th element of $R_{XX}^{-1} r_{XY}$, with R_{XX} given by true_cor_XX and r_{XY} by true_cor_YX, and a supplied true_std_reg_coef_j must equal it. A condition raised by ss_aipe_reg_coef, which makes the plans without assurance, names the arguments of this function (true_std_reg_coef_j, not true_reg_coef_j).

The interval planned for. Both intervals use the standard error $\sqrt{(1 - R^2)/[(1 - R^2_{X_j \cdot X_{-j}})(N - p - 1)]}$, formed from the sample squared multiple correlations of *Y* with the *p* predictors and of X_j with the other $p-1$, which treats the sample standard deviations as constants (see ci_src). With noncentral = TRUE the limits are the limits for the noncentrality parameter at the observed *t* statistic times that standard error, and the plan without assurance is the noncentral plan of ss_aipe_reg_coef (noncentral = TRUE). With noncentral = FALSE the interval is the estimate plus and minus *t* quantiles on $N - p - 1$ degrees of freedom times that standard error (Kelley & Maxwell, 2003, Equation 1), and the plan without assurance is theirs (Equation 2 with the *t* quantile, at the expected squared multiple correlations), the central plan of ss_aipe_reg_coef at unit standard deviations, with the *t* quantiles of its two tails.

The assurance step. With assurance the plan is by default (assurance_method = "analytic") the published approximation, computed without simulation: for the noncentral interval that of **MBESS**, for a lower confidence limit of the squared multiple correlation, and for the central interval the chi square approximation of Kelley and Maxwell (2003, Equation 3), the plan of ss_aipe_reg_coef without assurance with the variance of the coefficient multiplied by the assurance quantile of a chi square on $N - 1$ degrees of freedom over $N - p - 1$. Neither computes the probability it attains, so assurance_attained is NA, and the printed footer says so; both start from the plan without assurance, which can be a case or more smaller than that of **MBESS** (see ss_aipe_reg_coef).

With assurance_method = "simulation" the plan is the smallest sample size, not below the plan without assurance, at which the interval for the standardized coefficient is no wider than width with at least that probability, for random predictors from a multivariate normal population. The width of the noncentral interval depends on the observed *t* statistic and on both sample squared multiple correlations, and that of the central interval on the two squared multiple correlations, so for either interval the probability is estimated from G samples drawn from the exact sampling distribution of the sums of squares and products, the same samples at every sample size the search tries, so that the estimate changes smoothly with the sample size. The search assumes that the estimate does not decrease as the sample size grows, which held in every set checked (see "The assurance step" in ss_aipe_reg_coef); assurance_attained is the estimate at the returned sample size, at least assurance, with its Monte Carlo standard error in the attribute assurance_mc_se. The distribution of the width also depends on the correlation of *Y* with the targeted predictor: on the covariance structure path it is given, and on the squared multiple correlations path the step uses the smallest probability over 21 equally spaced values of it, from the smallest to the largest the three

planning values allow (the two ends with $p = 2$, the only values possible there), an approximate minimum over that range. Give seed for a plan that can be reproduced. This is the simulation Kelley and Maxwell (2008, pp. 186–187) described for the standardized coefficient, for which they wrote that no satisfactory method then existed.

The central interval for the standardized coefficient is not the central interval for the coefficient in the units of the data that ss_aipe_reg_coef plans for with noncentral = FALSE, and the exact probability that planner computes is that of the interval in the units of the data. The standard error of that interval is formed from the residual sums of squares of *Y* and of X_j; that of the standardized interval divides each by the total sum of squares of its variable, which gives $1-R^2$ and $1-R^2_{X_j \cdot X_{-j}}$, and these vary less from sample to sample, so the standardized interval needs fewer cases to be as narrow with the same probability. For two predictors correlated .3 with correlations of .4 and .3 with *Y*, a width of .3, and an assurance of .90, this function returns 171 with assurance_method = "simulation", seed = 1, and the default G, at which the central interval for the standardized coefficient is no wider than .3 with probability .914 (.895 at 170). The plan is simulated, so another seed can give one a case away: over seeds 1 to 30, 29 gave 171 and one gave 170. The central plan of ss_aipe_reg_coef there is 187 (.997 for this interval), and the chi square approximation of Kelley and Maxwell (2003, Equation 3, with the *t* quantile in place of the normal one), the default plan here, is 181 (.986). For their five exchangeable predictors (pp. 312–313), correlated .4 with one another and .3 with *Y*, at a width of .2 and an assurance of .80, the simulated plan is 478 (with each of seeds 1 to 30; .807, and .797 at 477), where the approximation gives 486 (.877) and ss_aipe_reg_coef 491 (.911). Each probability is the proportion of intervals no wider than the target among samples of raw multivariate normal data (50,000 or 100,000 samples at each *N*, Monte Carlo standard error at most .0014), standardized and fit by least squares without DMAR. Kelley and Maxwell's own simulation of that example (p. 317) reported 81.64 percent of the intervals no wider than .2 at their plan of 485 (made with the normal quantile) and a mean half-width of .098; raw multivariate normal data give .871 there with the *t* quantile of their Equation 1 (.887 with the normal one) and a mean half-width of .097 (50,000 samples, Monte Carlo standard error at most .0015).

Value

A data.frame with columns term and value: necessary_N, the necessary total sample size; with assurance, the rows assurance and assurance_attained, which is NA for the published approximation of assurance_method = "analytic" and, with "simulation", the simulated probability at necessary_N that the realized interval is no wider than width, with its Monte Carlo standard error in the attribute assurance_mc_se and the row G. With assurance the printed footer names the assurance method and says how assurance_attained was found, or that the approximation does not compute it; the attributes assurance_method and assurance_probability ("simulation" or "not computed") record the same.

Warning

As discussed in Kelley and Maxwell (2008), the sample size planning approach from the AIPE perspective used in this function is only an approximation.

Note

This function calls upon ss_aipe_reg_coef in DMAR, which plans the coefficient in the units of the data by default, but has a different naming scheme. See ss_aipe_reg_coef for more details.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

See Also

ss_aipe_src_sensitivity, ci_src, ci_reg_coef, ci_nc_t, ss_aipe_reg_coef, ss_aipe_rc

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Exchangeable correlation structure
true_cor_YX <- c(.3, .3, .3, .3, .3)
true_cor_XX <- rbind(c(1, .5, .5, .5, .5), c(.5, 1, .5, .5, .5), c(.5, .5, 1, .5, .5),
                c(.5, .5, .5, 1, .5), c(.5, .5, .5, .5, 1))

ss_aipe_src(width = .1, which_width = "Full", true_sd_Y = 1, true_sd_X = 1,
            true_cor_XX = true_cor_XX,
            true_cor_YX = true_cor_YX, which_predictor = 1, conf_level = 1 - .05)

# With an assurance of .85: the approximation with a lower confidence
# limit of the squared multiple correlation, and the plan found by
# simulation. G = 2000 keeps the example quick; a reported plan deserves
# the default G = 20000.
ss_aipe_src(width = .1, true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
            which_predictor = 1, assurance = .85)
plan <- ss_aipe_src(width = .1, which_width = "Full", true_sd_Y = 1, true_sd_X = 1,
                    true_cor_XX = true_cor_XX,
                   true_cor_YX = true_cor_YX, which_predictor = 1, conf_level = 1 - .05,
                    assurance = .85, assurance_method = "simulation",
                    G = 2000, seed = 113)
plan
attr(plan, "assurance_mc_se")

# The central t interval of Kelley and Maxwell (2003, Equation 1): the
# plan without assurance, and with an assurance of .85 by their chi
# square approximation and by simulation.
ss_aipe_src(width = .1, true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
```

```
            which_predictor = 1, noncentral = FALSE)
ss_aipe_src(width = .1, true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
            which_predictor = 1, noncentral = FALSE, assurance = .85)
ss_aipe_src(width = .1, true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
            which_predictor = 1, noncentral = FALSE, assurance = .85,
            assurance_method = "simulation", G = 2000, seed = 113)
```

`ss_aipe_src_sensitivity` *Sensitivity Analysis for Sample Size Planning From the Accuracy in Parameter Estimation Perspective for the Standardized Regression Coefficient*

Description

Performs a sensitivity analysis when planning sample size from the Accuracy in Parameter Estimation Perspective for the standardized regression coefficient. `ss_aipe_reg_coef_sensitivity` and `ss_aipe_rc_sensitivity` are the counterparts for the unstandardized coefficient.

Usage

```
ss_aipe_src_sensitivity(
  true_var_Y = NULL,
  true_cov_YX = NULL,
  true_cov_XX = NULL,
  supposed_var_Y = NULL,
  supposed_cov_YX = NULL,
  supposed_cov_XX = NULL,
  specified_N = NULL,
  which_predictor = 1,
  width = NULL,
  noncentral = TRUE,
  conf_level = 0.95,
  assurance = NULL,
  G = 1000,
  print_iter = TRUE,
  filename = NULL,
  assurance_method = c("analytic", "simulation"),
  seed = NULL
)
```

Arguments

`true_var_Y`	Population variance of the dependent variable (Y)
`true_cov_YX`	Population covariances vector between the p predictor variables and the dependent variable (Y)

`true_cov_XX` Population covariance matrix of the p predictor variables

`supposed_var_Y` Supposed variance of the dependent variable (Y), the planning value from which the sample size is planned

`supposed_cov_YX`
Supposed covariances vector (planning values) between the p predictor variables and the dependent variable (Y)

`supposed_cov_XX`
Supposed covariance matrix (planning values) of the p predictor variables

`specified_N` Directly specified sample size (instead of planning one from the supposed covariance structure)

`which_predictor`
identifies which of the p predictors is of interest

`width` Desired confidence interval width for the standardized regression coefficient of interest

`noncentral` `TRUE` (the default) for the noncentral t interval for the standardized coefficient (that of `ci_src`), `FALSE` for the central t interval of Kelley and Maxwell (2003, Equation 1). The interval of each replication is that interval, and a sample size planned from the supposed values is planned for it by `ss_aipe_src` with the same `noncentral`.

`conf_level` desired level of confidence for the computed interval (i.e., 1 - the Type I error rate)

`assurance` degree of certainty that the obtained confidence interval will be sufficiently narrow

`G` the number of generations/replication of the simulation study within the function

`print_iter` specify with a `TRUE/FALSE` statement if the iteration number should be printed as the simulation within the function runs

`filename` Optional path of a CSV file to receive the per-replication results (the standardized coefficient estimate `std_reg_coef_j`, its confidence limits `ci_lower` and `ci_upper`, the observed R^2 `R2`, the standard error `se_std_reg_coef_j`, and the t statistic `t_value`), appended when the file already exists and created otherwise; the default `NULL` writes nothing, and a throwaway run that wants the file should point it at `tempfile(fileext = ".csv")`.

`assurance_method`
Passed to `ss_aipe_src`, which plans the sample size from the supposed values with `assurance`: `"analytic"` (the default) for the published approximation, which does not compute the probability of an interval no wider than `width` (for the noncentral interval that of **MBESS**, and for the central one the chi square approximation of Kelley and Maxwell, 2003, Equation 3), at unit standard deviations; `"simulation"` for the smallest sample size at which a simulated estimate of that probability, made by the planner from its default `G` of 20000 samples with the random number stream that `seed` sets, is at least `assurance`. Not used with `specified_N` or without `assurance`.

`seed` `NULL` (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

Direct specification of `true_cov_YX` and `true_cov_XX` is necessary, even if one is interested in a single regression coefficient, so that the covariance/correlation structure can be specified when the simulation study within the function runs.

The coefficient. The data of every replication are standardized with their own sample standard deviations before the regression is fit, so each estimate and each interval is for the standardized coefficient, the quantity `true_std_reg_coef_j` of `ss_aipe_src` stands for. The true value the intervals are judged against is the population standardized coefficient

$$\beta_j = b_j \sigma_{X_j} / \sigma_Y,$$

the unstandardized coefficient b_j (the *j*th element of $\Sigma_{XX}^{-1}\sigma_{XY}$, from `true_cov_XX` and `true_cov_YX`) multiplied by the ratio of the population standard deviations (Kelley & Maxwell, 2003, p. 316), where σ_Y is the square root of `true_var_Y` and σ_{X_j} the square root of the *j*th diagonal element of `true_cov_XX`. The supposed coefficient is put on the same scale with the supposed standard deviations (`supposed_var_Y` and the diagonal of `supposed_cov_XX`), and a sample size planned from the supposed values is planned by `ss_aipe_src` for a coefficient on that scale, with the supposed correlations and the same `noncentral`, so that `width` is a width in the units of the intervals simulated and the plan is for the interval simulated.

The plan with assurance. With `noncentral = FALSE` and `assurance`, the plan is that of `ss_aipe_src` for the central interval for the standardized coefficient: by default the chi square approximation of Kelley and Maxwell (2003, Equation 3), and with `assurance_method = "simulation"` the smallest sample size at which that interval is no wider than `width` with at least that probability. For two predictors correlated .3 with correlations of .4 and .3 with *Y*, a width of .3, and an assurance of .90, the approximation is 181, at which .986 of the standardized intervals are no wider than .3, and the simulated plan is 171 (another seed can give one a case away; see `ss_aipe_src`).

The interval of each replication is that of `ci_reg_coef` with the standard error of the regression fitted to the standardized data. That standard error treats the sample standard deviations as constants, so it is approximate, and the predictors simulated here are random, so those of the predictors vary along with that of *Y* (Kelley & Maxwell, 2003, pp. 315–316; see `ci_src`); the coverage rows show how good the approximation is for the structure given.

Value

A `data.frame` with columns `term` and `value` summarizing the Monte Carlo sensitivity analysis: mean / median / SD summaries of the realized standardized regression coefficient (`mean_std_reg_coef_j`, `median_std_reg_coef_j`, `sd_std_reg_coef_j`), the realized interval widths (`mean_ci_width`, `median_ci_width`, `sd_ci_width`), and the realized squared multiple correlation coefficient (`mean_R2`, `median_R2`, `sd_R2`); the proportion of intervals at or below the planning target (`pct_ci_less_width`); the tail-specific and overall empirical non-coverage of the population standardized coefficient (`pct_ci_miss_low`, `pct_ci_miss_high`, `total_type_I_error`), all proportions on the 0 to 1 scale; and the input echoes (`total_N`, `p`, `which_predictor`, `true_std_reg_coef_j` and `supposed_std_reg_coef_j`, the population and planning values of the standardized coefficient implied by the covariance structures, `width`, `conf_level`, and, when one was supplied, `assurance`). The per-replication file written with `filename` has the columns `std_reg_coef_j`, `ci_lower`, `ci_upper`, `R2`, `se_std_reg_coef_j`, and `t_value`.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Note

Note that when the true and supposed covariance structures agree (`true_cov_YX` equals `supposed_cov_YX` and `true_cov_XX` equals `supposed_cov_XX`), the results are not literally from a sensitivity analysis, rather the function performs a standard simulation study. A simulation study can be helpful in order to determine if the sample size procedure under or overestimates necessary sample size. See `ss_aipe_reg_coef_sensitivity` in DMAR for more details.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

See Also

ss_aipe_src, ss_aipe_reg_coef_sensitivity, ss_aipe_rc_sensitivity, ss_aipe_reg_coef, ci_reg_coef

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Examples

```
# Sensitivity analysis for a standardized regression coefficient
# with two correlated predictors. A production run uses many more
# generations (G = 1000 is typical); G is reduced here so the
# example runs quickly.
set.seed(113)
cov_XX  <- matrix(c(1, 0.3, 0.3, 1), nrow = 2)
cov_YX  <- c(0.4, 0.3)
ss_aipe_src_sensitivity(
  true_var_Y = 1, true_cov_YX = cov_YX, true_cov_XX = cov_XX,
  supposed_var_Y = 1, supposed_cov_YX = cov_YX, supposed_cov_XX = cov_XX,
  which_predictor = 1, width = 0.20, conf_level = 0.95,
  G = 50, print_iter = FALSE
)
```

---

ss_aipe_two_phase *Sample Size Planning From the AIPE Perspective for a Mean Under Two-Phase Sampling*

Description

Plans the number of labeled units, n, on which a costly target measurement must be taken so that the confidence interval for the target mean from ci_two_phase is sufficiently narrow, when an inexpensive proxy measurement is available on every unit of a pool of M units. The plan follows the accuracy in parameter estimation (AIPE) approach: the goal is an interval no wider than width, either at the planning values (the expected width) or with a stated assurance that the realized interval will be that narrow (Kelley & Rausch, 2006). Beside the answer it reports the number of labeled units the same goal needs without the proxy, so the saving is in view.

Usage

```
ss_aipe_two_phase(
  true_sd_Y,
  true_cor,
  width,
  n_pool = Inf,
  conf_level = 0.95,
  assurance = NULL,
  true_kurtosis = 0
)
```

Arguments

`true_sd_Y`	Planning value of the standard deviation of the target measurement, σ_Y.
`true_cor`	Planning value of the correlation between the target and the proxy, ρ.
`width`	The desired full width of the confidence interval (the upper limit minus the lower limit), in the units of the target.
`n_pool`	The number of units in the pool, M, every one of which carries the proxy. The default `Inf` plans as though the pool were unlimited.
`conf_level`	Confidence level (default `0.95`).
`assurance`	`NULL` (default) to plan for the expected width, or a probability in (0, 1) that the realized width will be no wider than `width`, as in `ss_aipe_c`.
`true_kurtosis`	Excess kurtosis (the scale of `kurtosis`) of the residual of the target on the proxy; default `0`, its normal value. Used only with `assurance`, where it adjusts the degrees of freedom of the variance estimate (see Details).

Details

The interval planned for. The regression estimator of double sampling (Tamhane, 1978, Section 2.1, p. 420; the design is that of Neyman, 1938, who formed strata from the inexpensive variable) that `ci_two_phase` reports has, to first order, variance $\sigma_Y^2(1-\rho^2)/n + \rho^2\sigma_Y^2/M$, and its interval uses the t distribution on $n-2$ degrees of freedom. With ω the desired width (`width`), $1-\alpha$ the `conf_level`, and $t_n = t_{1-\alpha/2,\,n-2}$, the plan for the expected width is the smallest n with

$$2\,t_n\left\{\frac{\sigma_Y^2(1-\rho^2)}{n} + \frac{\rho^2\sigma_Y^2}{M}\right\}^{1/2} \le \omega.$$

In an evaluation of a system whose outputs an automated judge labels, for example, n is the number of outputs a human assessor must label when the judge has labeled M of them.

Assurance. The realized width replaces $\sigma_Y^2(1-\rho^2)$ by the residual mean square S_e^2 and $\rho^2\sigma_Y^2$ by $\hat{b}^2 S_X^2$. Treating the second, which is divided by the large M, as fixed at its planning value, and $\nu S_e^2/\sigma_e^2$ as chi square on ν degrees of freedom, the probability that the realized width is at most ω is

$$F_{\chi^2_\nu}\left(\frac{\nu\,n\,\{\omega^2/(4t_n^2) - \rho^2\sigma_Y^2/M\}}{\sigma_Y^2(1-\rho^2)}\right),$$

and the plan is the smallest n from which it is at least `assurance` at every larger n. Under normal residuals $\nu = n-2$. For residuals with excess kurtosis κ (`true_kurtosis`) the variance of S_e^2 is approximately $\sigma_e^4\{2/(n-2) + \kappa/n\}$, and matching it to a scaled chi square gives $\nu = 2/\{2/(n-2) + \kappa/n\}$: for an assurance of .80 or more, a lighter-tailed residual ($\kappa < 0$) needs fewer units and a heavier-tailed one more. A binary target is not light-tailed as a rule: a 0/1 variable with proportion p has excess kurtosis $\{1-6p(1-p)\}/\{p(1-p)\}$, which is negative only for p between about .21 and .79 and is about 5.1 at $p = .90$. The planning value is best taken from the residuals of a pilot regression of the target on the proxy, through `kurtosis`. With a large κ and a small n, ν is small, and the two-moment approximation can reach the assurance at a few small n, fall below it over a stretch of larger ones, and reach it again; the plan is the start of the stretch over which it holds for good, which the search finds exactly for any `assurance` in (0, 1). Under normal residuals and with an assurance of one half or more the two readings agree.

The pool sets a floor. The labeled units are drawn from the pool, and `ci_two_phase` needs at least one unit that carries the proxy alone, so $n \leq M - 1$. No number of labeled units can therefore bring the width below $2z_{1-\alpha/2}\{\sigma_Y^2(1-\rho^2)/(M-1) + \rho^2\sigma_Y^2/M\}^{1/2}$ (the `minimum_width` row), which is close to $2z_{1-\alpha/2}\,\sigma_Y/\sqrt{M}$, the width of the target mean were every unit of the pool labeled; a target at or below it is an error. A target just above it can still be out of reach once the *t* quantile and the assurance are accounted for; when the plan needs more than $M - 1$ labeled units, that too is an error, naming the pool as the constraint, rather than a missing value.

The target-only comparison. `necessary_n_target_only` is the planned size of the usual *t* interval for the target mean from labeled units alone, the plan `ss_aipe_mean()` makes for a single mean, so the two functions agree. Without an assurance it is the smallest n with $2\,t_{1-\alpha/2,\,n-1}\,c_4(n)\,\sigma_Y/\sqrt{n} \leq \omega$, where $c_4(n)\sigma_Y$ is the expected sample standard deviation under normality, so that the left side is the exact expected width; with `assurance` it is the same chi square argument on $n - 1$ degrees of freedom. It is not bounded by `n_pool`, because a design without the proxy has no use for a pool. The `true_kurtosis` supplied is the residual's, and the comparison uses it for the target as well, an approximation: the target's own kurtosis can differ, and the comparison for it is `ss_aipe_mean()` with that value. The labeled-unit plan, by contrast, sets the two estimates in its standard error at their planning values, since the expected value of the two-phase standard error has no closed form once the pool term enters; this overstates its expected width slightly, by a factor of at most about $1 + 1/\{4(n-2)\}$, and can add a unit to `necessary_n_labeled`.

Value

A `data.frame` (classes `dmar_ss_aipe` and `dmar_tbl`) with rows

`necessary_n_labeled`
: the planned number of labeled units.

`necessary_n_target_only`
: the number the same goal needs without the proxy.

`ratio`
: the first divided by the second.

`expected_width`
: the full width at `necessary_n_labeled` and the planning values.

`minimum_width`
: the narrowest width the pool allows, the normal-quantile width with all but one unit of the pool labeled (0 for an unlimited pool; see Details).

`width`, `true_sd_Y`, `true_cor`, `n_pool`, `conf_level`
: the inputs.

`assurance`, `true_kurtosis`
: the inputs, present only when an assurance was supplied.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Neyman, J. (1938). Contribution to the theory of sampling human populations. *Journal of the American Statistical Association, 33*(201), 101–116. doi:10.1080/01621459.1938.10503378

Tamhane, A. C. (1978). Inference based on regression estimator in double sampling. *Biometrika, 65*(2), 419–427. doi:10.1093/biomet/65.2.419

See Also

ci_two_phase for the interval, ss_aipe_two_phase_cost for the least costly numbers of labeled and pool units, ss_aipe_two_phase_sensitivity to evaluate a plan by simulation, and ss_seq_two_phase for the sequential rule.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(), ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(), ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivit ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(), ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase_cost(), ss_aipe_two_phase_sensitivity()

Examples

```
# Planning values from a pilot: a target standard deviation of 0.43 and
# a correlation of 0.55 with the proxy, which is available on a pool of
# 20,000 units. A 90% interval no wider than 0.10:
ss_aipe_two_phase(true_sd_Y = 0.43, true_cor = 0.55, width = 0.10,
                  n_pool = 20000,
                  conf_level = 0.90)

# The same goal with 80% assurance that the realized interval is no
# wider than 0.10:
ss_aipe_two_phase(true_sd_Y = 0.43, true_cor = 0.55, width = 0.10,
                  n_pool = 20000,
                  conf_level = 0.90, assurance = 0.80)

# A smaller pool raises the floor and the number of labeled units.
ss_aipe_two_phase(true_sd_Y = 0.43, true_cor = 0.55, width = 0.10,
                  n_pool = 1000,
                  conf_level = 0.90, assurance = 0.80)
```

---

ss_aipe_two_phase_cost

Sample Size Planning From the AIPE Perspective for the Least Costly Two-Phase Design

---

Description

Takes planning values for a two-phase design (the standard deviation of a costly target measurement, its correlation with an inexpensive proxy, and the cost of a proxy relative to a target measurement)

and a desired confidence interval width for the target mean. Returns the number of labeled units n, which carry both measurements, and the number of pool units M, which carry the proxy, of the least costly design in whole numbers whose interval from `ci_two_phase` has an expected width no larger than desired. The table also gives the cost relative to measuring the target alone and the correlation the proxy must have before it can save anything.

Usage

```
ss_aipe_two_phase_cost(
  true_sd_Y,
  true_cor,
  width,
  cost_ratio,
  conf_level = 0.95
)
```

Arguments

`true_sd_Y`	Planning value of the standard deviation of the target measurement, σ_Y.
`true_cor`	Planning value of the correlation between the target and the proxy, ρ. Only its absolute value matters.
`width`	The desired full width of the confidence interval (the upper limit minus the lower limit), in the units of the target.
`cost_ratio`	The cost of one proxy measurement relative to one target measurement, $r = c_P/c_T$, a number in (0, 1).
`conf_level`	Confidence level (default `0.95`).

Details

Estimand and interval. The quantity to be estimated is μ_Y, the population mean of the target measurement. The sampling unit is one unit of the pool: M units are drawn at random, the proxy is observed on all of them, and the target on a simple random subsample of n of them. The interval planned for is the one `ci_two_phase` reports for the regression estimator of double sampling, a Wald-type interval whose standard error estimates $\sigma_Y^2(1-\rho^2)/n + \rho^2\sigma_Y^2/M$ and whose reference distribution is t on $n-2$ degrees of freedom. This is a plan for the expected width, with no assurance: about half of all realized intervals are wider than `width`. `ss_aipe_two_phase` adds an assurance for a given pool.

The cost model. Every unit of the pool, the labeled ones included, costs c_P for its proxy, and each labeled unit costs c_T more for its target, so a design costs $c_T n + c_P M$, which is $n + rM$ in units of one target measurement. In an evaluation of a system whose outputs an automated judge labels, for example, c_T is the cost of a human assessor's label and c_P that of the judge's. The target alone on n units costs n.

Which designs are admissible. A two-phase design has $n \geq 3$ and $M \geq n + 1$, the least `ci_two_phase` accepts, and meets the width when, with $q = 1 - (1 - \text{conf_level})/2$ and ω the desired width (`width`),

$$2\, t_{q,\, n-2}\, \sigma_Y \left\{ \frac{1-\rho^2}{n} + \frac{\rho^2}{M} \right\}^{1/2} \leq \omega,$$

the criterion ss_aipe_two_phase uses for a given pool, so that planner, given n_pool = necessary_n_pool, returns necessary_n_labeled. The left side sets the two estimates in the standard error at their planning values, which overstates the expected width slightly (see ss_aipe_two_phase). The alternative without a proxy is the *t* interval for one mean on the smallest n with $2\,t_{q,\,n-1}\,c_4(n)\,\sigma_Y/\sqrt{n} \le \omega$, where $c_4(n)\sigma_Y$ is the expected sample standard deviation under normality; this is the necessary_n_target_only row here and in ss_aipe_two_phase. Because the *t* quantile is part of the requirement, a design with very few labeled units is admissible only when it is wide enough to pay for its quantile (12.7 at a 95% level with 3 labeled units), and no separate minimum is imposed.

The search. For each n the smallest admissible pool follows from the inequality above, so the search is over n alone. Since $t_{q,\,n-2}$ is at least the normal quantile z_q, no design with n labeled units can cost less than $n + r\max\{n+1,\ b/(1-a/n)\}$ for $n > a$, and none exists for $n \le a$, where $a = (1-\rho^2)s$ and $b = \rho^2 s$ with $s = (2z_q\sigma_Y/\omega)^2$. That bound is convex in n, so the function starts at its minimizer with the target alone as the design to beat, examines every n above and then below the minimizer for as long as the bound is less than the cost of the best design found, and returns that design. No design in whole numbers costs less: every n not examined has a bound at least as large as the cost returned. A two-phase design is returned only when it costs less than the target alone, and among designs of equal cost the first found is kept, so a tie with the target alone returns the target alone. The search and its bound were derived for DMAR; in 2,200 planning sets drawn at random (correlations of either sign from .02 to .999, cost ratios from .001 to .98, four confidence levels) the result equaled an exhaustive search over pairs (n, M) every time.

The fractional solution, for comparison. If the sizes could be fractional and the normal quantile were used, minimizing $n + rM$ subject to $a/n + b/M = 1$ (a Lagrange multiplier argument) gives $n = a + \sqrt{rab}$ and $M = b + \sqrt{ab/r}$, at a cost of $(\sqrt{1-\rho^2} + |\rho|\sqrt{r})^2$ times the s units the target alone would need. That factor is the relative_cost_continuous row. It is below 1 exactly when $|\rho| > 2\sqrt{r}/(1+r)$, the threshold_cor row; at or below that correlation the row is 1. The two sizes are in the ratio $M/n = \sqrt{b/(ra)}$: each is proportional to its share of the variance and inversely proportional to the square root of its cost. Neyman (1938, Equations 49, 52, and 53, pp. 110–111) derived that square root rule for double sampling in which the inexpensive variable forms strata and the variance is minimized for a fixed total cost; the form here, with a regression on the proxy and the cost minimized for a fixed width, is the same allocation problem stated the other way round. For the regression estimator itself, Tamhane (1978, Equation 2.2, p. 421) gives the same ratio of the two sizes for a fixed budget, and the correlation above which double sampling is no worse than the target alone, $\rho^2 \ge 4c_1c_2/(c_1+c_2)^2$ with c_1 and c_2 the costs of the proxy and of the target, which is threshold_cor; the result is for a bivariate normal population with a known covariance matrix. The fractional solution is a lower bound: no design in whole numbers that meets the width costs less than relative_cost_continuous times s target measurements. The design returned costs more than that bound because sizes are whole numbers and because the interval uses the *t* quantile (a median of 7% more, and far more when only a few units are needed, in the 891 random planning sets above in which a two-phase design was returned).

What can happen at the edges.

- A correlation at or below threshold_cor: the target alone is returned (necessary_n_pool is 0 and relative_cost is 1). In the random planning sets above no two-phase design was ever the least costly there.
- A correlation above threshold_cor: the threshold is exact for fractional sizes and the normal quantile only. The target alone can still be the least costly design in whole numbers, most often when few units are needed or the correlation is a little above the threshold (376 of the 1,267 random planning sets above the threshold), and it is then returned.

- `true_cor = 0`: the proxy carries no information and the target alone is returned.
- A proxy that costs as much as the target or more (`cost_ratio` of 1 or more) never saves anything and is refused with an error of class `dmar_cost_ratio_invalid`.
- A very wide `width`: the target alone needs at least 2 units, the fewest from which a standard deviation can be estimated.
- Sizes that cannot be represented as whole numbers (above 2^{52}, about 4.5e15), or a search that would examine more than 10^8 values of n: refused with an error of class `dmar_two_phase_no_design`. Neither arises for a width within a factor of a million of `true_sd_Y`.

Value

A `data.frame` (classes `dmar_ss_aipe` and `dmar_tbl`) with rows

`necessary_n_labeled`
: the number of labeled units, n; when `necessary_n_pool` is 0, the number of units for the target alone.

`necessary_n_pool`
: the number of units in the pool, M, every one of which carries the proxy; 0 when the design measures the target alone.

`relative_cost`
: the cost of the returned design, $n + rM$, over the cost of the target alone, `necessary_n_target_only`, both in whole numbers under the criteria above; 1 when the returned design is the target alone.

`relative_cost_continuous`
: the same ratio for fractional sizes under the normal quantile, $(\sqrt{1-\rho^2}+|\rho|\sqrt{r})^2$, or 1 at or below `threshold_cor`; reported for comparison.

`expected_width`
: the planned full width at the returned sizes and the planning values: the left side of the two-phase inequality above, or of the target-alone inequality when the pool is 0. It is at most `width`.

`necessary_n_target_only`
: the number of units the target alone needs for the same width.

`threshold_cor`
: the absolute correlation above which the proxy lowers the cost of the fractional solution, $2\sqrt{r}/(1+r)$.

`width`, `true_sd_Y`, `true_cor`, `cost_ratio`, `conf_level`
: the inputs.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Neyman, J. (1938). Contribution to the theory of sampling human populations. *Journal of the American Statistical Association, 33*(201), 101–116. doi:10.1080/01621459.1938.10503378

Tamhane, A. C. (1978). Inference based on regression estimator in double sampling. *Biometrika, 65*(2), 419–427. doi:10.1093/biomet/65.2.419

See Also

ss_aipe_two_phase for the labeled units given a pool, with or without an assurance, ci_two_phase for the interval, and ss_aipe_two_phase_sensitivity to evaluate a design by simulation.

Other AIPE sample size planning: ss_aipe_c_sensitivity(), ss_aipe_cliff_delta(), ss_aipe_cliff_delta_sensit
ss_aipe_composite_sem(), ss_aipe_crossed(), ss_aipe_crossed_sensitivity(), ss_aipe_equivalence_r(),
ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(),
ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivit
ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(
ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(),
ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(),
ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(),
ss_aipe_two_phase_sensitivity()

Examples

```
# A proxy measurement costs 1% of a target measurement. Planning values
# of 0.43 for the target's standard deviation and 0.55 for its
# correlation with the proxy; a 90% interval no wider than 0.10. The
# least costly design labels 151 units of a pool of 957 and costs 79%
# of what the 202 units of the target alone would.
ss_aipe_two_phase_cost(true_sd_Y = 0.43, true_cor = 0.55, width = 0.10,
                       cost_ratio = 0.01, conf_level = 0.90)

# At 10% of the cost the same proxy no longer pays: a correlation of 0.55
# is below the threshold, and the design measures the target alone.
ss_aipe_two_phase_cost(true_sd_Y = 0.43, true_cor = 0.55, width = 0.10,
                       cost_ratio = 0.10, conf_level = 0.90)

# A worked example in standard deviation units: a proxy that correlates
# 0.90 with the target at 5% of its cost, and a 95% interval of width
# 0.20. The design is 109 labeled units in a pool of 1011, a cost of
# 109 + 0.05 * 1011 = 159.55 target measurements against 387.
plan <- ss_aipe_two_phase_cost(true_sd_Y = 1, true_cor = 0.90, width = 0.20,
                               cost_ratio = 0.05)
plan

# The companion planner, given that pool, returns the same number of
# labeled units.
ss_aipe_two_phase(true_sd_Y = 1, true_cor = 0.90, width = 0.20, n_pool = 1011)
```

---

ss_aipe_two_phase_sensitivity

Sensitivity Analysis for Sample Size Planning From the AIPE Perspective for a Mean Under Two-Phase Sampling

---

Description

Evaluates a two-phase design by simulation: how wide the confidence intervals from `ci_two_phase` turn out to be, how often they are no wider than planned, and how often they miss the population mean of the target, when the design is carried out in a population that may differ from the one the plan assumed. The design is either planned by `ss_aipe_two_phase` from planning values, so that the distance between planning values and population values is what is studied, or fixed by the user.

Usage

```
ss_aipe_two_phase_sensitivity(
  true_sd_Y = NULL,
  true_cor = NULL,
  supposed_sd_Y = NULL,
  supposed_cor = NULL,
  specified_n_labeled = NULL,
  n_pool,
  width,
  population = NULL,
  conf_level = 0.95,
  assurance = NULL,
  true_kurtosis = 0,
  G = 1000,
  print_iter = FALSE,
  filename = NULL,
  seed = NULL
)
```

Arguments

`true_sd_Y`, `true_cor`
: The population standard deviation of the target measurement and its correlation with the proxy, from which the data are generated when `population` is `NULL`.

`supposed_sd_Y`, `supposed_cor`
: Planning values passed to `ss_aipe_two_phase` to choose the number of labeled units; supply both, or `specified_n_labeled` instead.

`specified_n_labeled`
: The number of labeled units to evaluate, in place of a planned one.

`n_pool`
: The number of units in the pool, M, a whole number larger than the number of labeled units.

`width`
: The desired full width of the confidence interval, the planning target and the yardstick for `pct_ci_less_width`.

`population`
: `NULL` (default) for a bivariate normal population, or a data frame of observed (target, proxy) pairs with numeric columns `y` and `x`, to be resampled with replacement as the population; `true_sd_Y` and `true_cor` are then computed from it and must not be supplied.

`conf_level`
: Confidence level (default `0.95`).

assurance, true_kurtosis
: Passed to ss_aipe_two_phase when the design is planned; see that page.

G
: The number of Monte Carlo replications (default 1000).

print_iter
: Logical; print the replication number as the loop runs.

filename
: Optional path for a comma separated file recording every replication (the estimate mean_Y, the two confidence limits, the interval width, and two indicators of whether the interval missed the population mean below or above): nothing is written when filename is NULL (the default), a new file with a header row is created otherwise, an existing file at that path is appended to, and a throwaway run should point it at tempfile(fileext = ".csv").

seed
: NULL (default) to draw from the current random number stream, or a number that makes the run reproducible without disturbing the stream of the calling script.

Details

The data generating mechanism. With population = NULL, each replication draws the n labeled units as (target, proxy) pairs from a bivariate normal distribution with standard deviations true_sd_Y and 1, correlation true_cor, and means 0, and the $M - n$ unlabeled units' proxies from the same proxy distribution. The interval's width and coverage do not depend on the two means or on the proxy's scale, so these choices lose no generality; the population target mean the intervals are scored against is then 0. Because the regression estimator uses the unlabeled proxies only through their mean and their sum of squares, the simulation draws those two from their exact joint distribution (independent normal and scaled chi square) rather than drawing every unit of the pool, which keeps a large pool cheap. With a population data frame, each replication resamples the labeled pairs and the unlabeled proxies with replacement from its rows, a population that keeps whatever skew, bounds, or nonlinearity the observed pairs have; the target mean scored against is then the mean of its y column. In the evaluation of a system whose outputs an automated judge labels, for example, a pilot set of outputs carrying both a human assessor's label and the judge's makes such a population.

What is evaluated. Each replication computes the regression interval of ci_two_phase (the interval the planner plans for) and records its width and whether it missed the population target mean below or above. A replication whose labeled units all share one proxy value has no regression interval, and one whose labeled units all share one target value would have an interval of zero width, which ci_two_phase refuses. Either can happen only with a population data frame, chiefly one whose proxy takes few distinct values or whose target is binary with a proportion near 0 or 1. Such replications are dropped, with a single warning that counts each kind, and the number kept is attr(x, "simulation")$returned (see the section on the record of the simulation). Many dropped replications say that the design has too few labeled units for that population, and the summaries, which describe the replications that produced an interval, then flatter the design.

Value

A data.frame (class dmar_tbl) with term and value columns: mean_mean_Y, median_mean_Y, and sd_mean_Y for the estimates of the target mean; the realized widths (mean_ci_width, median_ci_width, sd_ci_width); the proportion of intervals no wider than width (pct_ci_less_width); the proportions that missed the population mean from above (pct_ci_miss_low, the mean below the lower limit) and from below (pct_ci_miss_high) and their sum (total_type_I_error), all on the 0 to 1 scale; then n_labeled, total_N (the pool size M, the design's total number of units, which the

rest of the two-phase family calls `n_pool`; the sensitivity family's shared schema names the size of a whole design `total_N`), `true_mean_Y` (the population target mean), `true_sd_Y`, `true_cor`, `supposed_sd_Y` and `supposed_cor` (`NA` when `specified_n_labeled` was given), `width`, `conf_level`, and, only when one was supplied, `assurance` and `true_kurtosis`. The number of replications used is in the `simulation` attribute (its element `returned`), described below.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted - returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Neyman, J. (1938). Contribution to the theory of sampling human populations. *Journal of the American Statistical Association, 33*(201), 101–116. doi:10.1080/01621459.1938.10503378

See Also

`ss_aipe_two_phase`, `ci_two_phase`, `ss_aipe_two_phase_cost`

Other AIPE sample size planning: `ss_aipe_c_sensitivity()`, `ss_aipe_cliff_delta()`, `ss_aipe_cliff_delta_sensit`
`ss_aipe_composite_sem()`, `ss_aipe_crossed()`, `ss_aipe_crossed_sensitivity()`, `ss_aipe_equivalence_r()`,

ss_aipe_equivalence_r_sensitivity(), ss_aipe_equivalence_smd(), ss_aipe_equivalence_smd_sensitivity(), ss_aipe_icc(), ss_aipe_icc_sensitivity(), ss_aipe_indirect_effect(), ss_aipe_indirect_effect_sensitivit ss_aipe_mean(), ss_aipe_mean_sensitivity(), ss_aipe_mixed_effects_sensitivity(), ss_aipe_omega_squared(ss_aipe_omega_squared_sensitivity(), ss_aipe_partial_r(), ss_aipe_partial_r_sensitivity(), ss_aipe_pcm_sensitivity(), ss_aipe_r(), ss_aipe_r_sensitivity(), ss_aipe_reliability_sensitivity(), ss_aipe_semipartial_r(), ss_aipe_semipartial_r_sensitivity(), ss_aipe_two_phase(), ss_aipe_two_phase_cost()

Examples

```
# A plan made with a correlation of 0.55, carried out where the
# correlation is only 0.45. G = 200 keeps the example quick; a reported
# evaluation deserves the default G = 1000 or more.
ss_aipe_two_phase_sensitivity(true_sd_Y = 0.43, true_cor = 0.45,
                              supposed_sd_Y = 0.43, supposed_cor = 0.55,
                              n_pool = 20000, width = 0.10,
                              conf_level = 0.90, G = 200, seed = 113)

# A fixed design, 150 labeled units in a pool of 5,000, evaluated in a
# population of binary (target, proxy) pairs resampled from a pilot of
# 300 units: how often is the 90% interval narrower than 0.085?
set.seed(113)
proxy  <- rbinom(300, 1, 0.6)
target <- ifelse(runif(300) < 0.8, proxy, rbinom(300, 1, 0.6))
ss_aipe_two_phase_sensitivity(specified_n_labeled = 150, n_pool = 5000,
                              width = 0.085, conf_level = 0.90,
                              population = data.frame(y = target, x = proxy),
                              G = 200, seed = 113)
```

---

ss_power_c *Sample Size or Power for an Unstandardized Contrast in a One-Way Between-Subjects ANOVA*

Description

Determine the necessary per-group sample size to achieve a desired level of statistical power for the test of a single planned (unstandardized) contrast in a one-way between-subjects analysis of variance, or, given a per-group sample size, return the realized statistical power.

Usage

```
ss_power_c(
  true_contrast,
  c_weights,
  true_sd,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_group = NULL,
```

```
  directional = FALSE
)
```

Arguments

`true_contrast`	The population unstandardized contrast effect, $\psi = \sum c_j \mu_j$
`c_weights`	Vector of contrast weights (must sum to zero); use fractional weights so that the positive weights sum to 1 (e.g., `c(0.5, 0.5, -0.5, -0.5)`)
`true_sd`	Within-group population standard deviation, σ in Details
`desired_power`	Desired statistical power (default 0.85)
`alpha_level`	Type I error rate (default 0.05)
`specified_n_per_group`	Per-group sample size (assumed balanced); if specified, returns the realized power
`directional`	Logical: `TRUE` for a one-sided test (in the same sign as `true_contrast`), `FALSE` (default) for a two-sided test

Details

Under the alternative hypothesis the contrast *t*-statistic follows a noncentral *t*-distribution with degrees of freedom $N - J$ (where $N = nJ$ is the total sample size and J the number of groups, taken as `length(c_weights)`) and noncentrality parameter $\lambda = \psi/(\sigma\sqrt{\sum c_j^2/n})$.

The function searches over per-group sample sizes n until power first reaches `desired_power`; when `specified_n_per_group` is supplied it instead returns the realized power.

Value

A `data.frame` with rows for `necessary_n_per_group` (or `specified_n_per_group`), `actual_power`, and `noncentral_t_parm`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ss_power_sc, ss_power_one_way_anova, ss_power_c_ancova, ci_c, ss_aipe_c

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sc, ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# Power for the contrast (Group 1 + Group 2) / 2 vs (Group 3 + Group 4) / 2
# with population contrast = 0.5, within-group sigma = 1, desired power = .80
ss_power_c(true_contrast = 0.5, c_weights = c(0.5, 0.5, -0.5, -0.5), true_sd = 1,
           desired_power = 0.80)

# Realized power for n = 30 per group
ss_power_c(true_contrast = 0.5, c_weights = c(0.5, 0.5, -0.5, -0.5), true_sd = 1,
           specified_n_per_group = 30)
```

---

ss_power_composite_ancova

Sample Size or Composite Power for a One-Way or Factorial ANCOVA

Description

Determine the necessary per-cell sample size to achieve a desired level of composite statistical power in a balanced analysis of covariance with a groups (a one-way design) or a factorial arrangement of factors, or, given a per-cell sample size, return the realized composite power. Composite power is the probability that every effect named in `effects` is statistically significant in the same study, the quantity a design must be planned against when its conclusion requires more than one result to hold at once.

Usage

```
ss_power_composite_ancova(
  factor_levels,
  effects,
  slopes = c("homogeneous", "heterogeneous"),
  true_means = NULL,
  true_sd = NULL,
  true_covariate_R2 = 0,
```

```
  n_covariates = 0,
  true_cor = NULL,
  true_sd_X = 1,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_cell = NULL
)
```

Arguments

factor_levels	Integer vector of the number of levels of each factor, one entry per factor (each at least 2). A single value a is a one-way design with a groups; c(2, 3) is a 2 by 3 factorial.
effects	A non-empty list naming the effects in the composite. For slopes = "homogeneous" each element has factors (the factor indices the effect spans) and, unless true_means is supplied, its effect size true_f or true_eta_squared_partial. For slopes = "heterogeneous" each element also has a type, one of "mean" (a factorial mean effect, the default), "covariate" (the average slope, spanning no factors), or "slope" (a factor-by-covariate slope heterogeneity). An optional label names the effect. See the forwarded functions for the exact grammar.
slopes	The covariate-slope model, "homogeneous" (one common slope, the default) or "heterogeneous" (the slope may differ across cells, making the covariate and slope-heterogeneity effects testable).
true_means	Optional array of population cell means (dimensions factor_levels, or a vector in array order) from which the mean effects' sizes are read; enables the mean-pattern figure.
true_sd	The common within-cell population standard deviation of the outcome, required with the population-values interface.
true_covariate_R2	For slopes = "homogeneous" only: proportion of the outcome's within-cell variance the covariate or covariates explain, in $[0, 1)$, which raises every effect's noncentrality through $f/\sqrt{1 - R^2}$. Defaults to 0.
n_covariates	For slopes = "homogeneous" only: number of covariates, each spending one residual degree of freedom. Must be positive when true_covariate_R2 is. Defaults to 0.
true_cor	For slopes = "heterogeneous" only: optional array of the population covariate-outcome correlation within each cell (dimensions factor_levels), which supplies the covariate and slope effects' sizes with the population-values interface.
true_sd_X	For slopes = "heterogeneous" only: population standard deviation of the covariate, the units the slopes and the figure are drawn in. Scale free, so it changes no power. Default 1.
desired_power	Desired composite statistical power (default 0.85). Used only when specified_n_per_cell is NULL.
alpha_level	Type I error rate for each individual F test (default 0.05), the per-test rate, not a rate for the composite event.

specified_n_per_cell
: Per-cell sample size (balanced); if supplied, the realized composite power is returned rather than a sample size planned.

Details

This is the general entry point for the ANCOVA composite. The two-group design has its own simpler interface in ss_power_composite_ancova_2group, stated through a single true_smd and one or two correlations; use that when there are exactly two groups. Use this function for more than two groups or for a factorial design. With no covariate, name the effects through ss_power_composite_anova instead.

Homogeneous or heterogeneous slopes. The slopes argument chooses the model. "homogeneous" (the default) assumes one common covariate slope across the cells; the covariate is a variance reducer and the composite is over the factorial mean effects. "heterogeneous" lets the slope differ across cells, which makes the average covariate slope and the factor-by-covariate slope heterogeneity into testable effects that can join the mean effects in the composite. The heterogeneous one-way case is the a-group generalization of the two-group ANCOVA composite: a group mean effect, the covariate effect, and the group-by-covariate slope heterogeneity.

This function is a thin dispatcher. slopes = "homogeneous" forwards to ss_power_composite_factorial_ancova and slopes = "heterogeneous" to ss_power_composite_factorial_ancova_het; see those for the full method, the exactness discussion, and the shape of the returned object. The population effects can be stated as effect sizes or as population values (cell means, and for heterogeneous slopes a covariate-outcome correlation per cell) with a common within-cell standard deviation, from which the plot() method draws the population pattern.

Value

The data.frame the forwarded planner returns, with term and value columns and the dmar_ss_power class for tidy and glance, plus the dmar_composite_power_factorial (homogeneous) or dmar_composite_power_facto (heterogeneous) class for plot(). See ss_power_composite_factorial_ancova for the row-by-row description. With slopes = "heterogeneous" the result carries the cell correlations as the attribute true_cor.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E. (2004). The persistence of underpowered studies in psychological research: Causes, consequences, and remedies. *Psychological Methods, 9*, 147–163.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on the analysis of covariance, Chapter 7 on factorial designs, and Chapter 3 on statistical power.)

See Also

ss_power_composite_ancova_2group for the two-group special case with the simple true_smd/true_cor interface; ss_power_composite_anova for the no-covariate design; ss_power_composite_factorial_ancova

and ss_power_composite_factorial_ancova_het, the planners this forwards to; ss_power_factorial_ancova for a single effect

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_se ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other composite power: ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem()

Examples

```
# A four-group (one-way) ANCOVA with heterogeneous slopes: the a-group
# generalization of the two-group composite. The conclusion needs the group
# mean effect, a covariate effect, and evidence that the covariate slope
# differs across the groups, so the design is planned against all three.
ss_power_composite_ancova(
  factor_levels = 4, slopes = "heterogeneous",
  effects = list(list(type = "mean",      factors = 1, true_f = 0.30),
                 list(type = "covariate",              true_f = 0.40),
                 list(type = "slope",      factors = 1, true_f = 0.20)),
  desired_power = 0.80)

# A 2 by 3 factorial ANCOVA with one common slope. Both main effects must
# hold; the covariate explains 25 percent of the within-cell variance.
ss_power_composite_ancova(
  factor_levels = c(2, 3),
  effects = list(list(factors = 1, true_f = 0.25),
                 list(factors = 2, true_f = 0.20)),
  true_covariate_R2 = 0.25, n_covariates = 1,
  desired_power = 0.80)

# The population effects can instead be a full pattern of cell means with a
# common within-cell SD; plot() then draws the mean pattern itself.
cell_means <- matrix(c(10, 12, 11,
                       13, 12, 16), nrow = 2, byrow = TRUE)
fit <- ss_power_composite_ancova(
  factor_levels = c(2, 3), true_means = cell_means, true_sd = 4,
  effects = list(list(factors = 1, label = "A"),
                 list(factors = 2, label = "B")),
  specified_n_per_cell = 30)
fit
plot(fit)

# A one-row broom summary of a plan.
generics::tidy(ss_power_composite_ancova(
  factor_levels = c(2, 3),
```

```
  effects = list(list(factors = 1, true_f = 0.25),
                 list(factors = 2, true_f = 0.20)),
  desired_power = 0.80))

# The two-group case matches the dedicated two-group planner. Here factor 1
# has two levels, so the heterogeneous one-way composite reproduces
# ss_power_composite_ancova_2group with one correlation per group.
ss_power_composite_ancova(
  factor_levels = 2, slopes = "heterogeneous",
  true_means = c(0, 0.5), true_cor = c(0.1, 0.5), true_sd = 1,
  effects = list(list(type = "mean", factors = 1),
                 list(type = "covariate"),
                 list(type = "slope", factors = 1)),
  specified_n_per_cell = 95)
ss_power_composite_ancova_2group(true_smd = 0.5, true_cor = c(0.1, 0.5),
                                 specified_n_per_group = 95)
```

`ss_power_composite_ancova_2group` *Sample Size or Composite Power for a Two-Group ANCOVA With a Covariate*

Description

Determine the necessary per-group sample size to achieve a desired level of composite statistical power in a two-group analysis of covariance, or, given a per-group sample size, return the realized composite power. Composite power is the probability that every effect named in `composite_terms` is statistically significant in the same study, which is the quantity a design has to be planned against when its conclusion requires more than one result to hold at once.

Usage

```
ss_power_composite_ancova_2group(
  true_smd = 0,
  true_cor = 0,
  true_sd = 1,
  true_sd_X = 1,
  composite_terms = c("group", "covariate", "group_by_covariate"),
  include_interaction = TRUE,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_group = NULL,
  directional = FALSE
)

## S3 method for class 'dmar_composite_power'
plot(x, ...)
```

Arguments

true_smd	Supposed standardized mean difference (Cohen's d) between the two groups at the mean of the covariate, standardized by true_sd: a value the researcher posits for the population, either a minimally important effect or a value believed to be true, never a sample estimate. Defaults to 0, which is no group effect and therefore power equal to alpha_level for that test.
true_cor	Within-group population correlation between the covariate and the outcome (ρ_g in the formulas below). Length 1 for the same correlation in both groups, in which case the two slopes are equal and the interaction is zero, or length 2 for one correlation per group, in which case the slopes differ and the interaction carries the difference. Each element must lie in (-1, 1). Defaults to 0.
true_sd	Within-group population standard deviation of the outcome: the same σ as in a one-way ANOVA on the outcome, the same in both groups, and the standardizer of true_smd. Defaults to 1, which puts true_smd and the coefficients on a standardized scale.
true_sd_X	Population standard deviation of the covariate. Defaults to 1. A correlation is scale free, so true_sd_X does not affect power; it sets the units the slopes and the figure are expressed in.
composite_terms	Character vector naming the effects that must all be statistically significant. Any subset of "group", "covariate", and "group_by_covariate". A single term returns that test's ordinary power. Defaults to all three.
include_interaction	Logical: whether the fitted model contains the group by covariate interaction. Defaults to TRUE. Dropping it returns one residual degree of freedom.
desired_power	Desired composite statistical power (default 0.85). Used only when specified_n_per_group is NULL.
alpha_level	Type I error rate for each individual test (default 0.05). This is the per-test rate, not a rate for the composite event.
specified_n_per_group	Per-group sample size (assumed balanced); if specified, the realized power is returned rather than a sample size planned.
directional	Logical: TRUE for one-sided tests, each in the direction of its own supposed effect, FALSE (default) for two-sided tests.
x	An object returned by ss_power_composite_ancova_2group.
...	Further arguments passed to the figure: cov_range, palette, group_labels, and show_power.

Details

This is the two-group special case, kept under its own name for the simple true_smd/true_cor interface it allows. For a one-way design with more than two groups, or a factorial design, use the general ss_power_composite_ancova.

The model is

$$Y = b_0 + b_{group}G + b_{cov}X + b_{group\times cov}GX + e.$$

The group effect, the covariate effect, and the group by covariate interaction can each be named in the composite, alone or in any combination.

Calling `plot` on the result draws the population effects the planning values describe. The figure needs only the effect sizes, and nothing is simulated to draw it.

Coding the group factor as -1/2 and +1/2 and centering the covariate makes each coefficient read directly: b_{group} is the difference between the group means at the covariate mean, b_{cov} is the average of the two within-group slopes, and $b_{group \times cov}$ is the difference between them. With balanced groups and a covariate whose distribution does not differ across them, the columns of the design are mutually orthogonal, so the three tests are orthogonal and the coefficient estimates are uncorrelated.

Orthogonal effects do not give independent tests. Every test divides by the same estimated error standard deviation, so an error estimate that lands low inflates all of the test statistics together. The tests are positively dependent, and composite power is strictly larger than the product of the marginal powers. Multiplying the marginal powers understates the composite; the gap closes as the residual degrees of freedom grow and the error estimate stabilizes. Composite power can never exceed the least powerful test in the set, so the weakest effect governs the design.

Conditional on the error estimate the tests are independent, which reduces the composite to a one-dimensional integral over the chi square distribution of that estimate. Adaptive quadrature evaluates the integral, so no data are simulated and the result is deterministic to quadrature precision.

Within group g the slope is $\rho_g \sigma / \sigma_X$ and the residual variance is $\sigma^2(1 - \rho_g^2)$, so correlations that differ across the groups give slopes that differ. The error variance the ANCOVA pools and estimates is the average of the two, $\sigma^2_{adj} = \sigma^2(1 - \bar{\rho^2})$ with $\bar{\rho^2}$ the mean of ρ_1^2 and ρ_2^2, which is the familiar $\sigma\sqrt{1-\rho^2}$ whenever the two correlations are equal in absolute value. Correlations that are not make the residual variance differ across the groups, and that has consequences the section on unequal residual variances below spells out.

The noncentralities are formed as $\sqrt{N} f$, the convention the rest of the `ss_power_*` family uses (see `ss_power_reg_coef`), and the residual degrees of freedom are $N - p - 1$. When the correlations are equal and the interaction is dropped, the group test reproduces `ss_power_c_ancova` with contrast weights `c(1, -1)`.

Two approximations are worth separating, because they have different causes. The first is the conditioning on the covariate, which is present at every set of planning values and is described next. The second appears only when the correlations differ in absolute value, and has its own section below.

All three noncentralities condition on the covariate, substituting the expected cross-product matrix for the expectation of its inverse. Inversion is convex, so the assumed sampling variance is too small and every power in the table is overstated, by an amount of order $1/N$. The group test is not exempt: b_{group} is the difference at the covariate mean, and the realized group covariate means are not exactly equal, so the group coefficient inherits their sampling variability even under balanced groups and a covariate independent of them. What the group test's noncentrality does not involve is `true_sd_X`, because a correlation is scale free; that is not the same as exactness. The Type I error rate is unaffected, so it is specifically power that is overstated, not the calibration of the tests. This is the same approximation, and the same convention, that `ss_power_c_ancova` and `ss_power_reg_coef` already use, and the correction `ss_power_c_ancova` names in its own documentation is one instance of it.

The size of that overstatement was measured against simulation for a two-term composite with equal correlations, which isolates the conditioning from everything else, at 200,000 replications per cell: about 2 points of composite power at n = 25 per group, under half a point at n = 50, and within

Monte Carlo error from n = 100 up. The constant grows with the number of tests in the composite, because each contributes the same approximation, so a three-term composite is worse than this at any given n. Treat these as the scale of the effect rather than a correction to apply: for planned samples of a few dozen per group the reported power is optimistic by a point or two, and for the sample sizes a three-term composite usually requires the effect has died away.

Because every test divides by the same error estimate, composite power is not monotone in n at the smallest residual degrees of freedom: an error estimate that lands low at one or two degrees of freedom inflates all of the statistics at once, so the composite there can exceed its value at slightly larger n. `necessary_n_per_group` (or `approximate_n_per_group`, see the section on unequal residual variances) is the smallest n attaining `desired_power`, which for a target below `alpha_level` need not be a size that every larger n also attains.

Value

A `data.frame` with `term` and `value` columns. The design result comes first, then the marginal power and noncentrality of each test in the composite, then rows echoing the planning values, so the assumptions the power was evaluated under travel with the result. The `tails` row is 2 for nondirectional tests and 1 for directional tests. The names of the composite terms, the implied coefficients, the per-group slopes, and σ_{adj} are carried as attributes rather than rows, keeping the `value` column numeric. Call `plot` on the result to draw the population effects.

When the two correlations differ in absolute value the powers are approximations, and the row names say so: `composite_power` is reported as `approximate_composite_power`, each `power_<term>` as `approximate_power_<term>`, and a planned size as `approximate_n_per_group` and `approximate_total_N`. The section on unequal residual variances explains why. An `approximate` attribute carries the same flag for a program to test without parsing row names. `tidy` and `glance` read both sets of names, so a relabeled table still summarizes to its sample size and its power, and `glance()` keeps the `approximate_` names on the columns it carries through. Nothing changes when the correlations are equal in absolute value, which is the exact case.

Functions

- `plot(dmar_composite_power)`: Draw the population effects a result was planned on, reached from a result already in hand. With `show_power = TRUE` (the default) the figure is annotated with the power the plan delivers.

Unequal Residual Variances When the Correlations Differ

Within group g the residual variance is $\tau_g^2 = \sigma^2(1 - \rho_g^2)$. Correlations that differ in absolute value therefore leave the two groups with different residual variances, and two things the composite integral assumes stop being true.

The pooled error is no longer one scaled chi square. The residual sum of squares is $\tau_1^2 Q_1 + \tau_2^2 Q_2$ with Q_1 and Q_2 independent chi square variables on $n - 2$ degrees of freedom each, a mixture of two scaled chi squares. Averaging the squared correlations gets its mean exactly right and its spread wrong: the mixture's variance exceeds that of the single scaled chi square the integral uses by $(n-2)(\tau_1^2 - \tau_2^2)^2$, which is zero exactly when the two residual variances agree.

The numerators stop being uncorrelated. The covariate coefficient is the average of the two within-group slopes and the interaction coefficient is their difference, and slopes estimated with different residual variances leave those two estimators correlated: $\mathrm{Cov}(\hat{b}_{cov}, \hat{b}_{group \times cov}) = (\tau_2^2 - \tau_1^2)/(2n\sigma_X^2)$,

again zero exactly when the residual variances agree. The tests are then not independent even given the error estimate, which is the step that reduced the composite to a one-dimensional integral in the first place.

What the function reports in that case is therefore the composite power of a design whose pooled error is a single scaled chi square with the right mean and whose tests are conditionally independent, which is a near neighbor of the design described but not that design. It is an approximation, and the output says so: every row carrying a power is renamed with an `approximate_` prefix, and a planned sample size is reported as `approximate_n_per_group` and `approximate_total_N` rather than as `necessary_n_per_group` and `total_N`, because it is the smallest n at which the approximation reaches `desired_power`, not an n known to attain it. The numbers are the same numbers; only the names change, and only in this case.

The sign of the error is not guaranteed. Two fixed-covariate simulations, run with the covariate values held to the population moments so that the conditioning approximation above plays no part, bracket it. With `true_smd = 0.30`, `true_cor = c(0, 0.9)` and $n = 8$ per group, the composite of the group and covariate effects is reported as 0.0752 against a simulated 0.0865 (Monte Carlo standard error 0.0002), so the report is conservative. With `true_smd = 2.50`, `true_cor = c(0, 0.95)` and $n = 6$ per group, the group effect alone is reported as 0.9990 against a simulated 0.9979 (Monte Carlo standard error 0.0001), so the report is optimistic. Both are deliberately severe: a correlation gap of 0.9 and a handful of cases per group. At a gap a covariate plausibly shows, `true_smd = 0.50` with `true_cor = c(0.1, 0.5)` and $n = 25$ per group, the composite of the group effect and the interaction is reported as 0.1515 against a simulated 0.1508 (Monte Carlo standard error 0.0006), a difference inside simulation error.

What to do with the number, then. Read it as an approximation whose accuracy degrades with the gap between the correlations and not with N, and confirm a design you intend to run by simulating it: draw each group's errors with its own residual standard deviation $\sigma\sqrt{1-\rho_g^2}$, fit the same model, and count the replications in which every test in the composite rejects. Equal absolute correlations need none of this. Two correlations of the same magnitude and opposite sign, `true_cor = c(0.5,-0.5)`, give a large interaction and still equal residual variances, so that design is exact and its rows keep the ordinary names.

Planning Without the Composite

Composite power is the right quantity only when the conclusion needs several results at once. When one effect carries the argument, DMAR already plans for it and this function is unnecessary.

For the group effect, `ss_power_c_ancova` is the planner: a two-group comparison is the contrast `c(1, -1)` on the adjusted means. Naming one term here reproduces it exactly, which is the check the tests assert:

```
ss_power_c_ancova(true_contrast = 0.5, c_weights = c(1, -1), true_sd = 1,
                  true_cor = 0.3, specified_n_per_group = 30)
ss_power_composite_ancova_2group(true_smd = 0.5, true_cor = 0.3,
                                 specified_n_per_group = 30,
                                 composite_terms = "group",
                                 include_interaction = FALSE)
```

Both return 0.5143. The interaction is dropped in the second call because `ss_power_c_ancova` plans for the model without it, and carrying a term the other function does not have would spend

a residual degree of freedom on nothing. With more than two groups, or a contrast other than a simple difference, ss_power_c_ancova is the only one of the two that applies.

For the design with no covariate at all, ss_power_smd is the planner, and comparing the two shows the power a covariate adds:

```
ss_power_smd(true_smd = 0.5, n_1 = 30)                 # no covariate
ss_power_c_ancova(true_contrast = 0.5, c_weights = c(1, -1),
                  true_sd = 1, true_cor = 0.5,
              specified_n_per_group = 30)              # covariate, true_cor = .5
```

Power rises from 0.4779 to 0.5942 because the covariate removes ρ^2 of the error variance, at the cost of one degree of freedom. That is the ANCOVA bargain, and it is worth making before reaching for a composite.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on analysis of covariance, and Chapter 3 on statistical power and the noncentral distributions the tests follow.)

See Also

ss_power_composite_ancova for the general one-way or factorial ANCOVA composite (of which this is the two-group case); ss_power_c_ancova for a single contrast on the adjusted means, which is the non-composite planner for this design; ss_power_smd for the two-group design with no covariate; ss_power_c for a contrast with no covariate; ss_power_reg_coef; ci_c_ancova and ci_sc_ancova for intervals on the adjusted means; ancova to fit the model the plan is for

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_s ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other composite power: ss_power_composite_ancova(), ss_power_composite_anova(), ss_power_composite_factor ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_s

Examples

```
# A covariate correlating 0.10 with the outcome in one group and 0.40 in the
# other. The correlations differ, so the slopes differ and there is an
# interaction to detect; nothing had to be assumed about that in advance.
# They also differ in absolute value, so the groups' residual variances
# differ, the powers are approximations, and the rows are named to say so.
```

```
ss_power_composite_ancova_2group(true_smd = 0.20, true_cor = c(0.10, 0.40),
                                 specified_n_per_group = 100)

# The covariate is nearly certain to be detected and the interaction is
# better than even, but the group effect is weak, and the composite of all
# three is far below any of them. No single marginal power reveals that.

# Equal correlations mean equal slopes, so the interaction is zero and its
# test rejects only at the Type I error rate. Asking for a composite that
# includes it therefore asks for something that cannot happen often.
ss_power_composite_ancova_2group(true_smd = 0.20, true_cor = 0.10,
                                 specified_n_per_group = 100)

# The composite of the two effects that are actually present.
ss_power_composite_ancova_2group(true_smd = 0.20, true_cor = 0.10,
                                 specified_n_per_group = 100,
                                 composite_terms = c("group", "covariate"))

# Per-group sample size for composite power of 0.80 on the group effect and
# the interaction together.
ss_power_composite_ancova_2group(true_smd = 0.50, true_cor = c(0.10, 0.50),
                         composite_terms = c("group", "group_by_covariate"),
                         desired_power = 0.80)

# Composite power is not the product of the marginal powers. The tests share
# one error estimate, so they are positively dependent and the composite is
# larger than the product. Multiplying would understate the design. These
# correlations differ in absolute value, so the rows carry the approximate_
# prefix; see the section on unequal residual variances for what that means.
plan <- ss_power_composite_ancova_2group(
  true_smd = 0.50, true_cor = c(0.10, 0.50), specified_n_per_group = 95,
  composite_terms = c("group", "group_by_covariate"))
plan$value[plan$term == "approximate_composite_power"]
prod(plan$value[plan$term %in% c("approximate_power_group",
                                 "approximate_power_group_by_covariate")])

# Correlations of equal magnitude and opposite sign leave the residual
# variances equal, so this design is exact and keeps the ordinary row names,
# even though the two slopes could hardly differ more.
exact <- ss_power_composite_ancova_2group(
  true_smd = 0.50, true_cor = c(0.40, -0.40), specified_n_per_group = 95,
  composite_terms = c("group", "group_by_covariate"))
exact$value[exact$term == "composite_power"]

# Draw the population effects a result was planned on. The figure needs only
# the effect sizes, and nothing is simulated to draw it.
plot(ss_power_composite_ancova_2group(true_smd = 0.20,
                                      true_cor = c(0.10, 0.40),
                                      specified_n_per_group = 100))

# Planning without the composite: one term returns that test's ordinary
# power, and reproduces ss_power_c_ancova once the interaction is dropped.
ss_power_composite_ancova_2group(true_smd = 0.50, true_cor = 0.30,
```

```
                                  specified_n_per_group = 30,
                                  composite_terms = "group",
                                  include_interaction = FALSE)
ss_power_c_ancova(true_contrast = 0.50, c_weights = c(1, -1), true_sd = 1,
                  true_cor = 0.30, specified_n_per_group = 30)

# What the covariate buys, against the same design with no covariate.
ss_power_smd(true_smd = 0.50, n_1 = 30)

# The broom verbs summarize the plan in one row.
generics::tidy(ss_power_composite_ancova_2group(
  true_smd = 0.50, true_cor = c(0.10, 0.50),
  composite_terms = c("group", "group_by_covariate"),
  desired_power = 0.80))
```

`ss_power_composite_anova`
Sample Size or Composite Power for a One-Way or Factorial ANOVA

Description

Determine the necessary per-cell sample size to achieve a desired level of composite statistical power in a balanced analysis of variance with a groups (a one-way design) or a factorial arrangement of factors, or, given a per-cell sample size, return the realized composite power. Composite power is the probability that every effect named in `effects` is statistically significant in the same study, the quantity a design must be planned against when its conclusion requires more than one result to hold at once. Each effect is a main effect or an interaction, tested by its own F test, and any subset of them can make up the composite.

Usage

```
ss_power_composite_anova(
  factor_levels,
  effects,
  true_means = NULL,
  true_sd = NULL,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_cell = NULL
)
```

Arguments

`factor_levels`	Integer vector of the number of levels of each factor, one entry per factor (each at least 2). A single value `a` is a one-way design with a groups; `c(2, 3)` is a 2 by 3 factorial.

`effects` A non-empty list naming the effects in the composite. Each element is a list with `factors` (a vector of factor indices into `factor_levels`: one index for a main effect, several for an interaction) and, unless `true_means` is supplied, exactly one of `true_f` (Cohen's f) or `true_eta_squared_partial`. An optional `label` names the effect in the output and the figure; the default label is the factor indices joined by `x`. The purported effect sizes are population values the researcher posits, never sample estimates. A field under any other name is refused.

`true_means` Optional array of population cell means whose dimensions are `factor_levels` (a matrix for two factors), or a numeric vector of length `prod(factor_levels)` in array order. When supplied, each named effect's Cohen's f is computed from the means and `true_sd`, and `plot()` draws the mean pattern.

`true_sd` The common within-cell population standard deviation of the outcome, required with `true_means` and used only there.

`desired_power` Desired composite statistical power (default 0.85). Used only when `specified_n_per_cell` is `NULL`.

`alpha_level` Type I error rate for each individual F test (default 0.05), the per-test rate, not a rate for the composite event.

`specified_n_per_cell` Per-cell sample size (balanced); if supplied, the realized composite power is returned rather than a sample size planned.

Details

The population effects can be stated two ways: as effect sizes, a Cohen's f or partial eta squared per effect, or as a full array of population cell means together with a common within-cell standard deviation, from which each named effect's f is read off the analysis of variance decomposition of the means. Supplying means lets the `plot()` method draw the mean pattern itself, so the size of the population effects is on the page.

If the design includes one or more covariates, use `ss_power_composite_ancova`.

Value

A `data.frame` with `term` and `value` columns: the `necessary_n_per_cell` (or the supplied `specified_n_per_cell`) and the total sample size it implies, `total_N`, the `composite_power`, the `df_error`, then for each effect its marginal `power_<label>`, purported `true_f_<label>`, numerator `df_<label>`, and `noncentral_f_parm_<label>`, followed by `cells` and `alpha_level`. The result carries the `dmar_ss_power` class for `tidy` and `glance`, and a `dmar_composite_power_factorial` class so `plot()` draws the figure.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E. (2004). The persistence of underpowered studies in psychological research: Causes, consequences, and remedies. *Psychological Methods, 9*, 147–163.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 7 on factorial designs and Chapter 3 on statistical power.)

See Also

ss_power_composite_ancova for the design with one or more covariates; ss_power_factorial_anova and ss_power_one_way_anova for a single effect

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sc, ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other composite power: ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem()

Examples

```
# A 2 by 3 factorial ANOVA whose conclusion needs both main effects, so the
# design is planned against their composite.
ss_power_composite_anova(
  factor_levels = c(2, 3),
  effects = list(list(factors = 1, true_f = 0.25),
                 list(factors = 2, true_f = 0.20)),
  desired_power = 0.80)

# Realized composite power at 40 per cell for a main effect and the
# interaction of a 2 by 2 design, sizes given as partial eta squared.
ss_power_composite_anova(
  factor_levels = c(2, 2),
  effects = list(list(factors = 1,       true_eta_squared_partial = 0.06),
                 list(factors = c(1, 2), true_eta_squared_partial = 0.04)),
  specified_n_per_cell = 40)

# The effects can instead be a full pattern of population cell means with a
# common within-cell SD; plot() then draws the mean pattern itself.
cell_means <- matrix(c(10, 12, 11,
                       13, 12, 16), nrow = 2, byrow = TRUE)
fit <- ss_power_composite_anova(
  factor_levels = c(2, 3), true_means = cell_means, true_sd = 4,
  effects = list(list(factors = 1, label = "A"),
                 list(factors = 2, label = "B")),
  specified_n_per_cell = 30)
fit
plot(fit)
```

`ss_power_composite_factorial_ancova` *Sample Size or Composite Power for a Factorial ANCOVA*

Description

Determine the necessary per-cell sample size to achieve a desired level of composite statistical power in a balanced factorial analysis of covariance, or, given a per-cell sample size, return the realized composite power. Composite power is the probability that every effect named in `effects` is statistically significant in the same study, the quantity a design must be planned against when its conclusion requires more than one result to hold at once. Each effect is a main effect or an interaction of the factorial design, tested by its own F test, and any subset of them can make up the composite.

Usage

```
ss_power_composite_factorial_ancova(
  factor_levels,
  effects,
  true_means = NULL,
  true_sd = NULL,
  true_covariate_R2 = 0,
  n_covariates = 0,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_cell = NULL
)

## S3 method for class 'dmar_composite_power_factorial'
plot(x, ...)
```

Arguments

`factor_levels`	Integer vector of the number of levels of each factor, one entry per factor (each at least 2). A `c(2, 3, 2)` argument is a 2 by 3 by 2 design.
`effects`	A non-empty list naming the effects in the composite. Each element is itself a list with `factors` (a vector of factor indices into `factor_levels`: one index for a main effect, several for an interaction) and exactly one of `true_f` (Cohen's f for that effect) or `true_eta_squared_partial`. An optional `label` names the effect in the output and the figure; the default label is the factor indices joined by `x` (for example `"1x2"` for the interaction of factors 1 and 2). Each element also carries the effect size, one of `true_f` or `true_eta_squared_partial`, unless `true_means` is supplied, in which case the sizes come from the means and no effect size is given here. The purported effect sizes are population values the researcher posits, never sample estimates, which is why the two fields carry `true_`. Every field of an element is named, and a field under any other name is

refused with an error of class `dmar_argument_invalid`, so that an effect size is never dropped without a word.

`true_means` Optional array of population cell means whose dimensions are `factor_levels` (a matrix for two factors), or a numeric vector of length `prod(factor_levels)` in array order (the first factor varying fastest). When supplied, each named effect's Cohen's f is computed from the means and `true_sd`, and `plot()` draws the mean pattern. The means are the population values the researcher posits, on the raw scale of the outcome.

`true_sd` The common within-cell population standard deviation of the outcome (the square root of the error variance), required with `true_means` and used only there. Cohen's f for an effect is the spread of its cell-mean component relative to `true_sd`.

`true_covariate_R2` Proportion of the outcome's within-cell variance the covariate or covariates explain, in $[0, 1)$. The covariate removes that fraction of the error variance, which raises every effect's noncentrality through $f/\sqrt{1 - R^2}$. Defaults to 0.

`n_covariates` Number of covariates, a non-negative integer. Each spends one residual degree of freedom. Must be positive when `true_covariate_R2` is. Defaults to 0.

`desired_power` Desired composite statistical power (default 0.85). Used only when `specified_n_per_cell` is `NULL`.

`alpha_level` Type I error rate for each individual F test (default 0.05). This is the per-test rate, not a rate for the composite event.

`specified_n_per_cell` Per-cell sample size, assumed balanced across cells; if supplied, the realized composite power is returned rather than a sample size planned.

`x` An object returned by `ss_power_composite_factorial_ancova` or `ss_power_composite_factorial_`

`...` Further arguments to the figure: `palette` (a palette name, default `"okabe_ito"`) and `title`.

Details

The population effects can be stated two ways: as effect sizes, a Cohen's f or partial eta squared per effect, or as a full array of population cell means together with a common within-cell standard deviation, from which each named effect's f is read off the analysis of variance decomposition of the means. Supplying means lets the `plot()` method draw the mean pattern itself, so the size of the population effects is on the page.

With no covariate (the defaults `true_covariate_R2 = 0` and `n_covariates = 0`) this is a factorial ANOVA; the wrapper `ss_power_composite_factorial_anova` is that case named directly.

In a balanced factorial design the effect sums of squares are mutually orthogonal, so the effects are uncorrelated. Orthogonal effects do not give independent tests: every F test divides by the same error mean square, so an error estimate that lands low inflates all of the test statistics together. The tests are positively dependent, and composite power is strictly larger than the product of the marginal powers, bounded above by the least powerful test in the set, so the weakest effect governs the design.

Conditional on the error estimate the tests are independent, which reduces the composite to a one dimensional integral over the chi square distribution of that estimate. Effect j has numerator degrees

of freedom $\prod(a-1)$ over the factors it spans and noncentrality Nf^2_{adj} with $f_{\text{adj}} = f/\sqrt{1-R^2}$ and N the total sample size, the same convention and covariate adjustment ss_power_factorial_ancova uses. The residual degrees of freedom are $N - \text{cells} - \text{covariates}$. Adaptive quadrature evaluates the integral, so nothing is simulated and the result is deterministic to quadrature precision. A single-effect composite reproduces the ordinary noncentral F power, and naming one effect reproduces ss_power_factorial_anova (or ss_power_factorial_ancova with a covariate) exactly.

Exactness. With no covariate the composite is exact to quadrature precision: the balanced factorial F tests are exactly noncentral F and exactly independent given the error estimate. A covariate introduces the one approximation ss_power_factorial_ancova already carries, treating true_covariate_R2 as a fixed reduction of the error variance rather than an estimated one; the departure is of order $1/N$ and is negligible at the sample sizes a multi-effect composite usually needs.

Because every test divides by the same error estimate, composite power is not strictly monotone in specified_n_per_cell at the smallest residual degrees of freedom. necessary_n_per_cell is the smallest per-cell size attaining desired_power.

Value

A data.frame with term and value columns: the necessary_n_per_cell (or the supplied specified_n_per_cell) and the total sample size it implies, total_N, the composite_power, the df_error, then for each effect its marginal power_<label>, purported true_f_<label>, numerator df_<label>, and noncentral_f_parm_<label>, followed by rows echoing true_covariate_R2, n_covariates, cells, and alpha_level. The result carries the dmar_ss_power class, so tidy and glance summarize the per-cell size and the composite power in broom convention, and a dmar_composite_power_factorial class so plot() draws the figure.

Functions

- plot(dmar_composite_power_factorial): Draw the purported population effect sizes a result was planned on, each annotated with its marginal power, and the composite power in the subtitle.

The figure

The plot() method draws the purported population values. When true_means were supplied it draws the mean pattern itself: a profile of the population cell means over the first factor, one line per level of the second, faceted by any further factors, with an error bar of plus or minus one within-cell standard deviation at each mean so the effect sizes read against the noise. When effect sizes were supplied instead, the cell means are not pinned (many mean patterns share one Cohen's f), so it draws the effect sizes: one lollipop per named effect at its partial eta squared, colored and labeled by that effect's marginal power. Either way the composite power is in the subtitle and nothing is simulated. Requires **ggplot2**.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E. (2004). The persistence of underpowered studies in psychological research: Causes, consequences, and remedies. *Psychological Methods, 9*, 147–163.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 7 on factorial designs, Chapter 9 on the analysis of covariance, and Chapter 3 on statistical power.)

See Also

ss_power_composite_factorial_anova for the no-covariate case; ss_power_composite_ancova_2group for the two-group ANCOVA composite of the group effect, the covariate effect, and their interaction; ss_power_factorial_ancova and ss_power_factorial_anova for a single effect

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_se ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other composite power: ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_a ss_power_composite_sem()

Examples

```
# A 2 by 3 factorial ANCOVA. The conclusion needs both main effects to hold,
# so the design is planned against their composite. The covariate explains
# 25 percent of the within-cell variance (one covariate).
ss_power_composite_factorial_ancova(
  factor_levels = c(2, 3),
  effects = list(list(factors = 1, true_f = 0.25),
                 list(factors = 2, true_f = 0.20)),
  true_covariate_R2 = 0.25, n_covariates = 1,
  desired_power = 0.80)

# Realized composite power at 40 per cell for a main effect and the
# interaction of a 2 by 2 design, effect sizes given as partial eta squared.
ss_power_composite_factorial_ancova(
  factor_levels = c(2, 2),
  effects = list(list(factors = 1,        true_eta_squared_partial = 0.06),
                 list(factors = c(1, 2),  true_eta_squared_partial = 0.04)),
  specified_n_per_cell = 40)

# Naming one effect reproduces the single-effect planner exactly.
ss_power_composite_factorial_ancova(
  factor_levels = c(2, 3), effects = list(list(factors = 2, true_f = 0.25)),
  specified_n_per_cell = 20)
ss_power_factorial_anova(factor_levels = c(2, 3), effect_indices = 2,
```

```
                          true_f = 0.25, specified_n_per_cell = 20)

# The composite is not the product of the marginal powers. The tests share
# one error estimate, so they are positively dependent and the composite is
# the larger of the two.
plan <- ss_power_composite_factorial_ancova(
  factor_levels = c(2, 2, 3),
  effects = list(list(factors = 1, true_f = 0.30, label = "A"),
                 list(factors = c(1, 3), true_f = 0.25, label = "AxC")),
  specified_n_per_cell = 15)
plan$value[plan$term == "composite_power"]
prod(plan$value[plan$term %in% c("power_A", "power_AxC")])

# A one-row broom summary of the plan.
generics::tidy(ss_power_composite_factorial_ancova(
  factor_levels = c(2, 3),
  effects = list(list(factors = 1, true_f = 0.25),
                 list(factors = 2, true_f = 0.20)),
  desired_power = 0.80))

# The figure of the purported population values, annotated with the power a
# given sample size delivers.
plot(ss_power_composite_factorial_ancova(
  factor_levels = c(2, 3),
  effects = list(list(factors = 1, true_f = 0.25),
                 list(factors = 2, true_f = 0.20)),
  specified_n_per_cell = 30))

# The effects can instead be stated as a full pattern of population cell means
# with a common within-cell SD. Rows are the 2-level factor, columns the
# 3-level factor. plot() then draws the mean pattern itself.
cell_means <- matrix(c(10, 12, 11,
                       13, 12, 16), nrow = 2, byrow = TRUE)
fit <- ss_power_composite_factorial_ancova(
  factor_levels = c(2, 3), true_means = cell_means, true_sd = 4,
  effects = list(list(factors = 1, label = "A"),
                 list(factors = 2, label = "B")),
  specified_n_per_cell = 30)
fit
plot(fit)
```

`ss_power_composite_factorial_ancova_het`
Composite Power for a Factorial ANCOVA With Heterogeneous Slopes

---

Description

Determine the necessary per-cell sample size, or the realized composite power at a supplied per-cell size, for a balanced factorial analysis of covariance in which the covariate's slope may differ

across the cells. When the slopes differ, the covariate main effect (the average slope) and the factor-by-covariate slope heterogeneity are themselves testable effects, and any of them, together with the factorial mean effects, can make up the composite. This is the factorial generalization of ss_power_composite_ancova_2group, whose two-group model with a correlation per group is the one-factor, two-level case here.

Usage

```
ss_power_composite_factorial_ancova_het(
  factor_levels,
  effects,
  true_means = NULL,
  true_cor = NULL,
  true_sd = NULL,
  true_sd_X = 1,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_cell = NULL
)

## S3 method for class 'dmar_composite_power_factorial_het'
plot(x, ...)
```

Arguments

factor_levels	Integer vector of the number of levels of each factor (each at least 2).
effects	A non-empty list naming the effects in the composite. Each element has a type, one of "mean" (a factorial main effect or interaction on the cell means; the default), "covariate" (the average covariate slope), or "slope" (a factor-by-covariate slope heterogeneity). A "mean" or "slope" effect also gives factors, the factor indices it spans; a "covariate" effect spans none. Each element carries its effect size, true_f or true_eta_squared_partial, unless the population values (true_means and true_cor) are supplied, in which case the sizes come from them. An optional label names the effect; defaults are the factor indices joined by x for a mean effect, "covariate", and "cov_x_<factors>" for a slope effect. A field under any other name is refused with an error of class dmar_argument_invalid.
true_means	Optional array of population cell means (dimensions factor_levels), needed when a "mean" effect is in the composite. Supplies the mean effects' sizes.
true_cor	Optional array of the population covariate-outcome correlation within each cell (dimensions factor_levels, each in $(-1, 1)$); the correlation in cell c is ρ_c in the formulas below, and $\bar{\rho^2}$ is the mean of their squares. The cell slopes it implies supply the covariate and slope effects' sizes, and its spread across cells sets the slope heterogeneity. Required with the population-values interface, since the pooled error depends on every cell's correlation.
true_sd	The common within-cell population standard deviation of the outcome (before adjustment), σ in the formulas below; required with the population-values interface.

`true_sd_X` Population standard deviation of the covariate. Default 1. A correlation is scale free, so `true_sd_X` does not change any power; it sets the units the slopes and the figure are drawn in.

`desired_power` Desired composite statistical power (default 0.85). Used only when `specified_n_per_cell` is `NULL`.

`alpha_level` Type I error rate for each individual test (default 0.05).

`specified_n_per_cell` Per-cell sample size (balanced); if supplied, the realized composite power is returned rather than a sample size planned.

`x` An object returned by `ss_power_composite_factorial_ancova_het`.

`...` Further arguments to the figure: `palette`, `title`, and, for the regression-line figure, `cov_range` (covariate range in SDs either side of the mean, default 2).

Details

Kept separate from `ss_power_composite_factorial_ancova`, which assumes one common slope and treats the covariate only as a variance reducer. Use this function when the covariate slope is expected to differ across conditions, or when the test of that difference is part of the design.

The model fits the factorial mean structure, the covariate, and every factor-by-covariate slope term, so it has $2 \times$ cells parameters and residual degrees of freedom $N - 2\,\text{cells}$. Under balance and a covariate with a common distribution across the cells, those terms are mutually orthogonal, so the tests share only the pooled residual and the composite is the shared-error integral of `ss_power_composite_factorial_ancova`. A "mean" effect on factor set S has numerator df $\prod(a-1)$ and is read from the cell means against the pooled adjusted error $\sigma^2(1-\bar{\rho}^2)$; a "covariate" effect is the grand slope on 1 df; a "slope" effect on S is the slope heterogeneity across those factors, with the same df as the matching mean effect, read from the cell slopes.

The two approximations are those of `ss_power_composite_ancova_2group`. The covariate-related tests condition on the covariate cross-products, overstating power by an amount of order $1/N$, which is negligible past a few dozen per cell. Separately, correlations that differ in absolute value across cells make the pooled error a mixture of scaled chi squares rather than the single one the integral assumes; that one does not shrink with *N* and is the subject of the section below.

Value

A `data.frame` with `term` and `value` columns: the `necessary_n_per_cell` (or the supplied `specified_n_per_cell`) and the total sample size it implies, `total_N`, the `composite_power`, the `df_error`, then for each effect its marginal `power_<label>`, purported `true_f_<label>`, numerator `df_<label>`, and `noncentral_f_parm_<label>`, followed by `cells` and `alpha_level`. Carries the `dmar_ss_power` class for `tidy` / `glance` and a `dmar_composite_power_factorial_het` class for `plot()`.

When the supplied cell correlations differ in absolute value the powers are approximations and the row names say so, with an `approximate_` prefix on every power and on a planned sample size; see the section on unequal residual variances. An `approximate` attribute carries the same flag for a program to test without parsing row names, and `tidy` and `glance` read both sets of names, so a relabeled table still summarizes to its per-cell size and its composite power.

Functions

- `plot(dmar_composite_power_factorial_het)`: Draw the purported population values: the per-cell regression lines when population values were supplied, or the effect size lollipop otherwise.

Unequal Residual Variances When the Cell Correlations Differ

In cell c the residual variance is $\tau_c^2 = \sigma^2(1 - \rho_c^2)$, so cell correlations that differ in absolute value leave the cells with different residual variances. Averaging the squared correlations, which is what $\sigma^2(1 - \bar{\rho^2})$ does, gets the expected pooled error right and its distribution wrong: the residual sum of squares is $\sum_c \tau_c^2 Q_c$ with the Q_c independent chi square variables, a mixture of scaled chi squares with the same mean and a larger spread than the single scaled chi square the shared-error integral integrates over. The numerators are affected too. The covariate effect is the average cell slope and a slope effect is a contrast among the cell slopes, and cell slopes estimated with different residual variances give estimators of those two that are correlated, so the tests are not independent even given the error estimate.

Every power the function reports is then an approximation, and the output says so. `composite_power` is reported as `approximate_composite_power`, each `power_<label>` as `approximate_power_<label>`, and a planned sample size as `approximate_n_per_cell` and `approximate_total_N`, because that size is the smallest one at which the approximation reaches `desired_power` rather than a size known to attain it. The numbers do not change; the names do, and only in this case. Equal absolute cell correlations, including cells whose correlations share a magnitude and differ in sign, are the exact case and keep the ordinary names.

The error has no guaranteed sign, and it grows with the spread of the cell correlations rather than shrinking with *N*. Treat the number as an approximation and confirm a design you intend to run by simulating it: draw each cell's errors with its own residual standard deviation $\sigma\sqrt{1 - \rho_c^2}$, fit the same model, and count the replications in which every effect in the composite is significant. The two-group help page reports the size of the departure over a range of correlation gaps.

The effect size interface is a special case worth naming. Supplying `true_f` or `true_eta_squared_partial` for each effect states the effects directly and says nothing about the cell correlations, so the function has nothing to detect unequal residual variances from and labels the table exact. If the design those effect sizes came from has cell correlations that differ in absolute value, the same approximation applies and the labels will not tell you; supply `true_cor` instead when you want the function to keep track of it.

The figure

When the population values are supplied, `plot()` draws the population regression line in each cell over the covariate, so heterogeneous slopes show as lines of different angle and mean effects as vertical separation, colored by the first factor and faceted by any others. When effect sizes are supplied instead, it draws the effect size lollipop of `ss_power_composite_factorial_ancova`. Either way the composite power is in the subtitle. Requires **ggplot2**.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on the analysis of covariance and heterogeneity of regression, and Chapter 7 on factorial designs.)

See Also

ss_power_composite_factorial_ancova for the common-slope version; ss_power_composite_ancova_2group for the two-group case

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other composite power: ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_anova(), ss_power_composite_sem()

Examples

```
# A 2 by 2 design whose conclusion needs a main effect and evidence that the
# covariate slope differs across the first factor. Effect sizes stated
# directly: the main effect (mean), the average covariate effect, and the
# factor 1 by covariate slope heterogeneity.
ss_power_composite_factorial_ancova_het(
  factor_levels = c(2, 2),
  effects = list(list(type = "mean",      factors = 1,  true_f = 0.25),
                 list(type = "covariate",               true_f = 0.40),
                 list(type = "slope",      factors = 1,  true_f = 0.20)),
  desired_power = 0.80)

# The same design from population values: cell means, a covariate-outcome
# correlation per cell (they differ across factor 1, so the slopes do), and a
# common within-cell SD. plot() then draws the per-cell regression lines.
cell_means <- matrix(c(10, 12,
                       11, 13), nrow = 2, byrow = TRUE)
cell_rho   <- matrix(c(0.55, 0.55,
                       0.15, 0.20), nrow = 2, byrow = TRUE)
fit <- ss_power_composite_factorial_ancova_het(
  factor_levels = c(2, 2), true_means = cell_means, true_cor = cell_rho,
  true_sd = 4, true_sd_X = 2,
  effects = list(list(type = "mean",      factors = 1),
                 list(type = "covariate"),
                 list(type = "slope",      factors = 1)),
  specified_n_per_cell = 40)
fit
plot(fit)
```

```
# The one-factor, two-level case is the two-group composite ANCOVA.
ss_power_composite_factorial_ancova_het(
  factor_levels = 2,
  true_means = c(-0.25, 0.25), true_cor = c(0.1, 0.5), true_sd = 1,
  effects = list(list(type = "mean", factors = 1),
                 list(type = "slope", factors = 1)),
  specified_n_per_cell = 100)
```

`ss_power_composite_factorial_anova` *Sample Size or Composite Power for a Factorial ANOVA*

Description

Determine the necessary per-cell sample size to achieve a desired level of composite statistical power in a balanced factorial analysis of variance, or, given a per-cell sample size, return the realized composite power. Composite power is the probability that every effect named in `effects` is statistically significant in the same study, the quantity a design must be planned against when its conclusion requires more than one result to hold at once.

Usage

```
ss_power_composite_factorial_anova(
  factor_levels,
  effects,
  true_means = NULL,
  true_sd = NULL,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_cell = NULL
)
```

Arguments

`factor_levels`	Integer vector of the number of levels of each factor, one entry per factor (each at least 2). A `c(2, 3, 2)` argument is a 2 by 3 by 2 design.
`effects`	A non-empty list naming the effects in the composite; see `ss_power_composite_factorial_ancova` for the format. Each element gives the `factors` an effect spans and its `true_f` or `true_eta_squared_partial`, unless `true_means` is supplied.
`true_means`	Optional array of population cell means (dimensions `factor_levels`), or a numeric vector of length `prod(factor_levels)` in array order, from which each effect's Cohen's f is read given `true_sd`. When supplied, `plot()` draws the mean pattern. See `ss_power_composite_factorial_ancova`.
`true_sd`	The common within-cell standard deviation of the outcome, required with `true_means` and used only there.

`desired_power` Desired composite statistical power (default 0.85). Used only when `specified_n_per_cell` is `NULL`.

`alpha_level` Type I error rate for each individual F test (default 0.05), the per-test rate rather than a rate for the composite event.

`specified_n_per_cell` Per-cell sample size, assumed balanced; if supplied, the realized composite power is returned rather than a sample size planned.

Details

This is the no-covariate case of `ss_power_composite_factorial_ancova`, named directly. It does not take a covariate; when a covariate belongs in the model, use `ss_power_composite_factorial_ancova`, which raises every effect's power for the variance the covariate explains. With no covariate the composite is exact to quadrature precision, since the balanced factorial F tests are exactly noncentral F and exactly independent given the shared error estimate.

See `ss_power_composite_factorial_ancova` for the model, the one-dimensional integral that evaluates the composite over the shared error estimate, and the figure the `plot()` method draws. Naming one effect reproduces `ss_power_factorial_anova` exactly.

Value

A `data.frame` with `term` and `value` columns, as `ss_power_composite_factorial_ancova` returns but with `true_covariate_R2` 0 and `n_covariates` 0. It carries the `dmar_ss_power` class for `tidy` / `glance` and the `dmar_composite_power_factorial` class for `plot()`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E. (2004). The persistence of underpowered studies in psychological research: Causes, consequences, and remedies. *Psychological Methods, 9*, 147–163.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 7 on factorial designs and Chapter 3 on statistical power.)

See Also

`ss_power_composite_factorial_ancova` for the version that admits a covariate; `ss_power_factorial_anova` for a single effect

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_sem()`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`,

ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other composite power: ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova ss_power_composite_sem()

Examples

```
# A 2 by 3 factorial ANOVA whose conclusion needs both main effects. Plan
# against their composite, not against either one alone.
ss_power_composite_factorial_anova(
  factor_levels = c(2, 3),
  effects = list(list(factors = 1, true_f = 0.25),
                 list(factors = 2, true_f = 0.20)),
  desired_power = 0.80)

# Realized composite power at 25 per cell for a main effect and the
# three-way interaction of a 2 by 2 by 2 design.
ss_power_composite_factorial_anova(
  factor_levels = c(2, 2, 2),
  effects = list(list(factors = 1,          true_f = 0.30, label = "A"),
                 list(factors = c(1, 2, 3), true_f = 0.25, label = "AxBxC")),
  specified_n_per_cell = 25)

# Naming one effect reproduces the single-effect planner exactly.
ss_power_composite_factorial_anova(
  factor_levels = c(2, 3), effects = list(list(factors = 2, true_f = 0.25)),
  specified_n_per_cell = 20)
ss_power_factorial_anova(factor_levels = c(2, 3), effect_indices = 2,
                         true_f = 0.25, specified_n_per_cell = 20)

# The figure of the purported population effect sizes.
plot(ss_power_composite_factorial_anova(
  factor_levels = c(2, 3),
  effects = list(list(factors = 1, true_f = 0.25),
                 list(factors = 2, true_f = 0.20)),
  specified_n_per_cell = 30))

# Stating the effects as population cell means with a common within-cell SD.
# plot() then draws the mean pattern, with error bars of one SD.
cell_means <- matrix(c(10, 12, 11,
                       13, 12, 16), nrow = 2, byrow = TRUE)
plot(ss_power_composite_factorial_anova(
  factor_levels = c(2, 3), true_means = cell_means, true_sd = 4,
  effects = list(list(factors = 1, label = "A"),
                 list(factors = 2, label = "B")),
  specified_n_per_cell = 30))
```

ss_power_composite_sem *Sample Size or Composite Power for a Set of SEM Parameters*

Description

Determine the necessary sample size for a structural equation model study so that every parameter of interest is statistically significant in the same study with a desired probability, or, given a sample size, return that probability. Composite power is the probability that all of the named parameters are significant jointly, the quantity a design must be planned against when its conclusion requires more than one result to hold at once: a study can have adequate power for each hypothesis on its own and still be underpowered for the conclusion that rests on all of them together (Maxwell, 2004). The parameters of interest are any labeled parameters of a lavaan analysis model, structural paths, loadings, covariances, or quantities defined with `:=` such as an indirect effect, and any subset of them can make up the composite.

Usage

```
ss_power_composite_sem(
  model,
  true_cov_matrix = NULL,
  true_model = NULL,
  true_mean = NULL,
  parameters = NULL,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_N = NULL,
  G = 1000,
  seed = NULL,
  ...
)
```

Arguments

`model`	A single character string giving the free analysis model in lavaan model syntax (see `model.syntax`), the model that would be fit to the data. Each parameter of interest must carry a parameter label so it can be referred to by name, for example `"f2 ~ b*f1"` labels the structural path `b`, and `"ab := a*b"` defines an indirect effect from the labeled paths `a` and `b`.
`true_cov_matrix`	Population covariance matrix of the observed variables, with row and column names matching the observed variables in `model`. It is typically obtained from a fully fixed population model via `cov_sem`. Supply exactly one of `true_cov_matrix` or `true_model`.
`true_model`	A single character string giving the population model in lavaan model syntax with every parameter fixed to its population value, from which `cov_sem` derives

`true_cov_matrix` (and the population means, when the model has a mean structure). This is where the purported population values of the parameters of interest are chosen; they are values the researcher posits (from theory, prior studies, or pilot data), never sample estimates. Supply exactly one of `true_cov_matrix` or `true_model`.

`true_mean` Optional population means of the observed variables, used with `true_cov_matrix`: a named numeric vector with one entry per observed variable, or an unnamed vector in the row order of `true_cov_matrix`. The default `NULL` is zero means. Means matter only when the analysis model has a mean structure (an intercept term such as `s ~ 1`, as in a latent growth curve model); when `true_model` is supplied its mean structure provides the means and `true_mean` must not also be given.

`parameters` Character vector of the parameter labels that make up the composite. The default `NULL` uses every user-labeled parameter in `model`, in order of appearance, so labeling exactly the parameters of interest is the simplest way to state the set.

`desired_power` Desired composite statistical power (default 0.85). Used only when `specified_N` is `NULL`.

`alpha_level` Type I error rate for each individual two-sided Wald z test (default 0.05), the per-test rate, not a rate for the composite event.

`specified_N` Total sample size N; if supplied, the realized composite power at that N is returned rather than a sample size planned.

`G` Number of converged Monte Carlo replications per evaluated sample size (default 1000). The simulation error of each estimated power is about $\sqrt{p(1-p)/G}$; raise `G` for a sharper answer.

`seed` Optional integer seed for reproducibility. The default `NULL` uses the current state of the random number generator; a supplied seed is set internally and the prior state restored on exit.

`...` Additional arguments passed to `sem`, both when the population values are resolved and for every Monte Carlo fit (for example `std.lv = TRUE` or `missing = "listwise"`). A robust estimator such as `"MLM"` cannot be used here: the same arguments reach the setup fit, which is always from summary statistics, and lavaan refuses a robust estimator there. The Monte Carlo data are multivariate normal by construction, so a robust estimator would buy nothing. Every name given here must be an argument or an option of **lavaan** (see `lavOptions`); any other name, and any argument without a name, is refused before a model is fitted, with a condition of class `dmar_argument_invalid` that gives the name as it was typed

Details

Analytic sample size planning methods in SEM exist for a single targeted parameter (Satorra & Saris, 1985; Lai & Kelley, 2011) or for overall model fit (MacCallum, Browne, & Sugawara, 1996; `ss_power_sem`), but most studies state several hypotheses and support their conclusion only when all of them hold. This function plans for that case by a priori Monte Carlo simulation (Muthén & Muthén, 2002; Maxwell, Kelley, & Rausch, 2008): for a candidate N, `G` data sets are drawn from the multivariate normal population with covariance matrix `true_cov_matrix`, the analysis model is fit

to each, and each parameter of interest is tested with its two-sided Wald z test at `alpha_level`. The proportion of replications in which every parameter is significant estimates the composite power, and the per-parameter proportions estimate the marginal powers. Because the estimates share one fitted model, the tests are dependent; the simulation reflects that dependence exactly, at the stated *N*, with no asymptotic shortcut.

The composite event is contained in each marginal event, so composite power is at most the smallest marginal power: the weakest parameter governs the design, and the marginal `power_<label>` rows show which parameter that is.

When `specified_N` is `NULL` the necessary sample size is searched for. The search starts where the product of the marginal Wald powers (an independence approximation computed from the asymptotic variances, spent before any simulation) reaches `desired_power`, brackets the crossing geometrically, and bisects to adjacent integers, each candidate evaluated with its own `G` replications. A planning call therefore fits the analysis model several thousand times at the default `G`, and even at the smallest admissible `G` the search runs for several seconds, so the example below evaluates a stated `specified_N`, which is the cheap half of the method. A planning call is the same call with `specified_N` left out and `desired_power` stated or left at its default, and the first row of the result is then `necessary_N` rather than `specified_N`. The vignette `vignette("composite_sem_planning", package = "DMAR")` works through the planning calls for a mediation model and a latent growth curve model, with reference values computed at `G = 10000`.

Value

A `data.frame` with `term` and `value` columns: the `necessary_N` (or supplied `specified_N`), the `composite_power` and its simulation standard error `composite_power_mc_se`, then for each parameter its marginal `power_<label>` and purported `true_<label>` value under the analysis model, followed by `alpha_level`, the requested `replications`, the `converged_replications` the summary is based on, and, when a size was planned, the `desired_power`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize the sample size and the composite power in broom convention.

Monte Carlo Precision

Each reported power is a proportion of `G` replications, with simulation standard error about $\sqrt{p(1-p)/G}$; the `composite_power_mc_se` row reports it for the composite. The necessary sample size inherits that uncertainty: near the target the power curve is flat enough that neighboring *N* are separated by less than the simulation error, so repeated calls with different seeds return slightly different sizes. Raising `G` narrows the spread; reporting the seed makes a plan reproducible. A proportion of `G` replications takes only the values $0, 1/G, \ldots, 1$, so a `desired_power` above $1 - 1/G$ is refused with a message saying how large `G` must be for that target; the same resolution guard applies to `ss_aipe_composite_sem`'s `assurance`.

Note

A replication whose fit does not converge, or converges without a usable standard error for some parameter of interest, is discarded and fresh data are drawn, up to `20 * G` attempts per evaluated sample size; the reported powers condition on convergence. A replication whose likelihood has no maximum (an error variance running off below zero while the likelihood is flat or still rising; see `fa_bifactor`), which **lavaan** reports as converged where its optimizer stopped, is discarded the

same way, and an analysis model with no maximum at the population moments stops the function with an error of class `dmar_no_maximum` that prints the **lavaan** code of that fit. When fewer than `G` replications converge within the cap, a single warning is issued and the summary is based on the converged replications (their count is the `converged_replications` row). Frequent nonconvergence at small *N* is itself design information: a sample size at which the model rarely converges is too small in a sense that precedes power.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2011). Accuracy in parameter estimation for targeted effects in structural equation modeling: Sample size planning for narrow confidence intervals. *Psychological Methods, 16*(2), 127–148. doi:10.1037/a0021764

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130–149. doi:10.1037/1082989X.1.2.130

Maxwell, S. E. (2004). The persistence of underpowered studies in psychological research: Causes, consequences, and remedies. *Psychological Methods, 9*(2), 147–163. doi:10.1037/1082989X.9.2.147

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on statistical power.)

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Muthén, L. K., & Muthén, B. O. (2002). How to use a Monte Carlo study to decide on sample size and determine power. *Structural Equation Modeling, 9*(4), 599–620. doi:10.1207/S15328007SEM0904_8

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. doi:10.18637/jss.v048.i02

Satorra, A., & Saris, W. E. (1985). Power of the likelihood ratio test in covariance structure analysis. *Psychometrika, 50*(1), 83–90.

See Also

`ss_aipe_composite_sem` for the same set of parameters planned for accuracy in parameter estimation (AIPE) instead of significance; `cov_sem` for deriving `true_cov_matrix` from a fully fixed population model; `ss_power_sem` for overall model fit; `ss_aipe_sem_path` for a single targeted path; `ss_power_composite_anova` and its siblings for composite power in ANOVA and ANCOVA designs, where the composite is evaluated by quadrature rather than simulation.

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`,

ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other composite power: ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova
ss_power_composite_factorial_anova()

Examples

```
# A three-factor model whose conclusion rests on three structural paths
# at once: f1 predicting f2, f2 predicting f3, and f1 predicting f3
# directly. The population model fixes every parameter to its purported
# population value.
true_model <- "
  f1 =~ 1*y1 + 0.8*y2 + 0.8*y3
  f2 =~ 1*y4 + 0.8*y5 + 0.8*y6
  f3 =~ 1*y7 + 0.8*y8 + 0.8*y9
  f2 ~ 0.4*f1
  f3 ~ 0.3*f2 + 0.25*f1
  f1 ~~ 1*f1
  f2 ~~ 0.84*f2
  f3 ~~ 0.8*f3
  y1 ~~ 0.5*y1; y2 ~~ 0.5*y2; y3 ~~ 0.5*y3
  y4 ~~ 0.5*y4; y5 ~~ 0.5*y5; y6 ~~ 0.5*y6
  y7 ~~ 0.5*y7; y8 ~~ 0.5*y8; y9 ~~ 0.5*y9
"

# The analysis model is free; the labels name the parameters of interest.
analysis_model <- "
  f1 =~ y1 + y2 + y3
  f2 =~ y4 + y5 + y6
  f3 =~ y7 + y8 + y9
  f2 ~ a*f1
  f3 ~ b*f2 + c*f1
"

# Realized composite power at N = 200. The probability that all three
# paths come out significant in the same study is lower than the marginal
# power of any one of them: the composite event sits inside each marginal
# event, so the weakest parameter governs the design. G = 10 keeps the
# example quick; a reported plan deserves the default G = 1000 or more.
ss_power_composite_sem(model = analysis_model, true_model = true_model,
                       specified_N = 200, G = 10, seed = 113)
```

---

ss_power_contrast *Sample Size and Statistical Power for a Contrast in a Fixed-Effects ANOVA*

Description

Determine the necessary per-group sample size for a contrast in a one-way fixed-effects ANOVA so as to achieve a desired level of statistical power, or, alternatively, compute the achieved power for a given sample size. The contrast is specified by a vector of weights and the population means (or a population contrast value) along with the within-group variance.

Usage

```
ss_power_contrast(
  c_weights,
  true_means = NULL,
  true_var_error = NULL,
  true_contrast = NULL,
  desired_power = 0.85,
  alpha_level = 0.05,
  directional = FALSE,
  specified_n_per_group = NULL,
  print_iter = FALSE
)
```

Arguments

c_weights	Vector of contrast weights. Required to satisfy sum(c_weights) == 0, sum(c_weights[c_weights > 0]) == 1, and sum(c_weights[c_weights < 0]) == -1; that is, write the contrast in normalized fractional form so that the positive coefficients sum to 1 and the negative coefficients sum to -1. The contrast estimate is then directly interpretable as a (weighted) mean difference.
true_means	Vector of population group means, of length length(c_weights). Either true_means or true_contrast must be supplied (but not both).
true_var_error	Population within-group variance (σ^2); must be a positive number.
true_contrast	Optional. Directly specify the population contrast value $\psi = \sum_j c_j \mu_j$ as a single number. Use this when the means themselves are not of interest, only the contrast value.
desired_power	Target statistical power for the test of the contrast (default 0.85). Used only when specified_n_per_group is NULL.
alpha_level	Type I error rate (default 0.05).
directional	Logical. FALSE (default) yields a two-sided test. TRUE yields a one-sided test in the direction of the population contrast.
specified_n_per_group	Optional per-group sample size at which to evaluate power. May be a scalar (the common per-group size, equal across groups) or a numeric vector of length length(c_weights) giving each group's size. When NULL (default), the function instead solves for the smallest per-group *n* achieving desired_power.
print_iter	If TRUE, print the trial n and the corresponding power as the iterative search proceeds.

Details

Let $\psi = \sum_j c_j \mu_j$ be the population contrast. Under the usual fixed-effects ANOVA model with common within-group variance σ^2 and per-group sizes n_j, the standard error of the contrast estimate is

$$\mathrm{SE}_{\hat{\psi}} = \sqrt{\sigma^2 \sum_j c_j^2 / n_j}\,,$$

the test statistic $t = \hat{\psi}/\mathrm{SE}_{\hat{\psi}}$ follows a central t distribution with $N - a$ degrees of freedom under $H_0 : \psi = 0$, and a noncentral t distribution with the same df and noncentrality parameter $\lambda = \psi/\mathrm{SE}_{\hat{\psi}}$ under the alternative.

For a two-sided test at level α,

$$\text{Power} = \Pr(t > t_{1-\alpha/2,\,df}) + \Pr(t < -t_{1-\alpha/2,\,df}),$$

computed exactly from the noncentral t distribution; for a one-sided test, only the appropriate tail contributes.

Cohen's f for a one-df contrast is reported as $f = |\lambda|/\sqrt{N}$, equivalently $f^2 = F_{\text{pop}}/N$ (Cohen, 1988, Ch. 8); this matches the value that `pwr.f2.test` expects when used with `u = 1` and `v = N - a`.

Value

A `data.frame` with two columns, `term` and `value`:

When solving for *n* (`specified_n_per_group = NULL`) Rows are `necessary_n_per_group`, `total_N`, `actual_power`, `noncentral_t_parm`, and `true_f` (Cohen's f for a one-degree-of-freedom contrast).

When evaluating power (`specified_n_per_group` **supplied**) Rows are `specified_n_per_group` (or `NA` for unequal *n*), `total_N`, `actual_power`, `noncentral_t_parm`, and `true_f`.

The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention; the reported size is the per-group *n* (or `NA` when unequal group sizes are supplied).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

contrast_test, ss_power_reg_coef, cv_t

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# Four-group example from Maxwell & Delaney's textbook tradition: contrast
# the average of three treatment means with a fourth (control), under
# population means (90, 92, 88, 81), within-group variance 144.
#
# 1. Power achieved at n = 20 per group (total N = 80). Should be ~ .80.
ss_power_contrast(
  c_weights           = c(1/3, 1/3, 1/3, -1),
  true_means          = c(90, 92, 88, 81),
  true_var_error      = 144,
  specified_n_per_group         = 20
)

# 2. Per-group sample size needed for power = .90.
ss_power_contrast(
  c_weights           = c(1/3, 1/3, 1/3, -1),
  true_means          = c(90, 92, 88, 81),
  true_var_error      = 144,
  desired_power       = 0.90
)

# 3. Same effect size specification using a directly given true_contrast.
ss_power_contrast(
  c_weights           = c(1/3, 1/3, 1/3, -1),
  true_contrast       = 9,
  true_var_error      = 144,
  specified_n_per_group         = 20
)

# 4. Unequal per-group sample sizes.
ss_power_contrast(
  c_weights           = c(0.5, 0.5, -0.5, -0.5),
  true_means          = c(90, 92, 88, 81),
  true_var_error      = 144,
  specified_n_per_group         = c(15, 25, 25, 15)
```

```
)
```

`ss_power_c_ancova` *Sample Size or Power for an Unstandardized Contrast in a One-Way ANCOVA*

Description

Determine the necessary per-group sample size to achieve a desired level of statistical power for the test of a single planned (unstandardized) contrast on the adjusted means in a one-way analysis of covariance, or, given a per-group sample size, return the realized statistical power.

Usage

```
ss_power_c_ancova(
  true_contrast,
  c_weights,
  true_sd,
  true_cor,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_group = NULL,
  directional = FALSE
)
```

Arguments

`true_contrast`	The population unstandardized contrast effect on the adjusted means, $\psi = \sum c_j \mu_j^{(adj)}$
`c_weights`	Vector of contrast weights (must sum to zero); use fractional weights so the positive weights sum to 1
`true_sd`	Within-group population standard deviation of the response (the same σ as in a one-way ANOVA on the response)
`true_cor`	Within-group population correlation between the response and the covariate, ρ in Details; must lie in (-1, 1)
`desired_power`	Desired statistical power (default 0.85)
`alpha_level`	Type I error rate (default 0.05)
`specified_n_per_group`	Per-group sample size (assumed balanced); if specified, returns the realized power
`directional`	Logical: `TRUE` for a one-sided test (in the same sign as `true_contrast`), `FALSE` (default) for a two-sided test

Details

This function uses the standard large-sample formulation in which the ANCOVA error variance is $\sigma^2_{adj} = \sigma^2(1-\rho^2)$ (σ is `true_sd` and ρ is `true_cor`), the contrast t-statistic has degrees of freedom $N - J - 1$ (one less than the corresponding ANOVA contrast because of the covariate), and the noncentrality parameter is $\lambda = \psi/(\sigma\sqrt{1-\rho^2}\sqrt{\sum c_j^2/n})$. This assumes the covariate means are equal across groups (the typical assumption under random assignment); for designs with substantial group differences in the covariate, the small-sample correction $1 + (\bar{X}_j - \bar{X}_.)^2/SS_X^{(within)}$ would slightly inflate the standard error and reduce power, an effect that is negligible for moderate or large n.

Value

A `data.frame` with rows for `necessary_n_per_group` (or `specified_n_per_group`), `actual_power`, and `noncentral_t_parm`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

See Also

`ss_power_c`, `ci_c_ancova`, `ss_aipe_c_ancova`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_composite_s`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`, `ss_power_reg_coef_sensitivity()`, `ss_power_rm_anova()`, `ss_power_sc()`, `ss_power_sem()`, `ss_power_smd()`, `ss_power_split_plot_anova()`

Examples

```
# Same population contrast as in the ANOVA example, with true_cor = 0.5 between
# outcome and covariate; ANCOVA is more efficient than ANOVA here.
ss_power_c_ancova(true_contrast = 0.5, c_weights = c(0.5, 0.5, -0.5, -0.5),
                  true_sd = 1, true_cor = 0.5, desired_power = 0.80)
```

```
# Realized power for n = 30 per group
ss_power_c_ancova(true_contrast = 0.5, c_weights = c(0.5, 0.5, -0.5, -0.5),
                  true_sd = 1, true_cor = 0.5, specified_n_per_group = 30)
```

`ss_power_equivalence_c` *Sample Size for Equivalence or Noninferiority of a Linear Contrast*

Description

Computes the smallest sample size (per group, or the number of units in a paired design) at which the two one-sided tests procedure (Schuirmann, 1987) for a linear contrast $\psi = \sum_j c_j \mu_j$, or the companion one-sided noninferiority test, attains a desired power. Power is computed exactly through `power_equivalence_c`. This is the declaration-probability route to planning; the accuracy in parameter estimation (AIPE) route, which targets the confidence interval width directly, is `ss_aipe_c`, and the two answer the same question whenever `true_contrast = 0` and the width target is calibrated to the bounds.

Usage

```
ss_power_equivalence_c(
  c_weights = NULL,
  true_sd,
  contrast_lower = NULL,
  contrast_upper = NULL,
  true_contrast = 0,
  desired_power = 0.85,
  alpha_level = 0.05,
  side = c("equivalence", "noninferiority"),
  design = c("independent", "paired")
)
```

Arguments

`c_weights`	The contrast weights, required for `design = "independent"`. The weights must sum to zero with the positive weights summing to 1 and the negative weights to -1, so that the bounds are on the raw scale of the response. Leave `NULL` for `design = "paired"`, where the weights are already in the per-unit contrast scores.
`true_sd`	For `design = "independent"`, the anticipated error standard deviation (the square root of the mean square error). For `design = "paired"`, the anticipated standard deviation of the per-unit contrast scores (for two conditions, of the difference scores). It is σ in the formulas below.

`contrast_lower`, `contrast_upper`
: Bounds on the raw scale of the response, each positive; the equivalence region is $(-\delta_L, +\delta_U)$, where δ_L is `contrast_lower` and δ_U is `contrast_upper`. For `side = "equivalence"`, `contrast_upper` is required and `contrast_lower` defaults to it (symmetric bounds). For `side = "noninferiority"`, the test is against $-\delta_L$ alone, so `contrast_lower` is the argument to give; a call that gives only the upper bound uses it as the margin.

`true_contrast`
: The population contrast (ψ) the design should be able to detect as equivalent (or noninferior). Default `0`. For equivalence it must lie strictly inside the bounds; for noninferiority, strictly above $-\delta_L$. The farther `true_contrast` sits from the center of the region, the larger the required sample size.

`desired_power`
: The target probability of declaring equivalence (or noninferiority) at `true_contrast`. Default `0.85`, matching `ss_power_contrast`.

`alpha_level`
: One-sided significance level for each test. Default `0.05`.

`side`
: `"equivalence"` (default) or `"noninferiority"`.

`design`
: `"independent"` (default) for groups of different units sharing one pooled error term, or `"paired"` for a contrast measured within each unit, analyzed as the mean of the per-unit contrast scores (see `power_equivalence_c`).

Details

Design. Planning for independent groups assumes equal allocation across the J groups named by `c_weights` and a pooled error term on $N - J$ degrees of freedom. Groups with zero weight still contribute error degrees of freedom, which is why they belong in `c_weights` when the fitted model will include them. Planning for a paired design counts units. Each condition is observed on every unit, so the answer is the total sample size N, and the mean of the N per-unit contrast scores has standard error $\sigma/\sqrt{N}$ on $N - 1$ degrees of freedom, where σ is the standard deviation of the scores. That standard deviation carries the correlation between the measures (see the paired designs paragraph of `power_equivalence_c`), and a positive correlation is what lets a paired design reach a given power with fewer observations than independent groups need.

The search. The function starts from the normal-theory approximation and moves to the smallest integer size (per group, or the number of units of a paired design) whose exact power reaches `desired_power`. Power is monotone in the size once the design is feasible, so the search is a short walk.

Relation to the half-width rule. With symmetric bounds $\pm\delta$, `true_contrast = 0`, and $\alpha = .05$, targeting a 90% CI half-width of $\delta/2$ yields a declaration probability of about .90; this function makes the probability the target directly rather than through the width.

Value

A `data.frame` with columns `term` and `value`. For `design = "independent"` the rows are `necessary_n_per_group` (the recommended sample size of each group the contrast involves), `total_N` (the number of units the design needs, $J \times n$ for the J groups named by `c_weights`), and `actual_power` (the exact power achieved at the recommendation). For `design = "paired"` the rows are `necessary_N` and `actual_power`: each condition is observed on every unit, so the answer is the number of units, which is the total sample size, and there is no per-group size to report beside it. The row is named for the quantity it holds; it is the value `power_equivalence_c` takes as `N`, as `necessary_n_per_group`

is the value it takes as `n_per_group`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention; `tidy()` reports the planned size (per group, or the number of units) and the power in the same columns for both designs.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., Bandyopadhyay, T., Kelley, K., & Padalunkal, J. J. (2025). A sequential approach for noninferiority or equivalence of a linear contrast under cost constraints. *Psychological Methods, 30*(2), 425–439. doi:10.1037/met0000570

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657–680.

See Also

`power_equivalence_c`, `ss_aipe_c`, `ss_power_contrast`, `equivalence_c`

Other equivalence testing: `equivalence_c()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `plot_equivalence()`, `power_density_equivalence_md()`, `power_equivalence_c()`, `power_equivalence_md()`, `power_equivalence_md_plot()`

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_composite_sem()`, `ss_power_contrast()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`, `ss_power_reg_coef_sensitivity()`, `ss_power_rm_anova()`, `ss_power_sc()`, `ss_power_sem()`, `ss_power_smd()`, `ss_power_split_plot_anova()`

Examples

```
# 1. Two-group equivalence with bounds of 5 raw-scale points and an
#    anticipated error SD of 15.67: n per group for 90% power at
#    true equivalence.
ss_power_equivalence_c(c_weights = c(1, -1), true_sd = 15.67,
                       contrast_upper = 5, desired_power = 0.90)

# 2. Noninferiority is cheaper than equivalence at the same bound.
ss_power_equivalence_c(c_weights = c(1, -1), true_sd = 15.67,
                       contrast_upper = 5, desired_power = 0.90,
                       side = "noninferiority")
```

```
# 3. A true contrast off center raises the requirement.
ss_power_equivalence_c(c_weights = c(1, -1), true_sd = 15.67,
                       contrast_upper = 5, true_contrast = 2,
                       desired_power = 0.90)

# 4. A paired noninferiority design: difference scores with an
#    anticipated SD of 1 and a margin of 0.5. Each unit is measured
#    under both conditions, so the answer is the number of units,
#    necessary_N.
ss_power_equivalence_c(true_sd = 1, contrast_lower = 0.5,
                       desired_power = 0.90, side = "noninferiority",
                       design = "paired")
```

`ss_power_factorial_ancova`

Sample Size Planning for Power in Factorial ANCOVA

Description

Power and sample size for any effect (a main effect or any interaction) in a between-subjects factorial design with covariates: the analysis of covariance generalization of `ss_power_factorial_anova`. Covariates earn their keep by absorbing error variance: with a joint squared multiple correlation R^2 between the covariates and the outcome within cells, the error variance falls by the factor $1 - R^2$, so an effect size f defined on the original (ANOVA) metric grows to $f/\sqrt{1 - R^2}$ in the covariate-adjusted analysis, at the price of one error degree of freedom per covariate.

Usage

```
ss_power_factorial_ancova(
  factor_levels,
  effect_indices,
  true_f = NULL,
  true_eta_squared_partial = NULL,
  true_covariate_R2 = 0,
  n_covariates = 0,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_cell = NULL
)
```

Arguments

`factor_levels`	Integer vector giving the number of levels of each factor, for example `c(2, 4, 3)` for a 2 x 4 x 3 design.
`effect_indices`	Integer vector identifying the factors that define the effect of interest: `1` for the first factor's main effect, `c(2, 3)` for the B x C interaction, and so on.

`true_f` Cohen's f for the chosen effect *on the unadjusted (ANOVA) metric*, that is, with the within-cell standard deviation of the outcome in its denominator. Supply this or `true_eta_squared_partial`, not both.

`true_eta_squared_partial` Population partial eta squared for the chosen effect on the unadjusted metric.

`true_covariate_R2` Joint squared multiple correlation between the covariates and the outcome within cells, in $[0, 1)$. `0` reproduces the ANOVA analysis (with the covariate degrees of freedom still spent if `n_covariates > 0`).

`n_covariates` Number of covariates, a non-negative integer.

`desired_power` Desired power; the per-cell sample size is solved when `specified_n_per_cell` is `NULL`. Defaults to 0.85 to match `ss_power_factorial_anova`.

`alpha_level` Type I error rate.

`specified_n_per_cell` Per-cell sample size; when supplied, the realized power at that size is returned instead of solving for size.

Details

The test of an effect with numerator degrees of freedom df_h (the product of the involved factors' levels each minus one) is a noncentral F with noncentrality $\lambda = N f_{\text{adj}}^2$, where N is the total sample size, $f_{\text{adj}} = f/\sqrt{1 - R^2}$, and error degrees of freedom $N - (\prod \text{levels}) - c$, where c is the number of covariates, `n_covariates` (the standard one-line ANCOVA adjustment; Maxwell, Delaney, & Kelley, 2027, Chapter 9). The covariate slopes are assumed homogeneous across cells and the covariates measured at baseline, so that adjusting does not bias the treatment effects in a randomized design.

The complete worked example for this function, a 2 x 4 x 3 ANCOVA with two baseline covariates, planned effect by effect and then analyzed with Type III sums of squares, interaction plots, and focused follow-up contrasts, is the "Power for factorial ANCOVA" vignette: `vignette("ancova_2x4x3_power", package = "DMAR")`.

Value

A `data.frame` with the per-cell and total sample sizes (or the supplied ones), the realized `actual_power`, the numerator and error degrees of freedom, the unadjusted and covariate-adjusted effect sizes (`true_f`, `true_f_adjusted`), `true_covariate_R2`, `n_covariates`, the noncentrality parameter (`noncentral_f_parm`), and `alpha_level`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention (the reported size is the per-cell count).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on designs with covariates.)

See Also

ss_power_factorial_anova for the no-covariate case this wraps; ancova and ci_sc_ancova for the analysis side; `vignette("ancova_2x4x3_power", package = "DMAR")` for the full worked design.

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# A 2 x 4 x 3 design, planning the f = .10 main effect of the
# two-level factor A. Two baseline covariates with a modest joint
# R^2 = .25 cut the required total N by roughly a quarter:
ss_power_factorial_ancova(factor_levels = c(2, 4, 3), effect_indices = 1,
                          true_f = 0.10, true_covariate_R2 = 0,    n_covariates = 0,
                          desired_power = 0.80)
ss_power_factorial_ancova(factor_levels = c(2, 4, 3), effect_indices = 1,
                          true_f = 0.10, true_covariate_R2 = 0.25, n_covariates = 2,
                          desired_power = 0.80)

# Realized power for the three-way interaction at 6 per cell.
ss_power_factorial_ancova(factor_levels = c(2, 4, 3),
                          effect_indices = c(1, 2, 3), true_f = 0.15,
                          true_covariate_R2 = 0.25, n_covariates = 2,
                          specified_n_per_cell = 6)
```

---

ss_power_factorial_anova

Sample Size or Power for a Factorial Between-Subjects ANOVA Effect

---

Description

Determine the necessary per-cell sample size to achieve a desired level of statistical power for a single *F* test (main effect or interaction) in a between-subjects factorial ANOVA, or, given a per-cell sample size, return the realized statistical power. The function handles two-way and higher-order factorial designs.

Usage

```
ss_power_factorial_anova(
  factor_levels,
  effect_indices,
  true_f = NULL,
  true_eta_squared_partial = NULL,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_cell = NULL
)
```

Arguments

`factor_levels`	Integer vector giving the number of levels of each factor (e.g., `c(2, 3)` for a 2 x 3 design, `c(2, 2, 4)` for a 2 x 2 x 4 design)
`effect_indices`	Integer vector identifying the factors that define the effect of interest. For example, `1` requests the main effect of the first factor; `c(1, 2)` requests the AxB two-way interaction; `c(1, 2, 3)` requests the three-way interaction
`true_f`	Cohen's f effect size for the chosen effect; supply this or `true_eta_squared_partial`, but not both
`true_eta_squared_partial`	Population partial eta squared for the chosen effect; supply this or `true_f`
`desired_power`	Desired statistical power (default 0.85)
`alpha_level`	Type I error rate (default 0.05)
`specified_n_per_cell`	Per-cell sample size; if specified, returns the realized power

Details

For a between-subjects factorial design, the F statistic for the chosen effect follows a noncentral F distribution under the alternative with numerator degrees of freedom $\prod_{i \in S}(k_i - 1)$ (where S is `effect_indices` and k_i is `factor_levels[i]`), denominator degrees of freedom $N - K$ (where $N = n_{cell} \prod k_i$ and $K = \prod k_i$ is the number of cells), and noncentrality parameter $\lambda = N f^2$. Cohen's f relates to partial eta squared via $f = \sqrt{\eta_p^2 / (1 - \eta_p^2)}$.

The function searches over per-cell sample sizes until power reaches `desired_power`; when `specified_n_per_cell` is supplied it returns the realized power.

For covariates, see `ss_power_factorial_ancova`. A complete worked three-factor example (a 2 x 4 x 3 design planned effect by effect, simulated, analyzed with Type III sums of squares, plotted, and followed up with focused contrasts) is the vignette `vignette("ancova_2x4x3_power", package = "DMAR")`.

Value

A `data.frame` with rows for `necessary_n_per_cell` (or `specified_n_per_cell`), `total_N`, `df_effect`, `df_error`, `noncentral_f_parm`, and `actual_power`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention (the reported size is the per-cell count).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ss_power_one_way_anova, ss_power_c, ci_nc_F

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# 2 x 3 design, main effect of factor B (the 3-level factor), f = 0.25, power = .80
ss_power_factorial_anova(factor_levels = c(2, 3), effect_indices = 2,
                         true_f = 0.25, desired_power = 0.80)

# 2 x 2 design, AxB interaction, partial eta squared = 0.06, power = .80
ss_power_factorial_anova(factor_levels = c(2, 2), effect_indices = c(1, 2),
                         true_eta_squared_partial = 0.06, desired_power = 0.80)

# 2 x 2 x 3 design, three-way interaction, f = 0.20
ss_power_factorial_anova(factor_levels = c(2, 2, 3), effect_indices = c(1, 2, 3),
                         true_f = 0.20, desired_power = 0.80)

# Realized power for specified_n_per_cell = 20 in a 2x3 design, AxB interaction, f = 0.25
ss_power_factorial_anova(factor_levels = c(2, 3), effect_indices = c(1, 2),
                         true_f = 0.25, specified_n_per_cell = 20)
```

`ss_power_indirect_effect` *Sample Size Planning for Power for the Indirect (Mediation) Effect*

Description

Power, or the sample size required for a desired power, for the test of the indirect effect ab in the simple mediation model, with paths specified in standardized metric (unit-variance X, M, and Y). The default test is joint significance (the indirect effect is declared when both $\hat{a}$ and $\hat{b}$ are individually significant), which tracks the resampling tests' power closely and far exceeds the Sobel test in small samples (Fritz & MacKinnon, 2007); the Sobel normal-theory test is available for comparison. Power comes from the exact sampling distribution of the two path t statistics, with X random by default. This is the power counterpart of the accuracy in parameter estimation (AIPE) planner `ss_aipe_indirect_effect`, and the planning complement of the analysis function `mediate`.

Usage

```
ss_power_indirect_effect(
  true_std_reg_coef_a,
  true_std_reg_coef_b,
  true_std_reg_coef_direct = 0,
  desired_power = NULL,
  specified_N = NULL,
  alpha_level = 0.05,
  method = c("joint_significance", "sobel"),
  random_predictors = TRUE
)
```

Arguments

`true_std_reg_coef_a`
: Population standardized regression coefficient of the $X \rightarrow M$ path a.

`true_std_reg_coef_b`
: Population standardized regression coefficient of the $M \rightarrow Y$ path b, holding X.

`true_std_reg_coef_direct`
: Population standardized regression coefficient of the direct effect c' of X on Y holding M. Defaults to 0; it enters only through the residual variance of Y.

`desired_power`
: Desired power; supply this to solve for N.

`specified_N`
: Total sample size N; supply this to evaluate the realized power. Specify exactly one of `desired_power` and `specified_N`.

`alpha_level`
: Two-sided Type I error rate for each component test. Defaults to 0.05.

`method`
: `"joint_significance"` (default) or `"sobel"`.

`random_predictors`
: `TRUE` (default) when X is random, as when it is measured, and `FALSE` when it is fixed by design, as when groups are assigned. The mediator is random under either setting.

Details

With unit-variance variables, $M = aX + e_M$ and $Y = c'X + bM + e_Y$, with normal errors of variances $1 - a^2$ and $\sigma^2_{e_Y} = 1 - (b^2 + c'^2 + 2abc')$, which must be positive, or the function stops. As in `mediate`, t_a tests the slope of M on X ($N - 2$ degrees of freedom) and t_b the coefficient of M in the regression of Y on X and M ($N - 3$). Given X, t_a is noncentral t with noncentrality $a\sqrt{SS_X/(1 - a^2)}$; given X and M, t_b is noncentral t with noncentrality $b\sqrt{SS_{M \cdot X}}/\sigma_{e_Y}$, where $SS_{M \cdot X}$ is the residual sum of squares of M on X. $SS_{M \cdot X}/(1 - a^2)$ is chi square on $N - 2$ degrees of freedom in every design, because M is random given X; SS_X is chi square on $N - 1$ with X random, and N with X fixed at variance 1 (divisor N; two groups of equal size, for example). Each power averages the noncentral t power over these chi squares, by Gauss-Hermite sums on their normal scores.

$SS_{M \cdot X}$ is the error sum of squares of one test and the predictor sum of squares of the other, so the tests are dependent: `power_ab` is the probability that both reject, which is below `power_a` times `power_b`. The first-order Sobel statistic of `mediate` equals $t_a t_b/\sqrt{t_a^2 + t_b^2}$, and its power is the probability, integrated numerically, that it exceeds the normal critical value; `power_a` and `power_b` are then `NA`. Because it refers t statistics to the normal, its power need not rise with N: with paths of .9 and .05 it falls from .0729 at $N = 10$ to .0562 at 59, and with paths of .8 and .02 and `alpha_level = .5` it is .5072 at 12 and .5035 at 59, and does not reach .507 again until 216. The joint significance power rose with N in each of 1,000 settings checked at every N from 10 to 1,500 (paths from .001 to .99 in absolute value, `alpha_level` from .0001 to .997, X random or fixed), which is not a proof. So the search assumes neither: `necessary_N` is the smallest N of at least 10 whose power reaches `desired_power`. It scans upward, passing over a range of N only where a bound shows that none reaches it: written as its quantile at a fixed normal score, each chi square rises with N, so the power at every N of the range is at most the power with the noncentralities at the top of the range and each test's threshold (critical value times scale) at its least over the range. Where a path is near zero, the power is also at most the rate at which the test of that path rejects when the path is zero plus $\phi(1)$ (the standard normal density at 1) times the expected square of its noncentrality (for the Sobel method, the test of $|t|$ against the normal critical value, since the Sobel statistic is at most the smaller of $|t_a|$ and $|t_b|$), which lets the search pass at once over a long range in which the power stays just below `desired_power`. It stops at N = 10,000,000 with an error of class `dmar_computation_failed`. The tests check `power_ab` against raw-data simulation within Monte Carlo error at the plan for power .80 and at one case fewer.

Fixing each noncentrality at its population standard error and multiplying the two powers overstates the joint significance power: with paths of .39 that approximation gives .8017 at $N = 65$, where the power is .7675, and the plan for .80 is 69.

Value

A tidy `data.frame` with `necessary_N` (or `specified_N`), `power_ab` (the power to detect the indirect effect ab, the quantity the sample size is planned against), the powers of the tests of the two paths (`power_a`, `power_b`; `NA` for the Sobel method), the paths (`true_std_reg_coef_a`, `true_std_reg_coef_b`, `true_std_reg_coef_direct`), their implied standardized indirect effect `true_std_indirect_effect`, and `alpha_level`. The method is recorded in the `"method"` attribute and the predictor model in the `"random_predictors"` attribute. The result carries the `dmar_ss_power` class, so `tidy` reports the sample size and the power to detect the indirect effect (the row `power_ab`), and `glance` adds the component powers and the planning inputs.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Fritz, M. S., & MacKinnon, D. P. (2007). Required sample size to detect the mediated effect. *Psychological Science, 18*(3), 233–239. doi:10.1111/j.14679280.2007.01882.x

MacKinnon, D. P., Lockwood, C. M., Hoffman, J. M., West, S. G., & Sheets, V. (2002). A comparison of methods to test mediation and other intervening variable effects. *Psychological Methods, 7*(1), 83–104. doi:10.1037/1082989X.7.1.83

See Also

`mediate` to analyze the study this plans; `ss_aipe_indirect_effect` to plan for confidence interval width instead of detection; `design_consequences` for what the chosen design delivers.

Other mediation: `mediate()`, `mediation_mbco()`

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_composite_sem()`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`, `ss_power_reg_coef_sensitivity()`, `ss_power_rm_anova()`, `ss_power_sc()`, `ss_power_sem()`, `ss_power_smd()`, `ss_power_split_plot_anova()`

Examples

```
# Fritz and MacKinnon's (2007) running scenario (a = b = .39), X random.
ss_power_indirect_effect(true_std_reg_coef_a = .39,
                         true_std_reg_coef_b = .39, desired_power = .80)

# X fixed by design (two groups of equal size); the mediator is random.
ss_power_indirect_effect(true_std_reg_coef_a = .39,
                         true_std_reg_coef_b = .39, desired_power = .80,
                         random_predictors = FALSE)

# A near-zero a path against a larger b: the weak link drives the requirement.
ss_power_indirect_effect(true_std_reg_coef_a = .14,
                         true_std_reg_coef_b = .39, desired_power = .80)

# Realized power at a given N, and the Sobel test at the same N.
ss_power_indirect_effect(true_std_reg_coef_a = .39,
                         true_std_reg_coef_b = .39, specified_N = 75)
ss_power_indirect_effect(true_std_reg_coef_a = .39,
                         true_std_reg_coef_b = .39, specified_N = 75,
                         method = "sobel")
```

`ss_power_mixed_effects` *Sample Size or Power for a Treatment Effect in a Two-Level Mixed-Effects Model*

Description

Determine the necessary number of clusters (i.e., Level 2 units) per arm to achieve a desired level of statistical power for a treatment-versus-control comparison in a two-level mixed-effects model with a random intercept (e.g., individuals nested within clusters in a cluster-randomized trial, or repeated measurements nested within subjects in a person-randomized longitudinal study). Alternatively, given a number of clusters per arm, return the realized statistical power.

Usage

```
ss_power_mixed_effects(
  true_smd,
  cluster_size,
  true_icc,
  specified_n_clusters_per_arm = NULL,
  desired_power = 0.85,
  alpha_level = 0.05,
  directional = FALSE
)
```

Arguments

`true_smd`	Population standardized treatment effect the test is powered for (d in the formulas below), defined as the population mean difference divided by the population standard deviation of the outcome among individual observations (i.e., Level 1 units)
`cluster_size`	Number of observations (i.e., Level 1 units) in each cluster (i.e., Level 2 unit); for example, individuals per cluster or measurements per subject, n in the formulas below; a single whole number of at least 2, assumed equal across clusters. It has no default
`true_icc`	Population intraclass correlation (the proportion of total outcome variance attributable to differences between clusters (i.e., Level 2 units)), ρ in the formulas below; a planning value that must be in [0, 1)
`specified_n_clusters_per_arm`	Number of clusters (i.e., Level 2 units) in each arm, J in the formulas below (i.e., `specified_n_clusters_per_arm` treatment clusters and as many control clusters, so the total number of clusters is twice this value); if specified, returns the realized power
`desired_power`	Desired statistical power (default 0.85)
`alpha_level`	Type I error rate (default 0.05)
`directional`	Logical: `TRUE` for a one-sided test (in the same sign as `true_smd`), `FALSE` (default) for a two-sided test

Details

This function computes power for the fixed treatment effect at the higher level of a two-level mixed-effects model with random intercept,

$$y_{ij} = \beta_0 + \beta_1 T_j + u_j + \epsilon_{ij},$$

where $u_j \sim N(0, \sigma_u^2)$ is the cluster (i.e., Level 2) random intercept and $\epsilon_{ij} \sim N(0, \sigma_e^2)$ is the within-cluster (i.e., Level 1) residual. The treatment indicator T_j varies between clusters (i.e., entire clusters or entire subjects are assigned to treatment or control). The intra-class correlation is $\rho = \sigma_u^2/(\sigma_u^2 + \sigma_e^2)$ and the total outcome variance is $\sigma_y^2 = \sigma_u^2 + \sigma_e^2$.

With n observations in each cluster (`cluster_size`) and J clusters in each arm (`specified_n_clusters_per_arm`), the standard error of the estimated treatment effect is $SE(\hat{\beta}_1) = \sigma_y \sqrt{2(1 + (n-1)\rho)/(Jn)}$, giving a noncentrality parameter of

$$\lambda = d\sqrt{Jn/(2(1 + (n-1)\rho))}$$

under a two-sample *t*-test with $2J - 2$ degrees of freedom.

The factor $1 + (n-1)\rho$ is the design effect: as the within-cluster correlation grows, the effective information per cluster shrinks, so more clusters are needed for a given level of power.

Value

A `data.frame` with rows for `necessary_n_clusters_per_arm` (or `specified_n_clusters_per_arm` when `specified_n_clusters_per_arm` is given), `total_N` (the total number of observations (i.e., Level 1 units), $2Jn$), `noncentral_t_parm` (the noncentrality parameter λ of the *t* distribution), and `actual_power`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention (the reported size is the number of clusters per arm).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Raudenbush, S. W. (1997). Statistical analysis and optimal design for cluster randomized trials. *Psychological Methods, 2*, 173–185. doi:10.1037/1082989X.2.2.173

See Also

`ss_power_split_plot_anova`, `ss_power_smd`, `ss_power_pcm`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`,

ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Other mixed models: R2_mixed_effects(), R2_mixed_effects_decomposition(), icc_lmer(), manova_split_plot(), mixed_anova(), ss_aipe_mixed_effects(), ss_aipe_mixed_effects_sensitivity(), ss_power_split_plot_anova()

Examples

```
# 30 individuals per cluster, ICC = 0.05, standardized effect 0.30, power = .80
ss_power_mixed_effects(true_smd = 0.30, cluster_size = 30, true_icc = 0.05, desired_power = 0.80)

# Same effect but with much higher ICC (e.g., schools or therapists)
ss_power_mixed_effects(true_smd = 0.30, cluster_size = 30, true_icc = 0.20, desired_power = 0.80)

# Realized power with 25 clusters per arm
ss_power_mixed_effects(true_smd = 0.30, cluster_size = 30, true_icc = 0.05,
                       specified_n_clusters_per_arm = 25)

# directional test
ss_power_mixed_effects(true_smd = 0.30, cluster_size = 30, true_icc = 0.05,
                       desired_power = 0.80, directional = TRUE)
```

---

ss_power_one_way_anova

Sample Size or Power for a One-Way Between-Subjects ANOVA Omnibus F *Test*

Description

Determine the necessary total sample size to achieve a desired level of statistical power for the omnibus *F* test in a one-way between-subjects analysis of variance, or, given a total sample size, return the realized statistical power.

Usage

```
ss_power_one_way_anova(
  a,
  true_f = NULL,
  true_eta_squared = NULL,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_N = NULL
)
```

Arguments

a	Number of groups (levels of the between-subjects factor)
true_f	Cohen's f effect size (the population value); supply this or true_eta_squared, but not both
true_eta_squared	Population eta squared (proportion of total variance accounted for by group membership); supply this or true_f
desired_power	Desired statistical power (default 0.85)
alpha_level	Type I error rate (default 0.05)
specified_N	Total sample size; if specified, the function returns the realized power (the ss_power_* family is not uniform here: ss_power_contrast takes a *per-group* size)

Details

Under the alternative hypothesis, the omnibus F statistic follows a noncentral F distribution with numerator df $a-1$, denominator df $N-a$, and noncentrality parameter $\lambda = Nf^2$. Cohen's f relates to eta squared via $f = \sqrt{\eta^2/(1-\eta^2)}$.

The function searches over total sample sizes N (treating per-group N/a as balanced) until power first reaches desired_power. When specified_N is supplied it instead reports the realized power at that total sample size.

Value

A data.frame. When a sample size is being planned (specified_N not supplied) the rows are necessary_N, n_per_group, a, noncentral_f_parm, and actual_power; the search constructs the total as a balanced design, so n_per_group is a whole-number per-group count. When specified_N is supplied, power is evaluated at that total sample size directly and the rows are specified_N, a, noncentral_f_parm, and actual_power (no n_per_group row, since balance is not assumed). The result carries the dmar_ss_power class, so tidy and glance summarize it in broom convention; the summarized sample size is the total sample size N.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ss_power_factorial_anova, ss_power_c, ss_power_sc, ci_nc_F

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# Three groups, f = 0.25, power = .80
ss_power_one_way_anova(a = 3, true_f = 0.25, desired_power = 0.80)

# Same effect specified via eta squared
ss_power_one_way_anova(a = 3, true_eta_squared = 0.0588, desired_power = 0.80)

# Realized power at N = 60 across 3 groups
ss_power_one_way_anova(a = 3, true_f = 0.25, specified_N = 60)
```

---

ss_power_pcm *Sample Size Planning for Power for Polynomial Change Models*

Description

Returns power given the sample size, or sample size given the desired power, for the group difference in a polynomial change coefficient (a flat-line intercept, a linear slope, a quadratic acceleration, or any higher-order trend) in a two-group longitudinal design, following Raudenbush and Liu (2001). The trend whose group difference is tested is selected with `trend`; `trend = "linear"` (the default) reproduces the straight-line case.

Usage

```
ss_power_pcm(
  true_reg_coef = NULL,
  true_std_reg_coef = NULL,
  true_var_trend,
  true_var_error,
  frequency,
  duration,
  desired_power = NULL,
```

```
  specified_n_per_group = NULL,
  alpha_level = 0.05,
  directional = FALSE,
  trend = "linear"
)
```

Arguments

true_reg_coef	The population value of the between-subject (i.e., Level 2) regression coefficient for the group by time interaction in the polynomial change coefficient selected by trend (linear by default), where the grouping variable is coded -.5 and .5 for the two groups: the difference between the two groups' mean change coefficients, in the units of the change coefficient (for a linear trend, the units of the outcome per unit of time). Give this or true_std_reg_coef, not both.
true_std_reg_coef	The same group difference standardized: the unstandardized difference divided by the square root of true_var_trend, the between-subject standard deviation of the change coefficient (β in the formulas below; Raudenbush & Liu, 2001, p. 391). Give this or true_reg_coef, not both; a call that gives both or neither is refused with a condition of class dmar_argument_invalid. Whichever is given, the table reports the difference on both scales, in its rows true_reg_coef and true_std_reg_coef.
true_var_trend	The true between-subject variance of the individuals' change coefficient for the trend selected by trend (the variance of the slopes for trend = "linear"); τ in the formulas below, and the quantity ss_aipe_pcm takes under the same name.
true_var_error	The true within-subject (i.e., Level 1) error variance (σ_e^2 in the formulas below), the quantity ss_aipe_pcm takes under the same name.
frequency	Number of measurements per unit of time, where the unit is the one in which duration is expressed. It need not be a whole number; for example, frequency = 0.5 means one measurement every two time units. Together with duration it fixes the number of equally spaced measurement occasions, $M = f \times D + 1$, with f the frequency and D the duration.
duration	Length of the study in the chosen time unit (for example, years, grades, or hours). Measurements are taken at $M = f \times D + 1$ equally spaced occasions spanning time 0 to duration.
desired_power	Desired power, a single number strictly between 0 and 1
specified_n_per_group	Sample size in each of the two groups (the total is twice this); if specified, the function returns the power achieved at that size
alpha_level	Type I error rate
directional	Should a one (TRUE) or two (FALSE) tailed test be performed.
trend	The polynomial change coefficient whose group difference is tested, given either as a name ("intercept", "linear", "quadratic", "cubic", "quartic", ...) or as a non-negative integer order (0 = intercept / flat line, 1 = linear, 2 = quadratic, ...). Defaults to "linear". The design must supply at least $p + 1$ measurement occasions to estimate a degree-p trend.

Details

The two groups each contain $N/2$ subjects measured on $M = f \times D + 1$ equally spaced occasions. Each subject's degree-p polynomial change coefficient is estimated within subject; the test compares the two group means of that coefficient. The change coefficient is taken in the derivative-scaled metric ($p!$ times the leading coefficient of t^p), the metric in which the Raudenbush and Liu (2001) constants apply.

The within-subject sampling variance of the estimated coefficient, the row `true_var_trend_error`, is (Raudenbush & Liu, 2001, p. 392)

$$V = \sigma_e^2 \, f^{2p} \, \frac{(M-p-1)!}{K_p \, (M+p)!}, \qquad \frac{1}{K_p} = \frac{(2p)! \, (2p+1)!}{(p!)^2},$$

so that $1/K_p = 1, 12, 720, 100800, \ldots$ for $p = 0, 1, 2, 3, \ldots$. This V equals $(p!)^2$ times the variance of the ordinary least squares estimate of the coefficient of t^p, reduces to σ_e^2/M at $p = 0$ and to $12\,\sigma_e^2 f^2/[M(M^2-1)]$ at $p = 1$. The slope reliability is $\tau/(\tau + V)$ (their Equation 15), the variance of the between-group difference is $4(\tau+V)/N$ (their Equation 10), and the *t* test has $N-2$ degrees of freedom and noncentrality $\sqrt{N\,\beta^2\,[\tau/(\tau+V)]/4}$ (their Equations 12 and 14), with β the standardized difference `true_std_reg_coef`; given `true_reg_coef` instead, the noncentrality is the same quantity written in it, $\sqrt{N\,\gamma^2/[4(\tau+V)]}$ with γ the unstandardized difference. The linear case reproduces the National Youth Survey benchmark in their Tables 1 and 2; the general-p formula has been checked against the exact $(X'X)^{-1}$ variance and against an end-to-end Monte Carlo power study for the quadratic trend.

Value

A `data.frame` (class `dmar_tbl`) with one row per reported quantity in a `term` / `value` layout: the per-group size per group (`necessary_n_per_group`, or `specified_n_per_group` when `specified_n_per_group` is supplied) and the total (`total_N`); the achieved power (`actual_power`); the measurement schedule (`frequency`, `duration`, `n_occasions`); the polynomial order of the tested change coefficient (`polynomial_order`, 0 = intercept, 1 = linear, 2 = quadratic, ...); the population change difference unstandardized (`true_reg_coef`) and standardized (`true_std_reg_coef`, the unstandardized difference divided by the square root of `true_var_trend`), one of them the argument given and the other the same difference on the other scale; the within-subject (i.e., Level 1) error variance (`true_var_error`); the variance of the individuals' true change coefficients (`true_var_trend`) and the sampling-error variance of one individual's estimated change coefficient (`true_var_trend_error`, V below, the quantity of Kelley & Rausch, 2011, Equation 19, that `ss_aipe_pcm` takes under the same name); the change-coefficient reliability (`reliability`); and the noncentrality parameter of the *t* test (`noncentral_t_parm`).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Kelley, K., & Rausch, J. R. (2011). Sample size planning for longitudinal models: Accuracy in parameter estimation for polynomial change parameters. *Psychological Methods, 16*(4), 391–405. doi:10.1037/a0023352

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapters 11, 15.)

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Raudenbush, S. W., & Liu, X.-F. (2001). Effects of study duration, frequency of observation, and sample size on power in studies of group differences in polynomial change. *Psychological Methods, 6*(4), 387–401. doi:10.1037/1082989X.6.4.387

See Also

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_composite_sem()`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`, `ss_power_reg_coef_sensitivity()`, `ss_power_rm_anova()`, `ss_power_sc()`, `ss_power_sem()`, `ss_power_smd()`, `ss_power_split_plot_anova()`

Examples

```
# The examples reproduce the National Youth Survey illustration of
# Raudenbush and Liu (2001, p. 393). One observation per year
# (frequency = 1) over a four-year study (duration = 4) gives
# M = frequency * duration + 1 = 5 equally spaced occasions. The
# standardized slope difference is -0.40, the slope variance is
# 0.003, and the within-subject (i.e., Level 1) error variance is
# 0.0262, so the slope reliability is 0.53.

# (1) Power at a given sample size. With N = 238 (119 per group) the
#     design has power 0.61 (Raudenbush and Liu, 2001, Table 1, D = 4,
#     f = 1).
ss_power_pcm(true_std_reg_coef = -.4, true_var_trend = .003, true_var_error = .0262,
             frequency = 1, duration = 4, specified_n_per_group = 119)

# (2) Sample size for a target power. Solving the same design for 0.80
#     power returns N = 370 (185 per group), between their Table 2 cells
#     N = 300 (power 0.71) and N = 400 (power 0.83).
ss_power_pcm(true_std_reg_coef = -.4, true_var_trend = .003, true_var_error = .0262,
             frequency = 1, duration = 4, desired_power = .80)

# (3) Unstandardized slope. The unstandardized slope difference is the
```

```
#     standardized one times the square root of true_var_trend, -0.40
#     times the square root of 0.003, about -0.0219. Giving it as
#     true_reg_coef reproduces the power of 0.61 from example (1).
ss_power_pcm(true_reg_coef = -.0219, true_var_trend = .003, true_var_error = .0262,
             frequency = 1, duration = 4, specified_n_per_group = 119)

# (4) Longer study, same number of occasions. Doubling the duration to
#     D = 8 while keeping M = 5 (so frequency = 0.5, one observation every
#     two years) raises the slope reliability to 0.82 and power to about
#     0.80. Spreading the same five occasions over a longer span sharply
#     increases power (Raudenbush & Liu, 2001, p. 393).
ss_power_pcm(true_std_reg_coef = -.4, true_var_trend = .003, true_var_error = .0262,
             frequency = .5, duration = 8, specified_n_per_group = 119)

# (5) More frequent sampling over a shorter span, same occasions. Halving
#     the duration to D = 2 while keeping M = 5 (so frequency = 2) drops
#     the slope reliability to 0.22 and power to about 0.31. Sampling more
#     often over a shorter study does little for power (Raudenbush & Liu,
#     2001, p. 393).
ss_power_pcm(true_std_reg_coef = -.4, true_var_trend = .003, true_var_error = .0262,
             frequency = 2, duration = 2, specified_n_per_group = 119)

# (6) One-sided test. A directional test of the base design places the
#     whole Type I error rate in the predicted tail, raising power from
#     0.61 to about 0.73.
ss_power_pcm(true_std_reg_coef = -.4, true_var_trend = .003, true_var_error = .0262,
            frequency = 1, duration = 4, specified_n_per_group = 119, directional = TRUE)

# (7) A higher-order trend. The same machinery plans power for the group
#     difference in any polynomial change coefficient. Here the target is
#     the quadratic trend (curvature / acceleration): with eight occasions
#     (frequency = 1, duration = 7), a between-subject quadratic-coefficient
#     variance 0.002, a within-subject (i.e., Level 1) error variance of
#     0.05, and a
#     standardized quadratic difference of 0.45, the design is planned for
#     0.80 power. A quadratic trend needs at least three occasions; a cubic
#     at least four (trend = "cubic" or trend = 3), and so on.
ss_power_pcm(true_std_reg_coef = 0.45, true_var_trend = 0.002, true_var_error = 0.05,
             frequency = 1, duration = 7, desired_power = .80,
             trend = "quadratic")
```

`ss_power_r` *Sample Size or Power for a Pearson Correlation Coefficient (Fisher Z Transformation)*

Description

Determine the necessary sample size to achieve a desired level of statistical power for the test of a Pearson correlation against a null value (typically zero), or, given a sample size, return the realized

statistical power. The computation uses the Fisher's Z transformation, which has a near-normal sampling distribution.

Usage

```
ss_power_r(
  true_cor,
  null_cor = 0,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_N = NULL,
  directional = FALSE
)
```

Arguments

`true_cor`	Population correlation coefficient the sample size or power is planned for (the value under the alternative hypothesis), in $(-1, 1)$. The planner treats this value as the truth.
`null_cor`	The null hypothesis value of the correlation (default 0)
`desired_power`	Desired statistical power (default 0.85)
`alpha_level`	Type I error rate (default 0.05)
`specified_N`	Total sample size (the *number of pairs*); if specified, the function returns the power at that sample size.
`directional`	Logical: `TRUE` for a one-sided test (in the same direction as the difference `true_cor - null_cor`), `FALSE` (default) for a two-sided test

Details

Under the alternative the Fisher-transformed correlation $Z_r = \tanh^{-1}(r)$ is approximately normal with mean $Z_\rho = \tanh^{-1}(\rho)$ and variance $1/(N-3)$. Power is computed from this normal approximation.

For sample size, a closed-form expression is used as the starting point,

$$N = ((z_\alpha + z_\beta)/(Z_\rho - Z_{\rho_0}))^2 + 3,$$

where ρ is `true_cor`, ρ_0 is `null_cor`, z_α is the standard normal critical value at `alpha_level` (one- or two-sided, as `directional` says), and z_β is the standard normal quantile at `desired_power`, which is then verified iteratively to ensure power exactly meets or exceeds `desired_power`. The search is bounded at $N = 10^7$. When `true_cor` and `null_cor` are so close that `desired_power` is unreachable within that bound, the function stops with an error rather than searching indefinitely.

Value

A `data.frame` with rows for `necessary_N` (or `specified_N`), `actual_power`, `true_cor`, and `null_cor`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron, 1*, 3–32.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on the one-way ANOVA and Chapter 4 on contrasts.)

See Also

ci_r, convert_r_Z, convert_Z_r

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# true_cor = 0.30, null_cor = 0, desired power = .80, two-sided
ss_power_r(true_cor = 0.30, desired_power = 0.80)

# Same with a directional alternative
ss_power_r(true_cor = 0.30, desired_power = 0.80, directional = TRUE)

# Realized power for N = 100 pairs
ss_power_r(true_cor = 0.30, specified_N = 100)

# Test against a non-zero null (null_cor = 0.20), looking for evidence that rho > 0.20
ss_power_r(true_cor = 0.40, null_cor = 0.20, desired_power = 0.80, directional = TRUE)
```

ss_power_R2 *Plan Sample Size to Make the Test of the Squared Multiple Correlation Coefficient Sufficiently Powerful*

Description

Determine the necessary sample size for the omnibus test of the squared multiple correlation coefficient (R^2), or the realized statistical power given a specified sample size, under either fixed or random predictors. The fixed-predictors path uses Cohen's (1988) noncentral F formulation; the random-predictors path uses the three-moment noncentral F approximation of Lee (1971, Section 5.1) to the sampling distribution of the sample R^2 under joint multivariate normality.

Usage

```
ss_power_R2(
  true_R2 = NULL,
  alpha_level = 0.05,
  desired_power = 0.85,
  p,
  specified_N = NULL,
  true_f2 = NULL,
  null_R2 = 0,
  random_predictors = TRUE,
  print_iter = FALSE
)
```

Arguments

true_R2	Population squared multiple correlation coefficient the sample size or power is planned for. The planner treats this value as the truth; ss_power_R2_sensitivity shows what the plan delivers when the truth (true_R2 there) differs from the value planned with (supposed_R2 there).
alpha_level	Type I error rate
desired_power	Desired degree of statistical power, a single number strictly between 0 and 1. The values 0 and 1 are refused, with a condition of class dmar_desired_power_invalid.
p	The number of predictor variables
specified_N	The sample size used to calculate power (rather than determine necessary sample size). This is the *total* sample size across all groups or observations.
true_f2	Cohen's (1988) effect size for multiple regression: true_R2/(1 - true_R2).
null_R2	The null value of the omnibus test, which must be 0. Any other value stops with an error of class dmar_null_R2_unsupported that explains why and gives the call to ss_power_R2_change; see Details.
random_predictors	Whether the predictor variables are treated as random (TRUE, the default) or fixed (FALSE). See Details.

`print_iter` If the progress of the iterative procedure is printed to the screen as the iterations are occurring

Details

Determine the necessary sample size given a particular `true_R2`, `alpha_level`, `p`, and `desired_power`. Alternatively, given `true_R2`, `alpha_level`, `p`, and `specified_N`, the function can be used to determine the statistical power.

Fixed vs.\ random predictors. The two regression models give *different* sampling distributions for the omnibus F-statistic, and so different power. Under fixed predictors the design matrix is treated as constant in hypothetical replications of the study, and F follows a noncentral F with p and $N - p - 1$ degrees of freedom and noncentrality $\lambda = N \cdot f^2$, where $f^2 = \rho^2/(1 - \rho^2)$ and ρ^2 is `true_R2` (Cohen, 1988). Cohen's noncentrality is $\lambda = f^2(u + v + 1)$, with u and v the numerator and denominator degrees of freedom (as given by Maxwell, 2000, Equation 1); for the omnibus test $u = p$ and $v = N - p - 1$, so $u + v + 1 = N$ and the two forms are the same. Under random predictors the design matrix is itself a draw from a joint multivariate normal distribution, and the unconditional distribution of the sample R^2 is due to Fisher (1928); `ss_power_R2()` uses the three-moment noncentral F approximation of Lee (1971, Section 5.1) to that distribution, the same approximation `ci_R2()` uses for random-predictor confidence intervals. Gatsonis and Sampson (1989) give the exact calculation for random predictors and compare it with Cohen's tables (the 1977 edition of the book cited below), whose sample sizes were slightly smaller than the exact ones for most designs with up to 10 predictors. The sample size this function returns under random predictors is the smallest at which the power by Lee's approximation attains `desired_power`. Against the exact distribution, the power by Lee's approximation differed by at most .0043 over the combinations examined (squared correlations up to .95, `alpha_level` from .001 to .10, and one or more error degrees of freedom) and by at most .0004 with 25 or more error degrees of freedom; the necessary sample size differed from the one the exact distribution gives, by one case, in 20 of 1,500 searches, and in those the power by the exact distribution at the size returned was within .0011 of `desired_power`. In the behavioral, educational, and social sciences predictor variables are almost always random, so the default is `random_predictors = TRUE`; pass `random_predictors = FALSE` for designs in which the predictor variables are fixed by design (for example, planned dosing levels).

Tests other than the omnibus test. `ss_power_R2()` plans the test that the population squared multiple correlation coefficient of all `p` predictors is zero. The test that predictors added to a model contribute nothing, a test of the change in R^2, is planned with `ss_power_R2_change`, which takes the number of predictors in both models.

`null_R2` exists so that a call written for MBESS with a nonzero null value is answered with an explanation; it must be 0. A nonzero value stops with an error of class `dmar_null_R2_unsupported` (see `dmar_metadata`), for these reasons. In MBESS a nonzero null value ρ_0^2 forms the effect size $(\rho^2 - \rho_0^2)/(1 - \rho^2)$, which is Cohen's (1988) f^2 for the increment in R^2 when predictors are added to a model that already explains ρ_0^2, and refers it to an F distribution with p and $N - p - 1$ degrees of freedom and noncentrality Nf^2. What that calculation is depends on how `p` is read. With `p` read as the number of predictors added, it is Cohen's calculation, $\lambda = f^2(u+v+1)$, for a study whose total sample size is N plus the number of predictors in the restricted model: the sample size it reports is Cohen's total sample size less that number. With `p` read as all of the predictors, as this function defines it, the calculation is not the test of the increment, whose numerator degrees of freedom are the number of predictors added and not `p`. The argument alone cannot say which calculation is

meant, because the test of an increment needs the number of predictors in the restricted model, so the value is refused and the message writes the call to `ss_power_R2_change()`.

The number that MBESS returns for a population value a and a null value b is `ss_power_R2(true_f2 = (a - b) / (1 - a), p = p, random_predictors = FALSE)`, the omnibus test of `p` predictors with an effect of that size; the last example shows it. Whether the population squared multiple correlation coefficient of all `p` predictors exceeds a stated value above zero is a third question. DMAR does not plan sample size for that test; the test itself rejects when the one-sided lower confidence limit from `ci_R2` (with `alpha_lower = alpha_level` and `alpha_upper = 0`) exceeds the stated value.

Limit of the calculation. R computes the noncentral F distribution accurately for a noncentrality parameter up to about one million, and warns beyond it. `ss_power_R2()` returns no power from an evaluation at which R gave that warning: a power requested at such a `specified_N` is refused, and a search whose answer lies there stops and says so. A search that only passes through such sizes on its way to a smaller answer returns that answer, which is evaluated without the warning. The search does not go beyond a sample size of ten million, and stops with an error when `desired_power` is not attained by then. With `true_R2` of 0 the null hypothesis is true: the power at a `specified_N` is `alpha_level`, and a search stops with an error of class `dmar_effect_zero` because no sample size attains `desired_power`.

Value

A `data.frame` with columns `term` and `value`. For an N search the rows are `necessary_N`, `actual_power`, `noncentral_f_parm` (only meaningful for `random_predictors = FALSE`; `NA` otherwise), and `true_f2` (Cohen's $f^2 = \rho^2/(1-\rho^2)$ at `true_R2`). For a power-at-specified-N computation the first row is `specified_N` instead of `necessary_N`.

Note

When determining sample size for a desired degree of power, there will always be a slightly larger degree of actual power. This is the case because the algorithm employed determines sample size until the actual power is no less than the desired power (given sample size is a whole number power will almost certainly not be exactly the specified value). This is the same as other statistical power procedures that return whole numbers for necessary sample size.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Fisher, R. A. (1928). The general sampling distribution of the multiple correlation coefficient. *Proceedings of the Royal Society of London, Series A, 121*(788), 654–673. doi:10.1098/rspa.1928.0224

Gatsonis, C., & Sampson, A. R. (1989). Multiple correlation: Exact power and sample size calculations. *Psychological Bulletin, 106*(3), 516–524.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*(4), 524–555. doi:10.1080/00273170802490632

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Lee, Y. S. (1971). Some results on the sampling distribution of the multiple correlation coefficient. *Journal of the Royal Statistical Society, Series B, 33*(1), 117–130.

Maxwell, S. E. (2000). Sample size and multiple regression analysis. *Psychological Methods, 5*(4), 434–458. doi:10.1037/1082989X.5.4.434

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Anderson, S. F., Kelley, K., & Maxwell, S. E. (2017). Sample-size planning for more accurate statistical power: A method adjusting sample effect sizes for publication bias and uncertainty. *Psychological Science, 28*(11), 1547–1562. doi:10.1177/0956797617723724

See Also

ss_power_R2_change for the test of predictors added to a model, ss_aipe_R2, ss_power_R2_sensitivity, ss_power_reg_coef, ci_nc_F, ci_R2

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_s… ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# Random predictors (default; appropriate for most behavioral / social
# science applications).
ss_power_R2(true_R2 = .5, alpha_level = .05, desired_power = .85, p = 5)

# Fixed predictors (Cohen 1988): predictor variables fixed by design.
ss_power_R2(true_R2 = .5, alpha_level = .05, desired_power = .85,
            p = 5, random_predictors = FALSE)

# Effect size input (Cohen's f^2).
ss_power_R2(true_f2 = 1, alpha_level = .05, desired_power = .85, p = 5)
```

```
# Realized power at a specified N.
ss_power_R2(true_R2 = .5, specified_N = 15, alpha_level = .05,
            desired_power = .85, p = 5)

# The number MBESS reported for a population value of .30 against a null
# value of .25 with p = 4: the omnibus test of four fixed predictors with
# Cohen's f squared for that increment. The test of the increment itself
# is planned with ss_power_R2_change().
ss_power_R2(true_f2 = (.30 - .25) / (1 - .30), p = 4, specified_N = 85,
            alpha_level = .01, random_predictors = FALSE)
```

`ss_power_R2_change` *Plan Sample Size to Make the Test of a Change in the Squared Multiple Correlation Coefficient Sufficiently Powerful*

Description

Determine the necessary sample size, or the statistical power at a specified sample size, for the model comparison F-test that a set of predictors adds nothing to a regression model that already contains other predictors. The restricted model has `p_restricted` predictors and the full model adds one or more predictors to it, for `p_full` in all. The null hypothesis is that the two models have the same population squared multiple correlation coefficient, so that the change in R^2 is zero. This is the test of an increment in R^2 in hierarchical regression (Cohen, Cohen, West, & Aiken, 2003), with the effect size f^2 of Cohen (1988). `ss_power_R2` plans the omnibus test of all predictors at once, and `ss_power_reg_coef` the test of one coefficient with fixed predictors.

Usage

```
ss_power_R2_change(
  true_R2_full = NULL,
  true_R2_restricted = NULL,
  true_R2_change = NULL,
  true_f2 = NULL,
  p_full,
  p_restricted,
  alpha_level = 0.05,
  desired_power = 0.85,
  specified_N = NULL,
  random_predictors = TRUE
)
```

Arguments

`true_R2_full`	Population squared multiple correlation coefficient of the outcome with all `p_full` predictors of the full model, at least 0 and below 1.

`true_R2_restricted`
: Population squared multiple correlation coefficient of the outcome with the `p_restricted` predictors of the restricted model. The restricted model's predictors are a subset of the full model's, so `true_R2_restricted` cannot exceed `true_R2_full`.

`true_R2_change`
: Population change in R^2 when the added predictors join the restricted model, `true_R2_full` minus `true_R2_restricted`. It is the squared semipartial correlation of the added predictor when one predictor is added, and the squared multiple semipartial correlation of the added set when several are. Give it with `true_R2_full` or with `true_R2_restricted`.

`true_f2`
: Cohen's (1988) effect size for the increment, $f^2 = (\rho^2_{full} - \rho^2_{restricted})/(1 - \rho^2_{full})$, a nonnegative number. Give it alone, in place of the R^2 values.

`p_full`
: Number of predictors in the full model, not counting the intercept. A whole number of at least 1.

`p_restricted`
: Number of predictors in the restricted model, not counting the intercept. A whole number from 0 to `p_full - 1`. Zero is the model with an intercept only, for which the test is the omnibus test of `ss_power_R2`.

`alpha_level`
: Type I error rate.

`desired_power`
: Desired statistical power, a single number strictly between 0 and 1 (0 and 1 themselves are refused, with a condition of class `dmar_desired_power_invalid`); used when a sample size is sought. With `specified_N` the power is computed and `desired_power` is not used, but a value outside (0, 1) is still refused.

`specified_N`
: Total sample size at which to report power, in place of a search for the necessary sample size. A whole number of at least `p_full + 2`, the smallest size that leaves an error degree of freedom.

`random_predictors`
: Whether the predictors are treated as random (`TRUE`, the default) or fixed (`FALSE`). See Details.

Details

The effect. Write ρ^2_{full} and $\rho^2_{restricted}$ (`true_R2_full` and `true_R2_restricted`) for the two population squared multiple correlation coefficients, q for the number of predictors removed from the full model to give the restricted model ($q = p_{full} - p_{restricted}$, the predictors being tested; row `q` of the result, and u in Cohen, 1988), and w for the number in the restricted model, `p_restricted`. In the partial and semipartial correlation functions (`var_partial_r`, `var_semipartial_r`, and their planners) p is `p_full`, the number of variables partialed out is $p - q$, and one predictor is of interest, so $q = 1$. The change $\rho^2_{full} - \rho^2_{restricted}$ is the squared multiple correlation of the outcome with the part of the added predictors that the restricted model's predictors do not explain: the squared multiple semipartial correlation of the added set (Cohen et al., 2003, Section 5.4.3), and with one added predictor its squared semipartial correlation (Kelley & Maxwell, 2008, Equation 20, p. 176; Aloe & Becker, 2012, Equation 1). So the answer to whether the change in R^2 is the squared semipartial correlation is yes, for one added predictor, and for a set it is the multiple version of the same quantity. Dividing the change by $1 - \rho^2_{restricted}$ gives the squared multiple partial correlation (the squared partial correlation for one added predictor). Cohen's effect size divides the change by the full model's unexplained variance,

$$f^2 = \frac{\rho^2_{full} - \rho^2_{restricted}}{1 - \rho^2_{full}},$$

which is also the squared multiple partial correlation divided by one minus itself. Power depends on the two R^2 values only through f^2, so the ways of giving the effect are equivalent and exactly one is given: any two of the three R^2 quantities (`true_R2_full`, `true_R2_restricted`, `true_R2_change`), or `true_f2` alone. The same change is a larger effect when the full model explains more: a change of .05 is $f^2 = .071$ when the full model explains .30 and $f^2 = .25$ when it explains .80.

The test. The statistic is

$$F = \frac{(R^2_{full} - R^2_{restricted})/q}{(1 - R^2_{full})/(N - p_{full} - 1)},$$

with q and $N - p_{full} - 1$ degrees of freedom (the form of Cohen et al., 2003, Equation 5.5.1), the general model comparison *F* of Maxwell, Delaney, and Kelley (2027, Chapter 3) applied to the two regression models. With one added predictor it is the square of the *t*-statistic of that predictor's coefficient. In the notation of Maxwell, Delaney, and Kelley (2027), the restricted model has $df_R = N - p_{restricted} - 1$ error degrees of freedom and the full model $df_F = N - p_{full} - 1$; the numerator degrees of freedom are $df_R - df_F = q$ (rows `q` and `df_1`), and the denominator degrees of freedom are df_F (row `df_2`). The function takes the numbers of predictors, since the error degrees of freedom depend on the sample size being planned. A categorical predictor with *a* levels counts as *a* - 1 predictors.

Fixed predictors. When the predictor values are held constant in hypothetical replications of the study, *F* follows a noncentral *F* distribution with q and $N - p_{full} - 1$ degrees of freedom and noncentrality $\lambda = Nf^2$, where the R^2 values are those of the design itself (computed from the design's sums of squares, with divisor N). The calculation is exact under normal errors. With one added predictor this is the square of the noncentrality of Kelley and Maxwell (2008, Equations 16 and 19, p. 175, and Equation 31, p. 181), who give $\lambda_j = f_j\sqrt{N}$ for the *t*-test of one coefficient with fixed predictors. That is the noncentrality of `ss_power_reg_coef`, whose power this function reproduces when one predictor is added: the noncentral *t* of that function has noncentrality $\sqrt{N}f$, whose square is Nf^2. For any number of added predictors Nf^2 is also the noncentrality of `ss_power_R2` with fixed predictors.

Random predictors. When the predictors are sampled along with the outcome from a multivariate normal population, the sample squared multiple partial correlation of the outcome with the q added predictors, given the w others, has the distribution that a sample squared multiple correlation has with q predictors and a sample size of $N - w$ (Gatsonis & Sampson, 1989, p. 518). The critical value is the same as with fixed predictors, since under the null hypothesis the statistic has the central *F* distribution in both models; only the power differs. The power is computed from that distribution with the noncentral *F* approximation of Lee (1971, Section 5.1) that `ss_power_R2` and `ci_R2` use. Against the exact distribution, the power by Lee's approximation differed by at most .0043 over the combinations examined (squared partial correlations up to .95, `alpha_level` from .001 to .10, and one or more error degrees of freedom) and by at most .0004 with 25 or more error degrees of freedom; the necessary sample size differed from the one the exact distribution gives, by one case, in 20 of 1,500 searches, and in those the power by the exact distribution at the size returned was within .0011 of `desired_power`. At the levels of power used in planning, power is lower and the necessary sample size larger than with fixed predictors, and the difference grows with the number of predictors in the restricted model relative to the sample size. When power is very low the order can reverse by a small amount (at most .0005 in 1,920 combinations examined, all with no predictors in the restricted model, `alpha_level` of .001, and power below .09). Predictors are random in most studies in the behavioral, educational, and social sciences, so `random_predictors = TRUE` is the default.

Relation to the test of one regression coefficient. With one added predictor and `random_predictors = FALSE` this function and `ss_power_reg_coef` plan the same test and return the same sample size and power; `true_R2_Y_X` there is `true_R2_full` here and `true_R2_Y_X_without_j` is `true_R2_restricted`, with `p` equal to `p_full` and `p_restricted = p_full - 1`. `ss_power_reg_coef()` has fixed predictors only, so with the default of this function (random predictors) the sample size here is larger: 56 against 50 for a change from .30 to .40 with five predictors and power .80 (50 is the example of Kelley & Maxwell, 2008, pp. 181–182). The two functions report the effect size differently: `true_f2` here is f^2, and `true_f` there is f.

Cohen's (1988) noncentrality. Cohen computes power from $\lambda = f^2(u + v + 1)$, with u the number of predictors tested (q here) and v the error degrees of freedom (as given by Maxwell, 2000, Equation 1). For the omnibus test $q + v + 1 = N$. For an increment $v = N - q - w - 1$, so Cohen's noncentrality is $f^2(N - w)$, smaller than Nf^2 by wf^2. Cohen's value equals the fixed-predictor calculation for q predictors, no restricted model, and `specified_N` set to $N - w$, and the last example reproduces it that way. That $N - w$ is only the number that makes the noncentrality agree; it is not the study's sample size, which remains N. The necessary sample size by Cohen's noncentrality exceeds the fixed-predictor answer of this function by at most w cases. Maxwell (2000, p. 436) notes that Nf^2 is the noncentrality for fixed predictors and reads Cohen's form as an adjustment toward random predictors; the random-predictor calculation here makes that adjustment from the sampling distribution itself. MBESS's `ss.power.R2()` with a nonzero `Null.R2` computes this quantity when `p` is given as the number of predictors added: the sample size it reports is Cohen's total sample size less w.

Error from a larger model. The error variance of the test can be taken from a model with further predictors that are not part of the comparison (the Model 2 error of Cohen et al., 2003, Section 5.5.4). With fixed predictors that calculation is obtained by giving `true_f2` with the larger model's unexplained variance in its denominator, `p_full` as the number of predictors in that larger model, and `p_restricted` as `p_full` minus the number of predictors tested. Set `random_predictors = FALSE` for this calculation. The random-predictor calculation is for the comparison of two nested models only.

Edge cases. When the change is zero the null hypothesis is true: at a `specified_N` the power returned is `alpha_level`, and a search stops with an error of class `dmar_effect_zero`, because no sample size attains `desired_power`. A `specified_N` below `p_full + 2` leaves no error degrees of freedom and is an error, as is one that is not a whole number. With `p_restricted = 0` the restricted model explains nothing, so `true_R2_restricted`, if given, must be 0 and `true_R2_change`, if given, must equal `true_R2_full`; the result is then that of `ss_power_R2`. A search starts at `p_full + 2` and returns that size when it already attains `desired_power`. The search does not go beyond a sample size of ten million and stops with an error when the power is not attained by then. R computes the noncentral F distribution accurately for a noncentrality parameter up to about one million and warns beyond it; no power is returned from an evaluation at which R gave that warning, and the error says so.

Estimation. To plan for a narrow confidence interval rather than for power when one predictor is added, see `ss_aipe_semipartial_r`; its `true_r_sp` squared is `true_R2_change` and its `p` is `p_full` (that function counts the predictor of interest and the variables partialed from it, and one predictor is added, so `p_restricted + 1 = p_full`). DMAR has no accuracy in parameter estimation planner for the change in R^2 from a set of predictors.

Value

A `data.frame` with columns `term` and `value` and nine rows, in this order:

`necessary_N` The smallest total sample size at which power is no less than `desired_power`. When `specified_N` was given the row is named `specified_N` and holds that value.

`actual_power` Power at that sample size.

`noncentral_f_parm` The noncentrality parameter Nf^2 under fixed predictors; `NA` under random predictors, where the power does not come from a single noncentral F with that parameter.

`true_f2` The planning value of Cohen's f^2 for the increment: `true_f2` as given, or the value the R^2 planning values imply.

`true_partial_R2` The population squared multiple partial correlation the planning values imply, $f^2/(1+f^2)$.

`true_R2_change` The population change in R^2; `NA` when `true_f2` was given, which does not determine it.

`q` The number of predictors removed from the full model to give the restricted model, `p_full` minus `p_restricted`: the predictors being tested.

`df_1` Numerator degrees of freedom, equal to `q`.

`df_2` Denominator degrees of freedom at the sample size reported, $N - p_{full} - 1$.

The result carries the `dmar_ss_power` class, so `tidy` and `glance` apply, and the attribute `random_predictors` records the predictor model.

Note

The sample size returned is the smallest whole number at which power is no less than `desired_power`, so `actual_power` is slightly above the target. With random predictors that power is the one Lee's approximation gives; the paragraph on random predictors in Details says how closely that approximation follows the exact distribution.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Aloe, A. M., & Becker, B. J. (2012). An effect size for regression predictors in meta-analysis. *Journal of Educational and Behavioral Statistics, 37*(2), 278–297. doi:10.3102/1076998610396901

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum. (See Chapter 9.)

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Lawrence Erlbaum. (See Sections 5.4 and 5.5.)

Gatsonis, C., & Sampson, A. R. (1989). Multiple correlation: Exact power and sample size calculations. *Psychological Bulletin, 106*(3), 516–524.

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Lee, Y. S. (1971). Some results on the sampling distribution of the multiple correlation coefficient. *Journal of the Royal Statistical Society, Series B, 33*(1), 117–130.

Maxwell, S. E. (2000). Sample size and multiple regression analysis. *Psychological Methods, 5*(4), 434–458. doi:10.1037/1082989X.5.4.434

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on the general form of the model comparison *F*-test.)

See Also

ss_power_R2, ss_power_reg_coef, ss_aipe_semipartial_r, var_semipartial_r, ci_R2

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_s ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# Five predictors are expected to explain .25 of the variance and four
# added predictors to raise that to .30. Necessary sample size for power
# .80 at alpha .01, with random predictors (the default).
ss_power_R2_change(true_R2_restricted = .25, true_R2_full = .30,
                   p_restricted = 5, p_full = 9, alpha_level = .01,
                   desired_power = .80)

# The same plan with predictors fixed by design.
ss_power_R2_change(true_R2_restricted = .25, true_R2_full = .30,
                   p_restricted = 5, p_full = 9, alpha_level = .01,
                   desired_power = .80, random_predictors = FALSE)

# The effect given as the change with the full model's value, and as
# Cohen's f squared. Power at a sample size of 90, fixed predictors.
ss_power_R2_change(true_R2_change = .05, true_R2_full = .30,
                   p_restricted = 5, p_full = 9, alpha_level = .01,
                   specified_N = 90, random_predictors = FALSE)
ss_power_R2_change(true_f2 = .05 / .70, p_restricted = 5, p_full = 9,
                   alpha_level = .01, specified_N = 90,
                   random_predictors = FALSE)

# The effect given as the restricted model's value and the change, the
# way a hierarchical regression is often described. Random predictors.
ss_power_R2_change(true_R2_restricted = .25, true_R2_change = .05,
                   p_restricted = 5, p_full = 9, alpha_level = .01,
                   desired_power = .80)

# One added predictor with fixed predictors is the test of its regression
# coefficient: the two functions return the same sample size.
ss_power_R2_change(true_R2_restricted = .30, true_R2_full = .40,
                   p_restricted = 4, p_full = 5, desired_power = .80,
```

```
                    random_predictors = FALSE)
ss_power_reg_coef(true_R2_Y_X = .40, true_R2_Y_X_without_j = .30, p = 5,
                  desired_power = .80)

# Cohen's (1988) noncentrality for the design of the third example,
# f squared times (N - w) with w = 5: four predictors and no restricted
# model, with specified_N set to 90 - 5 so the noncentrality matches
# Cohen's. The study's sample size is still 90.
ss_power_R2_change(true_f2 = .05 / .70, p_restricted = 0, p_full = 4,
                   alpha_level = .01, specified_N = 85,
                   random_predictors = FALSE)
```

`ss_power_R2_sensitivity` *Sensitivity Analysis for Sample Size Planning to Make the Omnibus Test of R^2 Sufficiently Powerful*

Description

Monte Carlo sensitivity analysis for the power of the omnibus F-test of the squared multiple correlation coefficient (R^2). Given a `supposed_R2` used for sample size planning and a `true_R2` that actually obtains in the population (the two need not agree), the function draws `G` replications, fits the regression, compares F to its critical value, and reports the realized empirical power and a summary of the realized R^2 and F distributions. The simulation honors the same `random_predictors` / `generate_random_predictors` crossing as `ss_aipe_R2_sensitivity`, so the user can examine the effect of planning under one regression model (fixed or random predictors) but actually realizing the other.

Usage

```
ss_power_R2_sensitivity(
  true_R2 = NULL,
  supposed_R2 = NULL,
  desired_power = 0.85,
  p = NULL,
  alpha_level = 0.05,
  random_predictors = TRUE,
  specified_N = NULL,
  generate_random_predictors = TRUE,
  true_cor_YX = 0.3,
  true_cor_XX = 0.3,
  G = 10000,
  print_iter = TRUE,
  filename = NULL,
  seed = NULL
)
```

Arguments

true_R2	Value of the population squared multiple correlation coefficient
supposed_R2	Value of the squared multiple correlation coefficient used for sample size planning. Either supposed_R2 or specified_N must be supplied (not both).
desired_power	Desired degree of statistical power used for planning
p	Number of predictors
alpha_level	Type I error rate
random_predictors	Whether the sample size planning step treats predictors as random (TRUE, the default) or fixed (FALSE)
specified_N	Sample size at which the realized power should be computed; alternative to specifying supposed_R2
generate_random_predictors	Whether the internal simulation should generate predictors as random (TRUE, the default) or fixed (FALSE)
true_cor_YX	Correlation between the dependent variable (Y) and each of the X variables
true_cor_XX	Correlation among the X variables (off-diagonal of the predictor correlation matrix)
G	Number of Monte Carlo replications
print_iter	Whether to print the iteration number during the simulation
filename	Optional path of a CSV file to receive the per-replication results (the observed R^2 R2 and F statistic F_value), overwriting any file already at that path; the default NULL writes nothing, and a throwaway run that wants the file should point it at tempfile(fileext = ".csv").
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

When supposed_R2 equals true_R2, the function performs a straight Monte Carlo evaluation of the planning procedure (no misspecification). Pass specified_N to evaluate realized power at a specified sample size; in that case supposed_R2 must not be supplied. The crossing of random_predictors (used in planning) with generate_random_predictors (used in the simulation) lets the user inspect the consequences of planning under one regression model but realizing the other. See Gatsonis and Sampson (1989) for the comparison of fixed and random predictor power for the omnibus test.

Value

A data.frame with columns term and value summarizing the Monte Carlo sensitivity analysis across G replications. The term entries are: total_N (the sample size evaluated), empirical_power (the proportion of replications on which F exceeded the critical value), analytic_power (computed from ss_power_R2 under the same model as planning), mean_R2 / median_R2 / sd_R2 and

`mean_F_value` / `median_F_value` / `sd_F_value` (summaries of the realized R^2 and F), `F_crit` (the critical value), and the input echoes `p`, `true_R2`, `supposed_R2` and `desired_power` (both NA when `specified_N` was supplied instead), and `alpha_level`. The result carries the `dmar_ss_power_sensitivity` class, so `tidy` reports the planned sample size beside the empirical and analytic power, and `glance` adds the simulated R^2 and F distribution beside the echoed inputs.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (`pct_ci_less_width`), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute `simulation`, a list read with `attr(x, "simulation")`:

`attempted` the number of replications asked for (`G`).

`returned` the number that produced an interval (a test).

`failed` the number that did not (`attempted` - `returned`).

`denominator` the number of replications the rows named above are taken over, which is `returned`.

`width_target_probability` the proportion of the replications with an interval whose interval met the width target, the value of the `pct_ci_less_width` row; `NA` for a function or a call without a width target, and when no replication produced an interval.

`width_target_mc_ci` the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as `G` grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives `G`, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so `failed` is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Gatsonis, C., & Sampson, A. R. (1989). Multiple correlation: Exact power and sample size calculations. *Psychological Bulletin, 106*(3), 516–524.

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Lee, Y. S. (1971). Some results on the sampling distribution of the multiple correlation coefficient. *Journal of the Royal Statistical Society, Series B, 33*(1), 117–130.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

See Also

ss_power_R2, ss_aipe_R2_sensitivity, ci_R2

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_s ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
set.seed(113)
# Realized power when planning under the fixed-predictor model but the
# data are actually generated with random predictors. G = 200 keeps the
# example quick; a reported analysis deserves the default G of 10000.
ss_power_R2_sensitivity(true_R2 = 0.30, supposed_R2 = 0.30,
                        desired_power = 0.80, p = 5,
                        random_predictors = FALSE,
                        generate_random_predictors = TRUE,
                        G = 200, print_iter = FALSE)
```

---

ss_power_rc *Sample Size for a Targeted Regression Coefficient*

Description

Determine the necessary sample size for a targeted regression coefficient or determine the degree of power given a specified sample size.

Usage

```
ss_power_rc(
  true_R2_Y_X = NULL,
  true_R2_Y_X_without_j = NULL,
  p = NULL,
  desired_power = 0.85,
  alpha_level = 0.05,
  directional = FALSE,
```

```
  true_reg_coef_j = NULL,
  true_sd_X = NULL,
  true_sd_Y = NULL,
  true_R2_j_X_without_j = NULL,
  true_cor_XX = NULL,
  true_cor_YX = NULL,
  which_predictor = NULL,
  true_f2 = NULL,
  specified_N = NULL,
  print_iter = FALSE
)
```

Arguments

true_R2_Y_X	Population squared multiple correlation coefficient predicting the dependent variable (i.e., Y) from the p predictor variables (i.e., the X variables)
true_R2_Y_X_without_j	Population squared multiple correlation coefficient predicting the dependent variable (i.e., Y) from the p-1 predictor variables, where the one not used is the predictor of interest
p	Number of predictor variables in the full model, a single whole number of at least 1. It is required when the effect is given through true_R2_Y_X, true_reg_coef_j, or true_f2, and is not given with true_cor_XX and true_cor_YX, whose dimension sets it
desired_power	Desired degree of statistical power for the test of targeted regression coefficient
alpha_level	Type I error rate
directional	Whether or not a direction or a nondirectional test is to be used (usually directional=FALSE)
true_reg_coef_j	Population value of the regression coefficient for the predictor of interest, the name it has in ss_aipe_rc and ss_aipe_reg_coef. It is on the scale that true_sd_X and true_sd_Y describe: with both standard deviations 1 it is the standardized coefficient
true_sd_X	Population standard deviation for the predictor variable of interest
true_sd_Y	Population standard deviation for the outcome variable
true_R2_j_X_without_j	Population squared multiple correlation coefficient predicting the predictor variable of interest from the remaining p-1 predictor variables
true_cor_XX	Population correlation matrix for the p predictor variables
true_cor_YX	Population vector of correlation coefficient between the p predictor variables and the criterion variable
which_predictor	Identifies the predictor of interest when true_cor_XX and true_cor_YX are specified: a single whole number from 1 to p, its position in true_cor_XX
true_f2	Cohen's (1988) definition for an effect size for a targeted regression coefficient: (true_R2_Y_X - true_R2_Y_X_without_j) / (1 - true_R2_Y_X)

specified_N	Sample size for which power should be evaluated. This is the *total* sample size.
print_iter	If the progress of the iterative procedure is printed to the screen as the iterations are occurring

Details

Determines the necessary sample size given a desired level of statistical power. Alternatively, determines the statistical power for a given a specified sample size. There are a number of ways that the specification regarding the size of the regression coefficient can be entered. The most basic, and often the simplest, is to specify true_R2_Y_X and true_R2_Y_X_without_j. See the examples section for several options.

The test is the case of ss_power_R2_change with one predictor added and fixed predictors; that function also plans the test with random predictors and the test of a set of added predictors.

The arguments are checked, and the result computed, by ss_power_reg_coef, whose Details say how a single predictor given through true_cor_XX and an effect of exactly 0 are handled.

Value

A tidy data.frame with a term column and a numeric value column, forwarded unchanged from ss_power_reg_coef: rows for necessary_N (the necessary total sample size) or specified_N (when specified_N is supplied), actual_power, noncentral_t_parm (the noncentrality of the t distribution), and true_f (Cohen's f at the planning values, the square root of true_f2, since true_f2 is the effect size on the F scale). The result carries the dmar_ss_power class, so tidy and glance summarize it in broom convention.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Anderson, S. F., Kelley, K., & Maxwell, S. E. (2017). Sample-size planning for more accurate statistical power: A method adjusting sample effect sizes for publication bias and uncertainty. *Psychological Science, 28*(11), 1547–1562. doi:10.1177/0956797617723724

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E. (2000). Sample size and multiple regression analysis. *Psychological Methods, 5*(4), 434–458. doi:10.1037/1082989X.5.4.434

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

See Also

ss_aipe_reg_coef, ss_power_R2, ss_power_R2_change, ci_nc_F

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
Cor.Mat <- rbind(
  c(1.00, 0.53, 0.58, 0.60, 0.46, 0.66),
  c(0.53, 1.00, 0.35, 0.07, 0.14, 0.43),
  c(0.58, 0.35, 1.00, 0.18, 0.29, 0.50),
  c(0.60, 0.07, 0.18, 1.00, 0.30, 0.26),
  c(0.46, 0.14, 0.29, 0.30, 1.00, 0.30),
  c(0.66, 0.43, 0.50, 0.26, 0.30, 1.00))

true_cor_XX <- Cor.Mat[2:6, 2:6]
true_cor_YX <- Cor.Mat[1, 2:6]

# Method 1
ss_power_rc(true_R2_Y_X = 0.7826786, true_R2_Y_X_without_j = 0.7363697, p = 5,
            alpha_level = .05, directional = FALSE, desired_power = .80)

# Method 2
ss_power_rc(alpha_level = .05, true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
            which_predictor = 5,
            directional = FALSE, desired_power = .80)

# Method 3
# Here, true_reg_coef_j is the standardized regression coefficient. Had true_reg_coef_j
# been the unstandardized regression coefficient, true_sd_X and true_sd_Y
# would have been the standard deviation for the X variable of interest
# and Y, respectively.
ss_power_rc(true_R2_Y_X = 0.7826786, true_R2_j_X_without_j = 0.3652136,
            true_reg_coef_j = 0.2700964, p = 5,
            alpha_level = .05, true_sd_X = 1, true_sd_Y = 1,
            directional = FALSE, desired_power = .80)
# Method 4
```

```
ss_power_rc(alpha_level = .05, true_f2 = 0.2130898, p = 5,
            directional = FALSE, desired_power = .80)
# Power given a specified N and squared multiple correlation coefficients.
ss_power_rc(true_R2_Y_X = 0.7826786, true_R2_Y_X_without_j = 0.7363697, specified_N = 25, p = 5,
            alpha_level = .05, directional = FALSE)

# Power given a specified N and effect size.
ss_power_rc(alpha_level = .05, true_f2 = 0.2130898, p = 5, specified_N = 25, directional = FALSE)

# Reproducing Maxwell's (2000, p. 445) Example
Cor.Mat.Maxwell <- rbind(
    c(1.00, 0.35,  0.20, 0.20, 0.20, 0.20),
    c(0.35, 1.00,  0.40, 0.40, 0.40, 0.40),
    c(0.20, 0.40,  1.00, 0.45, 0.45, 0.45),
    c(0.20, 0.40,  0.45, 1.00, 0.45, 0.45),
    c(0.20, 0.40,  0.45, 0.45, 1.00, 0.45),
    c(0.20, 0.40,  0.45, 0.45, 0.45, 1.00)
)

RHO.XX.Maxwell <- Cor.Mat.Maxwell[2:6, 2:6]
Rho.YX.Maxwell <- Cor.Mat.Maxwell[1, 2:6]
R2.Maxwell <- Rho.YX.Maxwell %*% solve(RHO.XX.Maxwell) %*% Rho.YX.Maxwell

RHO.XX.Maxwell.no.1 <- Cor.Mat.Maxwell[3:6, 3:6]
Rho.YX.Maxwell.no.1 <- Cor.Mat.Maxwell[1, 3:6]
R2.Maxwell.no.1 <- Rho.YX.Maxwell.no.1 %*% solve(RHO.XX.Maxwell.no.1) %*% Rho.YX.Maxwell.no.1


# Note that Maxwell arrives at N=113, whereas this procedure arrives at 111.
# This seems to be the case becuase of rounding error in calculations
# and tables (Cohen, 1988) used. The present procedure is correct and
# contains no rounding error in the application of the method.
ss_power_rc(true_R2_Y_X = R2.Maxwell, true_R2_Y_X_without_j = R2.Maxwell.no.1, p = 5,
            alpha_level = .05, directional = FALSE, desired_power = .80)
```

`ss_power_reg_coef` *Sample Size for a Targeted Regression Coefficient*

Description

Determine the necessary sample size for a targeted regression coefficient or determine the degree of power given a specified sample size

Usage

```
ss_power_reg_coef(
  true_R2_Y_X = NULL,
  true_R2_Y_X_without_j = NULL,
  p = NULL,
```

```
  desired_power = 0.85,
  alpha_level = 0.05,
  directional = FALSE,
  true_reg_coef_j = NULL,
  true_sd_X = NULL,
  true_sd_Y = NULL,
  true_R2_j_X_without_j = NULL,
  true_cor_XX = NULL,
  true_cor_YX = NULL,
  which_predictor = NULL,
  true_f2 = NULL,
  specified_N = NULL,
  print_iter = FALSE
)
```

Arguments

`true_R2_Y_X`	Population squared multiple correlation coefficient predicting the dependent variable (i.e., Y) from the `p` predictor variables (i.e., the X variables)
`true_R2_Y_X_without_j`	Population squared multiple correlation coefficient predicting the dependent variable (i.e., Y) from the `p`-1 predictor variables, where the one not used is the predictor of interest
`p`	Number of predictor variables in the full model, a single whole number of at least 1. It is required when the effect is given through `true_R2_Y_X`, `true_reg_coef_j`, or `true_f2`, and is not given with `true_cor_XX` and `true_cor_YX`, whose dimension sets it
`desired_power`	Desired degree of statistical power for the test of targeted regression coefficient
`alpha_level`	Type I error rate
`directional`	Whether or not a direction or a nondirectional test is to be used (usually `directional=FALSE`)
`true_reg_coef_j`	Population value of the regression coefficient for the predictor of interest, the name it has in `ss_aipe_reg_coef` and `ss_aipe_rc`. It is on the scale that `true_sd_X` and `true_sd_Y` describe: with both standard deviations 1 it is the standardized coefficient
`true_sd_X`	Population standard deviation for the predictor variable of interest
`true_sd_Y`	Population standard deviation for the outcome variable
`true_R2_j_X_without_j`	Population squared multiple correlation coefficient predicting the predictor variable of interest from the remaining p-1 predictor variables
`true_cor_XX`	Population correlation matrix for the `p` predictor variables: symmetric, with 1 on the diagonal, and positive definite; another matrix is refused with a condition of class `dmar_argument_invalid`
`true_cor_YX`	Population vector of correlation coefficient between the `p` predictor variables and the criterion variable; with `true_cor_XX` it must imply a squared multiple correlation below 1 (the correlation matrix of the outcome and the predictors is then positive definite)

`which_predictor`
: Identifies the predictor of interest when `true_cor_XX` and `true_cor_YX` are specified: a single whole number from 1 to `p`, its position in `true_cor_XX`

`true_f2`
: Cohen's (1988) definition for an effect size for a targeted regression coefficient: `(true_R2_Y_X-true_R2_Y_X_without_j)/(1-true_R2_Y_X)`

`specified_N`
: Sample size for which power should be evaluated. This is the *total* sample size.

`print_iter`
: If the progress of the iterative procedure is printed to the screen as the iterations are occurring

Details

Determines the necessary sample size given a desired level of statistical power. Alternatively, determines the statistical power for a given a specified sample size. There are a number of ways that the specification regarding the size of the regression coefficient can be entered. The most basic, and often the simplest, is to specify `true_R2_Y_X` and `true_R2_Y_X_without_j`. See the examples section for several options.

Power is computed from a noncentral t distribution with noncentrality $\sqrt{N}\, f$, which treats the predictors as fixed (their values held constant across hypothetical replications). This is the standard fixed-predictor power analysis; under random predictors, where the predictor values themselves vary from sample to sample, the sample size required for a given level of power is somewhat larger. The test is the case of `ss_power_R2_change` with one predictor added and fixed predictors (`true_R2_Y_X` is its `true_R2_full` and `true_R2_Y_X_without_j` its `true_R2_restricted`); that function also plans the test with random predictors and the test of a set of added predictors.

With `true_cor_XX` and `true_cor_YX`, the squared multiple correlation of the full model is $\rho'_{YX}\mathbf{R}_{XX}^{-1}\rho_{YX}$, where ρ_{YX} is `true_cor_YX` and $\mathbf{R}_{XX}$ is `true_cor_XX`, and that of the model without the predictor of interest is the same quantity with the predictor's row and column removed from $\mathbf{R}_{XX}$ and its element from ρ_{YX}. With a single predictor (`p` of 1) the model without it has no predictors and its squared multiple correlation is 0, so the effect is that of `true_f2` $= \rho^2/(1-\rho^2)$, where ρ is `true_cor_YX`.

When the planning values give the predictor an effect of exactly 0 (`true_f2` of 0, equal squared multiple correlations, or a `true_reg_coef_j` of 0), the null hypothesis is true and the t statistic has the central t distribution at every sample size, so the probability that the test rejects is `alpha_level`, for the two-sided test and for the directional one alike. A sample size is then defined only for a `desired_power` no greater than `alpha_level`, and it is the smallest admissible one, `p` + 2; a larger `desired_power` is refused at once with a condition of class `dmar_effect_zero`. With `specified_N` the power at that size is returned, which is `alpha_level`. For an effect that is not 0 the power rises with the sample size, and a `desired_power` that no sample size up to 10,000,000 attains is refused at once with a condition of class `dmar_computation_failed`.

Value

`ss`
: Either the necessary sample size or the specified sample size, depending if one is interested in determining the necessary sample size given a desired degree of statistical power or if one is interested in the determining the value of statistical power given a specified sample size, respectively

`actual_power`
: Actual power of the situation described

noncentral_t_parm
: Value of the noncentral distribution for the appropriate *t*-distribution

true_f
: Cohen's *f* at the planning values, the effect size for the noncentral *t*-distribution; this is the square root of true_f2, because true_f2 is the effect size using an *F*-distribution. When the effect is given through true_reg_coef_j it carries the sign of the coefficient.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E. (2000). Sample size and multiple regression analysis. *Psychological Methods, 5*(4), 434–458. doi:10.1037/1082989X.5.4.434

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

Anderson, S. F., Kelley, K., & Maxwell, S. E. (2017). Sample-size planning for more accurate statistical power: A method adjusting sample effect sizes for publication bias and uncertainty. *Psychological Science, 28*(11), 1547–1562. doi:10.1177/0956797617723724

See Also

ss_aipe_reg_coef, ss_power_R2, ss_power_R2_change, ci_nc_F

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
Cor.Mat <- rbind(
  c(1.00, 0.53, 0.58, 0.60, 0.46, 0.66),
  c(0.53, 1.00, 0.35, 0.07, 0.14, 0.43),
  c(0.58, 0.35, 1.00, 0.18, 0.29, 0.50),
  c(0.60, 0.07, 0.18, 1.00, 0.30, 0.26),
  c(0.46, 0.14, 0.29, 0.30, 1.00, 0.30),
  c(0.66, 0.43, 0.50, 0.26, 0.30, 1.00)
)

true_cor_XX <- Cor.Mat[2:6, 2:6]
true_cor_YX <- Cor.Mat[1, 2:6]

# Method 1
ss_power_reg_coef(true_R2_Y_X = 0.7826786, true_R2_Y_X_without_j = 0.7363697, p = 5,
                  alpha_level = .05, directional = FALSE, desired_power = .80)

# Method 2
ss_power_reg_coef(alpha_level = .05, true_cor_XX = true_cor_XX, true_cor_YX = true_cor_YX,
                  which_predictor = 5,
                  directional = FALSE, desired_power = .80)

# Method 3
# Here, true_reg_coef_j is the standardized regression coefficient. Had true_reg_coef_j
# been the unstandardized regression coefficient, true_sd_X and true_sd_Y
# would have been the standard deviation for the X variable of
# interest and Y, respectively.
ss_power_reg_coef(true_R2_Y_X = 0.7826786, true_R2_j_X_without_j = 0.3652136,
                  true_reg_coef_j = 0.2700964,
               p = 5, alpha_level = .05, true_sd_X = 1, true_sd_Y = 1, directional = FALSE,
                  desired_power = .80)

# Method 4
ss_power_reg_coef(alpha_level = .05, true_f2 = 0.2130898, p = 5,
                  directional = FALSE, desired_power = .80)

# Power given a specified N and squared multiple correlation coefficients.
ss_power_reg_coef(true_R2_Y_X = 0.7826786, true_R2_Y_X_without_j = 0.7363697, specified_N = 25,
                  p = 5, alpha_level = .05, directional = FALSE)

# Power given a specified N and effect size.
ss_power_reg_coef(alpha_level = .05, true_f2 = 0.2130898, p = 5, specified_N = 25,
                  directional = FALSE)

# Reproducing Maxwell's (2000, p. 445) Example
Cor.Mat.Maxwell <- rbind(
  c(1.00, 0.35, 0.20, 0.20, 0.20, 0.20),
  c(0.35, 1.00, 0.40, 0.40, 0.40, 0.40),
  c(0.20, 0.40, 1.00, 0.45, 0.45, 0.45),
  c(0.20, 0.40, 0.45, 1.00, 0.45, 0.45),
  c(0.20, 0.40, 0.45, 0.45, 1.00, 0.45),
  c(0.20, 0.40, 0.45, 0.45, 0.45, 1.00)
```

```
)

RHO.XX.Maxwell <- Cor.Mat.Maxwell[2:6, 2:6]
Rho.YX.Maxwell <- Cor.Mat.Maxwell[1, 2:6]
R2.Maxwell <- Rho.YX.Maxwell %*% solve(RHO.XX.Maxwell) %*% Rho.YX.Maxwell

RHO.XX.Maxwell.no.1 <- Cor.Mat.Maxwell[3:6, 3:6]
Rho.YX.Maxwell.no.1 <- Cor.Mat.Maxwell[1, 3:6]
R2.Maxwell.no.1 <-
  Rho.YX.Maxwell.no.1 %*% solve(RHO.XX.Maxwell.no.1) %*% Rho.YX.Maxwell.no.1

# This procedure arrives at N = 111, whereas Maxwell (2000, p. 445)
# reports N = 113. The two differ because of the noncentrality
# parameterization, not rounding: this function uses the fixed-predictor
# noncentrality sqrt(N) * f (see Details), while the tabled value rests on
# Cohen's (1988) convention. Neither is a random-predictor result; under
# random predictors, where the predictor values vary across replications,
# the sample size needed for the same power is larger still.
ss_power_reg_coef(true_R2_Y_X = R2.Maxwell, true_R2_Y_X_without_j = R2.Maxwell.no.1, p = 5,
                  alpha_level = .05, directional = FALSE, desired_power = .80)
```

`ss_power_reg_coef_sensitivity`

Sensitivity Analysis for the Power of a Targeted Regression Coefficient

---

Description

Monte Carlo sensitivity analysis for the statistical power of the *t*-test of a targeted regression coefficient. Given a planned (`supposed_*`) covariance structure and a true (`true_*`) covariance structure, the function draws `G` replications, fits the multiple regression, and reports the empirical proportion of replications on which the *t*-test of the targeted coefficient rejects, together with the realized distribution of the estimated unstandardized coefficient $\hat{b}_j$, its standard error, and the test statistic. `ss_power_reg_coef_sensitivity()` is the power-oriented sibling of `ss_aipe_reg_coef_sensitivity` (which is CI-width oriented).

Usage

```
ss_power_reg_coef_sensitivity(
  true_var_Y = NULL,
  true_cov_YX = NULL,
  true_cov_XX = NULL,
  supposed_var_Y = NULL,
  supposed_cov_YX = NULL,
  supposed_cov_XX = NULL,
  specified_N = NULL,
  which_predictor = 1,
  desired_power = 0.85,
  alpha_level = 0.05,
```

```
  directional = FALSE,
  G = 1000,
  print_iter = TRUE,
  filename = NULL,
  seed = NULL
)
```

Arguments

true_var_Y	Population variance of the dependent variable (Y)
true_cov_YX	Population covariance vector between the p predictor variables and the dependent variable (Y)
true_cov_XX	Population covariance matrix of the p predictor variables
supposed_var_Y	Supposed variance of the dependent variable (Y), the planning value used in sample size planning. Defaults to true_var_Y.
supposed_cov_YX	Supposed covariance vector between the predictor variables and the dependent variable, the planning values used in sample size planning. Defaults to true_cov_YX.
supposed_cov_XX	Supposed covariance matrix of the predictor variables, the planning values used in sample size planning. Defaults to true_cov_XX.
specified_N	Directly specified sample size; if supplied, sample size planning is skipped.
which_predictor	Index identifying which of the p predictors is the targeted predictor for the power test.
desired_power	Desired degree of statistical power used for planning
alpha_level	Type I error rate
directional	Whether a one-sided or two-sided test is used
G	Number of Monte Carlo replications
print_iter	Whether to print the iteration number during the simulation
filename	Optional path of a CSV file to receive the per-replication results (the coefficient estimate reg_coef_j, its standard error se_reg_coef_j, the t statistic t_value, and the observed R^2 R2), overwriting any file already at that path; the default NULL writes nothing, and a throwaway run that wants the file should point it at tempfile(fileext = ".csv").
seed	NULL (the default) to draw from the current random number stream, or a single number that makes the call reproducible; a supplied seed is set for this call only, and the caller's random number state is restored when the function returns.

Details

When the supposed and true covariance structures are identical, the function performs a Monte Carlo evaluation of the planning procedure (no misspecification); when they differ, it performs a sensitivity analysis on the consequences of misspecifying the population covariance structure for

the targeted coefficient's power. The planning step calls ss_power_reg_coef with the supposed covariance structure; the simulation step generates data from the true covariance structure.

The coefficient is the unstandardized one, b_j, the jth element of $\Sigma_{XX}^{-1}\sigma_{XY}$, from true_cov_XX and true_cov_YX for the true value and from their supposed counterparts for the supposed value, and every replication is fit in the units of the data. The t statistic of a coefficient is the same whether or not the data are standardized, so the power is that of the test of the standardized coefficient as well; the standardized coefficient itself, $b_j\sigma_{X_j}/\sigma_Y$ (Kelley & Maxwell, 2003, p. 316), and its precision are the subject of ss_aipe_src_sensitivity.

Value

A data.frame with columns term and value summarizing the Monte Carlo sensitivity analysis. The term entries are: total_N (the sample size evaluated), empirical_power, analytic_power (computed from ss_power_reg_coef), the mean / median / SD of the realized $\hat{b}_j$ (mean_reg_coef_j, median_reg_coef_j, sd_reg_coef_j), of its standard error (mean_se_reg_coef_j, median_se_reg_coef_j, sd_se_reg_coef_j), of the test statistic (mean_t_value, median_t_value, sd_t_value), and of the squared multiple correlation coefficient (mean_R2, median_R2, sd_R2), t_crit (the critical value), and the input echoes p, which_predictor, true_reg_coef_j and supposed_reg_coef_j (the population and planning values of the targeted coefficient implied by the supplied covariance structures), desired_power (NA when specified_N was supplied instead), and alpha_level. The result carries the dmar_ss_power_sensitivity class, so tidy reports the planned sample size beside the empirical and analytic power, and glance adds the simulated estimator distribution beside the echoed inputs.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

width_target_mc_ci the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as G grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives G, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so failed is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Kelley, K., & Maxwell, S. E. (2008). Sample size planning with applications to multiple regression: Power and accuracy for omnibus and targeted effects. In P. Alasuutari, L. Bickman, & J. Brannen (Eds.), *The Sage handbook of social research methods* (pp. 166–192). Sage.

Maxwell, S. E. (2000). Sample size and multiple regression analysis. *Psychological Methods, 5*(4), 434–458. doi:10.1037/1082989X.5.4.434

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons of means and Chapter 6 on trend analysis.)

See Also

ss_power_reg_coef, ss_aipe_reg_coef_sensitivity

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# Targeted coefficient power sensitivity with two predictors. The
# default G = 1000 replications is used in practice; G is reduced here
# so the example runs quickly.
set.seed(113)
cov_XX  <- matrix(c(1, 0.3, 0.3, 1), nrow = 2)
cov_YX  <- c(0.4, 0.3)
ss_power_reg_coef_sensitivity(
  true_var_Y = 1, true_cov_YX = cov_YX, true_cov_XX = cov_XX,
  which_predictor = 1, desired_power = 0.80,
  G = 100, print_iter = FALSE
)
```

`ss_power_rm_anova` *Sample Size or Power for a One-Way Repeated Measures ANOVA Omnibus* F *Test*

Description

Determine the necessary number of subjects (the total sample size N) to achieve a desired level of statistical power for the omnibus *F* test of the within-subjects factor in a one-way repeated measures ANOVA, or, given a number of subjects, return the realized statistical power.

Usage

```
ss_power_rm_anova(
  a,
  true_f = NULL,
  true_eta_squared = NULL,
  true_cor = 0,
  true_epsilon = 1,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_N = NULL
)
```

Arguments

`a`	Number of measurement occasions (levels of the within-subjects factor)
`true_f`	Cohen's *f* effect size for the within-subjects factor (the population value); supply this or `true_eta_squared`, but not both
`true_eta_squared`	Population eta squared (proportion of variance, on the relevant scale, accounted for by the within-subjects factor); supply this or `true_f`
`true_cor`	Average correlation among the repeated measures (default 0). With `true_cor > 0`, the within-subjects test gains efficiency relative to a between-subjects analog
`true_epsilon`	Greenhouse-Geisser / Huynh-Feldt sphericity adjustment in (0, 1] (default 1, sphericity assumed). When `true_epsilon < 1`, both numerator and denominator degrees of freedom are multiplied by `true_epsilon`
`desired_power`	Desired statistical power (default 0.85)
`alpha_level`	Type I error rate (default 0.05)
`specified_N`	Total sample size: the number of subjects, each measured at all `a` occasions; if specified, the function returns the realized power

Details

Under the alternative hypothesis with sphericity (`true_epsilon = 1`), the within-subjects *F* statistic follows a noncentral *F* distribution with numerator df $a - 1$, denominator df $(N - 1)(a - 1)$, and

noncentrality parameter $\lambda = Naf^2/(1-\rho)$, where N is the number of subjects, f is Cohen's f (`true_f`) for the within-subjects effect and ρ is the average correlation across the repeated measures (`true_cor`; Maxwell, Delaney, & Kelley, 2027). Setting `true_cor = 0` reduces to the between-subjects expression.

When sphericity is violated, supplying `true_epsilon` (e.g., a Greenhouse-Geisser estimate) rescales the test using the Muller-Barton convention: both numerator and denominator degrees of freedom are multiplied by `true_epsilon`, and the noncentrality parameter is likewise multiplied by `true_epsilon`. Smaller `true_epsilon` therefore reduces power and increases the necessary sample size.

Value

A `data.frame` with rows for `necessary_N` (or `specified_N`), the number of subjects, `a`, `df_effect`, `df_error`, `noncentral_f_parm`, and `actual_power`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention (the reported size is the number of subjects).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

See Also

`ss_power_one_way_anova`, `ss_power_pcm`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_composite_sem()`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`, `ss_power_reg_coef_sensitivity()`, `ss_power_sc()`, `ss_power_sem()`, `ss_power_smd()`, `ss_power_split_plot_anova()`

Examples

```
# 4 measurement occasions, f = 0.25, average within-subject correlation 0.5, power = .80
ss_power_rm_anova(a = 4, true_f = 0.25, true_cor = 0.5, desired_power = 0.80)

# Same but with Greenhouse-Geisser epsilon = 0.75
ss_power_rm_anova(a = 4, true_f = 0.25, true_cor = 0.5, true_epsilon = 0.75, desired_power = 0.80)
```

```
# Realized power at N = 20 subjects, a = 4
ss_power_rm_anova(a = 4, true_f = 0.25, true_cor = 0.5, specified_N = 20)
```

`ss_power_sc` *Sample Size or Power for a Standardized Contrast in a One-Way Between-Subjects ANOVA*

Description

Determine the necessary per-group sample size to achieve a desired level of statistical power for the test of a single planned standardized contrast in a one-way between-subjects analysis of variance, or, given a per-group sample size, return the realized statistical power.

Usage

```
ss_power_sc(
  true_std_contrast,
  c_weights,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_group = NULL,
  directional = FALSE
)
```

Arguments

`true_std_contrast`	The population standardized contrast effect, ψ/σ, where σ is the within-group standard deviation
`c_weights`	Vector of contrast weights (must sum to zero); use fractional weights so that the positive weights sum to 1 (e.g., `c(0.5, 0.5, -0.5, -0.5)`)
`desired_power`	Desired statistical power (default 0.85)
`alpha_level`	Type I error rate (default 0.05)
`specified_n_per_group`	Per-group sample size (assumed balanced); if specified, returns the realized power
`directional`	Logical: `TRUE` for a one-sided test (in the same sign as `true_std_contrast`), `FALSE` (default) for a two-sided test

Details

Under the alternative hypothesis the contrast *t*-statistic follows a noncentral *t*-distribution with degrees of freedom $N-J$ ($J =$`length(c_weights)`) and noncentrality parameter $\lambda = \psi^*/\sqrt{\sum c_j^2/n}$, where ψ^* is the standardized contrast, `true_std_contrast`.

The function searches over per-group sample sizes n until power first reaches `desired_power`; when `specified_n_per_group` is supplied it returns the realized power.

Value

A data.frame with rows for necessary_n_per_group (or specified_n_per_group), actual_power, and noncentral_t_parm. The result carries the dmar_ss_power class, so tidy and glance summarize it in broom convention.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Lai, K., & Kelley, K. (2012). Accuracy in parameter estimation for ANCOVA and ANOVA contrasts: Sample size planning via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 350–370. doi:10.1111/j.20448317.2011.02029.x

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

ss_power_c, ss_power_one_way_anova, ci_sc, ss_aipe_sc

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sem(), ss_power_smd(), ss_power_split_plot_anova()

Examples

```
# Power for a standardized contrast of 0.5 across 4 groups,
# contrast (G1 + G2)/2 vs (G3 + G4)/2, desired power = .80
ss_power_sc(true_std_contrast = 0.5, c_weights = c(0.5, 0.5, -0.5, -0.5),
            desired_power = 0.80)

# Realized power at n = 30 per group
ss_power_sc(true_std_contrast = 0.5, c_weights = c(0.5, 0.5, -0.5, -0.5),
            specified_n_per_group = 30)
```

ss_power_sem *Sample Size Planning for Structural Equation Modeling From the Power Analysis Perspective*

Description

Calculate the necessary sample size for an SEM study, so as to have enough power to reject the null hypothesis that (a) the model has perfect fit, or (b) the difference in fit between two nested models equal some specified amount.

Usage

```
ss_power_sem(
  true_F_ML = NULL,
  df = NULL,
  null_rmsea = NULL,
  true_rmsea = NULL,
  true_F_full = NULL,
  true_F_restricted = NULL,
  true_rmsea_full = NULL,
  true_rmsea_restricted = NULL,
  df_full = NULL,
  df_restricted = NULL,
  alpha_level = 0.05,
  desired_power = 0.85
)
```

Arguments

true_F_ML	The true maximum likelihood fit function value in the population for the model of interest. Leave this argument NULL if you are doing nested model significance tests
df	The degrees of freedom of the model of interest. Leave this argument NULL if you are doing nested model significance tests
null_rmsea	The model's population RMSEA under the null hypothesis. Leave this argument NULL if you are doing nested model significance tests
true_rmsea	The model's population RMSEA under the alternative hypothesis. This should be the model's true population RMSEA value. Leave this argument NULL if you are doing nested model significance tests
true_F_full	The population maximum likelihood fit function value for the full model
true_F_restricted	The population maximum likelihood fit function value for the restricted model
true_rmsea_full	The population RMSEA value for the full model

`true_rmsea_restricted`
: The population RMSEA value for the restricted model

`df_full`
: The degrees of freedom for the full model

`df_restricted`
: The degrees of freedom for the restricted model

`alpha_level`
: The Type I error rate. Defaults to 0.05.

`desired_power`
: The desired power. Defaults to 0.85, matching the rest of the `ss_power_*` family.

Value

A `data.frame` with a `necessary_N` row, the smallest integer N whose power reaches `desired_power` under the supplied fit-function or RMSEA alternative, and an `actual_power` row giving the realized power at that N.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130–149. doi:10.1037/1082989X.1.2.130

Lai, K., & Kelley, K. (2011). Accuracy in parameter estimation for targeted effects in structural equation modeling: Sample size planning for narrow confidence intervals. *Psychological Methods, 16*(2), 127–148. doi:10.1037/a0021764

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

See Also

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_composite_sem()`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`, `ss_power_reg_coef_sensitivity()`, `ss_power_rm_anova()`, `ss_power_sc()`, `ss_power_smd()`, `ss_power_split_plot_anova()`

Examples

```
# One-model test: necessary N to reject H0: RMSEA = 0 in favor of
# a model whose population RMSEA is 0.05 at 80% power, alpha = .05,
# with df = 20.
```

```
ss_power_sem(null_rmsea = 0, true_rmsea = 0.05, df = 20,
             alpha_level = 0.05, desired_power = 0.80)

# Equivalent input via the population fit function, which is the
# degrees of freedom times the squared population RMSEA.
ss_power_sem(true_F_ML = 20 * 0.05^2, df = 20, alpha_level = 0.05, desired_power = 0.80)

# Two-model nested test: necessary N to detect the difference
# between a full model (RMSEA = 0.04, df = 18) and a restricted
# model (RMSEA = 0.06, df = 22) at 80% power.
ss_power_sem(true_rmsea_full = 0.04, df_full = 18,
             true_rmsea_restricted = 0.06, df_restricted = 22,
             alpha_level = 0.05, desired_power = 0.80)
```

`ss_power_smd` *Sample Size or Power for a Standardized Mean Difference (Two Independent Groups)*

Description

Determine the necessary per-group sample size to achieve a desired level of statistical power for the two-sample (independent groups) *t*-test on a standardized mean difference (Cohen's *d*; equivalently Hedges' *g* and Glass's *g* for sample size purposes). Alternatively, given a per-group sample size, return the realized statistical power.

Usage

```
ss_power_smd(
  true_smd,
  desired_power = 0.85,
  alpha_level = 0.05,
  n_1 = NULL,
  n_2 = NULL,
  directional = FALSE
)
```

Arguments

`true_smd`	Supposed standardized mean difference (Cohen's *d*) the design is planned against: a value the researcher posits for the population, either a minimally important effect or a value believed to be true in the population, never a sample estimate. Echoed in the returned table as the `true_smd` row.
`desired_power`	Desired statistical power (default 0.85)
`alpha_level`	Type I error rate (default 0.05)
`n_1`	Sample size for group 1 (if specified, the function returns the realized power; assumes `n_2 = n_1` unless `n_2` is also given)

n_2	Sample size for group 2 (defaults to n_1 when n_1 is supplied)
directional	Logical: TRUE for a one-sided test (in the same sign as true_smd), FALSE (default) for a two-sided test

Details

The two-sample t-statistic with pooled standard deviation follows a noncentral t-distribution with $n_1 + n_2 - 2$ degrees of freedom and noncentrality parameter $\lambda = \delta\sqrt{n_1 n_2/(n_1 + n_2)}$, where δ is the population standardized mean difference. For balanced designs ($n_1 = n_2 = n$) this simplifies to $\lambda = \delta\sqrt{n/2}$.

Power is computed as the probability that the absolute value of the test statistic exceeds the critical value(s) under the alternative; the function returns the per-group sample size for which power first reaches desired_power.

Kelley and Rausch (2006) develop the accuracy in parameter estimation approach to planning the sample size for the standardized mean difference, implemented in ss_aipe_smd.

Value

A data.frame with term and value columns. The design result comes first, followed by rows that echo the user-supplied planning inputs, so the assumptions the power was evaluated under travel with the result. The true_smd row is the supposed effect the plan is built on: a value the researcher posits, either a minimally important effect or a value believed to be true in the population, never a sample estimate. The tails row is 2 for a nondirectional test and 1 for a directional test.

When n_1 **is** NULL Result rows necessary_n_per_group, actual_power, and noncentral_t_parm, then the planning inputs true_smd, desired_power, alpha_level, and tails.

When n_1 **is specified** Result rows specified_n_1, specified_n_2, actual_power, and noncentral_t_parm, then the planning inputs true_smd, alpha_level, and tails (the supplied group sizes are the specified_n_1 / specified_n_2 rows).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Kelley, K., Maxwell, S. E., & Rausch, J. R. (2003). Obtaining power or obtaining precision: Delineating methods of sample-size planning. *Evaluation & the Health Professions, 26*(3), 258–287. doi:10.1177/0163278703255242

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Maxwell, S. E., Kelley, K., & Rausch, J. R. (2008). Sample size planning for statistical power and accuracy in parameter estimation. *Annual Review of Psychology, 59*, 537–563. doi:10.1146/annurev.psych.59.103006.093735

See Also

ss_aipe_smd, ci_smd, smd, ci_nc_t

design_consequences for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: power_fisher_exact(), ss_aipe_mixed_effects(), ss_power_R2(), ss_power_R2_change(), ss_power_R2_sensitivity(), ss_power_c(), ss_power_c_ancova(), ss_power_composite_ancova(), ss_power_composite_ancova_2group(), ss_power_composite_anova(), ss_power_composite_factorial_ancova(), ss_power_composite_factorial_ancova_het(), ss_power_composite_factorial_anova(), ss_power_composite_sem(), ss_power_contrast(), ss_power_equivalence_c(), ss_power_factorial_ancova(), ss_power_factorial_anova(), ss_power_indirect_effect(), ss_power_mixed_effects(), ss_power_one_way_anova(), ss_power_pcm(), ss_power_r(), ss_power_rc(), ss_power_reg_coef(), ss_power_reg_coef_sensitivity(), ss_power_rm_anova(), ss_power_sc(), ss_power_sem(), ss_power_split_plot_anova()

Examples

```
# Per-group sample size for d = 0.5, alpha = .05, power = .80, two-sided
ss_power_smd(true_smd = 0.5, desired_power = 0.80)

# Same with a directional (one-sided) test
ss_power_smd(true_smd = 0.5, desired_power = 0.80, directional = TRUE)

# Realized power given balanced n = 30 per group
ss_power_smd(true_smd = 0.5, n_1 = 30)

# Realized power for unbalanced (n_1 = 30, n_2 = 50)
ss_power_smd(true_smd = 0.5, n_1 = 30, n_2 = 50)
```

---

ss_power_split_plot_anova

Sample Size or Power for a Mixed-Effects ANOVA (Between X Within Design)

Description

Determine the necessary per-group sample size to achieve a desired level of statistical power for one of the three F tests in a mixed-effects ANOVA with one between-subjects factor and one within-subjects factor (the between-subjects main effect, the within-subjects main effect, or the between x within interaction), or, given a per-group sample size, return the realized statistical power. (This design is also commonly called a split-plot factorial.)

Usage

```
ss_power_split_plot_anova(
  a,
  b,
  effect,
  true_f = NULL,
  true_eta_squared_partial = NULL,
  true_cor,
  true_epsilon = 1,
  desired_power = 0.85,
  alpha_level = 0.05,
  specified_n_per_group = NULL
)
```

Arguments

a	Number of levels of the between-subjects factor (i.e., number of groups)
b	Number of levels of the within-subjects factor (i.e., number of measurement occasions)
effect	Which F test to compute power for: "between", "within", or "interaction"
true_f	Cohen's f effect size for the chosen effect (the population value); supply this or true_eta_squared_partial, but not both
true_eta_squared_partial	Population partial eta squared for the chosen effect; supply this or true_f
true_cor	Average correlation among the repeated measures within a subject (must lie in (-1, 1)). Higher true_cor reduces power for the between-subjects test (because subject means contain more redundant information) and increases power for the within-subjects and interaction tests
true_epsilon	Greenhouse-Geisser / Huynh-Feldt sphericity adjustment in (0, 1] (default 1, sphericity assumed). Applied to the within-subjects and interaction tests but not the between-subjects test. Both numerator and denominator df, and the noncentrality, are multiplied by true_epsilon (Muller-Barton convention)
desired_power	Desired statistical power (default 0.85)
alpha_level	Type I error rate (default 0.05)
specified_n_per_group	Per-group (between-subjects) sample size; if specified, returns the realized power

Details

This is a two-factor mixed-effects design: one between-subjects factor with a levels and one within-subjects factor with b levels; n subjects are randomly assigned to each between-subjects level and each subject is measured at all b within-subjects levels, for $N = na$ subjects total and Nb observations. The covariance among the b within-subject observations is summarized by true_cor, the average pairwise correlation.

The three F tests have noncentrality parameters

$$\lambda_B = Nbf^2/(1 + (b-1)\rho)$$

for the between-subjects test (numerator df $a - 1$, denominator df $N - a$),

$$\lambda_W = Nbf^2\,\epsilon/(1-\rho)$$

for the within-subjects test (numerator df $(b-1)\epsilon$, denominator df $(N-a)(b-1)\epsilon$), and the same form as λ_W for the interaction (numerator df $(a-1)(b-1)\epsilon$, same denominator df). Cohen's f relates to partial eta squared via $f = \sqrt{\eta_p^2/(1-\eta_p^2)}$. In these formulas f is `true_f` (or the value implied by `true_eta_squared_partial`, which is η_p^2), ρ is `true_cor`, ϵ is `true_epsilon`, and n is the per-group size (`specified_n_per_group`, or the size the function plans).

This design is the compound-symmetry (random intercept) special case of the two-level linear mixed-effects model: `true_cor` is the intraclass correlation and the b occasions are the observations within a subject (i.e., the Level 1 units). For two between-subjects groups the between-subjects $F(1, .)$ test is therefore the two-level treatment t test of `ss_power_mixed_effects` squared, so the two planners agree on that shared case.

Value

A `data.frame` with rows for `necessary_n_per_group` (or `specified_n_per_group`), `total_N`, `df_effect`, `df_error`, `noncentral_f_parm`, and `actual_power`. The result carries the `dmar_ss_power` class, so `tidy` and `glance` summarize it in broom convention (the reported size is the per-group count).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Muller, K. E., & Barton, C. N. (1989). Approximate power for repeated measures ANOVA lacking sphericity. *Journal of the American Statistical Association, 84*, 549–555.

See Also

`ss_power_one_way_anova`, `ss_power_factorial_anova`, `ss_power_rm_anova`, `ss_power_mixed_effects`

`design_consequences` for what a chosen design delivers: power, the Type S (sign) and Type M (exaggeration) errors of the significance filter, and the expected confidence interval width.

Other sample size for power: `power_fisher_exact()`, `ss_aipe_mixed_effects()`, `ss_power_R2()`, `ss_power_R2_change()`, `ss_power_R2_sensitivity()`, `ss_power_c()`, `ss_power_c_ancova()`, `ss_power_composite_ancova()`, `ss_power_composite_ancova_2group()`, `ss_power_composite_anova()`, `ss_power_composite_factorial_ancova()`, `ss_power_composite_factorial_ancova_het()`, `ss_power_composite_factorial_anova()`, `ss_power_composite_sem()`, `ss_power_contrast()`, `ss_power_equivalence_c()`, `ss_power_factorial_ancova()`, `ss_power_factorial_anova()`, `ss_power_indirect_effect()`, `ss_power_mixed_effects()`, `ss_power_one_way_anova()`, `ss_power_pcm()`, `ss_power_r()`, `ss_power_rc()`, `ss_power_reg_coef()`, `ss_power_reg_coef_sensitivity()`, `ss_power_rm_anova()`, `ss_power_sc()`, `ss_power_sem()`, `ss_power_smd()`

Other mixed models: R2_mixed_effects(), R2_mixed_effects_decomposition(), icc_lmer(), manova_split_plot(), mixed_anova(), ss_aipe_mixed_effects(), ss_aipe_mixed_effects_sensitivity(), ss_power_mixed_effects()

Examples

```
# 2 groups, 4 occasions, between-subjects effect, f = 0.25,
# average within-subject correlation 0.5, power = .80
ss_power_split_plot_anova(a = 2, b = 4, effect = "between", true_f = 0.25,
                    true_cor = 0.5, desired_power = 0.80)

# Same design, within-subjects (occasion) main effect, f = 0.25
ss_power_split_plot_anova(a = 2, b = 4, effect = "within", true_f = 0.25,
                    true_cor = 0.5, desired_power = 0.80)

# Same design, between x within interaction, f = 0.25
ss_power_split_plot_anova(a = 2, b = 4, effect = "interaction", true_f = 0.25,
                    true_cor = 0.5, desired_power = 0.80)

# Realized power for n = 25 per group on the interaction test, partial eta^2 = 0.06
ss_power_split_plot_anova(a = 2, b = 4, effect = "interaction",
              true_eta_squared_partial = 0.06, true_cor = 0.5, specified_n_per_group = 25)

# Greenhouse-Geisser correction with epsilon = 0.7 on the within-subjects test
ss_power_split_plot_anova(a = 2, b = 4, effect = "within", true_f = 0.25,
                    true_cor = 0.5, true_epsilon = 0.7, desired_power = 0.80)
```

---

ss_seq_c *Sequential Sample Size for a Fixed-Width Contrast Interval*

Description

Implements a purely sequential fixed-width confidence interval procedure for a linear contrast $\psi = \sum_j c_j \mu_j$: the stopping rule of Chow and Robbins (1965) applied to a contrast, with the groups allocated by the cost-optimal allocation of Chattopadhyay, Bandyopadhyay, Kelley, and Padalunkal (2025, Theorem 1). The goal is a 100(1 - 2α)% confidence interval $\hat{\psi} \pm h$ whose half-width h is fixed in advance, which is what makes a noninferiority or equivalence verdict reachable by design: with bounds $\pm\delta$, equivalence can never be declared unless $h < \delta$, and targeting $h = \delta/2$ gives a truly equivalent contrast about a 90% chance of being declared equivalent at $\alpha = .05$. Because no fixed sample size can guarantee a bounded-width interval when the error variance is unknown (Dantzig, 1940), the procedure is sequential: begin with a pilot, then keep sampling, re-estimating the variance, until the stopping criterion is met. Used with `pilot = TRUE` the function plans the pilot and the cost-optimal allocation; with `pilot = FALSE` it evaluates the stopping criterion at the data in hand, in the style of mr_smd.

Usage

```
ss_seq_c(
  c_weights,
  half_width,
  s = NULL,
  n_per_group = NULL,
  cost = NULL,
  alpha_level = 0.05,
  quantile = c("t", "normal"),
  pilot = FALSE,
  m0 = 10
)
```

Arguments

c_weights	The contrast weights. The weights must sum to zero with the positive weights summing to 1 and the negative weights to -1, so that half_width is on the raw scale of the response.
half_width	The target half-width h of the 100(1 - 2α)% confidence interval, in raw units of the response. For a noninferiority or equivalence decision at bounds $\pm\delta$, the recommended target is $\delta/2$.
s	The current estimate(s) of the error standard deviation: either a single pooled value or one value per group (aligned with c_weights). Required when pilot = FALSE; optional planning values when pilot = TRUE (used only to shape the allocation).
n_per_group	The current per-group sample sizes, aligned with c_weights. Required when pilot = FALSE.
cost	Optional per-observation sampling costs, one per group, aligned with c_weights. When supplied, the reported allocation is the cost-optimal $n_j \propto \lvert c_j \rvert \, \sigma_j / \sqrt{\text{cost}_j}$ (Chattopadhyay et al., 2025); with equal costs and standard deviations this reduces to allocation proportional to $\lvert c_j \rvert$.
alpha_level	One-sided rate per bound; the interval is at confidence level 1 - 2α. Default 0.05 (a 90% interval).
quantile	"t" (default) uses the *t* quantile on the current error degrees of freedom in the stopping criterion, a finite-sample refinement; "normal" uses the normal quantile of the classical statement of the rule (Chow & Robbins, 1965). The normal rule stops slightly early at wide targets, the finite-sample undershoot anticipated by Woodroofe (1977); the *t* rule corrects it at a small cost in sample size.
pilot	TRUE to plan the pilot stage; FALSE (default) to evaluate the stopping criterion at the current data.
m0	The minimum pilot sample size per group. Default 10. A pilot that is too small makes the variance estimate that drives the early steps unstable.

Details

The stopping criterion. Sampling stops at the first N for which

$$q^2 \sum_j c_j^2 s_j^2 / n_j \;\le\; h^2,$$

where q is the t or normal quantile at $1 - \alpha$. With a pooled s and equal allocation this is the Chow and Robbins (1965) rule specialized to a contrast; the procedure is asymptotically consistent (coverage approaches 1 - 2α) and first-order efficient (the mean stopping size approaches the oracle n^* an investigator with known variance would use). See `ss_seq_c_sensitivity` for a Monte Carlo evaluation of both properties.

Degrees of freedom. With a pooled `s` the criterion uses $\nu = N - J$. With per-group `s` it uses the Satterthwaite approximation, which is the appropriate error law when the variances are not assumed homogeneous.

Batches. Observations may be added in batches rather than one at a time; the asymptotic properties survive batching. Re-call the function after each batch.

When it applies. The balanced procedure of Kelley, Darku, and Chattopadhyay (2018, Example 6), developed without distributional assumptions, is a second sequential procedure for a contrast, and the two are built for different questions. `ss_seq_c()` targets the half-width of a 100(1 - 2α)% interval, the form a noninferiority or equivalence decision at stated bounds takes; it lets the groups grow at different rates, allocated by sampling cost and by the groups' standard deviations; and it offers a *t* refinement of its criterion. The balanced procedure targets the full width of a 100(1 - α)% interval and adds an observation to every group at each stage. Use `ss_seq_c()` for an equivalence or noninferiority question or when sampling costs differ across groups, and the balanced procedure for a narrow interval for a contrast in a balanced design. For the mean of a costly measurement with an inexpensive proxy available on a pool of units, the sequential rule is `ss_seq_two_phase`.

Value

With `pilot = TRUE`, a `data.frame` with row `pilot_n_per_group` (the pilot size for each group with a nonzero weight) followed by one `allocation_j` row per group giving the recommended sampling proportions for the accrual stage. With `pilot = FALSE`, rows `stop` (1 = the criterion is met, stop sampling; 0 = continue), `half_width_current` (the half-width the interval would have now), `half_width_target`, `N_current`, `N_projected` (the approximate total at which the criterion would be met under the current allocation, from the normal approximation), and the `allocation_j` rows for the next round of sampling.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., Bandyopadhyay, T., Kelley, K., & Padalunkal, J. J. (2025). A sequential approach for noninferiority or equivalence of a linear contrast under cost constraints. *Psychological Methods, 30*(2), 425–439. doi:10.1037/met0000570

Chow, Y. S., & Robbins, H. (1965). On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *The Annals of Mathematical Statistics, 36*(2), 457–462.

Dantzig, G. B. (1940). On the non-existence of tests of "Student's" hypothesis having power functions independent of σ. *The Annals of Mathematical Statistics, 11*(2), 186–192.

Kelley, K., Darku, F. B., & Chattopadhyay, B. (2018). Accuracy in parameter estimation for a general class of effect sizes: A sequential approach. *Psychological Methods, 23*(2), 226–243. doi:10.1037/met0000127

Mukhopadhyay, N., & de Silva, B. M. (2009). *Sequential methods and their applications*. CRC Press.

Woodroofe, M. (1977). Second order approximations for sequential point and interval estimation. *The Annals of Statistics, 5*(5), 984–995.

See Also

ss_seq_c_sensitivity, ss_aipe_c, ss_power_equivalence_c, equivalence_c, mr_smd, ss_seq_two_phase

Other sequential estimation: ss_seq_c_sensitivity(), ss_seq_two_phase()

Examples

```
# 1. Plan the pilot for a two-group contrast, target half-width 2.5
#    (bounds of 5 with the h = delta/2 rule):
ss_seq_c(c_weights = c(1, -1), half_width = 2.5, pilot = TRUE)

# 2. Evaluate the stopping criterion mid-study: pooled s = 15.4 at
#    n = 60 per group. Too imprecise to stop; the projection says
#    roughly how much further to go.
ss_seq_c(c_weights = c(1, -1), half_width = 2.5,
         s = 15.4, n_per_group = c(60, 60))

# 3. Costs differ: sampling the second group costs four times as
#    much per observation, so its share of new observations drops.
ss_seq_c(c_weights = c(1, -1), half_width = 2.5,
         s = c(15.4, 15.4), n_per_group = c(60, 60), cost = c(1, 4))
```

---

ss_seq_c_sensitivity *Monte Carlo Sensitivity of the Sequential Fixed-Width Procedure*

Description

Simulates the purely sequential fixed-width confidence interval procedure of ss_seq_c under a known data generating mechanism, reporting the distribution of the stopping sample size and the empirical coverage of the fixed-width interval. The two quantities to read are the ratio of the mean stopping size to the oracle n^* (first-order efficiency: the ratio approaches 1 as the target half-width shrinks) and the coverage (asymptotic consistency: coverage approaches 1 - 2α). The normal quantile rule stops slightly early at wide targets, the finite-sample undershoot anticipated by Woodroofe (1977); the *t* quantile rule corrects it at a small cost in sample size.

Usage

```
ss_seq_c_sensitivity(
  c_weights,
  half_width,
  true_sd,
  true_means = NULL,
  alpha_level = 0.05,
  quantile = c("t", "normal"),
  m0 = 10,
  G = 1000,
  seed = NULL
)
```

Arguments

c_weights	The contrast weights. The weights must sum to zero with the positive weights summing to 1 and the negative weights to -1.
half_width	The target half-width h of the $100(1 - 2\alpha)\%$ interval, in raw units of the response.
true_sd	The data generating error standard deviation: a single value applied to every group.
true_means	Optional vector of data generating group means, aligned with c_weights. Default all zero, so the true contrast is 0.
alpha_level	One-sided rate per bound; the interval is at confidence level $1 - 2\alpha$. Default 0.05.
quantile	"t" (default) or "normal"; see ss_seq_c.
m0	Pilot sample size per group. Default 10.
G	Number of Monte Carlo replications. Default 1000.
seed	Optional integer seed. Default NULL (the current RNG state is used). When supplied, the caller's RNG state is restored on exit.

Details

Simulation design. Each replication samples the groups with nonzero weights in balanced fashion (one observation per group per step) from normal populations with common true_sd, starting at m0 per group, and stops at the first step satisfying the ss_seq_c criterion with the pooled variance estimate. Equal growth is the cost-optimal allocation of Chattopadhyay, Bandyopadhyay, Kelley, and Padalunkal (2025, Theorem 1) when the costs, the standard deviations, and the sizes of the weights are equal; unequal costs change the optimal allocation but not the logic.

The oracle. With known σ and balanced allocation over the J_0 groups with nonzero weights, the fixed-width requirement is $n^* = z^2_{1-\alpha}\,\sigma^2 J_0 \sum_j c_j^2 / h^2$ in total. The sequential procedure spends about n^* without knowing σ, which is its point.

Value

A data.frame with rows n_star (the oracle total sample size an investigator with known σ would use), mean_N, median_N, sd_N (the stopping total across replications), ratio_mean_N_n_star,

coverage (the proportion of replications whose $\hat{\psi}_N \pm h$ interval covered the true contrast), se_coverage (its simulation standard error), and the input echoes half_width, true_contrast (the population contrast implied by c_weights and true_means), true_sd, alpha_level, and m0.

The record of the simulation

Every sensitivity function of the package summarizes its replications over one denominator: the replications that produced a confidence interval (for a power sensitivity function, a test). The proportion of intervals that met the width target (pct_ci_less_width), the proportions of intervals that missed the true value, and the summaries of the widths are all taken over those replications. A replication without an interval is in none of these rows. It is counted, and the returned table carries the counts in its attribute simulation, a list read with attr(x, "simulation"):

attempted the number of replications asked for (G).

returned the number that produced an interval (a test).

failed the number that did not (attempted - returned).

denominator the number of replications the rows named above are taken over, which is returned.

width_target_probability the proportion of the replications with an interval whose interval met the width target, the value of the pct_ci_less_width row; NA for a function or a call without a width target, and when no replication produced an interval.

width_target_mc_ci the exact binomial (Clopper-Pearson) limits for that proportion, at the confidence level of the table. They describe the error of the simulation, which shrinks as G grows, and nothing else.

A function that records more about its run adds elements to the list, and its help page names them. The line "Replications" beneath the printed table gives G, and the three counts when a replication was without an interval. With many replications without an interval the rows describe only the studies in which the interval could be formed, so failed is to be reported beside them.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chattopadhyay, B., Bandyopadhyay, T., Kelley, K., & Padalunkal, J. J. (2025). A sequential approach for noninferiority or equivalence of a linear contrast under cost constraints. *Psychological Methods, 30*(2), 425–439. doi:10.1037/met0000570

Chow, Y. S., & Robbins, H. (1965). On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *The Annals of Mathematical Statistics, 36*(2), 457–462.

Ghosh, M., Mukhopadhyay, N., & Sen, P. K. (1997). *Sequential estimation*. Wiley.

Woodroofe, M. (1977). Second order approximations for sequential point and interval estimation. *The Annals of Statistics, 5*(5), 984–995.

See Also

ss_seq_c, ss_aipe_c, ss_power_equivalence_c

Other sequential estimation: ss_seq_c(), ss_seq_two_phase()

Examples

```
# A two-group contrast, target half-width 2.5, error SD 15.67:
# the t-quantile rule stops near the oracle with near-nominal
# coverage. (G kept small here for speed; use G = 2000 or more in
# earnest.)
ss_seq_c_sensitivity(c_weights = c(1, -1), half_width = 2.5,
                     true_sd = 15.67, G = 200, seed = 113)
```

`ss_seq_two_phase` *Sequential Sample Size for a Fixed-Width Interval for a Mean Under Two-Phase Sampling*

Description

Evaluates, at the data in hand, a sequential stopping rule for a two-phase design: the proxy measurement is available on every unit of a pool, the costly target measurement is added to units drawn at random from the pool a few at a time, and sampling stops at the first number of labeled units, at or after a minimum of `m0`, at which the confidence interval from `ci_two_phase` has a half-width no larger than `half_width`. Unlike `ss_aipe_two_phase`, the rule needs no planning values for the target's standard deviation or its correlation with the proxy: the data supply both as they accrue. It follows `ss_seq_c` in its stopping rule, in the `half_width` and `alpha_level` arguments, and in the rows it returns (with the sizes named for the labeled units), and like it keeps no state: call it again after each batch. It differs from `ss_seq_c()` in three ways: it takes the data rather than summary statistics; it has no `quantile` argument, since its criterion is the half-width of the interval `ci_two_phase` reports, which uses the *t* quantile (the default of `ss_seq_c()`); and its minimum `m0` counts labeled units and defaults to 30 rather than `ss_seq_c()`'s 10 per group, because the slope as well as the variance must be estimated before the rule can be trusted.

Usage

```
ss_seq_two_phase(y, x, x_unlabeled, half_width, alpha_level = 0.05, m0 = 30)
```

Arguments

`y`	The target measurement on the units labeled so far.
`x`	The proxy measurement on the same units, aligned with `y`.
`x_unlabeled`	The proxy measurement on the units of the pool not yet labeled.
`half_width`	The target half-width h of the 100(1 - 2α)% confidence interval, in the units of the target. For a noninferiority or equivalence decision at bounds $\pm\delta$, `ss_seq_c` explains why $\delta/2$ is the usual choice.
`alpha_level`	One-sided rate per bound; the interval is at confidence level 1 - 2α. Default `0.05` (a 90% interval), as in `ss_seq_c`.
`m0`	The minimum number of labeled units before the rule may stop, default `30`. The early variance and slope estimates are unstable, and a rule allowed to stop on them stops too early too often.

Details

The rule. With n labeled units and a pool of M, sampling stops when $n \geq m_0$ and

$$t_{1-\alpha,\, n-2} \left\{ \frac{S_e^2}{n} + \frac{\hat{b}^2 S_X^2}{M} \right\}^{1/2} \leq h,$$

the left side being the half-width of the regression interval of `ci_two_phase` at confidence level 1 - 2α (S_e^2 the residual mean square of the target on the proxy, $\hat{b}$ the slope, S_X^2 the proxy's variance over the pool). This carries the fixed-width sequential approach of Chow and Robbins (1965), which `ss_seq_c` applies to a contrast, over to the two-phase interval. Labeled units must be added at random from the units not yet labeled, so that at every step they remain a simple random subsample of the pool. In an evaluation of a system whose outputs an automated judge labels, for example, a human assessor labels outputs drawn at random from those the judge has labeled until the rule says stop.

The projection. `n_labeled_projected` is the number of labeled units at which the rule would be met if the current estimates held, from the normal approximation $n = S_e^2/\{(h/z_{1-\alpha})^2 - \hat{b}^2 S_X^2/M\}$, the same device `ss_seq_c` uses: a guide to how much further to go, not a commitment. When the current estimates say that labeling every unit of the pool but one (the interval needs at least one unit with the proxy alone) would not reach the target, the projection is `NA` with a warning: the pool, not the labeling, is then the constraint.

A constant target, or one the proxy reproduces exactly. While every labeled unit has the same target value (for example, a binary target that has been 1 on every unit so far), the residual variation and the slope are both zero, and the interval would have zero width. While the target is an exact linear function of the proxy on the labeled units (for example, a binary judge and a binary human label that have agreed on every unit so far), the residual variation is zero and the width of the interval would come from the size of the pool alone. In both cases the rule could stop on no evidence, so the function stops with an error asking for more labeled units, whether or not the rule would have been met, as it does while the proxy is constant on them. The residual variance is taken to be zero when it is no larger than `.Machine$double.eps` times the variance of the target on the labeled units, which allows for rounding error.

Coverage. The stopping size is random and the interval at stopping is computed as though it had been fixed, so its coverage is not guaranteed in finite samples; it can fall short of nominal when n at stopping is small, or for a bounded target (a proportion, for example) whose variance depends on its mean. The minimum `m0` is the guard against the worst of this, and a simulation of the whole procedure in a plausible population is the check for a given design.

Value

A `data.frame` (class `dmar_tbl`) with rows `stop` (1 when the rule is met, so labeling stops; 0 to continue), `half_width_current` (the half-width of the interval now), `half_width_target`, `n_labeled_current` (the labeled units used, after any with a missing value are set aside, as in `ci_two_phase`), `n_labeled_projected`, and `n_pool` (M).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Chow, Y. S., & Robbins, H. (1965). On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *The Annals of Mathematical Statistics, 36*(2), 457–462. doi:10.1214/aoms/1177700156

Neyman, J. (1938). Contribution to the theory of sampling human populations. *Journal of the American Statistical Association, 33*(201), 101–116. doi:10.1080/01621459.1938.10503378

See Also

`ss_seq_c`, the sequential rule for a contrast; `ci_two_phase`; `ss_aipe_two_phase` for a design fixed in advance from planning values.

Other sequential estimation: `ss_seq_c()`, `ss_seq_c_sensitivity()`

Examples

```
# A proxy is available on a pool of 5,000 units; the target has been
# measured on a random 40 of them so far. Is a 90% interval with
# half-width 1 reached?
set.seed(113)
proxy  <- rnorm(5000, mean = 50, sd = 10)
target <- 20 + 0.6 * proxy + rnorm(5000, sd = 5)
order  <- sample(5000)
first  <- order[1:40]
ss_seq_two_phase(y = target[first], x = proxy[first],
                 x_unlabeled = proxy[-first], half_width = 1)

# Not yet; the projection suggests about 50. After labeling the next 30
# units at random:
first <- order[1:70]
ss_seq_two_phase(y = target[first], x = proxy[first],
                 x_unlabeled = proxy[-first], half_width = 1)
```

`summary_t_test` *Two-Sample* t *Test From Summary Statistics*

Description

Computes a two-sample *t* test (pooled or Welch) directly from the per-group means, standard deviations, and sample sizes, without requiring access to the raw observations. Returns the test statistic, degrees of freedom, *p*-value, and a CI on the mean difference in a `data.frame`. Useful for re-analyses from published papers that report only the summary numbers.

Usage

```
summary_t_test(
  mean_1,
  sd_1,
  n_1,
  mean_2,
  sd_2,
  n_2,
  null_mean_diff = 0,
  var_equal = TRUE,
  alternative = c("two_sided", "less", "greater"),
  conf_level = 0.95
)
```

Arguments

`mean_1`, `mean_2`	Group sample means.
`sd_1`, `sd_2`	Group sample standard deviations.
`n_1`, `n_2`	Group sample sizes.
`null_mean_diff`	Null value of the mean difference $\mu_1 - \mu_2$, μ_0 in the formulas below. Default `0`.
`var_equal`	Logical. If `TRUE` (default), uses Student's pooled-variance *t*. If `FALSE`, uses Welch's separate- variance *t* with Satterthwaite degrees of freedom.
`alternative`	One of `"two_sided"` (default; the base-R spelling `"two.sided"` is accepted as an alias), `"less"`, or `"greater"`.
`conf_level`	Confidence level for the CI on the mean difference. Default `0.95`.

Details

Pooled-variance *t* (Student, 1908). Under $\sigma_1 = \sigma_2$, the pooled SD is $s_p = \sqrt{((n_1 - 1)s_1^2 + (n_2 - 1)s_2^2)/(n_1 + n_2 - 2)}$, the test statistic is $t = (\bar{x}_1 - \bar{x}_2 - \mu_0)/(s_p\sqrt{1/n_1 + 1/n_2})$, and $df = n_1 + n_2 - 2$.

Welch's *t* (Welch, 1947). Under unequal variances, $t = (\bar{x}_1 - \bar{x}_2 - \mu_0)/\sqrt{s_1^2/n_1 + s_2^2/n_2}$ with Satterthwaite degrees of freedom (see `welch_t`).

Choosing pooled vs Welch. Delacre, Lakens, and Leys (2017) and Ruxton (2006) recommend Welch's test as the default. Pooled-variance *t* is preserved here primarily for reproducing analyses from older sources that used it.

Value

A `data.frame` with rows for the mean difference, the *t* statistic, degrees of freedom, *p*-value, and the CI lower and upper limits on the mean difference.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Delacre, M., Lakens, D., & Leys, C. (2017). Why psychologists should by default use Welch's *t*-test instead of Student's *t*-test. *International Review of Social Psychology, 30*(1), 92–101. doi:10.5334/irsp.82

Ruxton, G. D. (2006). The unequal variance *t*-test is an underused alternative to Student's *t*-test and the Mann-Whitney *U* test. *Behavioral Ecology, 17*(4), 688–690. doi:10.1093/beheco/ark016

Snedecor, G. W., & Cochran, W. G. (1989). *Statistical methods* (8th ed.). Iowa State University Press.

Student. (1908). The probable error of a mean. *Biometrika, 6*(1), 1–25. doi:10.2307/2331554

Welch, B. L. (1947). The generalization of "Student's" problem when several different population variances are involved. *Biometrika, 34*(1/2), 28–35.

See Also

`welch_t`, `t.test`, `smd`

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `welch_t()`

Examples

```
# 1. Re-analysis from published summary statistics:
#        Group A: M = 100, SD = 15, n = 30
#        Group B: M = 108, SD = 18, n = 25
summary_t_test(mean_1 = 100, sd_1 = 15, n_1 = 30,
               mean_2 = 108, sd_2 = 18, n_2 = 25)

# 2. Welch version for the same data:
summary_t_test(mean_1 = 100, sd_1 = 15, n_1 = 30,
               mean_2 = 108, sd_2 = 18, n_2 = 25,
               var_equal = FALSE)
```

---

summary_t_test_broom *Broom-Style Tidy / Glance Methods for* summary_t_test()

Description

`tidy()` returns the single mean-difference estimate and its confidence interval in the **broom** convention; `glance()` coincides with it, since a two-sample *t* test reports one estimand and there are no extra model-level statistics to add.

Usage

```
## S3 method for class 'dmar_summary_t_test'
tidy(x, ...)

## S3 method for class 'dmar_summary_t_test'
glance(x, ...)
```

Arguments

x	A dmar_summary_t_test object returned by summary_t_test.
...	Unused.

Value

A one-row data.frame with columns term, estimate, ci_lower, ci_upper, statistic, df, p_value, and conf_level.

Author(s)

Ken Kelley <kkelley@nd.edu>

---

teacher_expectancy *Teacher Expectancy Meta-Analysis Data (Raudenbush, 1984)*

Description

The 19 effect sizes from Raudenbush's (1984) synthesis of 18 experiments testing the effect of teacher expectancy on pupil IQ, the question *Pygmalion in the Classroom* raised (Rosenthal & Jacobson, 1968; the data of that study are shipped separately as pygmalion). In each experiment, teachers were told that randomly selected children were likely to bloom intellectually; the synthesis asks how large the resulting IQ advantage was and, centrally, why it varied across studies. Raudenbush's hypothesis was that the longer teachers had known their pupils before the expectancy induction, the less credible the induction and the smaller the effect (the credibility of expectancy induction in the paper's title). In these data the effect sizes fall as the weeks of prior contact rise ($r = -.55$ over the 18 studies; see Details), and a contrast of the 18 effect sizes with weights $1/(\text{weeks} + 2)$, centered to sum to zero, gives $z = 2.76$, one-tailed $p = .003$; the teacher expectancy vignette shows the calculation, and the examples plot the effect sizes against the weeks of prior contact.

Usage

```
teacher_expectancy
```

Format

A data frame with 19 rows (18 experiments; Pellegrini and Hicks, 1972, contributes a tester-aware and a tester-blind condition) and 10 variables.

`study` Integer identifier, in the order of the paper's Table 1.

`author` Study authors (with the Pellegrini and Hicks condition noted).

`year` Year of publication.

`weeks` Estimated weeks of teacher-student contact prior to the expectancy induction, 0 to 24. The moderator at the heart of the paper.

`testing` Factor: `group` or `individual` IQ testing.

`tester` Factor: test administrator `aware` of or `blind` to the expectancy designations.

`n_experimental`, `n_control` Per-condition sample sizes (from the studies as tabulated in Raudenbush & Bryk, 1985; the 1984 table does not print them).

`d` Standardized mean difference: the treatment effect in IQ points divided by the control group's posttest standard deviation (positive when the expectancy children gained more). The 1984 paper's Table 1 values.

`p_one_tailed` One-tailed *p*-value reported for the study's expectancy effect.

Details

The study-level Pellegrini and Hicks values. For analyses with the 18 *studies* as units (the combined significance tests, the contrast on weeks of prior contact, and the heterogeneity statistic), Raudenbush merged the two Pellegrini and Hicks conditions into a single study-level entry with $d = 0.52$ and one-tailed $p = .010$ (Table 1 prints these on the study's header row above the two condition rows). Replace rows 4 and 5 with that pair to reconstruct his 18-study analyses, as the teacher expectancy vignette does. For the tester aware-versus-blind comparisons the two conditions enter separately, which is why the data ship at the condition level.

Relation to the 1985 version. Raudenbush and Bryk (1985) re-standardized the same literature for their empirical Bayes analysis (that version circulates as `dat.raudenbush1985` in **metafor**), so its effect sizes differ from the `d` column here, which preserves the 1984 paper's metric. The sample sizes are common to both.

The teacher expectancy vignette (`vignette("teacher_expectancy", package = "DMAR")`) reproduces the paper's analyses with `combine_p`, `meta_contrast`, and `meta_smd`, and then reanalyzes the data with a random effects meta-analysis.

Author(s)

Ken Kelley <kkelley@nd.edu>

Source

Raudenbush, S. W. (1984). Magnitude of teacher expectancy effects on pupil IQ as a function of the credibility of expectancy induction: A synthesis of findings from 18 experiments. *Journal of Educational Psychology, 76*(1), 85–97.

References

Pellegrini, R. J., & Hicks, R. A. (1972). Prophecy effects and tutorial instruction for the disadvantaged child. *American Educational Research Journal, 9*(3), 413–419. doi:10.3102/00028312009003413

Raudenbush, S. W. (1984). Magnitude of teacher expectancy effects on pupil IQ as a function of the credibility of expectancy induction: A synthesis of findings from 18 experiments. *Journal of Educational Psychology, 76*(1), 85–97. doi:10.1037/00220663.76.1.85

Raudenbush, S. W., & Bryk, A. S. (1985). Empirical Bayes meta-analysis. *Journal of Educational Statistics, 10*(2), 75–98.

Rosenthal, R., & Jacobson, L. (1968). *Pygmalion in the classroom: Teacher expectation and pupils' intellectual development*. Holt, Rinehart and Winston.

See Also

`pygmalion` for the single Rosenthal and Jacobson study this literature grew from; `meta_smd`, `meta_contrast`, and `combine_p` for the analyses the vignette reproduces.

Examples

```
data(teacher_expectancy)
head(teacher_expectancy)

# The paper's central picture: effect size against weeks of prior contact.
plot(d ~ weeks, data = teacher_expectancy,
     xlab = "Weeks of teacher-student contact before induction",
     ylab = "Effect size d")

# The study-level (18-study) data Raudenbush used for the combined tests:
# merge the two Pellegrini & Hicks conditions into their study row.
study_level <- teacher_expectancy[-c(4, 5), ]
ph <- data.frame(study = 4, author = "Pellegrini & Hicks", year = 1972,
                 weeks = 0, testing = "group", tester = "aware",
                 n_experimental = 22, n_control = 22,
                 d = 0.52, p_one_tailed = .010)
study_level <- rbind(study_level[1:3, ], ph, study_level[4:17, ])
round(c(mean = mean(study_level$d), sd = sd(study_level$d)), 2)  # .11, .20
```

---

test_market *Controlled Test-Market Experiment (Bryant & Bruvold, 1980)*

Description

The controlled test-market experiment of Bryant and Bruvold (1980), used to illustrate multiple-comparison procedures in the analysis of covariance (ANCOVA) when the covariate is *random*. A company compared $k = 6$ marketing strategies ("panels") for a brand, randomly assigning them to retail outlets within $s = 4$ blocks of outlets that were homogeneous in size, locality, and ownership (a randomized complete block design, one outlet per panel-by-block cell). During the experiment a

concomitant variable, the remaining category movement in each outlet, becomes available; it cannot be controlled by the experimenter and is best modeled as a random covariate. Adjusting brand movement for this covariate sharply reduces unexplained error, permitting far finer comparison of the panels than the raw outcome allows.

Usage

```
test_market
```

Format

A data frame with 24 observations (6 panels $\times$ 4 blocks) on 4 variables.

`panel` Factor with levels 1–6: the marketing strategy (treatment) randomly assigned to the outlet. Different panels entail different methods of packaging, displaying, or pricing.

`block` Factor with levels 1–4: the block of retail outlets, grouped to be homogeneous in size, locality, ownership, and other considerations that influence brand movement.

`brand_movement` Test-brand movement during the test period, in hundreds of statistical cases. The dependent variable (y in the source).

`category_movement` Remaining category movement, the random concomitant variable (covariate; x in the source). It is not identically distributed across blocks, which is precisely the setting Bryant and Bruvold's grouped-covariate extension was designed for.

Details

Why it is a benchmark for ANCOVA multiple comparisons. The model fitted by Bryant and Bruvold (their Eq. 3.1) is a randomized-block ANCOVA,

$$y_{ij} = \theta_i + \beta_j + (x_{ij} - \delta_j)\, u + e_{ij},$$

with θ_i the ith panel (adjusted) mean, β_j the jth block effect, and u the within-cell covariate slope. The point of the example is that the studentized range of the adjusted panel means does *not* follow the ordinary Tukey distribution, because the covariate is random and its adjustment must be estimated; the correct reference distribution is the Bryant–Paulson generalized studentized range (`bryant_paulson`).

Reproducible quantities. Fitting `lm(brand_movement ~ panel + block + category_movement)` gives a covariate slope of 0.4079 and an error mean square of 0.01326 on $\nu = 14$ degrees of freedom, with adjusted panel means $3.595, 3.619, 4.102, 4.515, 4.618, 4.876$, exactly the values reported in the paper. With $q_{.05;\,1,6,14} = 4.83$ (`qbryant_paulson`), every pairwise simultaneous 95% interval is a difference of adjusted panel means plus or minus 0.278, so two panels differ at the simultaneous 95% level exactly when their adjusted means are more than 0.278 apart. Had the covariate not been measured, the error mean square would have been 0.2368, roughly eighteen times larger, and the intervals about four times wider. See `data-raw/test_market.R` for the construction script and its verification checks.

Author(s)

Ken Kelley

Source

Bryant, J. L., & Bruvold, N. T. (1980). Multiple comparison procedures in the analysis of covariance. *Journal of the American Statistical Association, 75*(372), 874–880 (Table 1). doi:10.2307/2287175

References

Bryant, J. L., & Paulson, A. S. (1976). An extension of Tukey's method of multiple comparisons to experimental designs with random concomitant variables. *Biometrika, 63*, 631–638.

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9.)

See Also

ci_c_ancova_bp for the simultaneous intervals this data set illustrates, bryant_paulson for the critical values, and ancova for an ANCOVA fit.

Examples

```
data(test_market)
str(test_market)

# Reproduce the published ANCOVA (slope 0.4079, error MS 0.01326, df 14).
fit <- lm(brand_movement ~ panel + block + category_movement,
          data = test_market)
coef(fit)["category_movement"]
sum(residuals(fit)^2) / fit$df.residual

# Adjusted panel means at the covariate grand mean.
xbar <- mean(test_market$category_movement)
adj <- vapply(levels(test_market$panel), function(p) {
  nd <- data.frame(panel = factor(p, levels = levels(test_market$panel)),
                   block = factor(1:4, levels = levels(test_market$block)),
                   category_movement = xbar)
  mean(predict(fit, nd))
}, numeric(1))
adj  # 3.595 3.619 4.102 4.515 4.618 4.876

# Bryant-Paulson simultaneous 95% intervals (s = 4 blocks => n = 4,
# df 14), every one of them the difference plus or minus 0.278.
ci_c_ancova_bp(adj_means = adj, s_ancova = sqrt(0.01326),
               n_per_group = 4, n_covariates = 1, df = 14)
```

---

tidy.dmar_ci_R2 *Tidy / Glance Methods for ci_R2 Output*

Description

Returns the standard broom-style one-row summary of the R^2 confidence interval: `term` (always `"R2"`), `estimate`, `ci_lower`, `ci_upper`, `conf_level`.

Usage

```
## S3 method for class 'dmar_ci_R2'
tidy(x, ...)

## S3 method for class 'dmar_ci_R2'
glance(x, ...)
```

Arguments

`x`	A `dmar_ci_R2` object returned by `ci_R2`.
`...`	Unused.

Value

A one-row `data.frame` in broom convention, with `alpha_lower` and `alpha_upper` after `conf_level` for an interval whose tail areas differ (see `dmar_tidiers`).

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
res <- ci_R2(R2 = 0.25, N = 100, p = 5)
generics::tidy(res)
generics::glance(res)
```

---

tidy.dmar_ci_smd *Tidy / Glance Methods for ci_smd Output*

Description

Returns the standard broom-style one-row summary of the SMD confidence interval: `term` (always `"smd"`), `estimate`, `ci_lower`, `ci_upper`, `conf_level`.

Usage

```
## S3 method for class 'dmar_ci_smd'
tidy(x, ...)

## S3 method for class 'dmar_ci_smd'
glance(x, ...)
```

Arguments

x	A dmar_ci_smd object returned by ci_smd.
...	Unused.

Value

A one-row data.frame with columns term (always "smd"), estimate (the point estimate of d), ci_lower (the lower confidence limit on δ), ci_upper (the upper confidence limit on δ), and conf_level (the confidence level used in construction, 1 - alpha_lower - alpha_upper when the tail areas were supplied instead), followed, for an interval whose tail areas differ, by alpha_lower and alpha_upper (see dmar_tidiers). Column names follow the broom convention so the result composes with the broom ecosystem.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
res <- ci_smd(smd = 0.5, n_1 = 50, n_2 = 50)
generics::tidy(res)
generics::glance(res)
```

---

tidy.dmar_cliff_delta *Tidy and Glance Methods for DMAR Result Tables*

Description

A DMAR function returns a tidy data.frame built to be read: one row per quantity, a numeric value column, and a display layer that rounds the printed values, not the stored ones (see dmar_tbl). The tidy verbs tidy and glance give the same numbers in the two shapes a programmer usually wants instead: one row per term with a typed column for each quantity, and a one-row summary of the result as a whole. This page states that contract once, for the confidence interval family, the post hoc family, the contrast tests, and the power-based sample size planners.

Usage

```
## S3 method for class 'dmar_cliff_delta'
tidy(x, ...)

## S3 method for class 'dmar_cliff_delta'
glance(x, ...)

## S3 method for class 'dmar_contrast_test'
tidy(x, ...)

## S3 method for class 'dmar_contrast_test'
```

```
glance(x, ...)

## S3 method for class 'dmar_df_barnard_rubin'
tidy(x, ...)

## S3 method for class 'dmar_df_barnard_rubin'
glance(x, ...)

## S3 method for class 'dmar_df_missing'
tidy(x, ...)

## S3 method for class 'dmar_df_missing'
glance(x, ...)

## S3 method for class 'dmar_ci_long'
tidy(x, ...)

## S3 method for class 'dmar_ci_long'
glance(x, ...)

## S3 method for class 'dmar_ci_anova'
tidy(x, ...)

## S3 method for class 'dmar_ci_anova'
glance(x, ...)

## S3 method for class 'dmar_anova_es'
tidy(x, ...)

## S3 method for class 'dmar_anova_es'
glance(x, ...)

## S3 method for class 'dmar_post_hoc_ci'
tidy(x, ...)

## S3 method for class 'dmar_post_hoc_ci'
glance(x, ...)

## S3 method for class 'dmar_ss_power'
tidy(x, ...)

## S3 method for class 'dmar_ss_power'
glance(x, ...)

## S3 method for class 'dmar_ss_aipe'
tidy(x, ...)

## S3 method for class 'dmar_ss_aipe'
```

```
glance(x, ...)

## S3 method for class 'dmar_tbl'
tidy(x, ...)

## S3 method for class 'dmar_tbl'
glance(x, ...)

## S3 method for class 'dmar_content_validity'
tidy(x, ...)

## S3 method for class 'dmar_content_validity'
glance(x, ...)

## S3 method for class 'dmar_dmacs'
tidy(x, ...)

## S3 method for class 'dmar_dmacs'
glance(x, ...)

## S3 method for class 'dmar_measurement_invariance'
tidy(x, ...)

## S3 method for class 'dmar_measurement_invariance'
glance(x, ...)

## S3 method for class 'dmar_measurement_alignment'
tidy(x, ...)

## S3 method for class 'dmar_measurement_alignment'
glance(x, ...)

## S3 method for class 'dmar_factor_congruence'
tidy(x, ...)

## S3 method for class 'dmar_factor_congruence'
glance(x, ...)

## S3 method for class 'dmar_ss_power_sensitivity'
tidy(x, ...)

## S3 method for class 'dmar_ss_power_sensitivity'
glance(x, ...)

## S3 method for class 'dmar_factor_analysis'
tidy(x, ...)

## S3 method for class 'dmar_factor_analysis'
```

```
glance(x, ...)

## S3 method for class 'dmar_factor_count'
tidy(x, ...)

## S3 method for class 'dmar_parallel_analysis'
glance(x, ...)

## S3 method for class 'dmar_probability_of_superiority_paired'
tidy(x, ...)

## S3 method for class 'dmar_probability_of_superiority_paired'
glance(x, ...)

## S3 method for class 'dmar_ci_ratio_of_means'
tidy(x, ...)

## S3 method for class 'dmar_ci_ratio_of_means'
glance(x, ...)

## S3 method for class 'dmar_vargha_delaney_A'
tidy(x, ...)

## S3 method for class 'dmar_vargha_delaney_A'
glance(x, ...)
```

Arguments

x	A DMAR result object carrying one of the classes listed above.
...	Unused, present for consistency with the generics.

Details

What the verbs return. `tidy(x)` returns a `data.frame` with one row per term, where a term is whatever the family produces one of: a parameter estimate, a contrast, a planned design. Its columns follow the naming convention the broom ecosystem uses, which separates words with dots rather than the underscores DMAR uses everywhere else: `term`, `estimate`, `se`, `statistic`, `p_value`, `ci_lower`, `ci_upper`, and `conf_level`. A method reports the subset of those columns its family can fill, plus any column the family genuinely adds, such as `p_adjusted` for a multiplicity-adjusted set of comparisons or `power` for a sample size planner.

`glance(x)` returns a one-row `data.frame` summarizing the result as a whole, in the same dotted convention: how many comparisons were made, at what confidence level, with which planning inputs. When a result has a single estimand and nothing further to say at the model level, as for a lone effect size and its confidence interval, `glance()` coincides with `tidy()`. That is expected rather than a defect, since there is no model-level quantity that the single row does not already carry.

Neither verb rounds. The `dmar_tbl` layer formats what is printed, while `tidy()` and `glance()` return full precision, which is what makes them the right input to a downstream calculation or plot.

Why broom is not a dependency. The `tidy()` and `glance()` generics live in **generics**, a small package that holds the generics and little else. **broom** imports them from there, and so does DMAR, which registers its methods against `generics::tidy` and `generics::glance` rather than against **broom** itself. A user with **broom** or the tidymodels stack loaded gets DMAR methods on the generic they already call; a user with neither installed can still call `generics::tidy()` directly. DMAR never loads **broom**, and does not need it installed.

The families and the classes they carry. Each family tags its return with a leading S3 class, ahead of `dmar_tbl` and `data.frame`, so the verbs dispatch while printing and data-frame behavior are untouched.

S3 class	Family	One `tidy()` row is
`dmar_ci_long`	confidence intervals, long form	an estimate and its limits
`dmar_ci_anova`	ANOVA effect size intervals	an effect and estimand, its estimate and limits
`dmar_anova_es`	ANOVA effect size estimates	an effect and estimand, its estimate
`dmar_post_hoc_ci`	simultaneous intervals	one pairwise or one contrast comparison
`dmar_contrast_test`	contrast tests	one contrast, with its test and its interval
`dmar_ss_power`	sample size planners	a planned size and the power it buys
`dmar_ss_power_sensitivity`	planner sensitivity studies	a planned size and two powers

The confidence interval family. Two classes cover the two output shapes.

`dmar_ci_long` Long-format interval tables, with rows for `ci_lower` and `ci_upper` and, when the function reports one, an estimate row whose `term` is the name of the parameter. Carried by `ci_r`, `ci_smd_c`, `ci_pvaf`, `ci_reg_coef`, `ci_ratio_of_means`, and `ci_ratio_of_means_paired`.

`dmar_ci_anova` Wide-format ANOVA effect size interval tables, with one row per effect and columns for the effect name, the point estimate of each estimand with its limits (`eta_squared_partial`, `eta_squared_partial_ci_lower`, `eta_squared_partial_ci_upper`, and the same for the total), and the design metadata. Carried by `ci_eta_squared` and `ci_omega_squared`, and by `ci_eta_squared_generalized`, whose one estimate sits beside `ci_lower` and `ci_upper`.

Both produce a `data.frame` with `term`, `estimate`, `ci_lower`, `ci_upper`, and `conf_level` (`NA` when the object records no level). For `dmar_ci_long` it is one row, and `glance()` calls `tidy()`, since the row is already the whole result. For `dmar_ci_anova` it is one row per effect and estimand, the rows of one estimand together: `term` names the estimand (`"eta_squared_total"`, `"eta_squared_partial"`), an `effect` column names the effect, a `stratum` column follows it for a fit with an `Error()` term, and the limits of the estimand are `ci_lower` and `ci_upper` (`NA` where no interval is formed, as for the total of a factorial effect). Its `glance()` coincides with `tidy()` when that is one row, and otherwise is one row giving the estimands (`term`, separated by commas when there are two), the number of effects (`n_effects`), `N`, and `conf_level`. The point estimates of `eta_squared` and `omega_squared` carry `dmar_anova_es`, whose `tidy()` has the same rows with `term`, `effect`, `stratum` (for an `aovlist` fit), and `estimate`, and whose `glance()` gives `term`, `n_effects`, and `N` (`NA` for `eta_squared()`, which reports none). The two ratio of means functions carry `dmar_ci_ratio_of_means` ahead of `dmar_ci_long`, because a Fieller confidence set need not be an interval: their one row adds, after the limits, `excluded_lower` and `excluded_upper` (the ends of the gap of a set that is the complement of an interval, `NA` otherwise) and `confidence_set`, the form of the set, so that a complement is not read from its limits of `-Inf` and `Inf` as the whole line (`NA` when a bootstrap interval was withheld and the limits are `NA`). `vargha_delaney_A` carries `dmar_vargha_delaney_A`, whose one `tidy()` row, named

"A", adds the test of stochastic equality (se, statistic, p_value) and the two sample sizes; its glance() coincides with it. Its verbs have df and test after p_value and a logical column interval_degenerate. cliff_delta and probability_of_superiority_paired carry dmar_cliff_delta and dmar_probability_of_superiority_paired, whose verbs are the default ones with the same logical column, TRUE when the estimated variance is 0 and the limits are not a confidence interval.

A table with no method of its own falls to the default methods for dmar_tbl. Its tidy() keeps the confidence level the table records in a conf_level column: beside the interval of a wide table (NA when no interval was formed), and, for a long table of term and value rows with ci_lower and ci_upper rows, on the rows of the interval, its estimate and its two limits, with NA on every other row (an interval held in rows named for its estimate, as sem_fit_measures holds the RMSEA's in rmsea_ci_lower and rmsea_ci_upper, is marked the same way, at its own level, except the interval for the standardized mean difference that cles, nnt_from_smd, and proportion_of_superiority echo in smd_ci_lower and smd_ci_upper, which their own intervals are computed from and whose rows keep NA). The rows carry the level and say which estimate the limits belong to, so that results_sentence on a tidy() result, or on rows of several tidy() results stacked with rbind(), states the level and the estimate of its own interval, as it does on the table itself. In a wide table that carries an interval, the estimate is the column the interval belongs to (the table's own estimate column, or its first numeric column that is not a limit, standard error, test statistic, degrees of freedom, *p*-value, confidence level, or count), and the other numeric columns follow under their own names. When that column has a name of its own (kappa of fleiss_kappa, D2 of ci_mahalanobis, eta_squared_partial of simple_effects_AB), an estimand column after estimate keeps the name, since term names the row (from the table's label columns, or the row number when it has none) and not the quantity; results_sentence labels the sentence of the row with it, as it labels the sentence of the table's own row. The tidy() of content_validity_index, whose estimate is the item's i_cvi, carries the same column, and so does that of fraction_missing_information, whose estimate is the parameter's FMI ("fmi") while term names the parameter. The tidy() of factor_count, whose RMSEA intervals sit in its rmsea_ci_lower and rmsea_ci_upper columns beside the other fit measures of each count, keeps the count as estimate and adds a conf_level column holding the level of those intervals, 0.90 unless the call set another. The default glance() of a long table is one row with a column for each term; for a table with an interval it adds an estimand column, the name of the column that holds the estimate the interval belongs to (the corrected correlation of correction_for_attenuation, listed after the observed one, and NA for a table that reports its limits alone, as ci_rc and ci_snr do), and a conf_level column, and names the column of the *p*-value the table's sentence reports, so that results_sentence on the row gives the table's own sentence.

Unequal tail areas. An interval formed from alpha_lower and alpha_upper that differ records its level as the coverage $1 - \alpha_{lower} - \alpha_{upper}$ and the two tail areas beside it. Every verb above that returns a conf_level for such an interval then adds alpha_lower and alpha_upper columns after it, holding the two areas on the rows of the interval (NA on the other rows of a long tidy()), so that results_sentence on the rows states the split, or the one-sided bound, as it does on the table. An equal-tailed interval, the usual one, adds neither column, so its columns are those it has without the record of the split; a stack of an equal-tailed and an unequal-tailed result therefore has different columns, which rbind() refuses and dplyr::bind_rows() fills with NA, read as equal tails.

The post hoc family. ci_tukey_kramer, ci_games_howell, ci_dunnett_t3, ci_scheffe, and ci_dunnett all carry dmar_post_hoc_ci. Their source table is wide, with one row per comparison: a contrast label, a point estimate (mean_difference for the pairwise and many-to-one procedures, estimate for Scheffe), a standard error, a test statistic, the ci_lower and ci_upper

of the simultaneous interval, and the multiplicity-adjusted `p_adjusted`. `tidy()` maps that to `term`, `estimate`, `ci_lower`, `ci_upper`, `p_adjusted`, and `conf_level`, one row per comparison. `glance()` describes the family of comparisons as a whole: how many there were, and the simultaneous confidence level they hold jointly.

The contrast tests. `contrast_test` carries `dmar_contrast_test`. Its source table is wide, with one row per contrast: a `contrast` label, the estimate $\hat{\psi} = \sum_i c_i \bar{Y}_i$, its standard error, the t-statistic and the degrees of freedom it is referred to, the unadjusted p-value, the multiplicity-adjusted `p_adjusted`, and the `ci_lower` and `ci_upper` limits. `tidy()` renames those to `term`, `estimate`, `ci_lower`, `ci_upper`, `statistic`, `df`, `p_value`, `p_adjusted`, and `conf_level`, one row per contrast. Both p-values are kept, because the pair is what a contrast table is read for: what the contrast would show on its own, and what it shows once the family it belongs to is accounted for.

Where a post hoc procedure fixes its adjustment as part of the method, a contrast test chooses one, and the same weights tested under `adjust = "none"` and under `adjust = "tukey"` are two different inferences. `glance()` therefore records the choice alongside the family-level numbers: `n_contrasts`, `adjust`, `var_equal`, the smallest adjusted p-value `p_adjusted_min`, and `conf_level`. `adjust` and `var_equal` name a procedure rather than measure a quantity, so this one-row summary, unlike a DMAR result table, is not numeric throughout. The post hoc and contrast methods, and wide result tables that carry adjustment metadata, retain `ci_adjustment` and `p_adjustment` in both `tidy()` and `glance()`. These are distinct because an interval and a test can use different multiplicity procedures. `results_sentence()` reports an adjusted p-value beside a simultaneous interval only when the recorded procedures match.

The power-based sample size planners. A planner in the `ss_power_*` family returns a long table with a row for the recommended sample size, a row for the realized power, and rows echoing the planning inputs. A planner that reports one size and one power for one design tags its return `dmar_ss_power`. This covers the closed-form effect size planners (`ss_power_R2`, `ss_power_r`, `ss_power_reg_coef`, `ss_power_smd`, `ss_power_sem`), the contrast and ANCOVA planners (`ss_power_c`, `ss_power_c_ancova`, `ss_power_sc`, `ss_power_contrast`, `ss_power_equivalence_c`), the ANOVA and cluster designs (`ss_power_one_way_anova`, `ss_power_factorial_anova`, `ss_power_factorial_ancova`, `ss_power_split_plot_anova`, `ss_power_rm_anova`, `ss_power_mixed_effects`), and the mediation planner `ss_power_indirect_effect`, whose reported power is the joint power to detect the indirect effect (its row `power_ab`) and whose component path powers `glance()` carries as extra columns.

The size `tidy()` reports is the design's planning unit: per group, per cell, per subject, or per cluster. The one-way ANOVA planner, whose natural unit is the total, is summarized by its total N. A design that reports two group sizes reports one of them beside the realized power, falling through to the total N when the per-group sizes are unequal, and `glance()` keeps every group size as a column so none is lost. A planner whose result spans several effects, with no single size-and-power summary to give, returns a plain `dmar_tbl` and does not gain these verbs at all.

The Monte Carlo sensitivity siblings `ss_power_R2_sensitivity` and `ss_power_reg_coef_sensitivity` report two powers at one planned sample size, the empirical (simulated) power and the analytic power, and comparing the two is the object of the study. They carry `dmar_ss_power_sensitivity` instead: `tidy()` places both powers beside the planned `sample_size`, and `glance()` adds the simulated distribution of the estimator.

Adding a planner to the family. A planner opts in by setting `dmar_ss_power` as a leading class before routing its return through `.as_dmar_tbl()`. The rows the verbs read are named in the internal vectors `.SS_POWER_SIZE_TERMS` and `.SS_POWER_POWER_TERMS`. A planner whose size or power row is not named there reports `NA` rather than failing, so a new row name has to be added to

those vectors when a planner introduces one. `glance()` leaves out only the two rows `tidy()` reads, the size and the power, and keeps every other row as a column, other powers included: the marginal powers of the composite planners (`power_a`, `power_b`, and `power_ab` for parameters labeled `a`, `b`, and `ab` in `ss_power_composite_sem`) and the path powers of `ss_power_indirect_effect`.

Value

`tidy()` returns a `data.frame` with one row per term and broom-convention column names. `glance()` returns a one-row `data.frame` summarizing the result as a whole. Both return values at full precision.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

`dmar_tbl` for the printing layer these tables share, and the "Reading DMAR result tables" vignette for the wider output convention.

Examples

```
# A single interval: tidy() and glance() coincide, because there is
# nothing at the model level the one row does not already carry.
res <- ci_r(r = 0.5, N = 50)
generics::tidy(res)
generics::glance(res)

# A family of simultaneous intervals: one tidy() row per comparison,
# one glance() row describing the family.
set.seed(113)
y <- c(rnorm(10, 0), rnorm(10, 1), rnorm(10, 2))
g <- factor(rep(c("a", "b", "c"), each = 10))
gh <- ci_games_howell(y, group = g)
generics::tidy(gh)
generics::glance(gh)

# A set of contrasts: tidy() keeps both the unadjusted and the
# adjusted p-value, and glance() names the adjustment that produced
# the second of them.
fit <- aov(bdi_post ~ condition, data = depression_bdi)
ct <- contrast_test(fit, contrasts = "pairwise", adjust = "tukey")
generics::tidy(ct)
generics::glance(ct)

# A sample size planner: tidy() gives the size and the power it buys,
# glance() adds the planning inputs that produced them.
plan <- ss_power_smd(true_smd = 0.5, desired_power = 0.80)
generics::tidy(plan)
generics::glance(plan)
```

`tidy.dmar_fa_confirmatory` *Tidy a Multiple-Factor CFA Fit*

Description

Returns the parameter rows of a `fa_confirmatory` table (loadings, error variances, intercepts, latent correlations, and the defined measurement properties) in the column convention used by the **broom** ecosystem. Fit-information rows belong in `glance.dmar_fa_confirmatory`.

Usage

```
## S3 method for class 'dmar_fa_confirmatory'
tidy(x, ...)
```

Arguments

`x`	A `dmar_fa_confirmatory` object returned by `fa_confirmatory`.
`...`	Unused.

Value

A `data.frame` with columns `term`, `estimate`, `se`, `statistic`, `p_value`, `ci_lower`, `ci_upper`, `conf_level`, the level of the parameter intervals, and `wald_df`, the degrees of freedom of the *t* reference of the row's interval and *p*-value (`wald_df` in `fa_confirmatory`), `Inf` for the normal reference. Under `wald_df = "missing_information"` only the loadings and the omega rows take a *t* reference, so the column tells them from the other rows, as it tells the rows of one fit from another's after tables are stacked.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
data(holzinger_swineford)
res <- fa_confirmatory(holzinger_swineford,
             list(verbal = c("t6_paragraph_comprehension",
                             "t7_sentence", "t9_word_meaning"),
                  deduction = c("t20_deduction",
                                "t22_problem_reasoning",
                                "t23_series_completion")))
generics::tidy(res)
generics::glance(res)
```

tidy.dmar_fmi *Tidy and Glance Methods for the Fraction of Missing Information*

Description

`tidy()` returns one row per parameter of a `fraction_missing_information` table, with the FMI as the estimate, its percentile limits, and the effective sample size; `glance()` returns a one-row summary of the table.

Usage

```
## S3 method for class 'dmar_fmi'
tidy(x, ...)

## S3 method for class 'dmar_fmi'
glance(x, ...)
```

Arguments

`x`	A `dmar_fmi` object returned by `fraction_missing_information`.
`...`	Unused.

Value

`tidy()` returns a `data.frame` with columns `term` (the parameter), `group`, `estimate` (its FMI), `estimand` (`"fmi"`, which names the estimate, so that `results_sentence` on a row reports `"fmi = 0.24, 90% CI [0.18, 0.32]"` and not the FMI as the parameter's value), `ci_lower`, `ci_upper`, `conf_level` (all three `NA` without the percentile interval), and `N_effective`. `glance()` returns a one-row `data.frame` with `n_parameters`, `fmi_max`, `fmi_median`, `N` (the cases the fit used, summed over groups), and `information`.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
fit <- fa_confirmatory(data = holzinger_swineford,
             factors = list(spatial = c("t1_visual_perception",
               "t2_cubes", "t3_paper_form_board", "t4_lozenges",
               "t25_paper_form_board_r", "t26_flags")),
             missing = "fiml", output = "fit")
fmi <- fraction_missing_information(fit)
generics::tidy(fmi)
generics::glance(fmi)
```

tidy.dmar_mediation_mbco

Tidy an MBCO Mediation Table

Description

Returns the effect rows of a `mediation_mbco` table in the column convention used by the **broom** ecosystem. The `statistic` column is the MBCO likelihood ratio statistic.

Usage

```
## S3 method for class 'dmar_mediation_mbco'
tidy(x, ...)
```

Arguments

`x`	A `dmar_mediation_mbco` object returned by `mediation_mbco`.
`...`	Unused.

Value

A `data.frame` with columns `term`, `estimate`, `se`, `statistic`, `p_value`, `ci_lower`, `ci_upper`, and `conf_level`, the level of the intervals.

Author(s)

Ken Kelley <kkelley@nd.edu>

tidy.dmar_R2_mixed_effects

Tidy / Glance Methods for R2_mixed_effects Output

Description

Returns the broom-style summary of the marginal and conditional R^2: `term` (`"R2_marginal"` or `"R2_conditional"`), `estimate`, and, when a bootstrap interval was requested, `ci_lower`, `ci_upper`, and `conf_level`. `glance` returns the same information in one row per quantity.

Usage

```
## S3 method for class 'dmar_R2_mixed_effects'
tidy(x, ...)

## S3 method for class 'dmar_R2_mixed_effects'
glance(x, ...)
```

Arguments

x	A dmar_R2_mixed_effects object returned by R2_mixed_effects.
...	Unused.

Value

A data.frame in broom convention.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
fit <- lme4::lmer(Reaction ~ Days + (Days | Subject),
                  data = lme4::sleepstudy)
res <- R2_mixed_effects(fit)
generics::tidy(res)
generics::glance(res)
```

---

tidy.dmar_reliability *A Reliability Coefficient Estimate*

Description

Returns a one-row data.frame in the column convention used by the **broom** ecosystem (term, estimate, se, ci_lower, ci_upper, conf_level, wald_df). The term is the coefficient name ("alpha", "omega", etc.), conf_level is the level of the interval, so that results_sentence on the row states it, and wald_df the degrees of freedom of the *t* reference of a Wald interval (the wald_df argument of reliability_alpha, reliability_omega, and reliability), Inf for the normal reference or an interval that is not a Wald interval, so that rows on the two references can be told apart after they are stacked.

Usage

```
## S3 method for class 'dmar_reliability'
tidy(x, ...)
```

Arguments

x	A dmar_reliability object returned by any of reliability_alpha, reliability_kr20, reliability_omega, reliability_omega_categorical, or reliability.
...	Unused.

Value

A one-row data.frame.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
# Coefficient alpha for the three verbal tests of the Holzinger and
# Swineford battery, from their covariance matrix.
S <- cov(holzinger_swineford[, c("t6_paragraph_comprehension",
                                 "t7_sentence", "t9_word_meaning")])
res <- reliability_alpha(S = S, N = 301, ci_method = "feldt")
generics::tidy(res)
generics::glance(res)
```

`tidy.mlmr` *An Mlmr Fit*

Description

Returns a one-row-per-coefficient `data.frame` in the column convention used by the **broom** ecosystem (`term`, `estimate`, `se`, `statistic`, `p_value`, `wald_df`, and optionally `ci_lower`, `ci_upper`). `wald_df` is the degrees of freedom of the *t* reference of the row's statistic and *p*-value (the `wald_df` argument of `mlmr`), `Inf` for the normal reference, so that rows on the two references can be told apart after tables are stacked. Use `as.data.frame()` on the fit for the DMAR-style table (snake_case columns) stored at `fit$coef_table`.

Usage

```
## S3 method for class 'mlmr'
tidy(x, conf.int = FALSE, conf_level = NULL, standardized = FALSE, ...)
```

Arguments

`x`	An object of class `"mlmr"`.
`conf.int`	Logical; if `TRUE`, append `ci_lower` and `ci_upper` columns using the confidence intervals already computed at fit time, and a `conf_level` column holding the level they were computed at. Defaults to `FALSE`.
`conf_level`	Ignored; the confidence interval comes from the fit object at `x$conf_level`. Present for compatibility with the broom generic.
`standardized`	Logical; if `TRUE` and effect sizes were computed at fit time, append a `std_estimate` column. Defaults to `FALSE`.
`...`	Unused.

Value

A `data.frame`.

Author(s)

Ken Kelley <kkelley@nd.edu>

Examples

```
fit <- mlmr(t6_paragraph_comprehension ~ t5_general_information +
              t9_word_meaning,
            data = holzinger_swineford, ci_method = "wald")
generics::tidy(fit)
generics::tidy(fit, conf.int = TRUE)
generics::tidy(fit, conf.int = TRUE, standardized = TRUE)
```

`tidy.mlmr_mv` *A Multivariate FIML Regression Fit*

Description

Broom-style `tidy()` and `glance()` for `mlmr_mv` fits. `tidy()` returns one row per coefficient per outcome with the broom dotted columns plus a leading `response` column identifying the outcome; `glance()` returns a one-row model-level summary with the per-outcome R^2 averaged and the number of responses reported.

Usage

```
## S3 method for class 'mlmr_mv'
tidy(x, conf.int = FALSE, conf_level = NULL, standardized = FALSE, ...)

## S3 method for class 'mlmr_mv'
glance(x, ...)
```

Arguments

`x`	An `mlmr_mv` fit.
`conf.int`	Logical: include `ci_lower` / `ci_upper`, with a `conf_level` column holding the level of the intervals?
`conf_level`	Ignored (the interval level is fixed at fit time and stored on the object); present for broom signature compatibility.
`standardized`	Logical: include `std_estimate`?
`...`	Unused.

Value

For `tidy.mlmr_mv`, a `data.frame` with columns `response`, `term`, `estimate`, `se`, `statistic`, `p_value`, `wald_df` (the degrees of freedom of the t reference of the row's statistic and p-value, `Inf` for the normal reference), and optionally `ci_lower`, `ci_upper`, `std_estimate`. For `glance.mlmr_mv`, a one-row `data.frame` with `R2` (mean across outcomes), `df`, `logLik`, `AIC`, `BIC`, `deviance`, `nobs`, and `n.responses`.

Author(s)

Ken Kelley <kkelley@nd.edu>

See Also

mlmr_mv; mlmr for the univariate methods these mirror.

---

unbiased_R2 *Unbiased and Adjusted Estimators of the Population Squared Multiple Correlation*

Description

Estimates the population squared multiple correlation coefficient ρ^2 from an observed sample R^2, correcting the well-known positive (upward) bias of R^2. Two estimators are available: the (essentially) unbiased Olkin and Pratt (1958) estimator (the default), and the classic Ezekiel (1930) adjusted-R^2 shrinkage formula reported by summary.lm as adj.r.squared. This is the inverse direction of expected_R2, which gives the forward expectation $E[R^2 \mid \rho^2]$.

Usage

```
unbiased_R2(R2, N, p, method = c("olkin_pratt", "ezekiel"))
```

Arguments

R2	Observed sample squared multiple correlation coefficient, in $[0, 1]$.
N	Sample size.
p	Number of predictor variables.
method	Which estimator to compute: "olkin_pratt" (default) for the (essentially) unbiased Olkin-Pratt (1958) estimator, or "ezekiel" for the Ezekiel (1930) adjusted R^2 (the value summary.lm reports as adj.r.squared).

Details

The sample R^2 overestimates ρ^2; the bias is larger for smaller samples and for more predictors. Two corrections are offered.

The Ezekiel (1930) adjusted estimator is

$$\hat{\rho}^2_{\text{Ezekiel}} = 1 - \frac{N-1}{N-p-1}\,(1-R^2).$$

It reduces the bias but is not unbiased; it is exactly the quantity summary(lm(...))$adj.r.squared reports.

The Olkin and Pratt (1958) estimator is (essentially) unbiased:

$$\hat{\rho}^2_{\text{OP}} = 1 - \frac{N-3}{N-p-1}\,(1-R^2)\;{}_2F_1\left(1, 1; \frac{N-p+1}{2}; 1-R^2\right),$$

where ${}_2F_1$ is the Gaussian hypergeometric function (the same function `expected_R2` uses for the forward direction; see Stuart, Ord, & Arnold, 1999, section 28). Both estimators can fall below 0 for very small R^2; that is expected behavior for a bias-corrected estimator and is not truncated here (matching `adj.r.squared`, which is also allowed to be negative).

Value

A 1-row `data.frame` (class `dmar_tbl`) with columns `term` and `value`. The `term` is `"unbiased_population_R2"` when `method = "olkin_pratt"` and `"adjusted_population_R2"` when `method = "ezekiel"`; `value` is the corresponding estimate of ρ^2.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Ezekiel, M. (1930). *Methods of correlation analysis*. Wiley.

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*(4), 524–555. doi:10.1080/00273170802490632

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Olkin, I., & Pratt, J. W. (1958). Unbiased estimation of certain correlation coefficients. *The Annals of Mathematical Statistics, 29*(1), 201–211.

Stuart, A., Ord, J. K., & Arnold, S. (1999). *Kendall's advanced theory of statistics, volume 2A: Classical inference and the linear model* (6th ed.). Arnold.

See Also

`expected_R2` for the forward expectation $E[R^2 \mid \rho^2]$, and `ci_R2`, `var_R2`, `ss_aipe_R2` for interval, variance, and planning tools on the same effect size.

Examples

```
# An observed R^2 = .50 with N = 50 and p = 5 predictors overstates rho^2.
# The Olkin-Pratt (essentially unbiased) estimate is the default:
unbiased_R2(R2 = .50, N = 50, p = 5)

# The Ezekiel adjusted R^2 (what summary(lm) reports) over-shrinks slightly,
# so it typically sits a little below the Olkin-Pratt value:
unbiased_R2(R2 = .50, N = 50, p = 5, method = "ezekiel")

# The Ezekiel option reproduces summary(lm)$adj.r.squared exactly.
set.seed(113)
d   <- as.data.frame(matrix(rnorm(50 * 6), 50, 6))
fit <- lm(V1 ~ ., data = d)
s   <- summary(fit)
```

```
unbiased_R2(R2 = s$r.squared, N = 50, p = 5, method = "ezekiel")$value
s$adj.r.squared

# The bias (and so the correction) shrinks as N grows for fixed R^2 and p.
unbiased_R2(.50, 50, 5)
unbiased_R2(.50, 500, 5)
```

---

vargha_delaney_A *Vargha and Delaney's* A *(Stochastic-Superiority Effect Size)*

Description

Takes two independent samples of scores that can be ordered and returns Vargha and Delaney's (2000) A, the probability that a randomly drawn observation from the first population exceeds a randomly drawn observation from the second (a tie counting one half), with its standard error, a confidence interval, and a test of $A = 1/2$. The statistic uses only the order of the observations, so it needs no assumption of normality or of equal variances and suits ordinal and skewed outcomes.

Usage

```
vargha_delaney_A(
  x,
  y = NULL,
  data = NULL,
  conf_level = 0.95,
  ci_method = c("logit", "wald"),
  test = c("t", "permutation", "z"),
  exact = NULL,
  n_resamples = 10000L,
  seed = NULL
)
```

Arguments

x	Either a numeric or logical vector of observations from group 1, an ordered factor, or a two-sided formula of the form `outcome ~ group` (in which case `data` must be supplied and the grouping variable must have exactly two observed levels; its first level is group 1). Two ordered factors must have the same levels in the same order. An unordered factor and a character vector are refused; see *What is refused, and why*.
y	Observations from group 2, of the same kind as `x`. Ignored when `x` is a formula.
data	Optional data frame containing the variables named in the formula `x`.
conf_level	Confidence coverage for a symmetric interval (default `0.95`).
ci_method	Either `"logit"` (default; Wald interval on the logit of A with back-transformation, which keeps the limits inside (0, 1)) or `"wald"` (untransformed Wald on the original scale, clipped to [0, 1]).

`test` Reference distribution of the test of $A = 1/2$. `"t"` (default) refers $(\hat{A} - 1/2)/\mathrm{SE}$ to Student's *t* with Satterthwaite degrees of freedom, the test of Brunner and Munzel (2000). `"permutation"` refers the same statistic to its distribution over reassignments of the scores to the two groups (a studentized permutation test; Neubert and Brunner, 2007). `"z"` refers it to the standard normal distribution. See *The test*.

`exact` Logical. If `NULL` (default), the permutation test enumerates every reassignment when `choose(n_1 + n_2, n_1)` is at most 50,000 and samples reassignments otherwise. If `TRUE`, enumeration is forced (refused above 1,000,000 reassignments). If `FALSE`, random reassignments are forced. Used only when `test = "permutation"`.

`n_resamples` A single whole number of at least 1: the number of randomly drawn reassignments when enumeration is not used. Default `10000L`. Used only when `test = "permutation"`.

`seed` Optional integer seed for the randomly drawn reassignments. Default `NULL`, under which the reassignments are drawn from, and advance, the current random number stream; supply an integer for reproducibility. When a seed is supplied the RNG state in place before the call is restored on exit. Used only when `test = "permutation"`.

Details

Estimand and interval. For independent observations X from the first population and Y from the second,

$$A = \Pr(X > Y) + \tfrac{1}{2} \Pr(X = Y).$$

The sampling units are the $n_1 + n_2$ individuals, each observed once and each in one group. The interval is a large-sample (Wald) confidence interval for the population *A*, formed on the logit scale by default. A value of $A = 0.5$ is stochastic equality, and $A > 0.5$ says that the first group tends to score higher. Stochastic equality (Vargha & Delaney, 2000, p. 102) is weaker than equality of the two distributions: populations that differ in spread or shape can have $A = 0.5$. The test reported here is of $A = 0.5$. It is not a test that the distributions are identical, the null hypothesis of the Wilcoxon-Mann-Whitney test (pp. 108–109). Qualitative magnitude labels for *A* are not reported, in keeping with the DMAR convention of reporting effect sizes as numbers with confidence intervals.

Assumptions. The two samples are independent of each other and the observations within each are independent and identically distributed; for paired observations use `probability_of_superiority_paired`. The response is at least ordinal. The estimate needs nothing more: no distributional shape, no equality of variances, and ties are allowed. The standard error, the interval, and the *t* and *z* tests are large-sample results; *The test* and *When the estimated variance is zero* say how they behave in small samples.

Scale and missing values. *A* depends only on the order of the observations (Vargha & Delaney, 2000, pp. 101, 107), so any increasing rescoring of the response gives the same value, and an ordinal response may be supplied as an ordered factor. Missing values (`NA` and `NaN`) are removed group by group; the numbers removed are in the `n_missing` attribute. Infinite values are ordered as usual. Dates and date-times (classes `Date` and `POSIXct`) are ordered by their numeric values, as are two time differences (class `difftime`) in the same units. A one-column matrix and a one-dimensional array are read as the vectors they are. Each group needs at least two non-missing observations; with fewer the function stops, since no variance can be estimated.

What is refused, and why. The statistic uses the order of the observations and nothing else, so the function must be given objects whose order is the order meant. It refuses, with a condition of class `dmar_scores_not_ordered` (see `dmar_metadata`), an unordered factor (its codes follow the order of `levels()`, alphabetical unless set, so the levels low, mid, high are stored as 2, 3, 1, and *A* computed from the codes of the example in *Examples* is 0.406 where the ordered levels give 0.656); a character vector (strings sort alphabetically, `"10"` before `"9"`); an ordered factor beside anything that is not an ordered factor with the same levels in the same order (each factor numbers its own levels from 1); time differences in different units, or one beside plain numbers (each such object is stored in its own units); a `Date` beside a `POSIXct`; and a list, data frame, or matrix of several columns. Nothing is computed in any of these cases. The message says which case applies and what to supply: an ordered factor with its levels from lowest to highest, `factor(x, levels = c("low", "mid", "high"), ordered = TRUE)`, or numeric scores.

Sample estimate. With $\bar{R}_X$ the mean rank of the first sample in the pooled ranking, mid-ranks for ties,

$$\hat{A} = \frac{\bar{R}_X - (n_1 + 1)/2}{n_2}$$

(Vargha & Delaney, 2000, Equation 14, p. 107). Equivalently, $\hat{A} = U/(n_1 n_2)$, with U the Mann-Whitney statistic that counts the pairs with $X_i > Y_j$ and half of the tied pairs. The function computes the mean rank form, whose rounding can differ from the exact ratio $U/(n_1 n_2)$ in about the thirteenth decimal place.

Standard error. Define the placements

$$V_{10}(X_i) = \frac{1}{n_2} \sum_j \psi(X_i, Y_j), \qquad V_{01}(Y_j) = \frac{1}{n_1} \sum_i \psi(X_i, Y_j),$$

with $\psi(x, y) = 1, \frac{1}{2}, 0$ as $x > y, =, <$. The variance estimate is

$$\widehat{\mathrm{Var}}(\hat{A}) = \frac{S_{10}^2}{n_1} + \frac{S_{01}^2}{n_2},$$

where S_{10}^2 and S_{01}^2 are the sample variances (divisors $n_1 - 1$ and $n_2 - 1$) of the V_{10} and V_{01} placements. This is the placement (structural components) estimator of DeLong, DeLong, and Clarke-Pearson (1988, p. 840) for the area under a single receiver operating characteristic curve; Schüürhuis, Konietschke, and Brunner (2025) state that the variance estimator of Brunner and Munzel (2000) is identical to it. It is consistent for two independent samples from any pair of distributions, with no parametric or equal-variance assumption. It is not unbiased in general: its expectation exceeds the variance of $\hat{A}$ by $(\xi_{11} - \xi_{10} - \xi_{01})/(n_1 n_2) \geq 0$, where ξ_{11} is the variance of $\psi(X, Y)$ and ξ_{10} and ξ_{01} are the covariances of two values of ψ that share their X or their Y. The excess is zero when both variables take only two values. With three continuous observations per group from one distribution the variance is 7/108 and the estimator averages 8/108.

Confidence interval. `ci_method = "logit"` (the default) forms a Wald interval for $\mathrm{logit}(A) = \log\{A/(1 - A)\}$ with the delta method standard error $\widehat{\mathrm{SE}}(\hat{A})/\{\hat{A}(1 - \hat{A})\}$,

$$\mathrm{logit}(\hat{A}) \pm z_{1-\alpha/2} \frac{\widehat{\mathrm{SE}}(\hat{A})}{\hat{A}(1 - \hat{A})},$$

and transforms the two limits back with the inverse logit, so they stay inside (0, 1). Pohar Perme and Manevski (2019) conclude that the DeLong variance estimator is reliable and that the interval

should be formed on the logit scale. `ci_method = "wald"` returns $\hat{A} \pm z_{1-\alpha/2}\,\widehat{\mathrm{SE}}(\hat{A})$, clipped to $[0, 1]$. Both intervals use the normal quantile, whatever `test` is.

Relation to Cliff's delta. $A = (\delta + 1)/2$, and because $\operatorname{arctanh}(2A - 1) = \operatorname{logit}(A)/2$, the logit interval here and the interval of `cliff_delta` are the same construction on the two scales. The two functions estimate the variance differently, however (placement variances here; there the sample version of Feng and Cliff's, 2004, Equation 2, which has a third term), so their limits are not the same after the change of scale. For `x = 1:5` and `y = 3:7`, $\hat{A} = 0.18$ with logit limits 0.034 and 0.576, which are -0.931 and 0.151 on the scale of δ; `cliff_delta()` gives $\hat{\delta} = -0.64$ with limits -0.935 and 0.177.

The test. The statistic is

$$W = \frac{\hat{A} - 1/2}{\widehat{\mathrm{SE}}(\hat{A})},$$

reported in `test_statistic`. The three choices of `test` share this statistic and differ in what it is referred to. `"t"` gives $p = 2\,\Pr(T_f \geq |W|)$ for Student's *t* with

$$f = \frac{\left(S_{10}^2/n_1 + S_{01}^2/n_2\right)^2}{\dfrac{(S_{10}^2/n_1)^2}{n_1 - 1} + \dfrac{(S_{01}^2/n_2)^2}{n_2 - 1}}$$

degrees of freedom, reported in `df`. The statistic, the degrees of freedom, and the rejection rule are those of Schüürhuis, Konietschke, and Brunner (2025, Equations 3 and 7, the display that precedes Equation 8, and Equation 8), written here in placement variances; they present them as the test of Brunner and Munzel (2000), whose statistic is their Equation 4.8 (p. 21) and whose degrees of freedom are their Equation 5.9 (p. 22), an approximation they recommend for group sizes of at least 10. `"z"` gives $p = 2\,\Phi(-|W|)$. `"permutation"` recomputes W for reassignments of the $n_1 + n_2$ pooled scores to groups of sizes n_1 and n_2: every one of the `choose(n_1 + n_2, n_1)` reassignments when there are at most 50,000 (or when `exact = TRUE`), and `n_resamples` randomly drawn ones otherwise, by the rules of `randomization_test`. The *p*-value is the share of reassignments with $|W_b| \geq |W|$ under enumeration, and $(1+r)/(1+m)$ with r such reassignments among m random ones. A reassignment whose estimated variance is 0 has $W_b = 0$ when its $\hat{A}$ is 1/2 and W_b infinite, with the sign of $\hat{A} - 1/2$, otherwise. Neubert and Brunner (2007, p. 5195) define only the one-sided *p*-value of their studentized permutation test, the share of reassignments whose statistic is at least the observed one, with the variance recomputed for each. Schüürhuis et al. describe its two-sided *p*-value as twice the smaller tail share; the form used here, the share at least as extreme in absolute value, is the one `randomization_test()` uses, and the two can differ. The permutation test is exact when the two distributions are equal, since every reassignment is then equally likely, and approximate when they differ in spread or shape with $A = 1/2$.

The *z* reference rejects a true $A = 1/2$ too often in small samples, which is why it is not the default. Rejection rates at a nominal .05 with normal data and $A = 1/2$, from this function (10,000 samples per row for *z* and *t*, Monte Carlo standard error about .002; 2,000 samples for the permutation test, standard error about .005, with 399 random reassignments where the reassignments were not enumerated); the ratio is the standard deviation of the first group over that of the second:

n_1	n_2	SD ratio	*z*	*t*	permutation
5	5	1	.086	.071	.047
10	10	1	.073	.056	.047
8	20	1	.080	.062	.045

20	20	1	.060	.052	.046
5	5	3	.084	.068	.048
10	10	3	.079	.051	.052
8	20	3	.090	.062	.063
20	8	3	.065	.052	.034
20	20	3	.062	.050	.052

Over these and the other simulations run for this function, which include five-category ordinal and lognormal data and group sizes from 5 to 30, the rates were .056 to .099 for *z*, .049 to .071 for *t*, and .032 to .063 for the permutation test. The *t* reference removes most of the excess from 10 per group upward and part of it below that. These ranges describe the conditions that were run. With one group of 5 or fewer and the other five or more times larger, the *t* reference rejected a true $A = 1/2$ in .07 to .11 of samples when the two distributions were equal (.073 with 5 and 25, .085 with 4 and 20, .112 with 3 and 30; normal data, 20,000 samples each), where the permutation test is exact (the *z* reference rejected in .105, .124, and .172 of the same samples). When the smaller group has fewer than about 6 scores, `test = "permutation"` is therefore the test to use, unless the two groups also differ greatly in spread, the case described next. When the small group also had five or six times the standard deviation of the large one, so that the distributions were not equal, the permutation test rejected in .08 of samples with 5 and 25 and in .11 with 4 and 20 (4,000 samples each, 999 reassignments). When the groups are completely separated the estimated variance is 0 and only the permutation test gives a *p*-value: W is infinite, and the *p*-value is the share of reassignments that are also completely separated (2/252 for 5 and 5 untied scores).

The test and the default interval are separate procedures. The interval is formed on the logit scale so that it stays inside 0 and 1, and the test is on the scale of *A*. They can disagree in small samples: for `x = c(3.5, 5, 6, 7, 8)` and `y = c(1, 2, 3, 4, 4.2)`, $\hat{A} = .92$, the logit interval is [.486, .993], the *t* test gives *p* = .0033, and the exact permutation test gives *p* = .032. A test on the logit scale would agree with the interval but rejects far too rarely (.007 to .041 at a nominal .05 in simulation) and is not offered.

When the estimated variance is zero. Two kinds of data give a variance estimate of exactly 0, and they are the only two. *Complete separation*: every observation in one group exceeds every observation in the other, so the estimate is at the end of its range ($\hat{A}$ 0 or 1). *No variation*: every observation has the same value, so the estimate is 0.5. In both, the two limits are reported equal to the estimate under either `ci_method`, `se` is 0, `test_statistic` and `p_value` are `NA` (under `test = "permutation"` the test is reported, as described above), the table's `interval_degenerate` attribute names the case, and a warning of class `dmar_interval_degenerate` is given whose `reason` element is `"complete_separation"` or `"no_variation"`; the condition also carries `estimate`, `n` (the two group sizes), and `limit_feng_cliff`. The reported limits are not a confidence interval with the stated coverage. The zero describes the sample: the variance estimate measures how much the comparisons differ from one observation to the next, and without overlap they do not differ. Complete separation is common when the effect is large and the samples are small. With five observations per group from normal populations, it occurred in 19 percent of samples when the population value of *A* was 0.86 and in 59 percent when it was 0.96, and across all samples the reported 95 percent limits then covered the population value 79 and about 40 percent of the time, against 98 and 96 percent among the samples that were not separated (20,000 simulated samples each). Feng and Cliff (2004, p. 324, Equations 6 to 8) treat this case by assuming the largest variance $\hat{\delta}$ can have, which gives the limit $(n_b - z^2)/(n_b + z^2)$ when $\hat{\delta} = 1$ (and its negative when $\hat{\delta} = -1$), with n_b the smaller group size; they describe the result as conservative. On the scale of *A* that limit is $n_b/(n_b + z^2)$ when $\hat{A} = 1$ and $z^2/(n_b + z^2)$ when $\hat{A} = 0$: 0.566 and 0.434 at the

95 percent level with five in the smaller group. The warning states that limit. It is not the limit this function reports.

Particular cases. Infinite values are ordered like any others, so a group made up of `Inf` beside finite scores is completely separated, and two groups made up of the same infinite value have no variation. Two constant groups at different values are completely separated. Groups that only touch, the largest score of one equal to the smallest of the other, are not separated: the tied pair gives a positive variance and an ordinary interval, which is wide (`c(3, 4, 5)` against `c(1, 2, 3)` gives $\hat{A} = 0.944$ with limits 0.475 and 0.997). The mark travels with the result: `results_sentence` writes the estimate and says that there is no confidence interval, the printed table says so beneath itself, `tidy()` and `glance()` carry it in the column `interval_degenerate`, `randomization_test` passes on the warning of `cliff_delta`, and `ss_aipe_cliff_delta_sensitivity` counts a separated replication as one without an interval. The randomization test remains valid in both cases.

Relation to other measures. A is the area under the receiver operating characteristic curve (AUC) when the scores are used to tell the two groups apart, and $\hat{A} = U/(n_1 n_2)$ for the Mann-Whitney U. Cliff's δ is $2A - 1$. The common language effect size of McGraw and Wong (1992), as `cles` computes it, is $\Phi(\delta/\sqrt{2})$ with δ the standardized mean difference, which is $\Pr(X > Y)$ for normal distributions with a common variance; A estimates that probability (with ties split) without those assumptions.

Reading the output. In the second example below the first level of `treatment`, Control, is group 1, with 246 children against 64. $\hat{A} = 0.429$: a randomly drawn control child has the higher `iq_8` than a randomly drawn child from the bloomer group with estimated probability .43, ties counting one half. The 95 percent logit interval is [0.351, 0.511], and the *t* test of $A = 1/2$ has $W = -1.72$ on 92.5 degrees of freedom, $p = .089$.

Value

A one-row `data.frame` (classes `dmar_vargha_delaney_A` and `dmar_tbl`) with these columns, in this order:

`A` The estimate $\hat{A}$.

`se` Its standard error, the square root of the placement variance of DeLong, DeLong, and Clarke-Pearson (1988); 0 when the estimated variance is 0.

`ci_lower`, `ci_upper` The confidence limits at `conf_level`, formed as `ci_method` says. Both equal `A` when the estimated variance is 0, and are then not a confidence interval.

`test_statistic` The test statistic $W = (\hat{A} - 1/2)/\mathrm{SE}$, the same under each choice of `test`; the `test` column names the distribution it is referred to. The column has this one name under all three tests. `NA` when the estimated variance is 0, except under `test = "permutation"`, where it is `Inf` or `-Inf` for completely separated groups and 0 when every score is equal.

`p_value` The two-sided *p*-value of $A = 1/2$ (stochastic equality) from the reference distribution named in `test`. `NA` when the estimated variance is 0, except under `test = "permutation"`.

`n_1`, `n_2` The numbers of non-missing observations in the two groups.

`ci_method` The interval method used.

`df` The Satterthwaite degrees of freedom under `test = "t"`; `NA` under the other two tests and when the estimated variance is 0.

`test` The reference distribution used: `"t"`, `"permutation"`, or `"z"`.

The table carries these attributes: `conf_level`, which the console prints beneath the table and `results_sentence` reports; `n_missing`, the numbers of missing values removed from the two groups, as `c(group_1 = , group_2 = )`; `interval_degenerate`, present only when the estimated variance is 0, with the value `"complete_separation"` or `"no_variation"`; and `resampling`, present only under `test = "permutation"`, a list with `exact` (whether every reassignment was enumerated), `n_arrangements` (`choose(n_1 + n_2, n_1)`), and `n_resamples_used` (the number of reassignments compared). The `tidy()` and `glance()` verbs (see `dmar_tidiers`) give the row in their column convention, with `df` and `test` after `p_value` and a logical column `interval_degenerate`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Brunner, E., & Munzel, U. (2000). The nonparametric Behrens-Fisher problem: Asymptotic theory and a small-sample approximation. *Biometrical Journal, 42*(1), 17–25. doi:10.1002/(SICI)1521-4036(200001)42:1<17::AIDBIMJ17>3.0.CO;2U

DeLong, E. R., DeLong, D. M., & Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics, 44*(3), 837–845. doi:10.2307/2531595

Feng, D., & Cliff, N. (2004). Monte Carlo evaluation of ordinal d with improved confidence interval. *Journal of Modern Applied Statistical Methods, 3*(2), 322–332. doi:10.22237/jmasm/1099267560

McGraw, K. O., & Wong, S. P. (1992). A common language effect size statistic. *Psychological Bulletin, 111*(2), 361–365. doi:10.1037/00332909.111.2.361

Neubert, K., & Brunner, E. (2007). A studentized permutation test for the non-parametric Behrens-Fisher problem. *Computational Statistics & Data Analysis, 51*(10), 5192–5204. doi:10.1016/j.csda.2006.05.024

Pohar Perme, M., & Manevski, D. (2019). Confidence intervals for the Mann-Whitney test. *Statistical Methods in Medical Research, 28*(12), 3755–3768. doi:10.1177/0962280218814556

Schüürhuis, S., Konietschke, F., & Brunner, E. (2025). A new approach to the nonparametric Behrens-Fisher problem with compatible confidence intervals. *arXiv*, 2504.01796. https://arxiv.org/abs/2504.01796

Vargha, A., & Delaney, H. D. (2000). A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics, 25*(2), 101–132. doi:10.3102/10769986025002101

See Also

`cliff_delta`, `probability_of_superiority_paired`, `randomization_test`, `cles`, `smd`, `ci_smd`, `cohen_kappa`

Examples

```
# Two numeric vectors.
set.seed(113)
```

```
x <- rnorm(40, mean = 0.6)
y <- rnorm(40, mean = 0)
vargha_delaney_A(x, y)

# Formula interface on the pygmalion field experiment. The first factor
# level (Control) forms group 1, so A below 0.5 says a randomly drawn
# control child tends to score below a child from the bloomer group.
vargha_delaney_A(iq_8 ~ treatment, data = pygmalion)

# Wald (untransformed) interval rather than logit.
vargha_delaney_A(x, y, ci_method = "wald")

# The studentized permutation test and the normal reference. With
# 40 per group the reassignments are sampled, so a seed is given.
# n_resamples = 2000 keeps the example quick; a reported p-value
# deserves the default of 10,000.
vargha_delaney_A(x, y, test = "permutation", n_resamples = 2000,
                 seed = 113)
vargha_delaney_A(x, y, test = "z")

# An ordinal response is supplied as an ordered factor with its levels
# stated from lowest to highest. The unordered factors rating_1 and
# rating_2 themselves are refused, with a message that says why: their
# codes follow the alphabetical order of the labels (high, low, mid).
rating_1 <- factor(c("low", "mid", "high", "high"))
rating_2 <- factor(c("low", "low", "mid", "high"))
lev <- c("low", "mid", "high")
vargha_delaney_A(factor(rating_1, levels = lev, ordered = TRUE),
                 factor(rating_2, levels = lev, ordered = TRUE))

# Completely separated groups: the estimated variance is 0, the two
# limits equal the estimate, and a warning says that they are not a
# confidence interval. The permutation test still gives a p-value, the
# share of the 252 reassignments that are also completely separated.
separated <- vargha_delaney_A(6:10, 1:5, test = "permutation")
separated
results_sentence(separated)
```

`variance_components_mls` *Modified-Large-Sample Confidence Intervals on Variance Components*

Description

Computes modified-large-sample (MLS) confidence intervals on the between-group and within-group variance components of a balanced one-way random-effects ANOVA, following Burdick & Graybill (1992). MLS intervals have substantially better coverage than the Satterthwaite or simple-Wald intervals when the components are far from zero, and they are the standard interval method in generalizability theory (Brennan, 2001).

Usage

```
variance_components_mls(
  ms_between,
  ms_within,
  df_between,
  df_within,
  n_per_group,
  conf_level = 0.95
)
```

Arguments

`ms_between`	Mean square between groups (numerator of the ANOVA F).
`ms_within`	Mean square within groups (denominator of the ANOVA F).
`df_between`	Degrees of freedom for the between-group MS (typically $a - 1$ for a groups).
`df_within`	Degrees of freedom for the within-group MS (typically $a(n-1)$ for a groups of size n).
`n_per_group`	Number of observations per group (assumed balanced).
`conf_level`	Confidence level for the CIs. Default `0.95`.

Details

Point estimates. For a one-way random-effects ANOVA on a groups of size n, the method-of-moments estimators are

$$\hat{\sigma}_b^2 \;=\; \max(0,\, (MS_b - MS_w)/n), \qquad \hat{\sigma}_w^2 \;=\; MS_w.$$

Modified-large-sample CIs (Burdick-Graybill 1992). The MLS interval for σ_b^2 is

$$\left[\frac{MS_b - MS_w - \sqrt{V_L}}{n},\;\; \frac{MS_b - MS_w + \sqrt{V_U}}{n}\right],$$

with

$$V_L \;=\; G_1^2 MS_b^2 + H_2^2 MS_w^2 + G_{12} MS_b MS_w, \qquad V_U \;=\; H_1^2 MS_b^2 + G_2^2 MS_w^2 + H_{12} MS_b MS_w,$$

where the constants G_1, G_2, H_1, H_2 and the cross-term constants G_{12}, H_{12} depend on the degrees of freedom and on χ^2 and F quantiles at the chosen confidence level (Burdick & Graybill, 1992, equations 2.4.1–2.4.5 give the explicit formulas). The lower limit is truncated at zero. For the within-group component, the standard χ^2-based CI on MS_w (Searle, Casella, & McCulloch, 1992) is used.

Caveats. MLS intervals assume balanced data and homogeneous variances within groups. For unbalanced data the appropriate analog is the Burdick-Graybill MLS extension to unequal sample sizes (Burdick & Graybill, 1992, Section 2.5), which is not implemented here.

Value

A `data.frame` with rows for the point estimates and MLS lower / upper CIs of the between-group variance component (σ_b^2), the within-group component (σ_w^2), and the implied intraclass correlation ($\rho = \sigma_b^2/(\sigma_b^2 + \sigma_w^2)$): the rows `sigma2_between`, `sigma2_within`, and `icc`, each followed by its limits, `<estimate>_ci_lower` and `<estimate>_ci_upper`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Brennan, R. L. (2001). *Generalizability theory*. Springer.

Burdick, R. K., & Graybill, F. A. (1992). *Confidence intervals on variance components*. Marcel Dekker.

Searle, S. R., Casella, G., & McCulloch, C. E. (1992). *Variance components*. Wiley.

See Also

icc, var_icc, ss_aipe_icc

Other agreement and measurement: R2_mixed_effects(), content_validity_index(), gwet_ac(), icc_lmer(), krippendorff_alpha(), limits_of_agreement(), lin_ccc()

Examples

```
# 1. Balanced one-way random-effects ANOVA: a = 10 groups, n = 5.
#        Hypothetical MS_b = 6.0, MS_w = 1.5.
variance_components_mls(ms_between = 6.0, ms_within = 1.5,
                        df_between = 9, df_within = 40, n_per_group = 5)
```

---

var_alpha *Asymptotic Variance of Coefficient Alpha (Cronbach, Guttman)*

Description

Computes two large-sample approximations to the variance of the sample coefficient alpha (Guttman, 1945; Cronbach, 1951) under multivariate normality of the J item scores and compound symmetry of their covariance matrix (equal item variances and equal covariances, the parallel-items model): the asymptotic variance of van Zyl, Neudecker, and Nel (2000), $2J(1-\alpha)^2/((J-1)(N-1))$, and the variance that follows from Bonett (2002), $2J(1-\alpha)^2/((J-1)(N-2))$.

Usage

```
var_alpha(alpha, N, n_items)
```

Arguments

alpha	Population coefficient alpha. Numeric scalar in $[0, 1)$.
N	Total sample size (number of *respondents*).
n_items	Number of items contributing to the composite alpha coefficient (J in the formulas); at least 2.

Details

For an interval estimate of alpha from data see `reliability_alpha` and `reliability`; for sample size planning see `ss_aipe_reliability`.

van Zyl, Neudecker, and Nel (2000). Under multivariate normality and compound symmetry, van Zyl et al. show that $n \operatorname{Var}(\hat{\alpha})$ tends to $2J(1-\alpha)^2/(J-1)$ as n grows, where $n = N - 1$ is the degrees of freedom of the item covariance matrix (their Equation 13). The asymptotic variance is therefore

$$\operatorname{Var}(\hat{\alpha}) \approx \frac{2J(1-\alpha)^2}{(J-1)(N-1)}.$$

Without compound symmetry the asymptotic variance is a function of the whole item covariance matrix (their Equation 21); `reliability_alpha` uses that general form for its `"ml"` interval, and at a compound symmetric covariance matrix it equals the form above.

Bonett (2002). Bonett (2003, Equations 2 and 4), citing Bonett (2002), uses $2J/((J-1)(N-2))$ as the large-sample variance of $\ln(1-\hat{\alpha})$ under the same assumptions. Carried to the alpha scale by the delta method it gives

$$\operatorname{Var}(\hat{\alpha}) \approx \frac{2J(1-\alpha)^2}{(J-1)(N-2)}.$$

Comparing the two. The two forms differ only in the denominator: the form with $N-2$ is the form with $N-1$ times $(N-1)/(N-2)$, and a sample size chosen to reach a target variance is one larger with the $N-2$ denominator than with $N-1$. Both are large-sample approximations. Under multivariate normality and compound symmetry the exact variance of $\hat{\alpha}$ for $N > 5$ is

$$\frac{2(1-\alpha)^2 n(nJ-2)}{(J-1)(n-2)^2(n-4)}, \quad n = N-1$$

(van Zyl, Neudecker, & Nel, 2000, Equation 12). It is larger than both forms, and the form with $N-2$ is the closer of the two: with $J = 5$ the exact variance is 2.53 times that form at $N = 10$, 1.27 times at $N = 30$, and 1.07 times at $N = 100$.

Value

A `data.frame` with columns `term` and `value` and two rows: `var_alpha_van_zyl`, the asymptotic variance of van Zyl, Neudecker, and Nel (2000), and `var_alpha_bonett`, the variance that follows from Bonett (2002).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G. (2002). Sample size requirements for testing and estimating coefficient alpha. *Journal of Educational and Behavioral Statistics, 27*(4), 335–340. doi:10.3102/10769986027004335

Bonett, D. G. (2003). Sample size requirements for comparing two alpha coefficients. *Applied Psychological Measurement, 27*(1), 72–74. doi:10.1177/0146621602239477

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334.

Guttman, L. (1945). A basis for analyzing test-retest reliability. *Psychometrika, 10*(4), 255–282.

Kelley, K., & Cheng, Y. (2012). Estimation of and confidence interval formation for reliability coefficients of homogeneous measurement instruments. *Methodology, 8*, 39–50. doi:10.1027/1614-2241/a000036

Kelley, K., & Pornprasertmanit, S. (2016). Confidence intervals for population reliability coefficients: Evaluation of methods, recommendations, and software for composite measures. *Psychological Methods, 21*, 69–92. doi:10.1037/a0040086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

McDonald, R. P. (1999). *Test theory: A unified treatment*. Lawrence Erlbaum.

Terry, L. J., & Kelley, K. (2012). Sample size planning for composite reliability coefficients: Accuracy in parameter estimation via narrow confidence intervals. *British Journal of Mathematical and Statistical Psychology, 65*, 371–401. doi:10.1111/j.20448317.2011.02030.x

van Zyl, J. M., Neudecker, H., & Nel, D. G. (2000). On the distribution of the maximum likelihood estimator of Cronbach's alpha. *Psychometrika, 65*(3), 271–280. doi:10.1007/BF02296146

See Also

`reliability_alpha`, `reliability`, `ss_aipe_reliability`

Other variance utilities: `var_cv()`, `var_ete()`, `var_indirect_effect()`, `var_omega_squared()`, `var_r()`, `var_smd()`, `var_smd_trimmed()`

Examples

```
# 1. Variance of alpha = 0.80 from a 10-item test with N = 100.
var_alpha(alpha = 0.80, N = 100, n_items = 10)

# 2. Variance shrinks with N and grows as alpha moves away from 1:
var_alpha(alpha = 0.90, N = 50,  n_items = 5)
var_alpha(alpha = 0.90, N = 500, n_items = 5)
```

`var_cv` *Asymptotic Variance of the Coefficient of Variation*

Description

Computes the large-sample variance of the sample coefficient of variation $k = s/M$, the sample standard deviation divided by the sample mean, for a random sample from a normal population: $\kappa^2(1/2 + \kappa^2)/(N - 1)$. It takes the population coefficient of variation and the sample size and returns the variance. Companion to `ci_cv` and `ss_aipe_cv`.

Usage

```
var_cv(true_cv, N)
```

Arguments

`true_cv`	Population coefficient of variation $\kappa = \sigma/\mu$. A single finite number greater than 0. For κ above about 0.4 the value returned is too small, and more so the larger κ is (see Details); for an interval use `ci_cv`, which does not use this variance.
`N`	Total sample size. A single finite number of at least 3.

Details

Estimand. The variance, over repeated samples of N independent observations from a normal population with mean $\mu > 0$ and standard deviation σ, of the sample coefficient of variation $k = s/M$, evaluated at the population value $\kappa = \sigma/\mu$ given as `true_cv`. Strictly, the sample coefficient has no finite variance, because the sample mean can be arbitrarily close to zero; the expression below describes its spread when that event has negligible probability, that is, when $\sqrt{N}/\kappa$ is large. Substituting a sample coefficient for `true_cv` gives a plug-in estimate of the variance, which carries the sampling error of the coefficient it was computed from.

The variance. The first-order (delta method) variance is

$$\operatorname{Var}(k) \;\approx\; \frac{\kappa^2}{N-1}\left(\frac{1}{2}+\kappa^2\right).$$

It follows from $\operatorname{Var}(s) \approx \sigma^2/\{2(N-1)\}$, $\operatorname{Var}(M) = \sigma^2/N$, and the independence of s and M under normality, which give $\operatorname{Var}(k) \approx \kappa^2/\{2(N-1)\} + \kappa^4/N$. That expression is Expression 5 of Sokal and Braumann (1980, p. 52) without its third-order term. The function uses $N-1$ under both terms, which agrees with it to first order in $1/N$ and is larger by $\kappa^4/\{N(N-1)\}$. The form usually given, their Expression 7, has N under both terms: $\kappa^2(1/2+\kappa^2)/N$.

Accuracy under normality. The comparisons in this paragraph are with the variance of k when the sample mean is restricted to lie within 8 standard errors of μ and to be at least $\mu/50$. The restriction excludes a probability below 10^{-9} in every case reported, and the figures given are the same for any lower bound from $\mu/1000$ to $\mu/5$. For $\kappa \le 0.3$ and $N \ge 10$ the value is too large. The excess is largest for a small κ, where it is that of $\sigma^2/\{2(N-1)\}$ over the exact variance of s: to one decimal place, 3.0 percent at $N = 10$, 1.4 percent at $N = 20$, 0.9 percent at $N = 30$, and 0.5 percent at $N = 50$. For κ above about 0.4 the value is too small, because of the terms of higher order that a first-order variance omits: by 5.7 percent at $\kappa = 0.5$ and $N = 10$, and by 1.7 percent at $\kappa = 0.5$ and $N = 30$.

Normality. The value is for a normal population. For any population with a finite fourth moment, the large-sample variance of k is ξ^2/N, with

$$\xi^2 \;=\; \frac{\kappa^2}{2} + \kappa^4 - \kappa^3\gamma_1 + \frac{\kappa^2\gamma_2}{4},$$

where γ_1 is the skewness and γ_2 the excess kurtosis of the population (Kelley, Darku, & Chattopadhyay, 2018, Equation 30, which is written in central moments). Under normality both are 0 and $\xi^2 = \kappa^2(1/2+\kappa^2)$. For other populations the normal-theory value can be far off in either direction: $\kappa^2(1/2+\kappa^2)$ is 39 percent larger than ξ^2 for a gamma population with $\kappa = 0.8$, and 56 percent smaller for a Laplace (double exponential) population with $\kappa = 0.3$. These compare the limiting values, so a larger sample does not remove the difference. `mr_cv` estimates ξ^2 from the scores, without assuming a normal population.

Use. Neither ci_cv nor ss_aipe_cv uses this variance: the interval inverts the noncentral t distribution of $\sqrt{N}/k$, and the planner (Kelley, 2007) chooses the sample size for that interval. The value serves as a quick description of the sampling variability of k and for comparison with a simulated variance. McKay (1932) and Vangel (1996) approximate the distribution of a function of k by a chi square distribution, from which confidence limits follow; the variance here is not taken from those approximations.

Value

A data.frame with the columns term and value and one row, var_cv_normal: the first-order variance of the sample coefficient of variation under normality (see Details).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007). Sample size planning for the coefficient of variation from the accuracy in parameter estimation approach. *Behavior Research Methods, 39*(4), 755–766. doi:10.3758/BF03192966

Kelley, K., Darku, F. B., & Chattopadhyay, B. (2018). Accuracy in parameter estimation for a general class of effect sizes: A sequential approach. *Psychological Methods, 23*(2), 226–243. doi:10.1037/met0000127

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

McKay, A. T. (1932). Distribution of the coefficient of variation and the extended t distribution. *Journal of the Royal Statistical Society, 95*(4), 695–698.

Sokal, R. R., & Braumann, C. A. (1980). Significance tests for coefficients of variation and variability profiles. *Systematic Zoology, 29*(1), 50–66. doi:10.1093/sysbio/29.1.50

Vangel, M. G. (1996). Confidence intervals for a normal coefficient of variation. *The American Statistician, 50*(1), 21–26. doi:10.1080/00031305.1996.10473537

See Also

cv, ci_cv, ss_aipe_cv, ss_aipe_cv_sensitivity, mr_cv

Other variance utilities: var_alpha(), var_ete(), var_indirect_effect(), var_omega_squared(), var_r(), var_smd(), var_smd_trimmed()

Examples

```
# The variance of the sample coefficient of variation in samples of 30
# from a normal population whose coefficient is .20. The variance is
# small, so it is printed to six decimal places.
v <- var_cv(true_cv = 0.20, N = 30)
print(v, digits = 6)

# Its square root is the standard error of the sample coefficient, shown
# here beside the standard deviation of 2000 simulated coefficients.
```

```
set.seed(113)
k <- replicate(2000, {
  y <- rnorm(30, mean = 10, sd = 2)
  sd(y) / mean(y)
})
c(normal_theory = sqrt(v$value), simulated = sd(k))

# The same comparison for a skewed population, a gamma distribution
# with a coefficient of .80 and samples of 200: the normal-theory
# standard error is too large.
k_gamma <- replicate(2000, {
  y <- rgamma(200, shape = 1 / 0.80^2)
  sd(y) / mean(y)
})
c(normal_theory = sqrt(var_cv(true_cv = 0.80, N = 200)$value),
  simulated = sd(k_gamma))
```

`var_ete` *Variance of the Estimated Treatment Effect in Two-Group ANCOVA With Heterogeneous Slopes*

Description

Computes the variance of the estimated treatment effect (ETE) at a chosen covariate value in a two-group analysis of covariance with heterogeneity of regression and a random covariate, following Li, McLouth, and Delaney (2020). When the two groups' slopes differ, the treatment effect is a function of the covariate, and its sampling variance at the sample grand mean, one standard deviation from the mean, or a fixed covariate value must account for the covariate being a random variable rather than a set of fixed constants; the fixed-constant formulas understate or misstate that variability. This is the reimplementation of `var.ete()` from **MBESS**, contributed there by Li Li.

Usage

```
var_ete(
  error_variance,
  var_Z,
  n_1,
  n_2,
  reg_coef_1,
  reg_coef_2,
  mean_Z = 0,
  fixed_value = 0,
  type = c("sample", "population"),
  covariate_value = c("sample_mean", "sd", "fixed")
)
```

Arguments

error_variance	Residual error variance: the population value when type = "population", the sample estimate when type = "sample".
var_Z	Variance of the random covariate: population value or sample estimate, matching type.
n_1, n_2	Sample sizes of the two groups (each must exceed 3; the formulas involve $n-3$ in denominators).
reg_coef_1, reg_coef_2	Regression coefficients (slopes) of the covariate in group 1 and group 2: population values or sample estimates, matching type.
mean_Z	Mean of the covariate (population value or sample mean, matching type). Defaults to 0. Used when covariate_value = "fixed".
fixed_value	The fixed covariate value at which the treatment effect is assessed when covariate_value = "fixed". Defaults to 0.
type	"sample" (default) for the unbiased estimate of the variance from sample slopes and variances, or "population" for the variance from population values.
covariate_value	Where the treatment effect is assessed: "sample_mean" (default) at the sample grand mean of the covariate, "sd" at the grand mean plus or minus one sample standard deviation, or "fixed" at fixed_value.

Details

When the slopes differ across groups, the treatment effect depends on the value of the covariate, and this function gives its variance at the grand mean of the covariate, one standard deviation either side of it, or a chosen value (covariate_value). The variance expressions here treat the covariate as normally distributed rather than fixed, which Li, McLouth, and Delaney (2020) show can change the estimated standard error substantially when heterogeneity of regression is strong. The square root of the returned value is the standard error used for a confidence interval or test of the treatment effect at the chosen covariate value.

At the sample grand mean of the covariate, writing $N = n_1 + n_2$, σ^2 for error_variance, σ_Z^2 for var_Z, and β_1 and β_2 for reg_coef_1 and reg_coef_2, the population variance (their Equation 10) is

$$\mathrm{Var} = \sigma^2 C_0 + \frac{(\beta_1 - \beta_2)^2 \sigma_Z^2}{N}, \qquad C_0 = \frac{1}{n_1} + \frac{1}{n_2} + \frac{n_2}{N n_1 (n_1 - 3)} + \frac{n_1}{N n_2 (n_2 - 3)},$$

and with type = "sample" the returned value is their unbiased estimator (Equation C.7), which subtracts $\sigma^2\{(N-3)/(n_1-3)+(N-3)/(n_2-3)\}/\{N(N-1)\}$ so that plugging in sample estimates does not overstate the variance. The covariate_value = "sd" expressions are their Equations 12 and C.9, which add the variance contribution of estimating the covariate's standard deviation, and the "fixed" expressions are their Equations 14 and C.10.

The two "fixed" estimands differ in where the deviation of fixed_value is measured from. With type = "sample" the deviation is taken from the sample grand mean (Equation C.10), so mean_Z should be the sample mean of the covariate. With type = "population" the deviation is taken from a known population mean (Equation 14); evaluating the treatment effect at that known mean itself, as in the paper's worked example, sets fixed_value = mean_Z, which zeroes the deviation term.

Value

A `data.frame` (class `dmar_tbl`) with one row, `term = "var_ete"`, whose `value` is the variance of the estimated treatment effect at the chosen covariate value. The `type` and `covariate_value` choices are recorded as attributes of the same names.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2007a). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K. (2007b). Methods for the behavioral, educational, and social sciences: An R package. *Behavior Research Methods, 39*(4), 979–984. doi:10.3758/BF03192993

Li, L., McLouth, C. J., & Delaney, H. D. (2020). Analysis of covariance in randomized experiments with heterogeneity of regression and a random covariate: The variance of the estimated treatment effect at selected covariate values. *Multivariate Behavioral Research, 55*(6), 926–940. doi:10.1080/00273171.2019.1693953

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 9 on heterogeneity of regression.)

See Also

`ancova` for the model whose treatment effect this variance describes; `regions_of_significance` for the companion question of where a moderated effect is distinguishable from zero.

Other variance utilities: `var_alpha()`, `var_cv()`, `var_indirect_effect()`, `var_omega_squared()`, `var_r()`, `var_smd()`, `var_smd_trimmed()`

Examples

```
# Pygmalion data (Maxwell, Delaney, & Kelley, 2027): the treatment
# effect of the "Bloomer" expectation at the covariate grand mean,
# with heterogeneous pre-IQ slopes.
data(pygmalion)
fit <- lm(iq_8 ~ iq_pre * treatment, data = pygmalion)
s2  <- sum(residuals(fit)^2) / fit$df.residual
var_ete(error_variance = s2, var_Z = var(pygmalion$iq_pre),
        n_1 = sum(pygmalion$treatment == "Bloomer"),
        n_2 = sum(pygmalion$treatment == "Control"),
        reg_coef_1 = coef(fit)["iq_pre"] +
          coef(fit)["iq_pre:treatmentBloomer"],
        reg_coef_2 = coef(fit)["iq_pre"])
```

var_icc *Asymptotic Variance of the Intraclass Correlation Coefficient*

Description

Computes the asymptotic (large-sample) variance of the intraclass correlation coefficient (ICC) for any of the six classical Shrout-Fleiss (1979) forms, given a value of the population ICC, the number of subjects N, and the number of raters k. The single-rater forms use Smith's (1956) one-way-ANOVA asymptotic variance, equivalently the Fisher (1925) information-matrix result, and the average-of-k forms use a delta method transformation through the Spearman-Brown relation.

Usage

```
var_icc(icc, N, n_raters, type = "ICC(1,1)")
```

Arguments

`icc`	The intraclass correlation coefficient (ρ in the formulas) at which the asymptotic variance is evaluated, on the scale matching `type`: for the single-rater types (`"ICC(1,1)"`, `"ICC(2,1)"`, `"ICC(3,1)"`) supply the single-rater ICC; for the average-of-k types (`"ICC(1,k)"`, `"ICC(2,k)"`, `"ICC(3,k)"`) supply the average-of-k ICC. Must lie in $[0, 1]$. **Population value or sample value?** The formula is derived in terms of the unknown population value ρ. In applied work ρ is never known, so two conventions are common: (i) for *prospective* variance calculation (e.g., when planning a study and asking how precise an estimator will be at plausible truths), supply your *anticipated* population value motivated by prior literature, theory, or a pilot; (ii) for *post hoc* variance calculation (e.g., constructing a Wald standard error or a meta-analytic weight from an observed sample), supply the *sample* estimate $\hat{\rho}$ as a plug-in for the population value. The two postures look identical at the call site but have different conceptual content; the plug-in case (ii) yields a consistent (not exact) estimator of the variance.
`N`	Total sample size: the number of subjects (targets) rated.
`n_raters`	Number of raters (or repeated measurements) per subject, k in the formulas and in the names of the ICC forms.
`type`	Which Shrout-Fleiss ICC variant the population value represents. One of `"ICC(1,1)"`, `"ICC(2,1)"`, `"ICC(3,1)"`, `"ICC(1,k)"`, `"ICC(2,k)"`, `"ICC(3,k)"`. The shorthand aliases used in `icc` (`"1"`, `"2"`, `"3"`, `"1k"`, `"2k"`, `"3k"`) are also accepted.

Details

Single-rater forms. For the single-rater intraclass correlation in a balanced design with N subjects and k measurements per subject, the standard large-sample variance derived from the Fisher information matrix is (Smith, 1956; Donner, 1986; Searle, 1971, ch. 11):

$$\mathrm{Var}(\hat{\rho}) \;\approx\; \frac{2(1-\rho)^2\big(1+(k-1)\rho\big)^2}{N\,k\,(k-1)}.$$

This expression is exact under the one-way random-effects model (`ICC(1,1)`) and serves as the standard large-sample approximation for the two-way single-rater forms (`ICC(2,1)`, `ICC(3,1)`) as well; differences with the exact two-way variance vanish at order $1/N$ (Bonett, 2002; Burdick & Graybill, 1992). For small N or moderate k where exact two-way inference matters, prefer the F-distribution-based confidence intervals returned by `icc`, which follow Shrout and Fleiss (1979) directly.

Average-of-*k* forms. Applying the Spearman-Brown transformation $\rho_k = k\rho/[1 + (k-1)\rho]$ together with the delta method gives the closed-form

$$\mathrm{Var}(\hat{\rho}_k) \approx \frac{2\,k\,(1-\rho_k)^2}{N\,(k-1)},$$

expressed directly in the average-level ICC ρ_k (so the user need not invert Spearman-Brown when working with reliability of composites). The reduction to this form follows from the substitutions $1-\rho = k(1-\rho_k)/[k-(k-1)\rho_k]$ and $1+(k-1)\rho = k/[k-(k-1)\rho_k]$.

Use cases. The asymptotic variance is the natural ingredient for Wald-style inference, sample size planning for the width of an ICC confidence interval (compare with Bonett, 2002, which uses a Fisher-style transformation), and meta-analytic synthesis of ICCs across studies (the inverse of `value` weights each study). For confidence intervals themselves, prefer `icc`, which uses the exact F-distribution inversion of Shrout and Fleiss (1979, pp. 425–426).

Value

A one-row `data.frame` with columns `term` (always `"var_icc"`) and `value` (the asymptotic variance).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G. (2002). Sample size requirements for estimating intraclass correlations with desired precision. *Statistics in Medicine, 21*(9), 1331–1335. doi:10.1002/sim.1108

Burdick, R. K., & Graybill, F. A. (1992). *Confidence Intervals on Variance Components*. Marcel Dekker.

Donner, A. (1986). A review of inference procedures for the intraclass correlation coefficient in the one-way random effects model. *International Statistical Review, 54*(1), 67–82.

Fisher, R. A. (1925). *Statistical Methods for Research Workers*. Oliver & Boyd.

McGraw, K. O., & Wong, S. P. (1996). Forming inferences about some intraclass correlation coefficients. *Psychological Methods, 1*(1), 30–46. doi:10.1037/1082989X.1.1.30

Searle, S. R. (1971). *Linear Models*. Wiley.

Shrout, P. E., & Fleiss, J. L. (1979). Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin, 86*(2), 420–428.

Smith, C. A. B. (1956). On the estimation of intraclass correlation. *Annals of Human Genetics, 21*(4), 363–373.

See Also

icc, ss_aipe_reliability, var_R2

Examples

```
# Single-rater one-way ICC at rho = .60 with 30 subjects and 4 raters.
var_icc(icc = 0.60, N = 30, n_raters = 4, type = "ICC(1,1)")

# Same study, but expressed at the average-of-4-rater level. The
# Spearman-Brown relation carries the single-rater value of .60 up
# to the average-of-4 scale, about .857.
rho_k <- 4 * 0.60 / (1 + 3 * 0.60)
rho_k
var_icc(icc = rho_k, N = 30, n_raters = 4, type = "ICC(1,k)")

# Two-way mixed-model consistency ICC, which uses the same one-way
# asymptotic variance as a large-sample approximation (see Details).
var_icc(icc = 0.60, N = 30, n_raters = 4, type = "ICC(3,1)")
```

---

var_indirect_effect *Variance of the Mediated (Indirect) Effect ab*

Description

Computes the asymptotic variance of the product of two regression coefficients $\hat{a}\hat{b}$ (the mediated/indirect effect in a simple three-variable mediator model: $X \rightarrow M \rightarrow Y$) under four competing formulas: Sobel (1982) first-order, Aroian (1947) / Goodman (1960) second-order, and the full second-order delta method with optional cross-product covariance. All four are reported in a single output so the user can see the relative contributions of the higher-order terms.

Usage

```
var_indirect_effect(
  reg_coef_a,
  reg_coef_b,
  var_reg_coef_a,
  var_reg_coef_b,
  cov_reg_coef_a_b = 0
)
```

Arguments

`reg_coef_a`, `reg_coef_b`
: The regression coefficients a of $X \rightarrow M$ and b of $M \rightarrow Y$: anticipated population values or estimates, on whatever scale the analysis uses (raw or standardized); the variances below must be on the same scale.

`var_reg_coef_a`, `var_reg_coef_b`
: Variances (squared standard errors) of $\hat{a}$ and $\hat{b}$. For standardized regression with no covariates, $\mathrm{Var}(\hat{a}) \approx (1-a^2)/(n-2)$ and $\mathrm{Var}(\hat{b}) \approx (1-b^2)/\{(n-3)(1-a^2)\}$, the standardized-model variance accounting for the correlation the a path induces between the predictors of Y.

`cov_reg_coef_a_b`
: Optional covariance between $\hat{a}$ and $\hat{b}$. Defaults to 0 (the assumption underlying the standard Sobel formula). In practice the two estimators are nearly uncorrelated when the controls in the Y-on-M regression are uncorrelated with the M-on-X regression's predictors.

Details

In the formulas below a and b are `reg_coef_a` and `reg_coef_b`, $\mathrm{Var}(\hat{a})$ and $\mathrm{Var}(\hat{b})$ are `var_reg_coef_a` and `var_reg_coef_b`, and $\mathrm{Cov}(\hat{a}, \hat{b})$ is `cov_reg_coef_a_b`.

Sobel (1982) first-order. The delta method variance of $\hat{a}\hat{b}$ under independent $\hat{a}$, $\hat{b}$ is

$$\mathrm{Var}_{\mathrm{Sobel}}(\hat{a}\hat{b}) \;=\; a^2\mathrm{Var}(\hat{b}) + b^2\mathrm{Var}(\hat{a}).$$

This is the most cited form and is the variance used by the standard Sobel z-test (Sobel, 1982).

Aroian (1947). Aroian retains the second-order term:

$$\mathrm{Var}_{\mathrm{Aroian}}(\hat{a}\hat{b}) \;=\; a^2\mathrm{Var}(\hat{b}) + b^2\mathrm{Var}(\hat{a}) + \mathrm{Var}(\hat{a})\,\mathrm{Var}(\hat{b}).$$

Aroian shows this is exact under joint normality of the two independent estimators.

Goodman (1960). Goodman's "unbiased" variance subtracts the second-order term instead of adding it:

$$\mathrm{Var}_{\mathrm{Goodman}}(\hat{a}\hat{b}) \;=\; a^2\mathrm{Var}(\hat{b}) + b^2\mathrm{Var}(\hat{a}) - \mathrm{Var}(\hat{a})\,\mathrm{Var}(\hat{b}).$$

For small variances the three forms agree to leading order; they diverge for noisy $\hat{a}$, $\hat{b}$.

Second-order delta method (with covariance). When $\hat{a}$ and $\hat{b}$ share variability (e.g., they are both estimated from the same regression of Y on X and M), the cross-product covariance term enters:

$$\mathrm{Var}(\hat{a}\hat{b}) \;\approx\; a^2\mathrm{Var}(\hat{b}) + b^2\mathrm{Var}(\hat{a}) + 2ab\cdot\mathrm{Cov}(\hat{a}, \hat{b}).$$

MacKinnon et al.\ (2002) show this matters in models with covariates that simultaneously load on M and Y.

Connection to `ss_aipe_indirect_effect`. The Sobel (delta method) variance is what `ss_aipe_indirect_effect` builds on under `method = "closed_form"` for AIPE planning; this function makes the alternative formulas available for explicit comparison.

Value

A `data.frame` with rows for the four variance formulas; columns are `term` and `value`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Aroian, L. A. (1947). The probability function of the product of two normally distributed variables. *The Annals of Mathematical Statistics, 18*(2), 265–271.

Goodman, L. A. (1960). On the exact variance of products. *Journal of the American Statistical Association, 55*(292), 708–713.

Lachowicz, M. J., Preacher, K. J., & Kelley, K. (2018). A novel measure of effect size for mediation analysis. *Psychological Methods, 23*(2), 244–261. doi:10.1037/met0000165

MacKinnon, D. P., Lockwood, C. M., Hoffman, J. M., West, S. G., & Sheets, V. (2002). A comparison of methods to test mediation and other intervening variable effects. *Psychological Methods, 7*(1), 83–104. doi:10.1037/1082989X.7.1.83

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Preacher, K. J., & Kelley, K. (2011). Effect size measures for mediation models: Quantitative strategies for communicating indirect effects. *Psychological Methods, 16*(2), 93–115. doi:10.1037/a0022658

Sobel, M. E. (1982). Asymptotic confidence intervals for indirect effects in structural equation models. *Sociological Methodology, 13*, 290–312.

Tofighi, D., & Kelley, K. (2020). Improved inference in mediation analysis: Introducing the model-based constrained optimization procedure. *Psychological Methods, 25*(4), 496–515. doi:10.1037/met0000259

See Also

`ss_aipe_indirect_effect`

Other variance utilities: `var_alpha()`, `var_cv()`, `var_ete()`, `var_omega_squared()`, `var_r()`, `var_smd()`, `var_smd_trimmed()`

Examples

```
# 1. Coefficients of 0.40 for both paths, each with a sampling variance
#    of 0.02, and no covariance:
var_indirect_effect(reg_coef_a = 0.40, reg_coef_b = 0.40,
                    var_reg_coef_a = 0.02, var_reg_coef_b = 0.02)

# 2. With a positive covariance between a-hat and b-hat:
var_indirect_effect(reg_coef_a = 0.40, reg_coef_b = 0.40,
                    var_reg_coef_a = 0.02, var_reg_coef_b = 0.02,
                    cov_reg_coef_a_b = 0.005)
```

`var_omega_squared` *Asymptotic Variance of Omega Squared (ANOVA Effect Size)*

Description

Computes the large-sample (delta method) variance of the sample $\hat{\omega}^2$ (Hays' 1994 bias-corrected estimator) in a fixed-effects ANOVA. Fleishman (1980, Eq. 22, p. 669) gives the exact variance of the unbiased estimator of the signal-to-noise ratio $f^2 = \sigma_a^2/\sigma_e^2$ under the noncentral F sampling distribution of the observed F statistic; because $\omega^2 = f^2/(1+f^2)$ (his Eq. 8), the delta method carries that variance to the ω^2 scale with the Jacobian $\mathrm{d}\omega^2/\mathrm{d}f^2 = (1-\omega^2)^2$. Fleishman gives no variance on the ω^2 scale himself, so the transfer is this package's step rather than his. The result is the natural companion to `ci_omega_squared` (CI) and `omega_squared` (point estimate).

Usage

```
var_omega_squared(
  true_omega_squared = NULL,
  df_effect = NULL,
  df_error = NULL,
  N = NULL,
  object = NULL
)
```

Arguments

`true_omega_squared`	Population ω^2, the value at which the sampling variance is evaluated. Numeric scalar in $[0, 1)$. Ignored when `object` is supplied.
`df_effect`	Numerator degrees of freedom for the effect. Ignored when `object` is supplied.
`df_error`	Error (residual) degrees of freedom. Ignored when `object` is supplied.
`N`	Total sample size. Ignored when `object` is supplied.
`object`	Optional fitted `aov` or `lm` object. When supplied, the function loops over the non-`Residuals` rows of `anova(object)` and returns one row per effect, plugging in the sample $\hat{\omega}_p^2$ for each as the working population value.

Details

Derivation. In a fixed-effects ANOVA with numerator df df_1 and denominator df df_2, the observed F statistic follows a noncentral F distribution with noncentrality $\lambda = df_1(F-1)$ when $\hat{\omega}^2 = df_1(F-1)/[df_1(F-1)+N]$ is the population value (Hays, 1994). The asymptotic variance of $\hat{\omega}^2$ is obtained by the delta method on this relationship (Fleishman, 1980), yielding:

$$\mathrm{Var}(\hat{\omega}^2) \;\approx\; \frac{2\cdot df_1\cdot(df_2-2)(1-\omega^2)^2(1+\lambda^*/df_1)^2}{N^2(df_2-4)},$$

with $\lambda^* = \omega^2 N/(1-\omega^2)$ the noncentrality implied by the population value. This is the form used by ci_omega_squared when constructing a Wald-style interval; the noncentral F CI is generally preferred.

Caveats. The variance is a delta method approximation; it becomes inaccurate when df_2 is small (< 10), when ω^2 is near the boundaries 0 or 1, or when the residual distribution is heavy-tailed. For small-sample inference, the noncentral F CI (ci_omega_squared) is preferred over a Wald-style interval built on this variance.

Value

A 1-row data.frame with columns term ("var_omega_squared") and value (the asymptotic variance).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Fleishman, A. I. (1980). Confidence intervals for correlation ratios. *Educational and Psychological Measurement, 40*(3), 659–670.

Hays, W. L. (1994). *Statistics* (5th ed.). Fort Worth, TX: Harcourt Brace College Publishers.

Kelley, K. (2007). Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software, 20*(8), 1–24. doi:10.18637/jss.v020.i08

Kelley, K., & Preacher, K. J. (2012). On effect size. *Psychological Methods, 17*, 137–152. doi:10.1037/a0028086

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on η^2, Chapter 7 on factorial designs, and Chapter 11 on generalized η^2 for within-subjects designs.)

Olejnik, S., & Algina, J. (2003). Generalized eta and omega squared statistics: Measures of effect size for some common research designs. *Psychological Methods, 8*(4), 434–447. doi:10.1037/1082989X.8.4.434

Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods, 9*(2), 164–182. doi:10.1037/1082989X.9.2.164

See Also

omega_squared, ci_omega_squared, ss_aipe_omega_squared

Other variance utilities: var_alpha(), var_cv(), var_ete(), var_indirect_effect(), var_r(), var_smd(), var_smd_trimmed()

Examples

```
# 1. One way ANOVA: 3 groups (df_effect = 2), 60 total (df_error = 57),
#        population omega^2 = 0.10.
var_omega_squared(true_omega_squared = 0.10,
```

```
                  df_effect = 2,
                  df_error = 57,
                  N = 60)

# 2. Per-effect variance from a fitted lm() / aov() (pygmalion data:
#        expectancy treatment x grade, N = 310):
fit_factorial <- aov(iq_8 ~ treatment * factor(grade), data = pygmalion)
var_omega_squared(object = fit_factorial)
```

`var_partial_r` *Asymptotic Variance of the Partial Correlation Coefficient*

Description

Computes the large-sample variance of the sample partial correlation coefficient $r_{XY \cdot Z_1 \cdots Z_{p-1}}$ under multivariate normality. Two formulas are available: the first-order variance on the raw r scale (the default), and the variance $1/(N-p-2)$ of the Fisher's Z transformation (Fisher, 1921, 1924), which is the appropriate quantity for constructing a confidence interval by transformation and back-transformation.

Usage

```
var_partial_r(r, N, p = 2, fisher_z = FALSE)
```

Arguments

`r`	The sample partial correlation coefficient $r_{XY \cdot Z_1 \cdots Z_{p-1}}$. Must be in $[-1, 1]$.
`N`	Total sample size.
`p`	Number of predictors in the regression of Y on X and the variables partialed out, so `p - 1` variables are partialed out and `p` is at least 2. A categorical variable with a levels counts as a - 1 predictors. It is the `p` of `ss_power_R2`, `ci_R2`, and `ci_reg_coef`: the correlation concerns one predictor of a regression with `p` predictors. In the notation of a model comparison, q is the number of predictors removed from the full model to give the restricted model ($q = p_{full} - p_{restricted}$, the predictors being tested), and the number of variables partialed out is $p - q$; this correlation concerns one predictor, so $q = 1$ and $p - 1$ variables are partialed out. Defaults to 2 (one variable partialed out).
`fisher_z`	Logical. If `FALSE` (the default), the function returns the raw-scale first-order variance. If `TRUE`, it returns the variance of the Fisher's Z transformation $Z = \operatorname{arctanh}(r)$ under the Fisher (1924) reduction.

Details

Raw scale, first-order variance. The first-order variance of a simple correlation is $(1-\rho^2)^2/N$ (Olkin & Finn, 1995, Equation 3). The sample partial correlation given $p-1$ variables is distributed as a simple correlation from $N-(p-1)$ observations, and the functions here use

$$\mathrm{Var}(\hat{r}_{XY\cdot Z}) \;\approx\; \frac{(1-\rho^2_{XY\cdot Z})^2}{N-p},$$

which differs from that at order $1/N^2$. The function evaluates this with $\hat{r}$ substituted for ρ. Olkin and Finn (1995) give the variance of a simple correlation; the divisor $N-p$ is this package's choice and is not taken from that article.

Fisher's *Z* Transformation. Fisher (1921) showed that for a Pearson correlation, the transformation $Z = \frac{1}{2}\log\{(1+r)/(1-r)\} = \mathrm{arctanh}(r)$ is approximately normal with variance $1/(N-3)$. Fisher (1924, p. 330) showed that the partial correlation based on N observations with $p-1$ variables partialed out is distributed as a simple correlation from a sample reduced in size by $p-1$; combined with the Fisher (1921) variance of Z, the Fisher's *Z* transformation of $\hat{r}_{XY\cdot Z_1\cdots Z_{p-1}}$ is therefore approximately normal with variance $1/(N-p-2)$. This is the standard ingredient for constructing a confidence interval on $\rho_{XY\cdot Z}$ by transforming, building a Wald interval on Z, and back-transforming with $\tanh$.

Value

A one-row `data.frame` with columns `term` (either `"var_partial_r"` or `"var_fisher_z_partial_r"`) and `value` (the requested variance).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Lawrence Erlbaum.

Fisher, R. A. (1915). Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika, 10*(4), 507–521.

Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron, 1*, 3–32.

Fisher, R. A. (1924). The distribution of the partial correlation coefficient. *Metron, 3*, 329–332.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on contrasts, Chapter 5 on multiple comparisons, and Chapter 9 on ANCOVA.)

Olkin, I., & Finn, J. D. (1995). Correlations redux. *Psychological Bulletin, 118*(1), 155–164. doi:10.1037/00332909.118.1.155

See Also

var_semipartial_r, var_R2, convert_r_Z, ci_r

Examples

```
# First-order variance for r_p = .35, N = 80, and two variables
#     partialed out (p = 3 predictors: X and the two variables).
var_partial_r(r = 0.35, N = 80, p = 3)

# Variance of Fisher's Z transformation (Fisher, 1921, 1924) for the
#     same setting, useful for building a CI on rho_p.
var_partial_r(r = 0.35, N = 80, p = 3, fisher_z = TRUE)
```

---

var_r *Asymptotic Variance of the Pearson Correlation Coefficient*

Description

Computes the large-sample variance of the sample Pearson correlation under bivariate normality and the variance of its Fisher *Z* transform. This standalone function is useful for AIPE planning, meta-analytic weighting, and Wald-style inference on *r*.

Usage

```
var_r(true_cor, N)
```

Arguments

true_cor	Population correlation coefficient ρ. Numeric scalar or vector in $(-1, 1)$.
N	Total sample size on which the Pearson r would be computed. Scalar or vector.

Details

The variance of *r* is evaluated at the population correlation true_cor and the sample size N. Substituting a sample correlation for true_cor gives a plug-in estimate of that variance, which carries the sampling error of the correlation it was computed from. The variance of the Fisher *Z* transform, $1/(N-3)$, depends on N alone.

Normal-theory variance (default). Under bivariate normality the large-sample variance is

$$\mathrm{Var}(\hat{r}) \;\approx\; (1-\rho^2)^2/(N-1),$$

the leading term of the exact moment expansion (Hotelling, 1953, Section 7; the exact density of r is Fisher's, 1915). As N grows, N times this variance approaches $(1-\rho^2)^2$, the variance of the limiting distribution of $\sqrt{N}(\hat{r}-\rho)$. Carried to the Fisher Z scale by the delta method it becomes $1/(N-1)$; the interval of ci_r uses Fisher's $1/(N-3)$ on that scale instead (see below).

Non-normal margins. The normal-theory variance does not hold when the margins are not normal: marginal kurtosis can change the large-sample variance of r considerably when $\rho \neq 0$ (Bonett &

Wright, 2000, p. 24), and in general the variance depends on every fourth-order cumulant of the joint distribution (Duncan & Layard, 1973, Equation 1.1).

Connection to Fisher's Z transform. On the variance- stabilized scale $Z = \tanh^{-1}(r)$ the asymptotic variance is $1/(N-3)$ regardless of ρ (Fisher, 1921). This is reported alongside the raw-scale variance because it is the natural working scale for CI construction (`ci_r`).

Value

A `data.frame` with the rows

- `var_r_normal`, the normal-theory asymptotic variance $(1-\rho^2)^2/(N-1)$ (Hotelling, 1953, Section 7),
- `var_fisher_z`, the variance of the Fisher Z transform, $1/(N-3)$.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Bonett, D. G., & Wright, T. A. (2000). Sample size requirements for estimating Pearson, Kendall and Spearman correlations. *Psychometrika, 65*(1), 23–28. doi:10.1007/BF02294183

Duncan, G. T., & Layard, M. W. J. (1973). A Monte-Carlo study of asymptotically robust tests for correlation coefficients. *Biometrika, 60*(3), 551–558. doi:10.1093/biomet/60.3.551

Fisher, R. A. (1915). Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika, 10*(4), 507–521.

Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron, 1*, 3–32.

Hotelling, H. (1953). New light on the correlation coefficient and its transforms. *Journal of the Royal Statistical Society, Series B, 15*(2), 193–232. (Section 7 gives the exact moments of r.)

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge.

Olkin, I., & Finn, J. D. (1995). Correlations redux. *Psychological Bulletin, 118*(1), 155–164. doi:10.1037/00332909.118.1.155

See Also

`expected_r`, `ci_r`, `var_partial_r`, `var_semipartial_r`

Other variance utilities: `var_alpha()`, `var_cv()`, `var_ete()`, `var_indirect_effect()`, `var_omega_squared()`, `var_smd()`, `var_smd_trimmed()`

Examples

```
# 1. Normal-theory variance:
var_r(true_cor = 0.30, N = 50)
```

`var_R2` *Variance of the Squared Multiple Correlation Coefficient*

Description

Computes the sampling variance of the squared multiple correlation coefficient from the population value, the sample size, and the number of predictors, the quantity that governs how precisely R^2 is estimated at a given design size.

Usage

```
var_R2(true_R2, N, p)
```

Arguments

`true_R2`	Population squared multiple correlation coefficient, ρ^2, the value at which the variance is evaluated
`N`	Sample size
`p`	The number of predictor variables

Details

Uses the hypergeometric function as discussed in section 28 of Stuart, Ord, and Arnold (1999) in order to obtain the *correct* value for the variance of the squared multiple correlation coefficient.

Value

A 1-row `data.frame` with columns `term` and `value`. The `term` value is `"var_R2"` and `value` is the exact variance of R^2 under random sampling from a multivariate normal population.

Note

The Gauss hypergeometric function ${}_2F_1$ is computed in base R (see the internal `.hyperg_2F1`); no GSL system library is required.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Kelley, K. (2008). Sample size planning for the squared multiple correlation coefficient: Accuracy in parameter estimation via narrow confidence intervals. *Multivariate Behavioral Research, 43*, 524–555. doi:10.1080/00273170802490632

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 3 on R^2 as a model comparison effect size.)

Stuart, A., Ord, J. K., & Arnold, S. (1999). *Kendall's advanced theory of statistics, volume 2A: Classical inference and the linear model* (6th ed.). Arnold.

See Also

`expected_R2`, `ci_R2`, `ss_aipe_R2`

Examples

```
var_R2(.5, 10, 5)
var_R2(.5, 25, 5)
var_R2(.5, 50, 5)
var_R2(.5, 100, 5)
```

---

var_semipartial_r *Asymptotic Variance of the Semipartial (Part) Correlation Coefficient*

Description

Computes a variance for the sample semipartial (also known as the part) correlation coefficient $r_{Y(X \cdot Z_1 \cdots Z_{p-1})}$ under multivariate normality. The function provides two quantities, both of which need the full-model coefficient of multiple determination $R^2_{Y \cdot X Z_1 \cdots Z_{p-1}}$: the squared standard error in the *t* test of X's coefficient (the default when `R2_full` is supplied; Cohen, Cohen, West, & Aiken, 2003), and the first-order normal-theory variance of the sample semipartial correlation (`under_null = FALSE`), which is the quantity for a Wald confidence interval and for sample size planning. Without `R2_full` the function returns the first-order variance for the special case in which the variables partialed out explain none of the outcome, and warns.

Usage

```
var_semipartial_r(r_sp, N, p = 2, R2_full = NULL, under_null = TRUE)
```

Arguments

`r_sp`	Sample semipartial correlation coefficient $r_{Y(X \cdot Z_1 \cdots Z_{p-1})}$, with the other variables partialed out of X only (not Y). Must be in $[-1, 1]$.
`N`	Total sample size.

p — Number of predictors in the regression of Y on X and the variables partialed out, so `p - 1` variables are partialed out and `p` is at least 2. A categorical variable with a levels counts as a - 1 predictors. It is the `p` of `ss_power_R2`, `ci_R2`, and `ci_reg_coef`: the correlation concerns one predictor of a regression with `p` predictors. In the notation of a model comparison, q is the number of predictors removed from the full model to give the restricted model ($q = p_{full} - p_{restricted}$, the predictors being tested), and the number of variables partialed out is $p - q$; this correlation concerns one predictor, so $q = 1$ and $p - 1$ variables are partialed out. Defaults to 2 (one variable partialed out).

R2_full — Coefficient of multiple determination $R^2_{Y \cdot X Z_1 \cdots Z_{p-1}}$ from the full model that includes X and every variable partialed out (`R2_Y_X` in `ci_reg_coef`; the squared semipartial correlation is `R2_full` minus the squared multiple correlation of the model without X). Must be in $[0, 1]$ and cannot be smaller than `r_sp` squared, because the squared semipartial correlation is the part of the full-model R^2 that X adds; a smaller value is refused with an error of class `dmar_semipartial_r_incoherent`, unless the two values could have come from rounding to two decimals (see Details). When `NULL` (the default), the value returned is the first-order variance that holds when the variables partialed out explain none of the outcome, with a warning of class `dmar_semipartial_r_no_R2_full`.

under_null — Logical, used only when `R2_full` is supplied. If `TRUE` (the default), the function returns $(1 - R^2)/(N - p - 1)$, the squared standard error in the t test of X's coefficient. If `FALSE`, it returns the first-order normal-theory variance of the sample semipartial correlation, evaluated at `r_sp` and `R2_full`.

Details

Background. The squared semipartial $r^2_{Y(X \cdot Z)}$ equals the increase in R^2 when X is added to a model already containing the variables partialed out, $Z_1, \ldots, Z_{p-1}$, that is, the unique variance in Y attributable to X. Unlike the partial, the semipartial is on the original scale of Y rather than on the partialed scale, which makes it the natural effect size companion to standardized regression coefficients in multiple regression reports (Cohen et al., 2003). Aloe and Becker (2012) propose it as an effect size for regression predictors in meta-analysis. `ss_power_R2_change` plans the power of the test of that change in R^2.

First-order variance (`under_null = FALSE`). Write ρ_{sp} for the population semipartial correlation, P^2 for the population full-model squared multiple correlation, and $P_r^2 = P^2 - \rho_{sp}^2$, the squared multiple correlation of Y with $Z_1, \ldots, Z_{p-1}$ alone. Under multivariate normality, to first order in $1/N$,

$$N \operatorname{Var}(\hat{r}_{sp}) \;\to\; (1 - \rho_{sp}^2)^2 \;-\; P_r^2 \,(1 - 2\rho_{sp}^2).$$

The variance depends on the population through ρ_{sp} and P_r^2 only. Three cases show how it behaves. When the variables partialed out explain none of the outcome ($P_r^2 = 0$) it is $(1 - \rho_{sp}^2)^2$, the form of a simple or partial correlation. When $\rho_{sp} = 0$ it is $1 - P^2$. For $P_r^2 > 0$, it is smaller than $(1 - \rho_{sp}^2)^2$ when $\rho_{sp}^2 < 1/2$ and larger when $\rho_{sp}^2 > 1/2$. Because the variance depends on P_r^2, no function of r_{sp} alone stabilizes it, and Fisher's Z transformation does not apply to a semipartial correlation as it does to a partial correlation. The expression is the first-order variance of Aloe and Becker (2012, Equation 4), which they obtain from Alf and Graf (1999), with $N - p$ in place of their n: their numerator, $P^4 - 2P^2 + P_{(f)}^2 + 1 - P_{(f)}^4$ with $P_{(f)}^2$ the squared multiple correlation of the model without X, equals the expression above with $P_r^2 = P_{(f)}^2$. The result follows from the delta method

variance of the difference between two nested squared multiple correlations (Model A of Olkin & Finn, 1995, for one variable partialed out; Alf & Graf, 1999, for any number), because r_{sp}^2 is that difference and so $\mathrm{Var}(\hat{r}_{sp}) \approx \mathrm{Var}(\widehat{\Delta R^2})/(4\rho_{sp}^2)$.

The function evaluates the expression at `r_sp` and `R2_full` and divides by $N - p$. The number of variables partialed out does not enter the first-order variance. The divisor $N - p$ is a convention, kept for agreement with `var_partial_r` and because, with estimated values, a Wald interval formed with it has coverage closer to the nominal level than one formed with N; it is not the most accurate estimate of the variance itself (in simulation neither $N - 1$ nor $N - p$ was closer to the observed variance in every condition).

The approximation has limits that simulation shows and that no theory given here corrects: it understates the variance when $|\rho_{sp}|$ is large and $N - p$ is small (in one simulated condition, $\rho_{sp} = .95$, $P_r^2 = 0$, $p = 2$, and $N = 40$, it was about 39 percent of the variance observed), and the sample semipartial correlation is biased toward zero by roughly $\rho_{sp}(p - 1)/(2N)$. A Wald interval formed from it also misses the population value too often when the full-model R^2 is near 1 and N is small, for any ρ_{sp} (in one simulated condition, $\rho_{sp} = .1$, a full-model R^2 of .99, $p = 4$, and $N = 12$, it missed in .13 of samples). When $|\rho_{sp}|$ is above about .9 and the variables partialed out explain some of the outcome, the interval misses more often on one side than the other even with several hundred observations (in one simulated condition, $\rho_{sp} = .95$, a full-model R^2 of .97, $p = 2$, and $N = 395$, it missed in .064 of samples, .051 of them on one side).

Values rounded to two decimals. Reported values are often rounded, and a rounded pair can have `R2_full` slightly below `r_sp` squared: a semipartial correlation reported as .46 has a square of .2116, and the full-model R^2 beside it may be reported as .21. When `R2_full + .005` is at least $(|r_{sp}| - .005)^2$ (with the difference in parentheses not below 0), the pair is consistent with rounding. The function then gives a warning of class `dmar_semipartial_r_rounded`, which carries the two values, and proceeds: the squared semipartial correlation cannot exceed the full-model R^2, so the variables partialed out are taken to explain none of the outcome ($P_r^2 = 0$) in the first-order variance, and the squared standard error of the *t* test uses `R2_full` as given. A pair further apart than rounding allows is refused. `ss_aipe_semipartial_r` and `ss_aipe_semipartial_r_sensitivity` apply the same rule to their values.

Without `R2_full`. The function returns $(1 - r_{sp}^2)^2/(N - p)$, the first-order variance at $P_r^2 = 0$. When the variables partialed out explain some of the outcome, this value is too large if $\rho_{sp}^2 < 1/2$ and too small if $\rho_{sp}^2 > 1/2$, which is why the call warns. At $r_{sp} = 0$ it is $1/(N - p)$ whatever the full-model R^2, where the first-order variance is $(1 - P^2)/N$.

Squared standard error of the *t* test (`R2_full` supplied, `under_null = TRUE`). In multiple regression the unique contribution of X is tested with

$$F \;=\; \frac{r_{Y(X\cdot Z)}^2\,(N - p - 1)}{1 - R_{Y\cdot XZ}^2} \;\sim\; F(1,\, N - p - 1)$$

under $H_0 : \rho_{Y(X\cdot Z)} = 0$ (Cohen et al., 2003). Equivalently $t = \hat{r}_{sp}\sqrt{(N - p - 1)/(1 - R_{Y\cdot XZ}^2)}$ is a *t*-statistic on $N - p - 1$ degrees of freedom, and the function returns the square of its denominator,

$$\frac{1 - R_{Y\cdot XZ}^2}{N - p - 1},$$

the squared standard error in the *t* test of X's coefficient. It uses the sample R^2 and equals the first-order variance at $\rho_{sp} = 0$; it is the ingredient for testing the significance of X's unique contribution, not a variance for an interval about a nonzero r_{sp}.

With `under_null = TRUE` the value is $(1-R^2_{full})/(N-p-1)$, which equals r^2_{sp}/t^2 for the *t* statistic of the predictor's coefficient, by the identity $r_{sp} = t\sqrt{(1-R^2_{full})/(N-p-1)}$ (Aloe & Becker, 2012, Equation 2).

Value

A one-row `data.frame` with columns `term` (either `"var_semipartial_r"` or `"var_semipartial_r_under_null"`) and `value` (the requested variance).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Alf, E. F., Jr., & Graf, R. G. (1999). Asymptotic confidence limits for the difference between two squared multiple correlations: A simplified approach. *Psychological Methods, 4*(1), 70–75. doi:10.1037/1082989X.4.1.70

Aloe, A. M., & Becker, B. J. (2012). An effect size for regression predictors in meta-analysis. *Journal of Educational and Behavioral Statistics, 37*(2), 278–297. doi:10.3102/1076998610396901

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Lawrence Erlbaum.

Kelley, K., & Maxwell, S. E. (2003). Sample size for multiple regression: Obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods, 8*(3), 305–321. doi:10.1037/1082989X.8.3.305

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on contrasts, Chapter 5 on multiple comparisons, and Chapter 9 on ANCOVA.)

Olkin, I., & Finn, J. D. (1995). Correlations redux. *Psychological Bulletin, 118*(1), 155–164. doi:10.1037/00332909.118.1.155

See Also

`var_partial_r`, `var_R2`, `ci_reg_coef`, `ss_aipe_semipartial_r`, `ss_power_R2_change`

Examples

```
# First-order variance of a semipartial r = .25 with N = 100, three
#     variables partialed out of X (p = 4 predictors in the full model),
#     and a full-model R^2 of .42.
var_semipartial_r(r_sp = 0.25, N = 100, p = 4, R2_full = 0.42,
                  under_null = FALSE)

# By default, with R^2 of the full model supplied, the function returns
#     the squared standard error in the t test of X's unique
#     contribution.
var_semipartial_r(r_sp = 0.25, N = 100, p = 4, R2_full = 0.42)
```

```
# A reported pair, r_sp = .46 with a full-model R^2 of .21, in which
#     rounding has left R^2 below .46^2 = .2116: the value is returned
#     with a warning that says how the pair was read. The handler keeps
#     the message and lets the call finish, so both are shown.
noted <- new.env()
rounded <- withCallingHandlers(
  var_semipartial_r(r_sp = 0.46, N = 100, p = 2, R2_full = 0.21),
  dmar_semipartial_r_rounded = function(w) {
    noted$message <- conditionMessage(w)
    invokeRestart("muffleWarning")
  })
rounded
noted$message
```

`var_smd` *Variance of the Standardized Mean Difference (Cohen's* d*) and Hedges'* g

---

Description

Computes the variance of the sample standardized mean difference (Cohen's d) of two independent groups whose scores are normally distributed with a common variance, using the *exact* noncentral t sampling distribution (Hedges, 1981) as the default and reporting the large-sample Hedges-Olkin (1985) approximation alongside for comparison. Optionally returns the variance of Hedges' g, the bias-corrected counterpart of d.

Usage

```
var_smd(true_smd, n_1, n_2 = NULL, unbiased = FALSE)
```

Arguments

`true_smd`	Population standardized mean difference, δ in the formulas, at which the variance is evaluated. Numeric scalar or vector.
`n_1`	Sample size in group 1. Scalar or vector.
`n_2`	Sample size in group 2. Scalar or vector. Defaults to `n_1` (balanced design).
`unbiased`	Logical. If `TRUE`, returns the variance of Hedges' g (the bias-corrected estimator); if `FALSE` (the default), returns the variance of Cohen's d.

Details

The exact variance and the large-sample approximation differ in finite samples, and the approximation is the smaller of the two. With `true_smd` (δ) equal to 0 the approximation is the exact variance times $(df-2)/df$, where $df = n_1 + n_2 - 2$: .75 with 5 per group and .97 with 30 per group. The ratio falls as $|\delta|$ grows (.72 and .96 at $|\delta| = 1$). The variance of Hedges' g is the variance of d times $J(df)^2$, the square of the bias-correction factor, which is below 1 (.81 with 5 per group, .96 with

20 per group) and approaches 1 as the degrees of freedom grow, so the variances of d and g are different quantities in small samples. Which variance is used matters when it enters meta-analytic weighting, AIPE planning, or a Wald-style standard error.

Exact noncentral *t* form. For $\hat{d} = (\bar{Y}_1 - \bar{Y}_2)/s_p$ with pooled s_p, the rescaled statistic $t = \hat{d}\sqrt{n_1 n_2/(n_1+n_2)}$ follows a noncentral t with $df = n_1 + n_2 - 2$ degrees of freedom and noncentrality parameter $\lambda = \delta\sqrt{n_1 n_2/(n_1+n_2)}$. The variance of a noncentral t is (Johnson, Kotz, & Balakrishnan, 1995, Sec.\ 31.3)

$$\mathrm{Var}(t) \;=\; \frac{df\,(1+\lambda^2)}{df-2} \;-\; \lambda^2\,c(df)^2,$$

where $c(df) = \sqrt{df/2}\,\Gamma((df-1)/2)\,/\,\Gamma(df/2)$; dividing by the design factor $n_1 n_2/(n_1+n_2)$ returns $\mathrm{Var}(\hat{d})$. For Hedges' g, multiply the result by $J(df)^2$ where $J(df) = 1/c(df)$ is the Hedges-Olkin (1985) bias-correction factor (see `expected_smd`).

Hedges-Olkin large-sample approximation. The large-sample approximation (Hedges & Olkin, 1985, equation 8) is

$$\mathrm{Var}(\hat{d}) \;\approx\; \frac{n_1+n_2}{n_1 n_2} \;+\; \frac{\delta^2}{2(n_1+n_2-2)}.$$

This approaches the exact form only as the degrees of freedom grow: even at $\delta = 0$ it returns $1/(n_1 n_2/(n_1+n_2))$ while the exact noncentral t variance is $[df/(df-2)]/(n_1 n_2/(n_1+n_2))$, so the approximation is biased downward by a factor of $(df-2)/df$, and the downward bias grows as $|\delta|$ grows and as n shrinks.

Companions. `var_smd()` is the variance partner of `expected_smd` (mean) and `ci_smd` (CI). For design-stage AIPE planning that solves for n given a target CI width on d, see `ss_aipe_smd`.

Value

A `data.frame` with rows for the exact (noncentral- t) variance and the Hedges-Olkin large-sample approximation. Columns are `term` (`"var_smd_exact"` or `"var_smd_approx"`) and `value`.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics, 6*(2), 107–128.

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. Academic Press.

Johnson, N. L., Kotz, S., & Balakrishnan, N. (1995). *Continuous univariate distributions, volume 2* (2nd ed.). Wiley.

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

See Also

smd, ci_smd, expected_smd, ss_aipe_smd

Other variance utilities: var_alpha(), var_cv(), var_ete(), var_indirect_effect(), var_omega_squared(), var_r(), var_smd_trimmed()

Examples

```
# 1. Balanced design, true_smd = 0.5, n = 20 per group.
var_smd(true_smd = 0.5, n_1 = 20)

# 2. Hedges-Olkin approximation drifts from exact when n is small.
var_smd(true_smd = 0.5, n_1 = 5)
var_smd(true_smd = 0.5, n_1 = 50)

# 3. Variance of Hedges' g (bias-corrected).
var_smd(true_smd = 0.5, n_1 = 20, unbiased = TRUE)
```

---

var_smd_trimmed *Asymptotic Variance of the Robust Trimmed SMD*

Description

Computes the asymptotic variance of the Algina-Keselman-Penfield (2005) robust standardized mean difference under the Yuen (1974) trimmed-mean framework, suitable for AIPE sample size planning for robust effect sizes (Keselman, Algina, Lix, Wilcox, & Deering, 2008).

Usage

```
var_smd_trimmed(true_smd_trimmed, n_1, n_2, trim = 0.2)
```

Arguments

true_smd_trimmed	Anticipated population value of the robust trimmed SMD δ_R.
n_1, n_2	Per-group sample sizes.
trim	Proportion to trim and Winsorize. Default 0.20.

Details

Variance formula. Under random sampling with trimming proportion γ from each tail, the variance of the trimmed-mean difference scales by $1/h_j$ (where $h_j = n_j - 2\lfloor \gamma n_j \rfloor$ is the number of retained observations in group j) rather than $1/n_j$. The large-sample variance of the standardized version, written on the d_R scale, is

$$\mathrm{Var}(\hat{d}_R) \approx \frac{h_1 + h_2}{h_1 h_2} + \frac{\delta_R^2}{2(h_1 + h_2)}.$$

For $\gamma = 0$ this reduces to the standard Hedges-Olkin (1985) variance of Cohen's d.

When to use. For AIPE planning of a robust effect size study, use `var_smd_trimmed()` in place of `var_smd()`. Pair with `smd_trimmed()` for the point estimate and noncentral t CI.

Value

A 1-row `data.frame` with columns `term` (`"var_smd_trimmed"`) and `value` (the variance).

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Algina, J., Keselman, H. J., & Penfield, R. D. (2005). An alternative to Cohen's standardized mean difference effect size: A robust parameter and confidence interval in the two independent groups case. *Psychological Methods, 10*(3), 317–328. doi:10.1037/1082989X.10.3.317

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. Academic Press.

Kelley, K., & Rausch, J. R. (2006). Sample size planning for the standardized mean difference: Accuracy in parameter estimation via narrow confidence intervals. *Psychological Methods, 11*(4), 363–385. doi:10.1037/1082989X.11.4.363

Keselman, H. J., Algina, J., Lix, L. M., Wilcox, R. R., & Deering, K. N. (2008). A generally robust approach for testing hypotheses and setting confidence intervals for effect sizes. *Psychological Methods, 13*(2), 110–129. doi:10.1037/1082989X.13.2.110

Maxwell, S. E., Delaney, H. D., & Kelley, K. (2027). *Designing experiments and analyzing data: A model comparison perspective* (4th ed.). Routledge. (See Chapter 4 on individual comparisons and Chapter 3 on one-way ANOVA.)

Yuen, K. K. (1974). The two-sample trimmed t for unequal population variances. *Biometrika, 61*(1), 165–170.

See Also

`smd_trimmed`, `var_smd`, `ss_aipe_smd`

Other variance utilities: `var_alpha()`, `var_cv()`, `var_ete()`, `var_indirect_effect()`, `var_omega_squared()`, `var_r()`, `var_smd()`

Examples

```
# 1. Population delta_R = 0.5, n = 30 per group, 20% trim:
var_smd_trimmed(true_smd_trimmed = 0.5, n_1 = 30, n_2 = 30)

# 2. Variance scales by h / n via the trimming proportion:
var_smd_trimmed(0.5, 30, 30, trim = 0.00)$value
var_smd_trimmed(0.5, 30, 30, trim = 0.20)$value
```

`welch_t` *Welch's Separate-Variance* t *Test*

Description

Computes Welch's (1947) separate-variance *t* test, the Satterthwaite (1946) approximation for unequal-variance two-sample inference, and returns the test statistic, Satterthwaite degrees of freedom, *p*-value, point estimate of the mean difference, and a confidence interval on the mean difference, all in a tidy `data.frame`. Unlike Student's pooled-variance *t* test (`stats::t.test(..., var.equal = TRUE)`), Welch's test does *not* assume the two populations have equal variances, and should be the default choice in applied work (Delacre, Lakens, & Leys, 2017).

Usage

```
welch_t(
  x,
  y,
  null_mean_diff = 0,
  alternative = c("two_sided", "less", "greater"),
  conf_level = 0.95
)
```

Arguments

`x, y`	Numeric vectors of observations from the two groups. The two groups are independent and need not be the same length; `NA`s are removed from each vector separately.
`null_mean_diff`	Null value of the mean difference $\mu_1 - \mu_2$, μ_0 in the formula below. Default `0`.
`alternative`	One of `"two_sided"` (default; the base-R spelling `"two.sided"` is accepted as an alias), `"less"`, or `"greater"`, defining the direction of the alternative hypothesis.
`conf_level`	Confidence level for the CI on the mean difference. Default `0.95`.

Details

Test statistic. Welch's *t* is

$$t \;=\; \frac{\bar{x} - \bar{y} - \mu_0}{\sqrt{s_1^2/n_1 + s_2^2/n_2}},$$

which is referred to a *t* distribution on the Satterthwaite (1946) approximate degrees of freedom

$$df \;=\; \frac{(s_1^2/n_1 + s_2^2/n_2)^2}{(s_1^2/n_1)^2/(n_1 - 1) + (s_2^2/n_2)^2/(n_2 - 1)}.$$

Why Welch by default. Student's pooled-variance *t* assumes $\sigma_1 = \sigma_2$; when that assumption fails it has both inflated and deflated Type I error rates depending on the $n_1 : n_2$ ratio (Ruxton, 2006). When the scores in each group are normally distributed, Welch's test keeps the Type I error rate

close to the nominal level across combinations of σ_1/σ_2 and n_1/n_2, and when the variances are equal its power is close to that of Student's test, a little lower when the group sizes differ. With skewed scores and small groups its Type I error rate can depart from the nominal level. Delacre, Lakens, and Leys (2017) and Lakens (2015) recommend Welch's test as the default.

Relation to `stats::t.test()`. The numerical results here match `stats::t.test(x, y, var.equal = FALSE)` to machine precision; this function differs only in returning a tidy `data.frame` that composes with the rest of DMAR.

Value

A `data.frame` with rows for the mean difference $\bar{x} - \bar{y}$, the Welch *t*-statistic, Satterthwaite degrees of freedom, *p*-value, the CI lower and upper limits on the mean difference, and the per-group means, SDs, and *n*.

Author(s)

Ken Kelley <kkelley@nd.edu>

References

Delacre, M., Lakens, D., & Leys, C. (2017). Why psychologists should by default use Welch's *t*-test instead of Student's *t*-test. *International Review of Social Psychology, 30*(1), 92–101. doi:10.5334/irsp.82

Lakens, D. (2015, January). Always use Welch's t-test instead of Student's t-test [Blog post]. The 20% Statistician. https://daniellakens.blogspot.com/2015/01/always-use-welchs-t-test-instead-of.html

Ruxton, G. D. (2006). The unequal variance *t*-test is an underused alternative to Student's *t*-test and the Mann-Whitney *U* test. *Behavioral Ecology, 17*(4), 688–690. doi:10.1093/beheco/ark016

Satterthwaite, F. E. (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin, 2*(6), 110–114.

Welch, B. L. (1947). The generalization of "Student's" problem when several different population variances are involved. *Biometrika, 34*(1/2), 28–35.

See Also

`t.test`, `summary_t_test`, `smd`, `ci_smd`

Other hypothesis tests: `adjusted_means()`, `ancova()`, `anova_within()`, `ci_dunnett()`, `ci_dunnett_t3()`, `ci_games_howell()`, `ci_scheffe()`, `ci_tukey_kramer()`, `compare_cov_structures()`, `contrast_test()`, `correlations_test()`, `equivalence_proportion_paired()`, `equivalence_r()`, `equivalence_rmsea()`, `equivalence_smd()`, `factorial_anova()`, `manova_split_plot()`, `mauchly_test()`, `mixed_anova()`, `obrien_test()`, `pairwise_within()`, `randomization_test()`, `randomization_test_paired()`, `regions_of_significance()`, `simple_effects_AB()`, `summary_t_test()`

Examples

```
# 1. Two groups with different variances:
set.seed(113)
x <- rnorm(20, mean = 100, sd = 15)
```

```
y <- rnorm(20, mean = 110, sd = 25)
welch_t(x, y)

# 2. One-sided test:
welch_t(x, y, alternative = "less")

# 3. Side-by-side comparison with base R's stats::t.test().
# The two functions implement the same Welch / Satterthwaite test, so
# the t-statistic, Satterthwaite degrees of freedom, p-value, and the
# CI on the mean difference match exactly. welch_t() differs only in
# what it returns: a data.frame(term, value) rather than a list-
# like htest object. The return composes with dplyr / ggplot2
# pipelines and avoids stringly-typed access like $statistic.
set.seed(113)
a <- rnorm(15, mean = 0,   sd = 1)
b <- rnorm(20, mean = 0.5, sd = 2)

# DMAR (data.frame):
dmar_res <- welch_t(a, b, conf_level = 0.95)
dmar_res

# Base R (htest list):
base_res <- stats::t.test(a, b, var.equal = FALSE, conf.level = 0.95)
base_res

# Verify the four key statistics agree numerically:
pick <- function(term) dmar_res$value[dmar_res$term == term]
stopifnot(
  all.equal(pick("t_value"), unname(base_res$statistic)),
  all.equal(pick("df"),          unname(base_res$parameter)),
  all.equal(pick("p_value"),     base_res$p.value),
  all.equal(pick("ci_lower"), base_res$conf.int[1]),
  all.equal(pick("ci_upper"), base_res$conf.int[2])
)
```

`welch_t_broom` *Broom-Style Tidy / Glance Methods for* `welch_t()`

Description

`tidy()` returns the single mean-difference estimate and its confidence interval in the **broom** convention; `glance()` coincides with it, since a two-sample *t* test reports one estimand and there are no extra model-level statistics to add.

Usage

```
## S3 method for class 'dmar_welch_t'
tidy(x, ...)
```

```
## S3 method for class 'dmar_welch_t'
glance(x, ...)
```

Arguments

x	A dmar_welch_t object returned by welch_t.
...	Unused.

Value

A one-row data.frame with columns term, estimate, ci_lower, ci_upper, statistic, df, p_value, and conf_level.

Author(s)

Ken Kelley <kkelley@nd.edu>

Index

∗ **AIPE sample size planning**
ss_aipe_c_sensitivity, 859
ss_aipe_cliff_delta, 809
ss_aipe_cliff_delta_sensitivity, 811
ss_aipe_composite_sem, 814
ss_aipe_crossed, 837
ss_aipe_crossed_sensitivity, 842
ss_aipe_equivalence_r, 863
ss_aipe_equivalence_r_sensitivity, 865
ss_aipe_equivalence_smd, 868
ss_aipe_equivalence_smd_sensitivity, 872
ss_aipe_icc, 875
ss_aipe_icc_sensitivity, 878
ss_aipe_indirect_effect, 882
ss_aipe_indirect_effect_sensitivity, 887
ss_aipe_mean, 890
ss_aipe_mean_sensitivity, 894
ss_aipe_mixed_effects_sensitivity, 902
ss_aipe_omega_squared, 906
ss_aipe_omega_squared_sensitivity, 908
ss_aipe_partial_r, 911
ss_aipe_partial_r_sensitivity, 916
ss_aipe_pcm_sensitivity, 923
ss_aipe_r, 926
ss_aipe_r_sensitivity, 979
ss_aipe_reliability_sensitivity, 967
ss_aipe_semipartial_r, 998
ss_aipe_semipartial_r_sensitivity, 1003
ss_aipe_two_phase, 1036
ss_aipe_two_phase_cost, 1039
ss_aipe_two_phase_sensitivity, 1043
∗ **Bayesian t analyses**
bayes_independent_t, 33
bayes_one_sample_t, 36
bayes_paired_t, 38
∗ **Cohen's U3**
proportion_of_superiority, 635
∗ **Cohen's d**
ci_smd, 165
smd, 799
∗ **Cronbach's alpha**
reliability_alpha, 662
∗ **Glass's delta**
ci_smd_c, 168
∗ **Hedges' g**
ci_smd, 165
smd, 799
∗ **Vargha-Delaney A**
vargha_delaney_A, 1177
∗ **agreement and measurement**
content_validity_index, 216
gwet_ac, 442
icc_lmer, 453
krippendorff_alpha, 465
limits_of_agreement, 470
lin_ccc, 473
R2_mixed_effects, 639
variance_components_mls, 1184
∗ **composite power**
ss_power_composite_ancova, 1049
ss_power_composite_ancova_2group, 1053
ss_power_composite_anova, 1060
ss_power_composite_factorial_ancova, 1063
ss_power_composite_factorial_ancova_het, 1067
ss_power_composite_factorial_anova, 1072

ss_power_composite_sem, 1075
* **confidence intervals for effect sizes**
ci_c, 71
ci_c_ancova, 83
ci_c_ancova_bp, 86
ci_correlation, 74
ci_cv, 80
ci_eta_squared, 95
ci_eta_squared_generalized, 101
ci_mahalanobis, 110
ci_omega_squared, 121
ci_pvaf, 129
ci_R2, 132
ci_ratio_of_means, 135
ci_ratio_of_means_paired, 141
ci_rc, 147
ci_reg_coef, 150
ci_rmsea, 153
ci_sc, 156
ci_sc_ancova, 160
ci_sm, 163
ci_smd, 165
ci_smd_c, 168
ci_snr, 170
ci_src, 172
ci_srsnr, 176
contrast_adjusted, 219
plot_smd, 599
* **confidence intervals**
ci_crossed, 78
ci_proportion, 125
ci_proportion_paired, 126
ci_two_phase, 181
* **correlation**
expected_partial_r, 350
expected_r, 352
* **critical values**
cv_bonferroni_f, 254
cv_bryant_paulson, 256
cv_chisq, 259
cv_dunnett, 261
cv_f, 263
cv_scheffe, 265
cv_smm, 266
cv_t, 269
cv_tukey_hsd, 270
cv_z, 273
* **data simulators**
simulate_ancova_data, 767
simulate_ancova_factorial_data, 769
simulate_anova_data, 773
simulate_longitudinal_gompertz, 774
simulate_longitudinal_logistic, 778
simulate_longitudinal_negative_exponential, 782
simulate_longitudinal_polynomial, 785
simulate_longitudinal_richards, 792
simulate_regression_data, 796
* **datagen**
simulate_ancova_data, 767
simulate_ancova_factorial_data, 769
simulate_anova_data, 773
simulate_longitudinal_gompertz, 774
simulate_longitudinal_logistic, 778
simulate_longitudinal_negative_exponential, 782
simulate_longitudinal_polynomial, 785
simulate_longitudinal_richards, 792
simulate_regression_data, 796
* **datasets**
bessel_errors, 41
depression_bdi, 274
diagnosis_agreement, 290
drinks_trial, 314
holzinger_swineford, 445
prime_time_achievement, 619
pygmalion, 637
teacher_expectancy, 1155
test_market, 1157
* **descriptive statistics**
descriptives, 276
kurtosis, 468
skewness, 798
* **design utilities**
design_consequences, 278
design_effect, 281

effects_coding, 322
helmert_coding, 444
is_orthogonal_set, 464
orthogonal_polynomial, 571
∗ **design**
adjusted_means, 14
ancova, 20
anova_within, 25
anova_within_two_way, 26
bryant_paulson, 49
ci_c, 71
ci_c_ancova, 83
ci_c_ancova_bp, 86
ci_correlation, 74
ci_dunnett_t3, 91
ci_eta_squared, 95
ci_eta_squared_generalized, 101
ci_games_howell, 107
ci_nc_chisq, 113
ci_nc_F, 115
ci_omega_squared, 121
ci_pvaf, 129
ci_rc, 147
ci_rmsea, 153
ci_sc, 156
ci_sc_ancova, 160
ci_sm, 163
ci_src, 172
ci_srsnr, 176
ci_two_phase, 181
compare_cov_structures, 210
contrast_adjusted, 219
contrast_test, 221
convert_cor_cov, 225
convert_d_or, 226
convert_d_r, 227
convert_R2, 231
convert_r_Z, 233
convert_t_smd, 234
convert_Z_r, 236
correction_for_attenuation, 237
cv, 252
cv_bonferroni_f, 254
cv_bryant_paulson, 256
cv_dunnett, 261
cv_scheffe, 265
cv_smm, 266
cv_tukey_hsd, 270
design_consequences, 278
design_effect, 281
effects_coding, 322
eta_squared, 341
eta_squared_generalized, 345
factorial_anova, 360
helmert_coding, 444
is_orthogonal_set, 464
manova_split_plot, 474
mixed_anova, 525
mr_cv, 549
mr_smd, 554
omega_squared, 567
orthogonal_polynomial, 571
pairwise_within, 574
power_density_equivalence_md, 606
power_equivalence_c, 608
power_equivalence_md, 611
power_equivalence_md_plot, 614
responder_analysis, 723
sd_unbiased, 727
simple_effects_AB, 761
simulate_ancova_data, 767
simulate_ancova_factorial_data, 769
simulate_anova_data, 773
simulate_longitudinal_polynomial, 785
simulate_regression_data, 796
ss_aipe_c, 806
ss_aipe_c_ancova, 854
ss_aipe_c_ancova_sensitivity, 856
ss_aipe_c_sensitivity, 859
ss_aipe_cliff_delta, 809
ss_aipe_cliff_delta_sensitivity, 811
ss_aipe_composite_sem, 814
ss_aipe_crossed, 837
ss_aipe_crossed_sensitivity, 842
ss_aipe_cv, 846
ss_aipe_cv_sensitivity, 850
ss_aipe_equivalence_r, 863
ss_aipe_equivalence_r_sensitivity, 865
ss_aipe_equivalence_smd, 868
ss_aipe_equivalence_smd_sensitivity, 872
ss_aipe_icc, 875

ss_aipe_icc_sensitivity, 878
ss_aipe_indirect_effect, 882
ss_aipe_indirect_effect_sensitivity, 887
ss_aipe_mean, 890
ss_aipe_mean_sensitivity, 894
ss_aipe_mixed_effects, 898
ss_aipe_mixed_effects_sensitivity, 902
ss_aipe_omega_squared, 906
ss_aipe_omega_squared_sensitivity, 908
ss_aipe_partial_r, 911
ss_aipe_partial_r_sensitivity, 916
ss_aipe_pcm, 920
ss_aipe_pcm_sensitivity, 923
ss_aipe_r, 926
ss_aipe_R2, 929
ss_aipe_R2_sensitivity, 933
ss_aipe_r_sensitivity, 979
ss_aipe_rc, 937
ss_aipe_rc_sensitivity, 940
ss_aipe_reg_coef, 944
ss_aipe_reg_coef_sensitivity, 952
ss_aipe_reliability, 956
ss_aipe_reliability_sensitivity, 967
ss_aipe_rmsea, 974
ss_aipe_sc, 982
ss_aipe_sc_ancova, 986
ss_aipe_sc_ancova_sensitivity, 991
ss_aipe_sc_sensitivity, 994
ss_aipe_sem_path, 1007
ss_aipe_sem_path_sensitivity, 1009
ss_aipe_semipartial_r, 998
ss_aipe_semipartial_r_sensitivity, 1003
ss_aipe_sm, 1014
ss_aipe_sm_sensitivity, 1023
ss_aipe_smd, 1016
ss_aipe_smd_sensitivity, 1020
ss_aipe_src, 1027
ss_aipe_src_sensitivity, 1032
ss_aipe_two_phase, 1036
ss_aipe_two_phase_cost, 1039
ss_aipe_two_phase_sensitivity, 1043
ss_power_c, 1047
ss_power_c_ancova, 1083
ss_power_composite_ancova, 1049
ss_power_composite_ancova_2group, 1053
ss_power_composite_anova, 1060
ss_power_composite_factorial_ancova, 1063
ss_power_composite_factorial_ancova_het, 1067
ss_power_composite_factorial_anova, 1072
ss_power_composite_sem, 1075
ss_power_contrast, 1079
ss_power_equivalence_c, 1085
ss_power_factorial_ancova, 1088
ss_power_factorial_anova, 1090
ss_power_indirect_effect, 1093
ss_power_mixed_effects, 1096
ss_power_one_way_anova, 1098
ss_power_pcm, 1100
ss_power_r, 1104
ss_power_R2, 1107
ss_power_R2_change, 1111
ss_power_R2_sensitivity, 1117
ss_power_rc, 1120
ss_power_reg_coef, 1124
ss_power_reg_coef_sensitivity, 1129
ss_power_rm_anova, 1133
ss_power_sc, 1135
ss_power_sem, 1137
ss_power_smd, 1139
ss_power_split_plot_anova, 1141
ss_seq_c, 1144
ss_seq_c_sensitivity, 1147
ss_seq_two_phase, 1150
var_icc, 1194
var_partial_r, 1201
var_R2, 1205
var_semipartial_r, 1206

∗ **distribution**

bryant_paulson, 49
convert_F_chisq, 229
cv_bonferroni_f, 254
cv_chisq, 259
cv_f, 263
moments_nc_chisq, 543
moments_nc_F, 545

moments_nc_t, 547
* **effect size estimates**
cles, 184
cliff_delta, 187
correction_for_attenuation, 237
eta_squared, 341
eta_squared_generalized, 345
expected_partial_r, 350
expected_r, 352
expected_smd, 357
nnt_from_smd, 556
omega_squared, 567
probability_of_superiority_paired, 629
proportion_of_superiority, 635
responder_analysis, 723
smd_trimmed, 804
* **equivalence testing**
equivalence_c, 325
equivalence_proportion_paired, 329
equivalence_r, 333
equivalence_rmsea, 335
equivalence_smd, 338
plot_equivalence, 581
power_density_equivalence_md, 606
power_equivalence_c, 608
power_equivalence_md, 611
power_equivalence_md_plot, 614
ss_power_equivalence_c, 1085
* **hplot**
plot_ci, 579
plot_equivalence, 581
plot_fa_confirmatory, 584
plot_forest, 586
plot_irt_information, 588
plot_mediation_mbco, 590
plot_R2, 593
plot_randomization_test, 595
plot_regions_of_significance, 597
plot_smd, 599
plot_trajectories, 601
plot_trajectories_fitted, 604
power_equivalence_md_plot, 614
* **htest**
ancova, 20
anova_within, 25
anova_within_two_way, 26
bayes_independent_t, 33
bayes_one_sample_t, 36
bayes_paired_t, 38
ci_c_ancova_bp, 86
ci_crossed, 78
ci_cv, 80
ci_dunnett, 89
ci_dunnett_t3, 91
ci_eigenvalue, 94
ci_eta_squared, 95
ci_eta_squared_generalized, 101
ci_games_howell, 107
ci_mahalanobis, 110
ci_nc_t, 118
ci_omega_squared, 121
ci_proportion, 125
ci_proportion_paired, 126
ci_R2, 132
ci_ratio_of_means, 135
ci_ratio_of_means_paired, 141
ci_reg_coef, 150
ci_scheffe, 158
ci_smd, 165
ci_smd_c, 168
ci_tukey_kramer, 179
ci_two_phase, 181
cles, 184
cliff_delta, 187
cohen_kappa, 196
combine_p, 201
compare_cov_structures, 210
contrast_adjusted, 219
contrast_test, 221
correlations_test, 240
cv, 252
cv_bonferroni_f, 254
cv_bryant_paulson, 256
cv_chisq, 259
cv_dunnett, 261
cv_f, 263
cv_scheffe, 265
cv_smm, 266
cv_tukey_hsd, 270
df_barnard_rubin, 283
df_missing, 286
dunn_test, 318
epsilon_corrections, 324
equivalence_c, 325
equivalence_proportion_paired, 329

equivalence_r, 333
equivalence_rmsea, 335
equivalence_smd, 338
eta_squared, 341
eta_squared_generalized, 345
expected_partial_r, 350
expected_r, 352
expected_smd, 357
factorial_anova, 360
fleiss_kappa, 424
gwet_ac, 442
icc, 451
icc_lmer, 453
krippendorff_alpha, 465
limits_of_agreement, 470
lin_ccc, 473
manova_split_plot, 474
mauchly_test, 478
meta_contrast, 512
mixed_anova, 525
mr_cv, 549
mr_smd, 554
nnt_from_smd, 556
obrien_test, 564
omega_squared, 567
pairwise_within, 574
power_fisher_exact, 616
probability_of_superiority_paired, 629
procrustes_phi, 633
proportion_of_superiority, 635
R2_mixed_effects, 639
randomization_test, 644
randomization_test_paired, 650
regions_of_significance, 653
reliability, 657
reliability_alpha, 662
reliability_H, 676
reliability_kr20, 685
reliability_omega, 690
reliability_omega_categorical, 707
responder_analysis, 723
sd_unbiased, 727
sem_exact_fit_bootstrap, 729
signal_to_noise_R2, 759
simple_effects_AB, 761
smd, 799
smd_c, 802
smd_trimmed, 804
ss_aipe_c_sensitivity, 859
ss_aipe_cliff_delta_sensitivity, 811
ss_aipe_crossed_sensitivity, 842
ss_aipe_cv, 846
ss_aipe_cv_sensitivity, 850
ss_aipe_equivalence_r_sensitivity, 865
ss_aipe_equivalence_smd_sensitivity, 872
ss_aipe_icc_sensitivity, 878
ss_aipe_indirect_effect_sensitivity, 887
ss_aipe_mean_sensitivity, 894
ss_aipe_omega_squared_sensitivity, 908
ss_aipe_partial_r_sensitivity, 916
ss_aipe_r_sensitivity, 979
ss_aipe_reliability_sensitivity, 967
ss_aipe_semipartial_r_sensitivity, 1003
ss_aipe_sm_sensitivity, 1023
ss_aipe_smd_sensitivity, 1020
ss_aipe_two_phase_sensitivity, 1043
ss_power_c, 1047
ss_power_c_ancova, 1083
ss_power_composite_ancova, 1049
ss_power_composite_ancova_2group, 1053
ss_power_composite_anova, 1060
ss_power_composite_factorial_ancova, 1063
ss_power_composite_factorial_ancova_het, 1067
ss_power_composite_factorial_anova, 1072
ss_power_composite_sem, 1075
ss_power_factorial_anova, 1090
ss_power_mixed_effects, 1096
ss_power_one_way_anova, 1098
ss_power_r, 1104
ss_power_rm_anova, 1133
ss_power_sc, 1135
ss_power_smd, 1139
ss_power_split_plot_anova, 1141

summary_t_test, 1152
var_alpha, 1186
var_cv, 1188
var_ete, 1191
var_indirect_effect, 1196
var_omega_squared, 1199
var_r, 1203
var_smd, 1210
var_smd_trimmed, 1212
vargha_delaney_A, 1177
variance_components_mls, 1184
welch_t, 1214
∗ **hypothesis tests**
adjusted_means, 14
ancova, 20
anova_within, 25
ci_dunnett, 89
ci_dunnett_t3, 91
ci_games_howell, 107
ci_scheffe, 158
ci_tukey_kramer, 179
compare_cov_structures, 210
contrast_test, 221
correlations_test, 240
equivalence_proportion_paired, 329
equivalence_r, 333
equivalence_rmsea, 335
equivalence_smd, 338
factorial_anova, 360
manova_split_plot, 474
mauchly_test, 478
mixed_anova, 525
obrien_test, 564
pairwise_within, 574
randomization_test, 644
randomization_test_paired, 650
regions_of_significance, 653
simple_effects_AB, 761
summary_t_test, 1152
welch_t, 1214
∗ **manip**
missing_coverage, 522
missing_patterns, 524
∗ **mediation**
mediate, 496
mediation_mbco, 500
ss_power_indirect_effect, 1093
∗ **meta-analysis**
combine_p, 201
meta_contrast, 512
meta_es, 514
meta_r, 518
meta_smd, 521
plot_forest, 586
∗ **misc**
mr_cv, 549
mr_smd, 554
∗ **missing data**
cov_fiml, 246
df_barnard_rubin, 283
df_missing, 286
fraction_missing_information, 430
missing_coverage, 522
missing_patterns, 524
∗ **mixed models**
icc_lmer, 453
manova_split_plot, 474
mixed_anova, 525
R2_mixed_effects, 639
R2_mixed_effects_decomposition, 642
ss_aipe_mixed_effects, 898
ss_aipe_mixed_effects_sensitivity, 902
ss_power_mixed_effects, 1096
ss_power_split_plot_anova, 1141
∗ **mixed-effects R-squared**
R2_mixed_effects_decomposition, 642
∗ **models**
adjusted_means, 14
analysis_of_change, 17
ci_cv, 80
ci_nc_t, 118
ci_smd_c, 168
fraction_missing_information, 430
irt_grm, 455
measurement_alignment, 480
measurement_invariance, 485
mediate, 496
mediation_mbco, 500
meta_es, 514
meta_r, 518
meta_smd, 521
mlmr, 528
mlmr_mv, 537

R2_mixed_effects_decomposition, 642
regions_of_significance, 653
signal_to_noise_R2, 759
smd, 799
smd_c, 802
∗ **multilevel R-squared**
R2_mixed_effects_decomposition, 642
∗ **multivariate and latent variable methods**
average_variance_extracted, 29
bifactor_indices, 44
cfa_1, 53
cfa_2, 58
ci_bifactor_indices, 62
ci_eigenvalue, 94
common_method_adjusted_r, 203
common_method_marker, 206
common_method_single_factor, 208
compare_sem, 212
dmacs, 292
ecvi, 320
equivalence_rmsea, 335
fa_bifactor, 382
fa_confirmatory, 390
fa_second_order, 415
factor_congruence, 362
factor_scores, 371
htmt, 448
irt_grm, 455
irt_information, 460
measurement_alignment, 480
measurement_invariance, 485
procrustes_phi, 633
sem_exact_fit_bootstrap, 729
sem_fit_measures, 734
sem_information_criteria, 747
sem_residual_fit, 754
simple_structure, 765
∗ **multivariate**
average_variance_extracted, 29
bifactor_indices, 44
cfa_1, 53
cfa_2, 58
ci_bifactor_indices, 62
ci_mahalanobis, 110
ci_nc_chisq, 113
ci_nc_F, 115
ci_R2, 132
common_method_adjusted_r, 203
common_method_marker, 206
common_method_single_factor, 208
compare_sem, 212
content_validity_index, 216
convert_R2, 231
correlations_test, 240
cov_fiml, 246
cov_sem, 250
covmat_from_cfa, 244
descriptives, 276
design_effect, 281
dmacs, 292
ecvi, 320
fa_bifactor, 382
fa_confirmatory, 390
fa_second_order, 415
factor_scores, 371
fraction_missing_information, 430
htmt, 448
icc, 451
irt_grm, 455
irt_information, 460
measurement_alignment, 480
measurement_invariance, 485
mlmr_mv, 537
reliability, 657
reliability_alpha, 662
reliability_kr20, 685
reliability_omega, 690
reliability_omega_categorical, 707
sem_exact_fit_bootstrap, 729
sem_fit_measures, 734
sem_information_criteria, 747
sem_residual_fit, 754
simple_structure, 765
ss_aipe_composite_sem, 814
ss_aipe_mixed_effects_sensitivity, 902
ss_aipe_pcm, 920
ss_aipe_pcm_sensitivity, 923
ss_aipe_sem_path, 1007
ss_aipe_sem_path_sensitivity, 1009
ss_power_composite_sem, 1075
ss_power_sem, 1137
∗ **noncentral distribution confidence intervals**

ci_nc_chisq, 113
ci_nc_F, 115
ci_nc_t, 118
* **noncentral distribution moments**
moments_nc_chisq, 543
moments_nc_F, 545
moments_nc_t, 547
* **nonparametric**
dunn_test, 318
randomization_test, 644
vargha_delaney_A, 1177
* **package**
DMAR-package, 10
* **parameterization conversions**
convert_cor_cov, 225
convert_d_or, 226
convert_d_r, 227
convert_F_chisq, 229
convert_R2, 231
convert_r_Z, 233
convert_t_smd, 234
convert_z_normal, 235
convert_Z_r, 236
* **plotting**
plot_ci, 579
plot_equivalence, 581
plot_fa_confirmatory, 584
plot_forest, 586
plot_irt_information, 588
plot_mediation_mbco, 590
plot_R2, 593
plot_randomization_test, 595
plot_regions_of_significance, 597
plot_smd, 599
plot_trajectories, 601
plot_trajectories_fitted, 604
power_equivalence_md_plot, 614
* **probability of superiority**
cles, 184
vargha_delaney_A, 1177
* **regression**
analysis_of_change, 17
ci_correlation, 74
ci_nc_chisq, 113
ci_nc_F, 115
ci_R2, 132
mlmr, 528
mlmr_mv, 537
var_partial_r, 1201
var_semipartial_r, 1206
* **reliability**
cohen_kappa, 196
diagnosis_agreement, 290
fleiss_kappa, 424
icc, 451
reliability, 657
reliability_alpha, 662
reliability_H, 676
reliability_kr20, 685
reliability_omega, 690
reliability_omega_categorical, 707
* **sample size for power**
power_fisher_exact, 616
ss_aipe_mixed_effects, 898
ss_power_c, 1047
ss_power_c_ancova, 1083
ss_power_composite_ancova, 1049
ss_power_composite_ancova_2group, 1053
ss_power_composite_anova, 1060
ss_power_composite_factorial_ancova, 1063
ss_power_composite_factorial_ancova_het, 1067
ss_power_composite_factorial_anova, 1072
ss_power_composite_sem, 1075
ss_power_contrast, 1079
ss_power_equivalence_c, 1085
ss_power_factorial_ancova, 1088
ss_power_factorial_anova, 1090
ss_power_indirect_effect, 1093
ss_power_mixed_effects, 1096
ss_power_one_way_anova, 1098
ss_power_pcm, 1100
ss_power_r, 1104
ss_power_R2, 1107
ss_power_R2_change, 1111
ss_power_R2_sensitivity, 1117
ss_power_rc, 1120
ss_power_reg_coef, 1124
ss_power_reg_coef_sensitivity, 1129
ss_power_rm_anova, 1133
ss_power_sc, 1135
ss_power_sem, 1137

ss_power_smd, 1139
ss_power_split_plot_anova, 1141
∗ **sequential estimation**
ss_seq_c, 1144
ss_seq_c_sensitivity, 1147
ss_seq_two_phase, 1150
∗ **univar**
ci_smd, 165
descriptives, 276
kurtosis, 468
skewness, 798
var_icc, 1194
∗ **utilities**
dmar_bootstrap, 296
dmar_metadata, 301
∗ **variance utilities**
var_alpha, 1186
var_cv, 1188
var_ete, 1191
var_indirect_effect, 1196
var_omega_squared, 1199
var_r, 1203
var_smd, 1210
var_smd_trimmed, 1212
∗ **within-subjects analysis**
anova_within, 25
anova_within_two_way, 26
epsilon_corrections, 324
mauchly_test, 478
pairwise_within, 574
plot_trajectories_fitted, 604

adjusted_means, *12*, 14
adjusted_means(), *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
alpha, *671*, *675*, *688*, *689*
analysis_of_change, *12*, 17
ancova, *12*, *16*, 20, *88*, *362*, *639*, *656*, *1058*, *1090*, *1159*, *1193*
ancova(), *16*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
Anova, *474–477*
anova, *212*, *1199*
anova.mlmr, 22, *24*
anova.mlmr_mv, 24
anova_within, *12*, 25, *28*, *325*, *479*, *527*, *576*
anova_within(), *16*, *22*, *28*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *325*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *606*, *649*, *653*, *656*, *764*, *1154*, *1215*
anova_within_two_way, *12*, 26, *477*, *527*
anova_within_two_way(), *26*, *325*, *479*, *576*, *606*
aov, *14, 15*, *26*, *89*, *91*, *96, 97*, *102*, *108*, *121, 122*, *159*, *179*, *219–222*, *318*, *341, 342*, *346, 347*, *567, 568*, *572*, *761, 762*, *1199*
as_kable, *13*, *370*, *372*, *408*
as_kable *(*dmar_output_helpers*)*, 310
average_variance_extracted, *12*, 29, *48*, *62*, *300, 301*, *308*, *404*, *450*, *766*
average_variance_extracted(), *49*, *57*, *62*, *70*, *95*, *206, 207*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*

bartlett.test, *566*
bayes_independent_t, *12*, 33, *38*, *40*
bayes_independent_t(), *38*, *40*
bayes_one_sample_t, *12*, *33–35*, 36, *39, 40*
bayes_one_sample_t(), *35*, *40*
bayes_paired_t, *12*, *35*, *38*, 38
bayes_paired_t(), *35*, *38*
bessel_errors, *13*, 41
bifactor_indices, *12*, 44, *63*, *68*, *70*, *308*, *384*, *386–389*, *411*, *680*, *684*
bifactor_indices(), *33*, *57*, *62*, *70*, *95*, *206, 207*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
binom.test, *217*
bootMer, *640, 641*
bryant_paulson, 49, *86*, *88*, *1158, 1159*

cfa, *459*, *488*
cfa_1, *12*, *29*, *32*, 53, *62*, *239*, *245*, *295*, *302*, *305*, *307*, *363*, *395*, *404*, *459*, *484*, *486*, *494*, *662*, *675*, *695*, *705*, *964, 965*
cfa_1(), *33*, *49*, *62*, *70*, *95*, *206, 207*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*,

404, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
cfa_2, *12*, *57*, 58, *302*, *305*, *307*, *363*, *395*, *404*
cfa_2(), *33*, *49*, *57*, *70*, *95*, *206, 207*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
cfa_k, *12*
cfa_k *(*fa_confirmatory*)*, 390
chisq.test, *725*
ci_bifactor_indices, *11, 12*, *46–49*, 62, *297, 298*, *300, 301*, *307–309*, *384*, *387–389*, *680*, *742*
ci_bifactor_indices(), *33*, *49*, *57*, *62*, *95*, *206, 207*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
ci_c, *11*, 71, *85*, *120*, *158*, *326*, *328*, *573*, *808*, *860*, *862*, *1049*
ci_c(), *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_c_ancova, *11*, *22*, 83, *86–88*, *158*, *162*, *221*, *275*, *769*, *856*, *1058*, *1084*
ci_c_ancova(), *73*, *77*, *83*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_c_ancova_bp, *11*, *52*, *83–85*, 86, *220*, *258*, *1159*
ci_c_ancova_bp(), *73*, *77*, *83*, *85*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_correlation, *73*, 74, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_crossed, *11*, 78, *838, 839*, *841, 842*, *844, 845*
ci_crossed(), *126*, *128*, *184*
ci_cv, *11*, 80, *118*, *120*, *252–254*, *550*, *553*, *846–848*, *852, 853*, *1188–1190*
ci_cv(), *73*, *77*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_dunnett, *11*, *16*, 89, *93*, *109*, *159*, *180*, *1166*
ci_dunnett(), *16*, *22*, *26*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
ci_dunnett_t3, *11*, 91, *109*, *267, 268*, *1166*
ci_dunnett_t3(), *16*, *22*, *26*, *90*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
ci_eigenvalue, *11*, 94
ci_eigenvalue(), *33*, *49*, *57*, *62*, *70*, *206, 207*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
ci_eta_squared, *11*, *21*, 95, *103–106*, *122–124*, *307*, *310, 311*, *342*, *344*, *361, 362*, *569*, *580*, *764*, *1165*
ci_eta_squared(), *73*, *77*, *83*, *85*, *88*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_eta_squared_generalized, *11*, *100*, 101, *297*, *301*, *346–348*, *580*, *1165*
ci_eta_squared_generalized(), *73*, *77*, *83*, *85*, *88*, *100*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_games_howell, *11*, *92, 93*, 107, *319*, *1166*
ci_games_howell(), *16*, *22*, *26*, *90*, *93*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
ci_mahalanobis, *11*, 110, *1166*
ci_mahalanobis(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_nc_chisq, *13*, 113, *116–118*, *121*, *154*, *230, 231*, *260, 261*, *321, 322*, *336*, *544*, *736*, *977*
ci_nc_chisq(), *117*, *121*
ci_nc_F, *13*, *76*, *97*, *100*, *104*, *111–113*, *115*, 115, *118*, *121*, *123, 124*, *130, 131*, *133*, *135*, *172*, *177, 178*, *230–232*,

264, *308*, *546*, *763, 764*, *1092*, *1100*, *1110*, *1123*, *1127*
ci_nc_F(), *115*, *121*
ci_nc_t, *13*, *72*, *75*, *77*, *81*, *113*, *115–117*, 118, *148, 149*, *151*, *153*, *157, 158*, *161*, *164–166*, *168–170*, *174*, *176*, *232*, *801*, *803–806*, *846, 847*, *860*, *866*, *926*, *932*, *937*, *939*, *950*, *980*, *983*, *985*, *987*, *997*, *1014*, *1016–1019*, *1031*, *1141*
ci_nc_t(), *115*, *117*
ci_omega_squared, *11*, *21*, *97*, *100*, *104*, 121, *307*, *310*, *348*, *361, 362*, *568–570*, *580, 581*, *906–908*, *910*, *1165*, *1199, 1200*
ci_omega_squared(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_proportion, *11*, 125, *128*, *195*, *724*, *726*, *732*
ci_proportion(), *79*, *128*, *184*
ci_proportion_paired, *11*, 126, *329, 330*, *332*, *724*
ci_proportion_paired(), *79*, *126*, *184*
ci_pvaf, *11*, *97*, *100*, *104*, *116*, *123, 124*, 129, *344*, *570*, *1165*
ci_pvaf(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_R, *11*, *580, 581*, *595*
ci_R *(*ci_correlation*)*, 74
ci_r, *11*, *120*, *204*, *206*, *228*, *234*, *237*, *239*, *243*, *334*, *355*, *453*, *863–865*, *1106*, *1165*, *1203, 1204*
ci_r *(*ci_correlation*)*, 74
ci_R2, *11*, *76, 77*, *112*, *116, 117*, 132, *232*, *350*, *357*, *580, 581*, *594, 595*, *761*, *797*, *912*, *916*, *930*, *932*, *937*, *998*, *1004*, *1109, 1110*, *1113*, *1116*, *1120*, *1176*, *1201*, *1206, 1207*
ci_R2(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_ratio_of_means, *11*, 135, *142*, *146*, *298*, *301*, *305*, *313*, *1165*
ci_ratio_of_means(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_ratio_of_means_paired, *11*, *136*, *140*, 141, *298*, *301*, *305*, *313*, *1165*
ci_ratio_of_means_paired(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_rc, *11*, *120*, 147, *150*, *153*, *172*, *176*, *535*, *939*, *944, 945*, *947*, *949*, *1166*
ci_rc(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_reg_coef, *11*, *120*, *147–149*, 150, *172*, *174–176*, *350*, *535*, *912*, *917*, *943*, *947*, *954, 955*, *998*, *1004*, *1027*, *1031*, *1034*, *1036*, *1165*, *1201*, *1207*, *1209*
ci_reg_coef(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_rmsea, *11*, 153, *302*, *321, 322*, *337*, *742*, *746*, *974–977*, *979*
ci_rmsea(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_sc, *11*, *73*, *120*, 156, *162*, *982–985*, *995, 996*, *1136*
ci_sc(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_sc_ancova, *11*, *22*, *85*, *120*, 160, *275*, *856*, *987–989*, *1058*, *1090*
ci_sc_ancova(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_scheffe, *11*, *90*, *109*, 158, *180*, *465*, *1166*
ci_scheffe(), *16*, *22*, *26*, *90*, *93*, *109*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*

ci_sm, *11*, *37, 38*, *120*, *158*, 163, *1015, 1016*
ci_sm(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *168*, *170*, *172*, *176*, *178*, *221*, *601*
ci_smd, *11*, *35*, *37*, *112*, *118*, *120, 121*, *158*, 165, *170*, *186*, *235*, *316*, *339, 340*, *359*, *522*, *556–558*, *561*, *580, 581*, *600, 601*, *635, 636*, *648, 649*, *801*, *804, 805*, *870, 871*, *874*, *1017–1019*, *1141*, *1183*, *1211, 1212*, *1215*
ci_smd(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *170*, *172*, *176*, *178*, *221*, *601*
ci_smd_c, *11*, *118*, *120, 121*, *158*, *166*, *168*, 168, *801*, *1019*, *1165*
ci_smd_c(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *172*, *176*, *178*, *221*, *601*
ci_snr, *11*, *116*, 170, *178*, *194*, *1166*
ci_snr(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *176*, *178*, *221*, *601*
ci_src, *11*, *120*, *147*, *149–151*, *153*, *158*, 172, *535*, *944, 945*, *947*, *1027–1029*, *1031*, *1033, 1034*
ci_src(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *178*, *221*, *601*
ci_srsnr, *11*, *172*, 176, *193, 194*
ci_srsnr(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *221*, *601*
ci_tukey_kramer, *11*, *90*, *92, 93*, *108, 109*, *159*, 179, *319*, *575*, *1166*
ci_tukey_kramer(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
ci_two_phase, *11*, 181, *1036–1040*, *1043–1046*, *1150–1152*
ci_two_phase(), *79*, *126*, *128*
cles, *11*, 184, *190, 191*, *311*, *561*, *632*, *635, 636*, *648, 649*, *1166*, *1182, 1183*
cles(), *191*, *239*, *344*, *348*, *351*, *355*, *359*, *562*, *570*, *632*, *636*, *727*, *805*
cliff_delta, *11*, 187, *302*, *306*, *312*, *316*, *319*, *558*, *561*, *629, 630*, *632*, *648, 649*, *726*, *810, 811*, *814*, *1166*, *1180*, *1182, 1183*
cliff_delta(), *186*, *239*, *344*, *348*, *351*, *355*, *359*, *562*, *570*, *632*, *636*, *727*, *805*
cohen_f, *11*, *177*, 192, *195*
cohen_h, *11*, 194
cohen_kappa, *12*, 196, *291, 292*, *298*, *300, 301*, *427*, *443*, *465*, *468*, *1183*
cohen_kappa(), *292*, *427*, *453*, *662*, *675*, *684*, *689*, *705*, *721*
combine_p, *12*, 201, *514*, *517*, *1156, 1157*
combine_p(), *514*, *517*, *520*, *522*, *587*
common_method_adjusted_r, *12*, 203, *207*, *310*
common_method_adjusted_r(), *33*, *49*, *57*, *62*, *70*, *95*, *207*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
common_method_marker, *12*, *204*, *206*, 206, *208, 209*
common_method_marker(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
common_method_single_factor, *12*, *206, 207*, 208
common_method_single_factor(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
compare_cov_structures, *12*, 210, *494*
compare_cov_structures(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
compare_sem, *12*, 212, *304*, *368–371*, *742*, *744*, *746, 747*, *750*, *753*
compare_sem(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *295*, *322*, *337*, *366*, *381*,

389, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
content_validity_index, *12*, 216, *1166*
content_validity_index(), *443*, *454*, *468*, *472*, *474*, *642*, *1186*
contr.helmert, *444*
contr.poly, *572, 573*
contr.sum, *323*, *343*, *361*
contrast_adjusted, *12*, *15, 16*, 219, *255, 256*, *318*
contrast_adjusted(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *601*
contrast_test, *12*, *221*, 221, *263*, *266*, *316*, *328*, *514*, *573*, *575*, *764*, *774*, *1082*, *1167*
contrast_test(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
convert_chisq_F, *12*
convert_chisq_F *(*convert_F_chisq*)*, 229
convert_cor_cov, *12*, 225
convert_cor_cov(), *227, 228*, *231, 232*, *234–237*
convert_d_or, *12*, 226, *228*
convert_d_or(), *226*, *228*, *231, 232*, *234–237*
convert_d_r, *12*, *227*, 227
convert_d_r(), *226, 227*, *231, 232*, *234–237*
convert_delta_lambda, *12*
convert_delta_lambda *(*convert_t_smd*)*, 234
convert_F_chisq, *12*, 229
convert_F_chisq(), *226–228*, *232*, *234–237*
convert_f_R2, *12*
convert_f_R2 *(*convert_R2*)*, 231
convert_lambda_delta, *12*
convert_lambda_delta *(*convert_t_smd*)*, 234
convert_lambda_R2, *12*
convert_lambda_R2 *(*convert_R2*)*, 231
convert_or_d, *13*, *228*
convert_or_d *(*convert_d_or*)*, 226
convert_R2, *226–228*, *231*, 231, *234–237*
convert_R2_f, *13*, *760, 761*
convert_R2_f *(*convert_R2*)*, 231
convert_R2_lambda, *13*
convert_R2_lambda *(*convert_R2*)*, 231
convert_r_d, *13*, *227*
convert_r_d *(*convert_d_r*)*, 227
convert_r_Z, *13*, *77*, 233, *237*, *239*, *520*, *929*, *1106*, *1203*
convert_r_Z(), *226–228*, *231, 232*, *235–237*
convert_t_smd, *226–228*, *231, 232*, *234*, 234, *236, 237*
convert_z_normal, *13*, 235
convert_z_normal(), *226–228*, *231, 232*, *234, 235*, *237*
convert_Z_r, *13*, *77*, *233, 234*, 236, *239*, *520*, *1106*
convert_Z_r(), *226–228*, *231, 232*, *234–236*
cor, *243*, *277*, *355*, *634*
cor.test, *241–243*, *355*
correction_for_attenuation, *11*, 237, *312*, *449, 450*, *518*, *520*, *1166*
correction_for_attenuation(), *186*, *191*, *344*, *348*, *351*, *355*, *359*, *562*, *570*, *632*, *636*, *727*, *805*
correlations_test, *12*, 240, *429*, *926*, *929*, *979*
correlations_test(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
cov2cor, *209*, *277*
cov_fiml, *12*, 246, *369*, *407*, *436*, *531*, *577*, *664*, *668*, *678*, *692*, *700*
cov_fiml(), *285*, *289*, *438*, *523*, *525*
cov_sem, *12*, *248*, 250, *815*, *818*, *1007, 1008*, *1011*, *1013*, *1075*, *1078*
covmat_from_cfa, *12*, 244, *248*, *251*
cv, *12*, *82, 83*, 252, *553*, *728*, *849*, *853*, *1190*
cv_bonferroni_f, *12*, 254, *264*
cv_bonferroni_f(), *258*, *261*, *263, 264*, *266*, *268*, *270*, *273, 274*
cv_bryant_paulson, *12*, 256
cv_bryant_paulson(), *256*, *261*, *263, 264*, *266*, *268*, *270*, *273, 274*
cv_chisq, *12*, *231*, 259, *264*
cv_chisq(), *256*, *258*, *263, 264*, *266*, *268*, *270*, *273, 274*
cv_dunnett, *12*, *90*, *255–257*, 261, *265*, *268*, *271*
cv_dunnett(), *256*, *258*, *261*, *264*, *266*, *268*,

270, *273, 274*
cv_f, *12*, *231*, *255, 256*, *261*, 263
cv_f(), *256*, *258*, *261*, *263*, *266*, *268*, *270*, *273, 274*
cv_scheffe, *12*, *159*, *220*, *256*, *258*, *263, 264*, 265, *268*
cv_scheffe(), *256*, *258*, *261*, *263, 264*, *268*, *270*, *273, 274*
cv_smm, *12*, *91*, *93*, *109*, *257*, *263*, *265*, 266, *271*
cv_smm(), *256*, *258*, *261*, *263, 264*, *266*, *270*, *273, 274*
cv_t, *12*, *255*, *258*, *260–264*, *266*, *268*, 269, *271–273*, *1082*
cv_t(), *256*, *258*, *261*, *263, 264*, *266*, *268*, *273, 274*
cv_tukey_hsd, *12*, *50, 51*, *108, 109*, *180*, *223, 224*, *255–258*, *263*, *266*, *268*, 270
cv_tukey_hsd(), *256*, *258*, *261*, *263, 264*, *266*, *268*, *270*, *274*
cv_z, *12*, *236*, *271–273*, 273
cv_z(), *256*, *258*, *261*, *263, 264*, *266*, *268*, *270*, *273*

dbryant_paulson, *12*
dbryant_paulson *(*bryant_paulson*)*, 49
dchisq, *544*
depression_bdi, *13*, 274
descriptives, *11*, *243*, 276, *453*, *469*, *798, 799*
descriptives(), *469*, *799*
design_consequences, *12*, 278, *808*, *810*, *814*, *825*, *836*, *849*, *853*, *856*, *859*, *862*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *901*, *905*, *908*, *910*, *915*, *919*, *922*, *925*, *929*, *932*, *937*, *939*, *943*, *950*, *955*, *965*, *973*, *975*, *981*, *985*, *990*, *994*, *997*, *1002*, *1006*, *1009*, *1013*, *1016*, *1019*, *1023*, *1026*, *1031*, *1036*, *1049*, *1082*, *1084*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
design_consequences(), *283*, *323*, *444*, *465*, *573*
design_effect, *12*, 281
design_effect(), *280*, *323*, *444*, *465*, *573*
df, *546*
df_barnard_rubin, *12*, 283, *288, 289*
df_barnard_rubin(), *249*, *289*, *438*, *523*, *525*
df_missing, *12*, *285*, 286, *399*, *402*, *532*, *534*, *699*, *702*
df_missing(), *249*, *285*, *438*, *523*, *525*
diagnosis_agreement, *13*, *199*, 290, *427*, *453*, *662*, *675*, *684*, *689*, *705*, *721*
difftime, *563*
dmacs, *12*, 292, *308*
dmacs(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
DMAR *(*DMAR-package*)*, 10
dmar *(*DMAR-package*)*, 10
DMAR-package, 10
dmar.bootstrap_exploratory *(*dmar_bootstrap*)*, 296
dmar_bootstrap, *13*, *30*, *32*, *64*, *66*, *69*, *103*, *105*, *137*, *139*, *143*, *145*, *197*, *199*, 296, *303*, *309*, *368*, *425*, *427*, *431*, *436*, *449, 450*, *466, 467*, *497*, *503*, *507*, *529*, *533*, *538*, *540*, *578*, *591*, *640, 641*, *659*, *665*, *672, 673*, *678*, *686*, *688*, *692*, *703*, *708*, *719*, *729*, *732*, *883*, *885*, *888, 889*, *969*, *971, 972*
dmar_metadata, *13*, *53*, *58*, *65*, *98*, *189*, *199*, *297*, *301*, 301, *342*, *368*, *383*, *385*, *388*, *391*, *407*, *410*, *416*, *419*, *427*, *449*, *480*, *486*, *491*, *499*, *569*, *577*, *631*, *660*, *666*, *679*, *687*, *693*, *709*, *732*, *742*, *758*, *800*, *803*, *1108*, *1179*
dmar_output_helpers, 310
dmar_tbl, *13*, *16*, *22*, *28*, *92*, *99*, *109*, *123*, *223*, *241*, *295*, *309–311*, *314*, *319*, *343*, *361*, *365*, *368*, *372*, *408*, *429*, *476*, *527*, *569*, *1161*, *1168*
dmar_tidiers, *13*, *28*, *92, 93*, *99*, *109*, *123*, *140*, *145*, *190*, *223, 224*, *309*, *311–313*, *343*, *366*, *370*, *476*, *527*, *569*, *632*, *962*, *1160, 1161*, *1183*
dmar_tidiers *(*tidy.dmar_cliff_delta*)*, 1161
drinks_trial, *13*, 314
dt, *548*
dunn_test, *12*, *255*, 318

ecvi, *12*, *304*, 320, *742*

ecvi(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
efa_k, *12*
efa_k *(*fa_exploratory*)*, 406
effects_coding, *12*, 322, *444*, *465*, *573*
effects_coding(), *280*, *283*, *444*, *465*, *573*
eigen, *95*
epsilon_corrections, *12*, *26*, *28*, 324, *479*
epsilon_corrections(), *26*, *28*, *479*, *576*, *606*
equivalence_c, *12*, *312*, 325, *329*, *331, 332*, *581–583*, *611*, *1087*, *1147*
equivalence_c(), *332*, *334*, *337*, *340*, *583*, *608*, *611*, *613*, *616*, *1087*
equivalence_proportion_paired, *12*, *127, 128*, 329
equivalence_proportion_paired(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *328*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *583*, *608*, *611*, *613*, *616*, *649*, *653*, *656*, *764*, *1087*, *1154*, *1215*
equivalence_r, *12*, *312*, *328*, 333, *337*, *340*, *863, 864*, *866, 867*
equivalence_r(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *328*, *332*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *583*, *608*, *611*, *613*, *616*, *649*, *653*, *656*, *764*, *1087*, *1154*, *1215*
equivalence_rmsea, *12*, *154, 155*, *302*, 335, *733*, *737*, *746*, *757*
equivalence_rmsea(), *16*, *22*, *26*, *33*, *49*, *57*, *62*, *70*, *90*, *93*, *95*, *109*, *159*, *180*, *206*, *208, 209*, *212*, *215*, *224*, *243*, *295*, *322*, *328*, *332*, *334*, *340*, *362*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *477*, *479*, *484*, *494*, *527*, *566*, *576*, *583*, *608*, *611*, *613*, *616*, *634*, *649*, *653*, *656*, *733*, *746*, *753*, *757*, *764*, *766*, *1087*, *1154*, *1215*
equivalence_smd, *12*, *312*, *328*, *334*, *337*, 338, *868–873*, *875*
equivalence_smd(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *328*, *332*, *334*, *337*, *362*, *477*, *479*, *527*, *566*, *576*, *583*, *608*, *611*, *613*, *616*, *649*, *653*, *656*, *764*, *1087*, *1154*, *1215*
eta_squared, *11*, *95*, *97*, *99, 100*, *123*, *307*, 341, *348*, *567–570*, *764*, *1165*
eta_squared(), *186*, *191*, *239*, *348*, *351*, *355*, *359*, *562*, *570*, *632*, *636*, *727*, *805*
eta_squared_generalized, *11*, *102, 103*, *105, 106*, *343, 344*, 345
eta_squared_generalized(), *186*, *191*, *239*, *344*, *351*, *355*, *359*, *562*, *570*, *632*, *636*, *727*, *805*
expand.grid, *15*, *220*, *770, 771*
expected_partial_r, *11*, 350, *915*
expected_partial_r(), *186*, *191*, *239*, *344*, *348*, *355*, *359*, *562*, *570*, *632*, *636*, *727*, *805*
expected_r, *11*, *351*, 352, *359*, *1204*
expected_r(), *186*, *191*, *239*, *344*, *348*, *351*, *359*, *562*, *570*, *632*, *636*, *727*, *805*
expected_R2, *11*, 356, *359*, *930*, *1175, 1176*, *1206*
expected_smd, *11*, *280*, 357, *521*, *548*, *1211, 1212*
expected_smd(), *186*, *191*, *239*, *344*, *348*, *351*, *355*, *562*, *570*, *632*, *636*, *727*, *805*

fa_bifactor, *12*, *44, 45*, *47*, *49*, *66*, *302*, *305*, *307*, 382, *396, 397*, *404*, *412*, *418–422*, *424*, *446*, *483*, *492*, *508*, *681, 682*, *701*, *732*, *817*, *978*, *1008*, *1012*, *1077*
fa_bifactor(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *322*, *337*, *366*, *381*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
fa_confirmatory, *12*, *31*, *45*, *53*, *55–58*, *60–62*, *214*, *288, 289*, *302*, *304, 305*, *307, 308*, *311, 312*, *363*, *366*, *371*, *374*, *381, 382*, *384–389*, 390, *394*, *406–408*, *411*, *415*, *417–420*, *422–424*, *435*, *438, 439*, *449*, *487*, *490–492*, *494*, *584, 585*, *658*, *664*, *667, 668*, *671*, *678*, *681–684*, *691*, *695*, *698–700*, *740–742*, *751*, *964*, *1169*
fa_confirmatory(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*,

766
fa_exploratory, *12*, *302, 303*, *305*, *307*, *362, 363*, *366–369*, *371*, *387*, *395*, 406, *577*
fa_second_order, *12*, *302*, *305–307*, *389*, *397*, *404*, 415, *681*
fa_second_order(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
facet_wrap, *602, 603*
factanal, *209*
factor, *655*
factor_congruence, *12*, 362, *634*
factor_congruence(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *322*, *337*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
factor_count, *12*, *297*, *299*, *301, 302*, *305*, 367, *412*, *577*, *1166*
factor_scores, *12*, *46*, *49*, *303*, *311, 312*, 371, *384*, *389*, *411*, *415*, *680*, *684*
factor_scores(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *322*, *337*, *366*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
factorial_anova, *12*, *99*, *123*, *343*, 360
factorial_anova(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
fisher.test, *618*
fleiss_kappa, *12*, *199*, *219*, *298*, *300, 301*, 424, *443*, *465*, *468*, *1166*
fleiss_kappa(), *199*, *292*, *453*, *662*, *675*, *684*, *689*, *705*, *721*
format.dmar_tbl, *311*
format_p, *13*, *312*, *314*, 428, *628, 629*
formatC, *429*
formula, *360*, *528*, *538*
fraction_missing_information, *12*, *248, 249*, *287, 288*, *301*, *311*, *399, 400*, 430, *523–525*, *532*, *698*, *964*, *970*, *1166*, *1170*
fraction_missing_information(), *249*, *285*, *289*, *523*, *525*
ggsave, *603*
glance, *92*, *109*, *223*, *824*, *832*, *1048*, *1051*, *1056*, *1061*, *1065*, *1069*, *1073*, *1077*, *1081*, *1084*, *1087*, *1089*, *1091*, *1094*, *1097*, *1099*, *1115*, *1119*, *1122*, *1131*, *1134*, *1136*, *1143*, *1161*
glance.dmar_anova_es *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_ci_anova *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_ci_bifactor *(*ci_bifactor_indices*)*, 62
glance.dmar_ci_long *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_ci_R2 *(*tidy.dmar_ci_R2*)*, 1159
glance.dmar_ci_ratio_of_means *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_ci_smd *(*tidy.dmar_ci_smd*)*, 1160
glance.dmar_cliff_delta *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_common_method_adjusted_r *(*common_method_adjusted_r*)*, 203
glance.dmar_compare_sem *(*compare_sem*)*, 212
glance.dmar_content_validity *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_contrast_test *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_df_barnard_rubin *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_df_missing *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_dmacs *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_fa_bifactor *(*fa_bifactor*)*, 382
glance.dmar_fa_confirmatory, 439, *1169*
glance.dmar_fa_second_order *(*fa_second_order*)*, 415
glance.dmar_factor_analysis *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_factor_congruence *(*tidy.dmar_cliff_delta*)*, 1161
glance.dmar_factor_scores *(*factor_scores*)*, 371
glance.dmar_fmi, *437*

glance.dmar_fmi (tidy.dmar_fmi), 1170
glance.dmar_measurement_alignment (tidy.dmar_cliff_delta), 1161
glance.dmar_measurement_invariance (tidy.dmar_cliff_delta), 1161
glance.dmar_mediation_mbco, 440
glance.dmar_parallel_analysis (tidy.dmar_cliff_delta), 1161
glance.dmar_post_hoc_ci (tidy.dmar_cliff_delta), 1161
glance.dmar_probability_of_superiority_paired (tidy.dmar_cliff_delta), 1161
glance.dmar_R2_mixed_effects (tidy.dmar_R2_mixed_effects), 1171
glance.dmar_reliability, 440
glance.dmar_ss_aipe (tidy.dmar_cliff_delta), 1161
glance.dmar_ss_power (tidy.dmar_cliff_delta), 1161
glance.dmar_ss_power_sensitivity (tidy.dmar_cliff_delta), 1161
glance.dmar_summary_t_test (summary_t_test_broom), 1154
glance.dmar_tbl (tidy.dmar_cliff_delta), 1161
glance.dmar_vargha_delaney_A (tidy.dmar_cliff_delta), 1161
glance.dmar_welch_t (welch_t_broom), 1216
glance.mlmr, 441
glance.mlmr_mv (tidy.mlmr_mv), 1174
glht, *262*
gls, *210*, *212*
gwet_ac, *12*, *219*, 442
gwet_ac(), *219*, *454*, *468*, *472*, *474*, *642*, *1186*

helmert_coding, *12*, *323*, 444, *465*, *573*
helmert_coding(), *280*, *283*, *323*, *465*, *573*
holzinger_swineford, *13*, 445, *681*
htmt, *12*, *29*, *32*, *55*, *60*, *62*, *297*, *298*, *301*, *305*, *393*, *404*, 448, *484*, *766*
htmt(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*

icc, *12*, *79*, *199*, *282*, *283*, *427*, 451, *454*, *468*, *841*, *877*, *878*, *880*, *881*, *1186*, *1194–1196*
icc(), *199*, *292*, *427*, *662*, *675*, *684*, *689*, *705*, *721*
icc_lmer, *12*, *282*, *283*, 453, *640*, *642*, *644*, *905*
icc_lmer(), *219*, *443*, *468*, *472*, *474*, *477*, *527*, *642*, *644*, *901*, *905*, *1098*, *1144*, *1186*
integrate, *51*, *262*, *267*
irt_grm, *12*, *302*, 455
irt_grm(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
irt_information, *12*, 460, *588*, *589*
irt_information(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *484*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
is_orthogonal_set, *12*, *323*, *444*, 464, *573*
is_orthogonal_set(), *280*, *283*, *323*, *444*, *573*

kable, *242*, *311*
knit_print.dmar_tbl, *372*
knit_print.dmar_tbl (dmar_output_helpers), 310
krippendorff_alpha, *12*, *298*, *301*, *313*, *443*, 465
krippendorff_alpha(), *219*, *443*, *454*, *472*, *474*, *642*, *1186*
kruskal.test, *319*
kurtosis, *11*, *276*, 468, *798*, *799*, *807*, *891–893*, *895*, *897*, *1037*
kurtosis(), *277*, *799*

lavaan, *55*, *60*, *385*, *394*, *404*, *418*, *530*, *535*, *539*
lavOptions, *816*, *976*, *1008*, *1011*, *1076*
lavTestLRT, *22*, *24*, *397*, *404*, *489*, *492*
lavTestScore, *492*, *494*
lchoose, *217*
limits_of_agreement, *12*, 470, *474*
limits_of_agreement(), *219*, *443*, *454*, *468*, *474*, *642*, *1186*
lin_ccc, *12*, *472*, 473
lin_ccc(), *219*, *443*, *454*, *468*, *472*, *642*, *1186*

lm, *14, 15*, *89*, *91*, *96, 97*, *102*, *108*, *121, 122*, *159*, *179*, *219–222*, *318*, *341, 342*, *346, 347*, *528*, *535*, *537*, *541*, *567, 568*, *572*, *654*, *761, 762*, *1199*
lme, *640*
lmer, *27*, *453, 454*, *526, 527*, *640*, *642, 643*
loa, *12*
loa *(*limits_of_agreement*)*, 470
logLik, *212*

manova, *477*
manova_split_plot, *12*, *362*, 474
manova_split_plot(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *454*, *479*, *527*, *566*, *576*, *642*, *644*, *649*, *653*, *656*, *764*, *901*, *905*, *1098*, *1144*, *1154*, *1215*
mauchly.test, *478*
mauchly_test, *12*, *25, 26*, *28*, *325*, 478
mauchly_test(), *16*, *22*, *26*, *28*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *325*, *332*, *334*, *337*, *340*, *362*, *477*, *527*, *566*, *576*, *606*, *649*, *653*, *656*, *764*, *1154*, *1215*
mcnemar.test, *128*
measurement_alignment, *12*, *305*, *307*, 480
measurement_alignment(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *494*, *634*, *733*, *746*, *753*, *757*, *766*
measurement_invariance, *12*, *214*, *294, 295*, *304, 305*, *307, 308*, *376*, *381*, *394*, *398*, *404*, *480*, *484*, 485, *741, 742*, *751*
measurement_invariance(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *634*, *733*, *746*, *753*, *757*, *766*
mediate, *12*, *298–300*, *306*, 496, *511*, *1093–1095*
mediate(), *511*, *1095*
mediation_mbco, *12*, *298–300*, *308*, *312*, *440*, *499*, 500, *590–592*, *883–886*, *1171*
mediation_mbco(), *500*, *1095*
meta_contrast, *12*, *203*, 512, *517*, *1156, 1157*
meta_contrast(), *203*, *517*, *520*, *522*, *587*
meta_es, *12*, *201*, *203*, *513, 514*, 514, *518–522*, *586, 587*
meta_es(), *203*, *514*, *520*, *522*, *587*
meta_r, *12*, *514*, *517*, 518, *587*
meta_r(), *203*, *514*, *517*, *522*, *587*
meta_smd, *12*, *201*, *203*, *514*, *517*, 521, *587*, *1156, 1157*
meta_smd(), *203*, *514*, *517*, *520*, *587*
missing_coverage, *12*, *248, 249*, *397*, *438*, 522, *525*, *669*, *701*
missing_coverage(), *249*, *285*, *289*, *438*, *525*
missing_patterns, *12*, *248, 249*, *438*, *523*, 524
missing_patterns(), *249*, *285*, *289*, *438*, *523*
mixed_anova, *12*, *362*, 525
mixed_anova(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *454*, *477*, *479*, *566*, *576*, *642*, *644*, *649*, *653*, *656*, *764*, *901*, *905*, *1098*, *1144*, *1154*, *1215*
mlmr, *12*, *22, 23*, *289*, *298, 299*, *301*, *430*, *435*, *438*, *499*, 528, *537–541*, *1173*, *1175*
mlmr_mv, *12*, *24*, *299*, *301*, *430*, 537, *1174, 1175*
model.matrix, *528*, *533*, *540*
model.syntax, *250, 251*, *815*, *976*, *1007, 1008*, *1010, 1011*, *1075*
moments_nc_chisq, *13*, 543
moments_nc_chisq(), *546*, *548*
moments_nc_F, *13*, *544*, 545, *548*
moments_nc_F(), *544*, *548*
moments_nc_t, *13*, *544*, *546*, 547
moments_nc_t(), *544*, *546*
mr_cv, *12*, 549, *555, 556*, *1189, 1190*
mr_smd, *12*, *550*, *553*, 554, *1144*, *1147*
mvrnorm, *510*, *769*, *790*, *797*
mxSE, *506*

na.omit, *565*
nls, *18*
nnt_from_smd, *11*, *185, 186*, *191*, *303*, *311*, 556, *636*, *724–726*, *1166*
nnt_from_smd(), *186*, *191*, *239*, *344*, *348*, *351*, *355*, *359*, *570*, *632*, *636*, *727*, *805*
nobs, *96*, *103*, *122*
now, *13*, 562

obrien_test, *12*, 564
obrien_test(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *576*, *649*, *653*, *656*, *764*, *1154*, *1215*
omega, *702*, *705*, *718*, *721*
omega_squared, *11*, *22*, *121–124*, *307*, *344*, 567, *908*, *1165*, *1199*, *1200*
omega_squared(), *186*, *191*, *239*, *344*, *348*, *351*, *355*, *359*, *562*, *632*, *636*, *727*, *805*
oneway.test, *762*, *763*
optim, *482*
options, *353*
orthogonal_polynomial, *12*, 571
orthogonal_polynomial(), *280*, *283*, *323*, *444*, *465*

p.adjust, *222–224*, *255*, *318*, *319*, *762*, *763*
pairwise.t.test, *224*, *576*
pairwise_within, *12*, *223*, 574
pairwise_within(), *16*, *22*, *26*, *28*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *325*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *606*, *649*, *653*, *656*, *764*, *1154*, *1215*
parallel_analysis, *12*, *297*, *299–301*, *305*, *368–371*, *415*, 576
pbeta, *930*
pbryant_paulson, *12*
pbryant_paulson *(*bryant_paulson*)*, 49
pchisq(), *114*
pf, *930*
plot, *1055*, *1056*
plot.dmar_composite_power *(*ss_power_composite_ancova_2group*)*, 1053
plot.dmar_composite_power_factorial *(*ss_power_composite_factorial_ancova*)*, 1063
plot.dmar_composite_power_factorial_het *(*ss_power_composite_factorial_ancova_het*)*, 1067
plot_cfa_k, *13*
plot_cfa_k *(*plot_fa_confirmatory*)*, 584
plot_ci, *13*, 579, *583*, *585*, *595*, *598*, *601*
plot_ci(), *583*, *585*, *587*, *589*, *592*, *595*, *596*, *598*, *601*, *603*, *606*, *616*
plot_equivalence, *13*, *328*, 581
plot_equivalence(), *328*, *332*, *334*, *337*, *340*, *581*, *585*, *587*, *589*, *592*, *595*, *596*, *598*, *601*, *603*, *606*, *608*, *611*, *613*, *616*, *1087*
plot_fa_confirmatory, *13*, *404*, 584
plot_fa_confirmatory(), *581*, *583*, *587*, *589*, *592*, *595*, *596*, *598*, *601*, *603*, *606*, *616*
plot_forest, *13*, *517*, *522*, 586
plot_forest(), *203*, *514*, *517*, *520*, *522*, *581*, *583*, *585*, *589*, *592*, *595*, *596*, *598*, *601*, *603*, *606*, *616*
plot_irt_information, *13*, *463*, 588
plot_irt_information(), *581*, *583*, *585*, *587*, *592*, *595*, *596*, *598*, *601*, *603*, *606*, *616*
plot_mediation_mbco, *13*, *299*, *300*, *508*, *510*, *511*, 590
plot_mediation_mbco(), *581*, *583*, *585*, *587*, *589*, *595*, *596*, *598*, *601*, *603*, *606*, *616*
plot_R2, *13*, *581*, 593, *601*
plot_R2(), *581*, *583*, *585*, *587*, *589*, *592*, *596*, *598*, *601*, *603*, *606*, *616*
plot_randomization_test, *13*, 595, *648*, *649*
plot_randomization_test(), *581*, *583*, *585*, *587*, *589*, *592*, *595*, *598*, *601*, *603*, *606*, *616*
plot_regions_of_significance, *13*, 597, *656*
plot_regions_of_significance(), *581*, *583*, *585*, *587*, *589*, *592*, *595*, *596*, *601*, *603*, *606*, *616*
plot_smd, *13*, *168*, *581*, *595*, 599, *801*
plot_smd(), *73*, *77*, *83*, *85*, *88*, *100*, *106*, *112*, *124*, *131*, *135*, *140*, *146*, *149*, *153*, *155*, *158*, *162*, *165*, *168*, *170*, *172*, *176*, *178*, *221*, *581*, *583*, *585*, *587*, *589*, *592*, *595*, *596*, *598*, *603*, *606*, *616*
plot_trajectories, *13*, *19*, 601, *605*, *606*, *776*, *777*, *780*, *784*, *789*, *790*, *794*
plot_trajectories(), *581*, *583*, *585*, *587*, *589*, *592*, *595*, *596*, *598*, *601*, *606*, *616*
plot_trajectories_fitted, *13*, *603*, 604
plot_trajectories_fitted(), *26*, *28*, *325*,

479, *576*, *581*, *583*, *585*, *587*, *589*, *592*, *595, 596*, *598*, *601*, *603*, *616*
polychoric, *718*, *721*
power_density_equivalence_md, *12*, 606, *613*, *616*
power_density_equivalence_md(), *328*, *332*, *334*, *337*, *340*, *583*, *611*, *613*, *616*, *1087*
power_equivalence_c, *12*, *328*, 608, *1085–1087*
power_equivalence_c(), *328*, *332*, *334*, *337*, *340*, *583*, *608*, *613*, *616*, *1087*
power_equivalence_md, *12*, *606–608*, *610, 611*, 611, *615, 616*
power_equivalence_md(), *328*, *332*, *334*, *337*, *340*, *583*, *608*, *611*, *616*, *1087*
power_equivalence_md_plot, *12*, *608*, *613*, 614
power_equivalence_md_plot(), *328*, *332*, *334*, *337*, *340*, *581*, *583*, *585*, *587*, *589*, *592*, *595, 596*, *598*, *601*, *603*, *606*, *608*, *611*, *613*, *1087*
power_fisher_exact, *12*, 616
power_fisher_exact(), *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
prcomp, *95*
predict, *15*
prime_time_achievement, *13*, 619
print.dmar_fa_exploratory *(*fa_exploratory*)*, 406
print.dmar_factor_count *(*factor_count*)*, 367
print.dmar_factor_scores *(*factor_scores*)*, 371
print_anova, *13*, *429*, 627, *629*
print_summary, *13*, *429*, *628*, 628
probability_of_superiority_paired, *11*, *40*, *146*, *188*, *191*, *302*, *306*, *312*, 629, *636*, *653*, *1166*, *1178*, *1183*
probability_of_superiority_paired(), *186*, *191*, *239*, *344*, *348*, *351*, *355*, *359*, *562*, *570*, *636*, *727*, *805*
procrustes_phi, *12*, *299*, *365, 366*, 633
procrustes_phi(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208, 209*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *733*, *746*, *753*, *757*, *766*
proportion_of_superiority, *11*, *312*, *632*, 635, *726*, *1166*
proportion_of_superiority(), *186*, *191*, *239*, *344*, *348*, *351*, *355*, *359*, *562*, *570*, *632*, *727*, *805*
pt, *119*
ptukey, *49–52*, *108*, *180*, *223*, *271*
pwr.f2.test, *1081*
pygmalion, *13*, *98*, *105*, *569*, 637, *656*, *1155*, *1157*

qbryant_paulson, *12*, *87*, *257, 258*, *1158*
qbryant_paulson *(*bryant_paulson*)*, 49
qf, *265*
qr, *509*
qtukey, *49, 50*, *52*, *108*, *179*, *223*, *271*, *273*
quantile, *277*, *297*

R2_mixed_effects, *12*, *301*, 639, *643, 644*, *1172*
R2_mixed_effects(), *219*, *443*, *454*, *468*, *472*, *474*, *477*, *527*, *644*, *901*, *905*, *1098*, *1144*, *1186*
R2_mixed_effects_decomposition, *12*, 642
R2_mixed_effects_decomposition(), *454*, *477*, *527*, *642*, *901*, *905*, *1098*, *1144*
randomization_test, *12*, *186*, *189*, *191*, *299*, *306*, *595, 596*, 644, *1180*, *1182, 1183*
randomization_test(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *653*, *656*, *764*, *1154*, *1215*
randomization_test_paired, *12*, *40*, *299*, *306*, *631, 632*, *649*, 650
randomization_test_paired(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *656*, *764*, *1154*, *1215*
regions_of_significance, *12*, *592*, *597, 598*, 653, *1193*
regions_of_significance(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*,

527, *566*, *576*, *649*, *653*, *764*, *1154*, *1215*
reliability, *12*, *238, 239*, *300*, *518*, *520*, 657, *675*, *684*, *689*, *705*, *721*, *956*, *965*, *1172*, *1187, 1188*
reliability(), *199*, *292*, *427*, *453*, *675*, *684*, *689*, *705*, *721*
reliability_alpha, *12*, *249*, *289*, *300*, *657–659*, *661, 662*, 662, *685–689*, *692*, *705*, *719*, *956*, *961, 962*, *968–971*, *973*, *1172*, *1187, 1188*
reliability_alpha(), *199*, *292*, *427*, *453*, *662*, *684*, *689*, *705*, *721*
reliability_H, *12*, *46*, *49*, *68*, *70*, *300*, *307, 308*, *381*, *389*, *395*, *397*, *399*, *404*, *411*, *415*, *424*, *634*, *657, 658*, 676, *698*, *742*
reliability_H(), *199*, *292*, *427*, *453*, *662*, *675*, *689*, *705*, *721*
reliability_kr20, *12*, *300*, *657*, *662*, *671*, *675*, 685, *721*, *1172*
reliability_kr20(), *199*, *292*, *427*, *453*, *662*, *675*, *684*, *705*, *721*
reliability_omega, *12*, *31, 32*, *45*, *47*, *49*, *56, 57*, *66*, *68*, *70*, *238*, *249*, *289*, *300*, *307*, *381*, *389*, *396*, *399, 400*, *404*, *424*, *450*, *463*, *494*, *657–659*, *661, 662*, *665–667*, *670–672*, *675, 676*, *678*, *681–684*, 690, *709, 710*, *713*, *719*, *721*, *830*, *956*, *961, 962*, *968–971*, *973*, *1172*
reliability_omega(), *199*, *292*, *427*, *453*, *662*, *675*, *684*, *689*, *721*
reliability_omega_categorical, *12*, *47*, *56*, *68*, *300*, *394, 395*, *459*, *657–659*, *662*, *667*, *670*, *672*, *675*, *687*, *689*, *695, 696*, *702*, *705*, 707, *1172*
reliability_omega_categorical(), *199*, *292*, *427*, *453*, *662*, *675*, *684*, *689*, *705*
responder_analysis, *11*, *126*, *128*, *560, 561*, *723*
responder_analysis(), *186*, *191*, *239*, *344*, *348*, *351*, *355*, *359*, *562*, *570*, *632*, *636*, *805*
results_sentence, *13*, *72*, *75*, *81*, *97*, *99*, *103*, *111*, *122, 123*, *129*, *133*, *137, 138*, *143, 144*, *148*, *151*, *154*, *157*, *164*, *166*, *169*, *171*, *174*, *177*, *186*, *189*, *239*, *328*, *331*, *334*, *340*, *534*, *541*, *560*, *575*, *580*, *631*, *636*, *1166*, *1170*, *1172*, *1182, 1183*
results_sentence *(*dmar_output_helpers*)*, 310

sd, *728*
sd_unbiased, *11*, *547*, 727
sem, *251*, *535*, *539*, *541*, *816*, *976, 977*, *979*, *1007, 1008*, *1011*, *1013*, *1076*
sem_exact_fit_bootstrap, *12*, *64*, *299*, *301*, *303, 304*, *308*, *665*, *678*, *686*, *693*, *709*, 729, *742*
sem_exact_fit_bootstrap(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *210*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *746*, *753*, *757*, *766*
sem_fit_measures, *12*, *56*, *61*, *66*, *154, 155*, *214, 215*, *302*, *304*, *308*, *310–312*, *321*, *336, 337*, *368, 369*, *384*, *394*, *401*, *408, 409*, *412, 413*, *418*, *456*, *489*, *732, 733*, 734, *748*, *753–755*, *757*, *977*, *979*, *1166*
sem_fit_measures(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *210*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *753*, *757*, *766*
sem_information_criteria, *12*, *212–215*, *302*, *304*, *368, 369*, *371*, *398*, *742*, *746*, 747
sem_information_criteria(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *210*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *757*, *766*
sem_residual_fit, *12*, *304*, *738*, *742*, *746*, 754
sem_residual_fit(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *210*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *766*
sem_sam, *12*, *305*, 758
signal_to_noise_R2, *11*, 759
simple_effects_AB, *12*, 761, *1166*
simple_effects_AB(), *16*, *22*, *26*, *90*, *93*,

109, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *1154*, *1215*
simple_structure, *12*, 765
simple_structure(), *33*, *49*, *57*, *62*, *70*, *95*, *206*, *208*, *210*, *215*, *295*, *322*, *337*, *366*, *381*, *389*, *404*, *424*, *451*, *459*, *463*, *484*, *494*, *634*, *733*, *746*, *753*, *757*
simulate_ancova_data, *13*, 767, *771*, *774*, *797*
simulate_ancova_data(), *772*, *774*, *777*, *780*, *784*, *790*, *794*, *797*
simulate_ancova_factorial_data, *13*, 769
simulate_ancova_factorial_data(), *769*, *774*, *777*, *780*, *784*, *790*, *794*, *797*
simulate_anova_data, *13*, *771*, 773, *797*
simulate_anova_data(), *769*, *772*, *777*, *780*, *784*, *790*, *794*, *797*
simulate_longitudinal_gompertz, *13*, *19*, 774, *780*, *784*, *794*
simulate_longitudinal_gompertz(), *769*, *772*, *774*, *780*, *784*, *790*, *794*, *797*
simulate_longitudinal_logistic, *13*, *19*, *777*, 778, *784*, *794*
simulate_longitudinal_logistic(), *769*, *772*, *774*, *777*, *784*, *790*, *794*, *797*
simulate_longitudinal_negative_exponential, *13*, *19*, *777*, *780*, 782, *794*
simulate_longitudinal_negative_exponential(), *769*, *772*, *774*, *777*, *780*, *790*, *794*, *797*
simulate_longitudinal_polynomial, *13*, *17*, *19*, *776, 777*, *779, 780*, *783, 784*, 785, *793, 794*
simulate_longitudinal_polynomial(), *769*, *772*, *774*, *777*, *780*, *784*, *794*, *797*
simulate_longitudinal_richards, *13*, *19*, *776, 777*, *780*, *784*, 792
simulate_longitudinal_richards(), *769*, *772*, *774*, *777*, *780*, *784*, *790*, *797*
simulate_regression_data, *13*, *771*, *774*, 796
simulate_regression_data(), *769*, *772*, *774*, *777*, *780*, *784*, *790*, *794*
skewness, *11*, *276*, *469*, 798
skewness(), *277*, *469*
smd, *11*, *168*, *170*, *186*, *195*, *228*, *235*, *294, 295*, *306, 307*, *316*, *340*, *358, 359*, *521, 522*, *548*, *561*, *601*, *636*, *648, 649*, 799, *803*, *805*, *1019*, *1141*, *1154*, *1183*, *1212*, *1215*
smd_c, *11*, *168*, *170*, *306, 307*, *801*, 802, *1019*
smd_trimmed, *11*, 804, *1213*
smd_trimmed(), *186*, *191*, *239*, *344*, *348*, *351*, *355*, *359*, *562*, *570*, *632*, *636*, *727*
ss_aipe_c, *11*, *120*, *610, 611*, 806, *856*, *860*, *862*, *893*, *985*, *997*, *1037*, *1049*, *1085*, *1087*, *1147*, *1149*
ss_aipe_c(), *892*
ss_aipe_c_ancova, *11*, *769*, *808*, 854, *857*, *1084*
ss_aipe_c_ancova_sensitivity, *11*, 856
ss_aipe_c_sensitivity, *11*, *806, 807*, 859, *897*
ss_aipe_c_sensitivity(), *810*, *814*, *818*, *841*, *845*, *864*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1046*
ss_aipe_cliff_delta, *11*, *191*, 809, *812*, *814*
ss_aipe_cliff_delta(), *814*, *818*, *841*, *845*, *862*, *864*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1046*
ss_aipe_cliff_delta_sensitivity, *11*, *189*, *307*, 811, *1182*
ss_aipe_cliff_delta_sensitivity(), *810*, *818*, *841*, *845*, *862*, *864*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1046*
ss_aipe_composite_sem, *11*, *250*, *307*, 814, *885, 886*, *1077, 1078*
ss_aipe_composite_sem(), *810*, *814*, *841*, *845*, *862*, *864*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1046*
ss_aipe_crd, 819, *835*, *898*
ss_aipe_crd_both_fixed_budget, *11*
ss_aipe_crd_both_fixed_budget

(ss_aipe_crd), 819
ss_aipe_crd_both_fixed_width, *11*
ss_aipe_crd_both_fixed_width (ss_aipe_crd), 819
ss_aipe_crd_es, *303*, 826
ss_aipe_crd_es_both_fixed_budget, *11*
ss_aipe_crd_es_both_fixed_budget (ss_aipe_crd_es), 826
ss_aipe_crd_es_both_fixed_width, *11*
ss_aipe_crd_es_both_fixed_width (ss_aipe_crd_es), 826
ss_aipe_crd_es_n_clusters_fixed_budget, *11*
ss_aipe_crd_es_n_clusters_fixed_budget (ss_aipe_crd_es), 826
ss_aipe_crd_es_n_clusters_fixed_width, *11*
ss_aipe_crd_es_n_clusters_fixed_width (ss_aipe_crd_es), 826
ss_aipe_crd_es_n_individuals_fixed_budget, *11*
ss_aipe_crd_es_n_individuals_fixed_budget (ss_aipe_crd_es), 826
ss_aipe_crd_es_n_individuals_fixed_width, *11*
ss_aipe_crd_es_n_individuals_fixed_width (ss_aipe_crd_es), 826
ss_aipe_crd_n_clusters_fixed_budget, *11*
ss_aipe_crd_n_clusters_fixed_budget (ss_aipe_crd), 819
ss_aipe_crd_n_clusters_fixed_width, *11*
ss_aipe_crd_n_clusters_fixed_width (ss_aipe_crd), 819
ss_aipe_crd_n_individuals_fixed_budget, *11*
ss_aipe_crd_n_individuals_fixed_budget (ss_aipe_crd), 819
ss_aipe_crd_n_individuals_fixed_width, *11*
ss_aipe_crd_n_individuals_fixed_width (ss_aipe_crd), 819
ss_aipe_crossed, *11*, *78, 79*, 837, *842–845*
ss_aipe_crossed(), *810*, *814*, *818*, *845*, *862*, *864*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1046*
ss_aipe_crossed_sensitivity, *11*, *840, 841*, 842
ss_aipe_crossed_sensitivity(), *810*, *814*, *818*, *841*, *862*, *864*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1046*
ss_aipe_cv, *11*, *254*, *306*, *550*, *553*, 846, *851–853*, *1188*, *1190*
ss_aipe_cv_sensitivity, *11*, *303*, *307, 308*, *846*, *848, 849*, 850, *1190*
ss_aipe_equivalence_r, *11*, 863, *866, 867*
ss_aipe_equivalence_r(), *810*, *814*, *818*, *841*, *845*, *862*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1046*
ss_aipe_equivalence_r_sensitivity, *11*, *863, 864*, 865
ss_aipe_equivalence_r_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *864*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_equivalence_smd, *11*, *339, 340*, *864*, 868, *873*, *875*
ss_aipe_equivalence_smd(), *811*, *814*, *818*, *841*, *845*, *862*, *864*, *867*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_equivalence_smd_sensitivity, *11*, *339*, *867*, *871*, 872
ss_aipe_equivalence_smd_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *864*, *867*, *871*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *981*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_icc, *11*, *283*, *454*, 875, *879–881*, *901*, *1186*
ss_aipe_icc(), *811*, *814*, *818*, *841*, *845*, *862*, *864*, *867*, *871*, *875*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_icc_sensitivity, *11*, *876, 877*,

878
ss_aipe_icc_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *864*, *867*, *871*, *875*, *878*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_indirect_effect, *11*, *299*, *301*, *496*, *500*, *511*, 882, *887, 888*, *890*, *1093*, *1095*, *1197, 1198*
ss_aipe_indirect_effect(), *811*, *814*, *818*, *841*, *845*, *862*, *864*, *867*, *871*, *875*, *878*, *881*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_indirect_effect_sensitivity, *11*, *299*, *301*, *885, 886*, 887
ss_aipe_indirect_effect_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *864*, *867*, *871*, *875*, *878*, *881*, *886*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_mean, *11*, 890, *894, 895*, *897*
ss_aipe_mean(), *811*, *814*, *818*, *841*, *845*, *862*, *864*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_mean_sensitivity, *11*, *893*, 894
ss_aipe_mean_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_mixed_effects, *11*, 898, *902–905*
ss_aipe_mixed_effects(), *454*, *477*, *527*, *618*, *642*, *644*, *905*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097, 1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143, 1144*
ss_aipe_mixed_effects_sensitivity, *11*, *898*, *900, 901*, 902, *925*
ss_aipe_mixed_effects_sensitivity(), *454*, *477*, *527*, *642*, *644*, *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *901*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*, *1098*, *1144*
ss_aipe_omega_squared, *11*, 906, *908–910*, *1200*
ss_aipe_omega_squared(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_omega_squared_sensitivity, *11*, 908
ss_aipe_omega_squared_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_partial_r, *11*, *351*, *810*, *886*, 911, *916, 917*, *919*, *927–929*, *1000*, *1002*
ss_aipe_partial_r(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_partial_r_sensitivity, *11*, *911*, *914, 915*, 916, *981*, *1006*
ss_aipe_partial_r_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *881*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_pcm, *11*, 920, *923–925*, *1101, 1102*
ss_aipe_pcm_sensitivity, *11*, *922*, 923
ss_aipe_pcm_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_r, *11*, *77*, *120*, *864*, *914, 915*, 926, *979–981*
ss_aipe_r(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *973*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_R2, *11*, *77*, *117*, *135*, *232*, *308*, *357*,

761, *797*, *907, 908*, 929, *937*, *1002*, *1110*, *1176*, *1206*
ss_aipe_R2_sensitivity, *11*, *307*, *930–932*, 933, *1117*, *1120*
ss_aipe_r_sensitivity, *11*, *867*, *917*, *926*, *928, 929*, 979
ss_aipe_r_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_rc, *11*, *120*, *886*, 937, *940*, *952*, *1027, 1028*, *1031*, *1121*, *1125*
ss_aipe_rc_sensitivity, *11*, 940, *952*, *1032*, *1036*
ss_aipe_reg_coef, *11*, *149*, *153*, *176*, *797*, *938, 939*, *942, 943*, 944, *953–955*, *983*, *999*, *1017*, *1028–1031*, *1036*, *1121*, *1123*, *1125*, *1127*
ss_aipe_reg_coef_sensitivity, *11*, *939–943*, *948*, *950*, 951, *1032*, *1036*, *1129*, *1132*
ss_aipe_reliability, *11*, *245*, *432*, *438*, *662*, *675*, 956, *967–970*, *973*, *1187, 1188*, *1196*
ss_aipe_reliability_sensitivity, *11*, *299*, *301*, *307*, *963–965*, 967
ss_aipe_reliability_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *982*, *1002*, *1006*, *1039*, *1043*, *1047*
ss_aipe_rmsea, *11*, *154, 155*, *742*, 974, *976, 977*, *979*
ss_aipe_rmsea_sensitivity, *11*, *250, 251*, *308*, 975
ss_aipe_sc, *11*, *808*, 982, *988*, *990*, *995*, *997*, *1136*
ss_aipe_sc_ancova, *11*, *22*, *938*, *983*, 986, *991, 992*, *994*, *1017*
ss_aipe_sc_ancova_sensitivity, *11*, *307*, *989, 990*, 991
ss_aipe_sc_sensitivity, *11*, *862*, *994*, 994
ss_aipe_sem_path, *11*, *250, 251*, *308*, *432*, *816–818*, 1007, *1010, 1011*, *1013*, *1078*
ss_aipe_sem_path_sensitivity, *11*, *250, 251*, *308*, *818*, *1008*, 1009
ss_aipe_semipartial_r, *11*, *306*, *810*, *886*, *915*, 998, *1003–1006*, *1114*, *1116*, *1208, 1209*
ss_aipe_semipartial_r(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *910*, *915*, *919*, *925*, *929*, *973*, *982*, *1006*, *1039*, *1043*, *1047*
ss_aipe_semipartial_r_sensitivity, *11*, *306*, *919*, *998*, *1000, 1001*, 1003, *1208*
ss_aipe_semipartial_r_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *911*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1039*, *1043*, *1047*
ss_aipe_sm, *12*, *893*, 1014, *1024*, *1026*
ss_aipe_sm_sensitivity, *12*, 1023
ss_aipe_smd, *12*, *168*, *235*, *280*, *316*, *359*, *801*, *870, 871*, 1016, *1020, 1021*, *1023*, *1140, 1141*, *1211–1213*
ss_aipe_smd_sensitivity, *12*, *874, 875*, 1020
ss_aipe_src, *12*, *938, 939*, *944, 945*, *947*, *949*, *983*, *1017*, 1027, *1033, 1034*, *1036*
ss_aipe_src_sensitivity, *12*, *940*, *943*, *952–954*, *1031*, 1032, *1131*
ss_aipe_two_phase, *12*, *184*, 1036, *1040, 1041*, *1043–1046*, *1150*, *1152*
ss_aipe_two_phase(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *911*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1043*, *1047*
ss_aipe_two_phase_cost, *12*, *184*, *307*, *1039*, 1039, *1046*
ss_aipe_two_phase_cost(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*, *871*, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *911*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1047*
ss_aipe_two_phase_sensitivity, *12*, *184*, *1039*, *1043*, 1043
ss_aipe_two_phase_sensitivity(), *811*, *814*, *818*, *841*, *845*, *862*, *865*, *867*,

871, *875*, *878*, *882*, *886*, *890*, *893*, *897*, *905*, *908*, *911*, *915*, *919*, *925*, *929*, *973*, *982*, *1002*, *1006*, *1039*, *1043*
ss_power_c, *12*, 1047, *1058*, *1084*, *1092*, *1100*, *1136*, *1167*
ss_power_c(), *618*, *901*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_c_ancova, *12*, *1049*, *1055*, *1057*, *1058*, 1083, *1167*
ss_power_c_ancova(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_composite_ancova, *12*, 1049, *1054*, *1058*, *1061*, *1062*
ss_power_composite_ancova(), *618*, *901*, *1049*, *1058*, *1062*, *1066*, *1071*, *1073*, *1074*, *1078*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_composite_ancova_2group, *12*, *1051*, 1053, *1066*, *1068*, *1069*, *1071*
ss_power_composite_ancova_2group(), *618*, *901*, *1049*, *1052*, *1062*, *1066*, *1071*, *1073*, *1074*, *1078*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_composite_anova, *12*, *1051*, 1060, *1078*
ss_power_composite_anova(), *618*, *901*, *1049*, *1052*, *1058*, *1066*, *1071*, *1073*, *1074*, *1078*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_composite_factorial_ancova, *12*, *1051*, 1063, *1069–1073*
ss_power_composite_factorial_ancova(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1071*, *1073*, *1074*, *1078*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_composite_factorial_ancova_het, *12*, *1051*, *1052*, 1067
ss_power_composite_factorial_ancova_het(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1073*, *1074*, *1078*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_composite_factorial_anova, *12*, *1064*, *1066*, 1072
ss_power_composite_factorial_anova(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1078*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_composite_sem, *12*, *250*, *308*, *814*, *818*, 1075, *1168*
ss_power_composite_sem(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1074*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_contrast, *12*, *774*, *806*, 1079, *1086*, *1087*, *1099*, *1167*
ss_power_contrast(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_equivalence_c, *12*, *328*, *610*, *611*, 1085, *1147*, *1149*, *1167*
ss_power_equivalence_c(), *328*, *332*, *334*, *337*, *340*, *583*, *608*, *611*, *613*, *616*, *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*,

1132, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_factorial_ancova, *12*, *1052*, *1065*, *1066*, 1088, *1091*, *1167*
ss_power_factorial_ancova(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_factorial_anova, *12*, *764*, *1062*, *1065*, *1066*, *1073*, *1088–1090*, 1090, *1100*, *1143*, *1167*
ss_power_factorial_anova(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_indirect_effect, *12*, *496*, *500*, *511*, 1093, *1167*, *1168*
ss_power_indirect_effect(), *500*, *511*, *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_mixed_effects, *12*, *642*, *901*, 1096, *1143*, *1167*
ss_power_mixed_effects(), *454*, *477*, *527*, *618*, *642*, *644*, *901*, *905*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*, *1144*
ss_power_one_way_anova, *12*, *194*, *1049*, *1062*, *1092*, 1098, *1134*, *1136*, *1143*, *1167*
ss_power_one_way_anova(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_pcm, *12*, *785*, *788–790*, *922*, *1097*, 1100, *1134*
ss_power_pcm(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_r, *12*, *77*, *929*, 1104, *1167*
ss_power_r(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_R2, *12*, *194*, *307*, *308*, *350*, *912*, *916*, *998*, *1004*, 1107, *1111–1114*, *1116*, *1118*, *1120*, *1123*, *1127*, *1167*, *1201*, *1207*
ss_power_R2(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_R2_change, *12*, *306–308*, *1002*, *1107*, *1108*, *1110*, 1111, *1122*, *1123*, *1126*, *1127*, *1207*, *1209*
ss_power_R2_change(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_R2_sensitivity, *12*, *1107*, *1110*, 1117, *1167*
ss_power_R2_sensitivity(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1097*, *1100*, *1103*, *1106*, *1110*, *1116*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_rc, *12*, *307*, 1120
ss_power_rc(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1078*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_reg_coef, *12*, *307*, *1055*, *1058*, *1082*, *1110*, *1111*, *1113*, *1114*, *1116*, *1122*, 1124, *1131*, *1132*, *1167*

ss_power_reg_coef(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1073*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1132*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_reg_coef_sensitivity, *12*, 1129, *1167*
ss_power_reg_coef_sensitivity(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1074*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1134*, *1136*, *1138*, *1141*, *1143*
ss_power_rm_anova, *12*, 1133, *1143*, *1167*
ss_power_rm_anova(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1074*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1136*, *1138*, *1141*, *1143*
ss_power_sc, *12*, *1049*, *1100*, 1135, *1167*
ss_power_sc(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1074*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1138*, *1141*, *1143*
ss_power_sem, *12*, *742*, *1076*, *1078*, 1137, *1167*
ss_power_sem(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1074*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1141*, *1143*
ss_power_smd, *12*, *168*, *280*, *316*, *801*, *1016*, *1058*, *1097*, 1139, *1167*
ss_power_smd(), *618*, *901*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1074*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1143*
ss_power_split_plot_anova, *12*, *1097*, 1141, *1167*
ss_power_split_plot_anova(), *454*, *477*, *527*, *618*, *642*, *644*, *901*, *905*, *1049*, *1052*, *1058*, *1062*, *1066*, *1071*, *1074*, *1079*, *1082*, *1084*, *1087*, *1090*, *1092*, *1095*, *1098*, *1100*, *1103*, *1106*, *1110*, *1116*, *1120*, *1123*, *1127*, *1132*, *1134*, *1136*, *1138*, *1141*
ss_seq_c, *12*, 1144, *1147–1152*
ss_seq_c(), *1149*, *1152*
ss_seq_c_sensitivity, *12*, *1146*, *1147*, 1147
ss_seq_c_sensitivity(), *1147*, *1152*
ss_seq_two_phase, *12*, *184*, *1039*, *1146*, *1147*, 1150
ss_seq_two_phase(), *1147*, *1149*
stats::pchisq(), *115*
stats::pf(), *117*
stats::power.t.test(), *1019*
stats::pt(), *121*
stats::qchisq(), *115*
stats::qf(), *117*
stats::qt(), *121*
summary.lm, *1175*
summary_t_test, *12*, *140*, 1152, *1155*, *1215*
summary_t_test(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1215*
summary_t_test_broom, 1154
Sys.time, *563*

t.test, *91*, *108*, *645*, *649*, *653*, *1154*, *1215*
teacher_expectancy, *13*, *203*, *522*, *587*, 1155
test_market, *13*, 1157
tidy, *92*, *99*, *109*, *123*, *223*, *343*, *569*, *824*, *832*, *1048*, *1051*, *1056*, *1061*, *1065*, *1069*, *1073*, *1077*, *1081*, *1084*, *1087*, *1089*, *1091*, *1094*, *1097*, *1099*, *1115*, *1119*, *1122*, *1131*, *1134*, *1136*, *1143*, *1161*
tidy.dmar_anova_es *(*tidy.dmar_cliff_delta*)*, 1161
tidy.dmar_ci_anova *(*tidy.dmar_cliff_delta*)*, 1161
tidy.dmar_ci_bifactor *(*ci_bifactor_indices*)*, 62
tidy.dmar_ci_long *(*tidy.dmar_cliff_delta*)*, 1161
tidy.dmar_ci_R2, 1159

tidy.dmar_ci_ratio_of_means (tidy.dmar_cliff_delta), 1161
tidy.dmar_ci_smd, 1160
tidy.dmar_cliff_delta, 1161
tidy.dmar_common_method_adjusted_r (common_method_adjusted_r), 203
tidy.dmar_compare_sem (compare_sem), 212
tidy.dmar_content_validity (tidy.dmar_cliff_delta), 1161
tidy.dmar_contrast_test (tidy.dmar_cliff_delta), 1161
tidy.dmar_df_barnard_rubin (tidy.dmar_cliff_delta), 1161
tidy.dmar_df_missing (tidy.dmar_cliff_delta), 1161
tidy.dmar_dmacs (tidy.dmar_cliff_delta), 1161
tidy.dmar_fa_bifactor (fa_bifactor), 382
tidy.dmar_fa_confirmatory, 1169
tidy.dmar_fa_second_order (fa_second_order), 415
tidy.dmar_factor_analysis (tidy.dmar_cliff_delta), 1161
tidy.dmar_factor_congruence (tidy.dmar_cliff_delta), 1161
tidy.dmar_factor_count (tidy.dmar_cliff_delta), 1161
tidy.dmar_factor_scores (factor_scores), 371
tidy.dmar_fmi, *437*, 1170
tidy.dmar_measurement_alignment (tidy.dmar_cliff_delta), 1161
tidy.dmar_measurement_invariance (tidy.dmar_cliff_delta), 1161
tidy.dmar_mediation_mbco, 1171
tidy.dmar_post_hoc_ci (tidy.dmar_cliff_delta), 1161
tidy.dmar_probability_of_superiority_paired (tidy.dmar_cliff_delta), 1161
tidy.dmar_R2_mixed_effects, 1171
tidy.dmar_reliability, 1172
tidy.dmar_ss_aipe (tidy.dmar_cliff_delta), 1161
tidy.dmar_ss_power (tidy.dmar_cliff_delta), 1161
tidy.dmar_ss_power_sensitivity (tidy.dmar_cliff_delta), 1161
tidy.dmar_summary_t_test (summary_t_test_broom), 1154
tidy.dmar_tbl (tidy.dmar_cliff_delta), 1161
tidy.dmar_vargha_delaney_A (tidy.dmar_cliff_delta), 1161
tidy.dmar_welch_t (welch_t_broom), 1216
tidy.mlmr, 1173
tidy.mlmr_mv, 1174
TukeyHSD, *180*, *224*, *273*

unbiased_R2, *11*, 1175
uniroot, *51*, *111*, *114–117*, *119*, *121*, *133*, *262*, *267*, *788*

var.test, *566*
var_alpha, *13*, 1186
var_alpha(), *1190*, *1193*, *1198*, *1200*, *1204*, *1212*, *1213*
var_cv, *13*, *253*, *254*, 1188
var_cv(), *1188*, *1193*, *1198*, *1200*, *1204*, *1212*, *1213*
var_ete, *13*, 1191
var_ete(), *1188*, *1190*, *1198*, *1200*, *1204*, *1212*, *1213*
var_icc, *13*, *283*, *453*, *454*, *877*, *878*, *881*, *901*, *1186*, 1194
var_indirect_effect, *13*, *498*, *500*, *511*, *890*, 1196
var_indirect_effect(), *1188*, *1190*, *1193*, *1200*, *1204*, *1212*, *1213*
var_omega_squared, *13*, *570*, 1199
var_omega_squared(), *1188*, *1190*, *1193*, *1198*, *1204*, *1212*, *1213*
var_partial_r, *13*, *351*, *912*, *915*, *917*, *1112*, 1201, *1204*, *1208*, *1209*
var_r, *13*, *77*, 1203
var_r(), *1188*, *1190*, *1193*, *1198*, *1200*, *1212*, *1213*
var_R2, *13*, *357*, *1176*, *1196*, *1203*, 1205, *1209*
var_semipartial_r, *13*, *306*, *999*, *1000*, *1002*, *1003*, *1112*, *1116*, *1203*, *1204*, 1206
var_smd, *13*, *548*, 1210, *1213*
var_smd(), *1188*, *1190*, *1193*, *1198*, *1200*, *1204*, *1213*
var_smd_trimmed, *13*, *805*, 1212
var_smd_trimmed(), *1188*, *1190*, *1193*, *1198*, *1200*, *1204*, *1212*
VarCorr, *640*, *642*

vargha_delaney_A, *11*, *188*, *191*, *302*, *306*, *312, 313*, *558*, *561*, *629, 630*, *632*, *1165*, 1177
variance_components_mls, *12*, *79*, *453, 454*, 1184
variance_components_mls(), *219*, *443*, *454*, *468*, *472*, *474*, *642*
vcov, *541*

welch_t, *12*, *35*, *140*, *1153, 1154*, 1214, *1217*
welch_t(), *16*, *22*, *26*, *90*, *93*, *109*, *159*, *180*, *212*, *224*, *243*, *332*, *334*, *337*, *340*, *362*, *477*, *479*, *527*, *566*, *576*, *649*, *653*, *656*, *764*, *1154*
welch_t_broom, 1216
wilcox.test, *649*
writeLines, *241*