Package 'findn'

February 3, 2026

Type Package

Title Simulation Based Sample Size Estimation

Version 0.1.0

Maintainer Lukas Baumann <baumann@imbi.uni-heidelberg.de>

Description Estimates the sample size for a test or a trial based on repeated simulation using a model based approach.
Implements a method by Maruo et al. (2018) <doi:10.1080/19466315.2017.1349689> and an extension.

Imports ggplot2, rlang, scales

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Lukas Baumann [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7931-7470>),
Björn Bornkamp [ctb] (ORCID: <https://orcid.org/0000-0002-6294-8185>)

Repository CRAN

Date/Publication 2026-02-03 13:40:02 UTC

Contents

findn *Find the Sample Size for a trial based repeated simulation using a model based approach*

Description

findn estimates the sample size to achieve a pre-defined power, when the power can only be evaluated using simulations. findn uses a model-based approach for this purpose.

Usage

```
findn(
  fun,
  targ,
  start,
  k = 25,
  init_evals = 100,
  r = 4,
  stop = c("evals", "power_ci", "abs_unc", "rel_unc"),
  max_evals = 2000,
  level = 0.05,
  power_ci_tol = 0.02,
  abs_unc_tol = 10,
  rel_unc_tol = 0.1,
  var_a = 1,
  var_b = 1,
  alpha = 0.05,
  alternative = c("two.sided", "one.sided"),
  min_x = 2,
  verbose = FALSE,
  ...
)
```

Arguments

fun	A function that estimates the power of a trial. The function has to take at least two arguments: n, the sample size and k, the number of iterations.
targ	The target power.
start	An initial guess for the sample size.
k	Number of trial simulations to use in fun to estimate the power.
init_evals	How many evaluations the first model is based on.
r	A multiplicator for the range of the initial design points.
stop	The stopping criterion. One of "evals", "power_ci", "abs_unc", "rel_unc".
max_evals	The maximum number of simulations.

level	Significance level for the confidence intervals if stop is something other than "evals". Also used to determine the levels for the confidence intervals that are printed if verbose = TRUE.
power_ci_tol	Tolerance parameter if stop = "power_ci".
abs_unc_tol	Tolerance parameter if stop = "abs_unc".
rel_unc_tol	Tolerance parameter if stop is "rel_unc".
var_a	Variance of the prior distribution for the intercept.
var_b	Variance of the prior distribution for the slope.
alpha	The significance level of the underlying test. This is used to compute the mean of the prior distribution of the intercept.
alternative	Either "two.sided" or "one.sided". This is only used to determine the mean of the intercept prior.
min_x	The minimum sample size that fun can be evaluated for.
verbose	If TRUE, the current sample size estimate, the predicted power and its level percent confidence is returned after every iteration.
...	Further optional arguments.

Details

findn estimates the sample size for a target function that returns a simulated power value for a test or a trial. The target function must have at least two arguments, n, the sample size for which the trial is simulated, and k, that specifies how often the trial is simulated. Note that depending on how fun is written, n can either be the sample size per group or the total sample size. The function has to return an estimate for the power of the trial for the sample size n based on k Monte Carlo simulations.

findn uses an algorithm that assumes a probit model and computes Bayesian parameter estimates. The mean of the prior distribution of the intercept is computed from the significance level alpha of the underlying test and the alternative. The mean of the prior distribution of the slope is computed from the initial guess for the sample size - start. The variances of the prior distributions can be adjusted using the arguments var_a and var_b.

There are four different stopping criteria. When stop = "evals" the algorithm stops when the target function was evaluated max_evals times. When stop = "power_ci"the algorithm stops when the level percent confidence interval of the predicted power at the current sample size estimate is within the interval targ plus and minus power_ci_tol. When stop = "abs_unc" the algorithm stops when the number of sample sizes in the uncertainty set smaller than abs_unc_tol. The uncertainty set is defined as the set that contains all sample sizes for which the level percent confidence interval for the predicted power contains targ. When stop = "rel_unc" the algorithm stops when the relative uncertainty range is smaller than rel_unc_tol. The relative uncertainty range is defined as the greatest integer in the uncertainty set minus the smallest integer in the uncertainty set, divided by the smallest number in the uncertainty set. The algorithm also stops when stop is either "power_ci", "abs_unc" or "rel_unc" and the stopping criterion couldn't be satisfied within max_evals evaluations.

Value

findn returns an object of class findn. By default, a list containing the point estimate for the sample size, the minimum sufficient sample size (i.e. the smallest sample size for which the lower limit of the confidence interval for the estimated power is larger than the target power) and a message whether the stopping criterion was reached is printed. See print.findn for details.

Examples

```
# Function that simulates the outcomes of a two-sample t-test
ttest <- function(n, k, mu1 = 0, mu2 = 1, sd = 2) {
  sample1 <- matrix(rnorm(n = ceiling(n) * k, mean = mu1, sd = sd),
    ncol = k)
  mean1 <- apply(sample1, 2, mean)
  sd1_hat <- apply(sample1, 2, sd)
  sample2 <- matrix(rnorm(n = ceiling(n) * k, mean = mu2, sd = sd),
    ncol = k)
  mean2 <- apply(sample2, 2, mean)
  sd2_hat <- apply(sample2, 2, sd)
  sd_hat <- sqrt((sd1_hat^2 + sd2_hat^2) / 2)
  teststatistic <- (mean1 - mean2) / (sd_hat * sqrt(2 / n))
  crit <- qt(1 - 0.025, 2 * n - 2)
  return(mean(teststatistic < -crit))
}

findn(fun = ttest, targ = 0.8, k = 25, start = 100,
  init_evals = 100, r = 4, stop = "evals", max_evals = 2000,
  level = 0.05, var_a = 1, var_b = 0.1, alpha = 0.025,
  alternative = "one.sided", verbose = FALSE)
```

---

findn_maruo *Find the Sample Size Using the Algorithm by Maruo et al.*

Description

findn_maruo estimates the sample size for a certain target function based on repeated simulations using a model based approach proposed by Maruo et al. (2018).

Usage

```
findn_maruo(fun, targ, start = 10, k = 100, ...)
```

Arguments

fun	A function that estimates the power of a trial. The function has to take at least two arguments: n, the sample size and k, the number of iterations.
targ	The target power. Must be either 0.8 or 0.9.
start	Starting value for the algorithm. Maruo et al. suggest to use 10.
k	Number of trial simulations to use in fun to estimate the power.
...	Further optional arguments.

Value

findn_maruo returns a list containing the point estimate for the sample size and a list of all sample sizes that for which the trial function was evaluated.

References

Maruo, K., Tada, K., Ishil, R. and Gosho M. (2018) An Efficient Procedure for Calculating Sample Size Through Statistical Simulations, Statistics in Biopharmaceutical Research 10, 1-8.

Examples

```
# Function that simulates the outcomes of a two-sample t-test
ttest <- function(n, k, mu1 = 0, mu2 = 1, sd = 2) {
  sample1 <- matrix(rnorm(n = ceiling(n) * k, mean = mu1, sd = sd),
    ncol = k)
  mean1 <- apply(sample1, 2, mean)
  sd1_hat <- apply(sample1, 2, sd)
  sample2 <- matrix(rnorm(n = ceiling(n) * k, mean = mu2, sd = sd),
    ncol = k)
  mean2 <- apply(sample2, 2, mean)
  sd2_hat <- apply(sample2, 2, sd)
  sd_hat <- sqrt((sd1_hat^2 + sd2_hat^2) / 2)
  teststatistic <- (mean1 - mean2) / (sd_hat * sqrt(2 / n))
  crit <- qt(1 - 0.025, 2 * n - 2)
  return(mean(teststatistic < -crit))
}

findn_maruo(fun = ttest, targ = 0.8)
```

---

plot.findn *Plot of a findn Object*

Description

Plot of a findn Object

Usage

```
## S3 method for class 'findn'
plot(x, min_n = 1, max_n = NULL, power_lim = 0.95, ...)
```

Arguments

x	object of class findn.
min_n	lower limit of the x-axis.
max_n	upper limit of the x-axis. The default is NULL.

power_lim	if max_n is NULL then the upper limit of the x-axis is the smallest sample size for which the lower limit of the level percent confidence interval for the predicted power exceeds the value of power_lim. The default is 0.95.
...	Further arguments.

Value

None.

Examples

```
# Function that simulates the outcomes of a two-sample t-test
ttest <- function(mu1 = 0, mu2 = 1, sd, n, k) {
  sample1 <- matrix(rnorm(n = ceiling(n) * k, mean = mu1, sd = sd),
    ncol = k)
  mean1 <- apply(sample1, 2, mean)
  sd1_hat <- apply(sample1, 2, sd)
  sample2 <- matrix(rnorm(n = ceiling(n) * k, mean = mu2, sd = sd),
    ncol = k)
  mean2 <- apply(sample2, 2, mean)
  sd2_hat <- apply(sample2, 2, sd)
  sd_hat <- sqrt((sd1_hat^2 + sd2_hat^2) / 2)
  teststatistic <- (mean1 - mean2) / (sd_hat * sqrt(2 / n))
  crit <- qt(1 - 0.025, 2*n - 2)
  return(mean(teststatistic < -crit))
}

# Create a findn object
res.ttest <- findn(fun = ttest, targ = 0.8, k = 25, start = 100,
  init_evals = 100, r = 4, stop = "evals", max_evals = 2000,
  level = 0.05, var_a = 0.05, var_b = 1, alpha = 0.025,
  alternative = "one.sided", sd = 2, verbose = FALSE)

# plot with default settings
plot(res.ttest, power_lim = 0.95)
```

---

print.findn *Printing a findn Object*

Description

Displays details about a sample size estimation from a findn object.

Usage

```
## S3 method for class 'findn'
print(
  x,
  details = c("low", "high"),
```

```
  max_n = NULL,
  digits = 3,
  invisible = FALSE,
  ...
)
```

Arguments

x	Object of class findn.
details	Either "low" (default) or "high". See also 'Details'.
max_n	If details = "high" the predicted power values and confidence intervals are shown for all sample sizes from 1 to max_n if max_n is non-NULL. See also 'Details'.
digits	Number of decimal places to be shown.
invisible	Whether the results should be printed or only assigned.
...	Further arguments.

Details

When details = "low", only the point estimate (i.e., the smallest sample for which the predicted power exceeds the target power), the "minimum sufficient sample size" (i.e., the smallest sample size for which the lower limit of the level interval for the predicted power exceeds the target power) and an exit message. The exit message shows whether the chosen stopping rule was satisfied. If details = "high" then the default behaviour (i.e. when max_n = NULL) is to display all sample sizes, their predicted power values and the alpha whether their power exceeds the target power, and the three largest sample sizes that are smaller than the smallest sample size that is rated uncertain and the three smallest sample sizes which are greater than the smallest sample size that is rated uncertain. If details = "high" and max_n is non-NULL, then the sample sizes, their predicted power values and the confidence intervals for the predicted power values from 1 to max_n are displayed.

Value

findn returns an object of class findn which contains the following elements:

sample_size	the sample size estimate
fit	the model coefficients and covariance matrix from the last Bayesian probit regression model
all_evals	all evaluated sample sizes
targ	the target power
level	the significance level for the confidence intervals used for the stopping criteria
exit.mes	a message about wheter the stopping criterion was reached with the number of simulations given by max_evals

By default, a list containing the point estimate for the sample size, the minimum sufficient sample size (i.e. the smallest sample size for which the lower limit of the confidence interval for the estimated power is larger than the target power) and a message whether the stopping criterion was reached is printed.

Examples

```
# Function that simulates the outcomes of a two-sample t-test
ttest <- function(mu1 = 0, mu2 = 1, sd, n, k) {
  sample1 <- matrix(rnorm(n = ceiling(n) * k, mean = mu1, sd = sd),
    ncol = k)
  mean1 <- apply(sample1, 2, mean)
  sd1_hat <- apply(sample1, 2, sd)
  sample2 <- matrix(rnorm(n = ceiling(n) * k, mean = mu2, sd = sd),
    ncol = k)
  mean2 <- apply(sample2, 2, mean)
  sd2_hat <- apply(sample2, 2, sd)
  sd_hat <- sqrt((sd1_hat^2 + sd2_hat^2) / 2)
  teststatistic <- (mean1 - mean2) / (sd_hat * sqrt(2 / n))
  crit <- qt(1 - 0.025, 2*n - 2)
  return(mean(teststatistic < -crit))
}

# Create a findn object
res_ttest <- findn(fun = ttest, targ = 0.8, k = 25, start = 100,
  init_evals = 100, r = 4, stop = "evals", max_evals = 2000,
  level = 0.05, var_a = 0.05, var_b = 1, alpha = 0.025,
  alternative = "one.sided", sd = 2, verbose = FALSE)

# print with default settings
print(res_ttest, details = "low", digits = 3)
```

Index