Package 'ForceChoice'

September 12, 2026

Title Forced-Choice Modeling Based on Item Response Theory and Cognitive Diagnostic Models

Version 1.0.1

Description Fits, simulates, and evaluates forced-choice and traditional item response theory (IRT) models for noncognitive assessment. Eight model families are supported, spanning dominance (multidimensional IRT (MIRT) 1PL--4PL; multidimensional generalized partial credit model (MGPCM)), ideal-point unfolding (multidimensional generalized graded unfolding model (MGGUM)), and forced-choice designs (forced-choice multidimensional IRT (FCMIRT), forced-choice generalized graded unfolding model (FCGGUM), Thurstonian IRT (TIRT), forced-choice diagnostic classification model (FCDCM), forced-choice generalized deterministic inputs, noisy ``and" gate model (FCGDINA)) that mitigate response biases such as acquiescence and social desirability. Core estimation backends include full Bayesian inference via Hamiltonian Monte Carlo (Stan) and a fast improved stochastic expectation-maximization (iStEM) algorithm suitable for large-scale data; FCGDINA also provides a deterministic expectation-maximization (EM) estimator. Comprehensive model evaluation uses the limited-information M2 family of goodness-of-fit statistics (Maydeu-Olivares and Joe, 2005 <doi:10.1198/016214504000002069>; 2006 <doi:10.1007/s11336-005-1295-9>) together with root mean square error of approximation (RMSEA), comparative fit index (CFI), Tucker-Lewis index (TLI), and standardized root mean square residual (SRMSR).

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation yes

Biarch true

Depends R (>= 4.1.0)

Imports coda, GPArotation, MASS, methods, numDeriv, parallel, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), rstantools (>= 2.7.1)

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppArmadillo, RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

Suggests knitr, loo, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Author Haijiang Qin [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0000-6721-5653>),
Lei Guo [aut, cph] (ORCID: <https://orcid.org/0000-0002-8273-3587>)

Maintainer Haijiang Qin <haijiang133@outlook.com>

Repository CRAN

Date/Publication 2026-09-12 17:10:02 UTC

Contents

`ForceChoice-package` *ForceChoice: Forced-Choice Modeling Based on IRT and CDM*

Description

The **ForceChoice** package provides a unified framework for fitting, simulating, and evaluating forced-choice and traditional item response models. It supports eight model families spanning dominance (MIRT), ideal-point/unfolding (MGGUM), polytomous (MGPCM), and forced-choice (FCMIRT, FCGGUM, TIRT, FCDCM, FCGDINA) paradigms, with full Bayesian estimation via Stan, fast improved stochastic EM (iStEM), and a deterministic EM backend for FCGDINA.

Model Families

Traditional (single-stimulus) models:

`fit.MIRT` Multidimensional IRT (1PL–4PL). Binary responses with optional Q-matrix structure.

`fit.MGPCM` Multidimensional Generalized Partial Credit Model. Polytomous ordered-category responses.

`fit.MGGUM` Multidimensional Generalized Graded Unfolding Model. Ideal-point polytomous responses.

Forced-choice (comparative) models:

`fit.FCMIRT` Forced-choice MIRT. Item-level dominance model with sequential ranking over item endorsement logits at the block level.

`fit.FCGGUM` Forced-choice GGUM. Item-level ideal-point model with sequential ranking over binary GGUM endorsement logits.

`fit.TIRT` Thurstonian IRT for forced-choice. Pairwise probit comparison of latent utility differences.

`fit.FCDCM` Forced-choice DCM. Higher-order cognitive diagnostic model with exact attribute-profile marginalization.

`fit.FCGDINA` Forced-choice GDINA. General CDM item-response structure with forced-choice block likelihood.

Estimation Methods

Model families support a common estimation interface:

Stan (`method = "stan"`)**:** Full Bayesian inference via Hamiltonian Monte Carlo (NUTS/HMC). Provides posterior means, standard deviations, and convergence diagnostics ($\hat{R}$).

iStEM (`method = "iStEM"`)**:** Improved Stochastic EM (iStEM) algorithm combining finite-grid block Gibbs person sampling with L-BFGS-B item-parameter optimization. Scales well to moderate data sets.

EM (`method = "EM"`)**:** Deterministic posterior-weight EM for FCGDINA. Useful as a reproducible baseline or fast diagnostic estimator.

Core Workflow

A typical analysis follows four steps:

1. **Simulate** (or load) data with `sim.data.*()`.
2. **Fit** a model with `fit.*()`.
3. **Rotate** the solution if needed with the `rotate` S3 generic.
4. **Evaluate** fit with `get.fit.index()` and `summary.good.of.fit()`.

Goodness-of-Fit

The package uses the limited-information M2 framework (Maydeu-Olivares & Joe, 2005, 2006) implemented in `good.of.fit`. Fit indices include:

- Information criteria: AIC, AICc, BIC, CAIC, SABIC, HQIC
- Absolute fit: M2, RMSEA (with CI), SRMSR, McDonald NCI
- Comparative fit: CFI, TLI, IFI
- Pseudo-R^2: McFadden, Cox–Snell, Nagelkerke, etc.
- Local dependence: Yen's Q3 (raw and adjusted)
- Classification: posterior entropy, mean max posterior

Reproducibility

The package is designed so that simulation, estimation, extraction, and model checking can be scripted end to end. Simulation functions return the true parameters and the effective call arguments; fitted objects retain the original input arguments, estimates, convergence diagnostics, and likelihood-related quantities used by `logLik` and `get.fit.index`. For computationally expensive Stan or iStEM workflows, manuscript replication code should set seeds explicitly and use a small demonstration configuration in addition to any full-scale analysis.

Author(s)

Maintainer: Haijiang Qin <haijiang133@outlook.com> (ORCID) [copyright holder]

Authors:

- Haijiang Qin <haijiang133@outlook.com> (ORCID) [copyright holder]
- Lei Guo <happygl1229@swu.edu.cn> (ORCID) [copyright holder]

Examples

```
set.seed(123)
sim <- sim.data.FCGDINA(N.person = 12, N.block = 2, I.block = 2,
                        D = 2, model = "DINA")
str(sim$response)
head(sim$Q.matrix)
```

---

coef *S3 Methods: coef*

Description

Extract estimated model coefficients (item, statement, or block parameters) from fitted **ForceChoice** model objects. This provides a standard R interface for retrieving parameter estimates, standard errors, and convergence diagnostics in a consistent format across all eight model families.

Usage

```
## S3 method for class 'MIRT'
coef(object, type = c("par", "theta", "Corr", "all"), ...)

## S3 method for class 'MGPCM'
coef(object, type = c("par", "theta", "Corr", "all"), ...)

## S3 method for class 'MGGUM'
coef(object, type = c("par", "theta", "Corr", "all"), ...)

## S3 method for class 'FCMIRT'
coef(object, type = c("par", "theta", "Corr", "all"), ...)

## S3 method for class 'FCDCM'
coef(object, type = c("par", "theta", "Corr", "all"), ...)

## S3 method for class 'FCGDINA'
coef(object, type = c("par", "theta", "Corr", "all"), ...)

## S3 method for class 'FCGGUM'
coef(object, type = c("par", "theta", "Corr", "all"), ...)

## S3 method for class 'TIRT'
coef(object, type = c("par", "theta", "Corr", "all"), ...)
```

Arguments

`object` An object of one of the following classes:

- `"MIRT"` — Multidimensional IRT model.

- "MGPCM" — Multidimensional Generalized Partial Credit Model.
- "MGGUM" — Multidimensional Generalized Graded Unfolding Model.
- "FCMIRT" — Forced-Choice Multidimensional IRT model.
- "FCDCM" — Forced-Choice Diagnostic Classification Model.
- "FCGDINA" — Forced-Choice GDINA model.
- "FCGGUM" — Forced-Choice Generalized Graded Unfolding Model.
- "TIRT" — Thurstonian IRT for Forced-Choice.

type — Character string specifying which parameters to extract: "par" (default) for item/statement parameters, "theta" for person trait estimates, "Corr" for factor correlations, or "all" for a list with all three.

... — Additional arguments (currently ignored).

Details

For FCDCM objects, type = "par" returns a list with components delta (higher-order parameters, $D \times 2$) and par (block η parameters, $B \times 2$), reflecting the hierarchical structure of the model.

Value

type = "par" A matrix of item/statement/block parameter estimates. Rows index items/statements/blocks and columns index parameter types. Column names indicate the parameter (e.g., "a1", "b", "d0", "tau1", "eta0").

type = "theta" An $N \times D$ matrix of person parameter (latent trait) estimates.

type = "Corr" A $D \times D$ inter-trait correlation matrix.

type = "all" A named list with components par, theta, and Corr.

Methods (by class)

- coef(MIRT): Extract coefficients from MIRT objects. Returns $I \times (D+3)$ matrix with columns a1..aD, b, c, d.
- coef(MGPCM): Extract coefficients from MGPCM objects. Returns $I \times (D + K_{\max})$ matrix with columns a1..aD, d0..d(K-1).
- coef(MGGUM): Extract coefficients from MGGUM objects. Returns matrix with columns a1..aD, delta1..deltaD, tau0..tau(K-1).
- coef(FCMIRT): Extract coefficients from FCMIRT objects. Returns statement parameter matrix with columns a1..aD, b, c, d.
- coef(FCDCM): Extract coefficients from FCDCM objects. Returns a list with delta ($D \times 2$ higher-order parameters) and par ($B \times 2$ block η parameters). type = "theta" returns the higher-order trait, and type = "Corr" returns a scalar 1.
- coef(FCGDINA): Extract coefficients from FCGDINA objects. type = "par" returns CDM delta parameters, type = "theta" returns posterior attribute mastery probabilities, type = "Corr" returns NA (no continuous latent trait).
- coef(FCGGUM): Extract coefficients from FCGGUM objects. Returns matrix with columns a1..aD, delta1..deltaD, tau0..tau(K-1).
- coef(TIRT): Extract coefficients from TIRT objects. Returns $I \times (D+1)$ matrix with columns lambda1..lambdaD, psi2.

Examples

```
sim <- sim.data.MIRT(N = 20, I = 6, D = 2, model = "m2pl")
fit <- fit.MIRT(
  sim$response, model = "m2pl", D = 2, method = "iStEM",
  control.method = list(
    vis = FALSE, seed = 123,
    M = 2, B = 2, burnin.maxitr = 2,
    maxitr = 3, eps1 = 10, eps2 = 10,
    estimate.se = FALSE)
)

coef(fit)                  # item parameter estimates
coef(fit, type = "theta")  # person trait estimates
coef(fit, type = "Corr")   # factor correlation matrix
coef(fit, type = "all")    # everything in a list
```

`confint` *S3 Methods: confint*

Description

Computes Wald-type confidence intervals for the model parameters of a fitted **ForceChoice** model object. Intervals are constructed as $\hat{\psi}_k \pm z_{1-\alpha/2}\,\mathrm{SE}_k$, where $\hat{\psi}_k$ is the point estimate and SE_k is the standard error of the k-th free parameter.

Usage

```
## S3 method for class 'MIRT'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'MGPCM'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'MGGUM'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'FCMIRT'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'FCDCM'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'FCGDINA'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'FCGGUM'
confint(object, parm = NULL, level = 0.95, ...)
```

```
## S3 method for class 'TIRT'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

`object`	A fitted model object.
`parm`	Optional numeric indices or character names selecting free parameters. By default, intervals are returned for all free parameters.
`level`	Numeric; confidence level (default: `0.95` for 95% intervals).
`...`	Additional arguments (currently ignored).

Value

A numeric matrix with two columns, `2.5%` and `97.5%` (column names adjust automatically for non-default `level`). Each row corresponds to one free parameter.

Methods (by class)

- `confint(MIRT)`: Confidence intervals for `MIRT` parameters.
- `confint(MGPCM)`: Confidence intervals for `MGPCM` parameters.
- `confint(MGGUM)`: Confidence intervals for `MGGUM` parameters.
- `confint(FCMIRT)`: Confidence intervals for `FCMIRT` parameters.
- `confint(FCDCM)`: Confidence intervals for `FCDCM` parameters.
- `confint(FCGDINA)`: Confidence intervals for `FCGDINA` parameters.
- `confint(FCGGUM)`: Confidence intervals for `FCGGUM` parameters.
- `confint(TIRT)`: Confidence intervals for `TIRT` parameters.

`deviance` *S3 Methods: deviance*

Description

Returns the model deviance, defined as -2ℓ where ℓ is the marginal log-likelihood evaluated at the parameter estimates.

Usage

```
## S3 method for class 'MIRT'
deviance(object, ...)

## S3 method for class 'MGPCM'
deviance(object, ...)

## S3 method for class 'MGGUM'
```

```
deviance(object, ...)

## S3 method for class 'FCMIRT'
deviance(object, ...)

## S3 method for class 'FCDCM'
deviance(object, ...)

## S3 method for class 'FCGDINA'
deviance(object, ...)

## S3 method for class 'FCGGUM'
deviance(object, ...)

## S3 method for class 'TIRT'
deviance(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Additional arguments (currently ignored).

Value

A numeric scalar: $-2 \log L$.

Methods (by class)

- `deviance(MIRT)`: Model deviance (-2ℓ) for `MIRT` objects.
- `deviance(MGPCM)`: Model deviance for `MGPCM` objects.
- `deviance(MGGUM)`: Model deviance for `MGGUM` objects.
- `deviance(FCMIRT)`: Model deviance for `FCMIRT` objects.
- `deviance(FCDCM)`: Model deviance for `FCDCM` objects.
- `deviance(FCGDINA)`: Model deviance for `FCGDINA` objects.
- `deviance(FCGGUM)`: Model deviance for `FCGGUM` objects.
- `deviance(TIRT)`: Model deviance for `TIRT` objects.

`extract` *S3 Methods: extract*

Description

A generic S3 extractor function designed to retrieve internal components from fitted model objects produced by the **ForceChoice** package. This function provides a consistent interface across all eight model classes, allowing users to access estimated parameters, fit statistics, latent trait estimates, convergence diagnostics, and model configuration data without directly manipulating the internal list structure.

Usage

```
## S3 method for class 'MIRT'
extract(object, what, ...)

## S3 method for class 'MGPCM'
extract(object, what, ...)

## S3 method for class 'MGGUM'
extract(object, what, ...)

## S3 method for class 'FCMIRT'
extract(object, what, ...)

## S3 method for class 'FCDCM'
extract(object, what, ...)

## S3 method for class 'FCGDINA'
extract(object, what, ...)

## S3 method for class 'FCGGUM'
extract(object, what, ...)

## S3 method for class 'TIRT'
extract(object, what, ...)
```

Arguments

`object`	An object of one of the following classes: • `"MIRT"` — Multidimensional IRT model. • `"MGPCM"` — Multidimensional Generalized Partial Credit Model. • `"MGGUM"` — Multidimensional Generalized Graded Unfolding Model. • `"FCMIRT"` — Forced-Choice Multidimensional IRT model. • `"FCDCM"` — Forced-Choice Diagnostic Classification Model. • `"FCGDINA"` — Forced-Choice GDINA model. • `"FCGGUM"` — Forced-Choice Generalized Graded Unfolding Model. • `"TIRT"` — Thurstonian IRT for Forced-Choice.
`what`	A character string specifying the name of the component to extract. Valid choices depend on the class of `object`. See Details section for full listings.
`...`	Additional arguments passed to methods (currently ignored).

Details

This function supports extraction from the eight ForceChoice model classes. Below are the available components for each:

`MIRT`, `MGPCM`, `MGGUM` Traditional (single-stimulus) models. Available components:

`par` List (`est`, `se`, `Rhat`, `free`) of item parameter arrays.

`theta` List (`est`, `se`, `Rhat`) of $N \times D$ person parameter matrices.

`Corr` List (`est`, `se`, `Rhat`) of $D \times D$ inter-trait correlation matrices.

`logLik` Marginal log-likelihood (class `"logLik"`).

`npar` Number of free parameters.

`method` Estimation method (`"stan"` or `"iStEM"`).

`Q.matrix` The $I \times D$ Q-matrix.

`length.poly` Vector of category counts per item (MGPCM, MGGUM only).

`stan.obj` The `stanfit` object (Stan only).

`MCMC.obj` The MCMC extract list (Stan only).

`iStEM` List of iStEM diagnostics (iStEM only).

`call` The matched call.

`arguments` List of arguments used for fitting.

`FCMIRT`, `FCGGUM` Forced-choice models. In addition to the traditional-model components listed above, the following are available:

`block.items` List of item indices in each block.

`response` The $N \times B$ forced-choice response matrix.

`patterns` List of permissible ranking patterns per block.

`patterns.total` List of all possible ranking patterns per block.

`fc.type` Character vector of forced-choice formats (`"RANK"`, `"MOLE"`, `"PICK"`).

`FCDCM` Forced-Choice DCM. Available components:

`par` List (`est`, `se`, `Rhat`, `free`) of block η parameter matrices ($B \times 2$).

`delta` List (`est`, `se`, `Rhat`) of higher-order δ parameters ($D \times 2$).

`theta` List (`est`, `se`, `Rhat`) of $N \times 1$ higher-order trait estimates.

`alpha` Posterior mean attribute profile ($N \times D$).

`alpha.patterns` Full enumeration of 2^D attribute mastery patterns.

`zeta.patterns` Condensation outputs for each pattern.

`dcm.type` DCM condensation rule (`"DINA"` or `"DINO"`).

`response` The $N \times B$ binary block response matrix.

`block.items` List of two-statement blocks.

`patterns` List of binary response patterns per block.

`logLik` Marginal log-likelihood (class `"logLik"`).

`npar` Number of free parameters.

`method` Estimation method.

`Q.matrix` The $I \times D$ Q-matrix.

`stan.obj`, `MCMC.obj`, `iStEM`, `EM` Estimation backend objects.

`call` The matched call.

`arguments` List of arguments used for fitting.

`FCGDINA` Forced-Choice GDINA. Available components:

`delta` List (`est`, `se`, `Rhat`) of item-level CDM delta parameters.

`alpha` Posterior attribute summaries, including posterior class probabilities when available.

`alpha.patterns` Full enumeration of 2^D attribute mastery patterns.

`design.matrix.list` Per-item CDM design matrices.

`response`, `block.items`, `patterns`, `patterns.total`, `fc.type` Forced-choice data structures.

`TIRT` Thurstonian IRT. Available components:

`par` List (`est`, `se`, `Rhat`, `free`) of statement-level parameter matrices ($I \times (D+1)$).

`theta` List (`est`, `se`, `Rhat`) of $N \times D$ person parameter matrices.

`gamma` List (`est`, `se`, `Rhat`, `free`) of pairwise γ parameters.

`gamma.matrix` List (`est`, `se`, `Rhat`) of $I \times I$ skew-symmetric γ matrices.

`Corr` List (`est`, `se`, `Rhat`) of $D \times D$ inter-trait correlation matrices.

`pairs.matrix` Matrix of pairwise comparison indices.

`pairs.value` Person-specific pair data (MOLE/PICK).

`response` The $N \times I_{pairs}$ pairwise response matrix.

`block.items` List of item indices in each block.

`fc.type` Character vector of forced-choice formats.

`Q.matrix` The statement-level Q-matrix.

`logLik` Marginal log-likelihood (class `"logLik"`).

`npar` Number of free parameters.

`method` Estimation method.

`stan.obj`, `MCMC.obj`, `iStEM` Estimation backend objects.

`call` The matched call.

`arguments` List of arguments used for fitting.

Value

The requested component. Return type varies depending on `what` and the class of `object`. If an invalid `what` is provided, an informative error is thrown listing valid options.

Methods (by class)

- `extract(MIRT)`: Extract components from `MIRT` objects.
- `extract(MGPCM)`: Extract components from `MGPCM` objects.
- `extract(MGGUM)`: Extract components from `MGGUM` objects.
- `extract(FCMIRT)`: Extract components from `FCMIRT` objects.
- `extract(FCDCM)`: Extract components from `FCDCM` objects.
- `extract(FCGDINA)`: Extract components from `FCGDINA` objects.
- `extract(FCGGUM)`: Extract components from `FCGGUM` objects.
- `extract(TIRT)`: Extract components from `TIRT` objects.

Examples

```
sim <- sim.data.MIRT(N = 20, I = 6, D = 2, model = "m2pl")
fit <- fit.MIRT(
  sim$response, model = "m2pl", D = 2, method = "iStEM",
  control.method = list(
```

```
    vis = FALSE, seed = 123,
    M = 2, B = 2, burnin.maxitr = 2,
    maxitr = 3, eps1 = 10, eps2 = 10,
    estimate.se = FALSE)
)

extract(fit, "par")       # item parameter estimates
extract(fit, "theta")     # person trait estimates
extract(fit, "Corr")      # factor correlation matrix
extract(fit, "npar")      # number of free parameters
extract(fit, "logLik")    # marginal log-likelihood
extract(fit, "iStEM")     # iStEM convergence diagnostics
```

`fit.FCDCM` *Fit the Forced-Choice Diagnostic Classification Model (FCDCM)*

Description

Fits a higher-order cognitive diagnostic model for forced-choice (paired- comparison) data. Each block presents two statements, and the respondent selects the one they agree with more. The discrete attribute mastery profile $\boldsymbol{\alpha}_j$ for each person is marginalized over all 2^D possible patterns, linking a continuous higher-order latent trait θ_j to the DCM condensation rule at the statement level.

Usage

```
fit.FCDCM(
  data,
  Q.matrix,
  block.items = NULL,
  dcm.type = "DINA",
  method = c("iStEM", "stan"),
  control.model = NULL,
  control.method = NULL
)
```

Arguments

`data`	An $N \times B$ forced-choice data matrix. Each cell must be a two-statement ranking string such as `"1>3"` (first is selected over second), or `0/1` coded responses where `1` means selecting the first statement in the block.
`Q.matrix`	An $I \times D$ binary Q-matrix for the statement- level DCM. $q_{id} = 1$ if attribute d is required by statement i. Must have exactly $2B$ rows.
`block.items`	A list of length B; each element is an integer vector of the two global statement indices in that block. If `NULL`, parsed from `data` assuming `"A>B"` format where `A` and `B` are statement indices.
`dcm.type`	Character vector (length 1 or I): `"DINA"` (default, conjunctive) or `"DINO"` (disjunctive). Recycled if scalar.

`method` Estimation method: `"iStEM"` (default) or `"stan"`.

`control.model` A named list of model-level hyperparameters for the higher-order DCM. Supported entries:

`delta1.mu`, `delta1.sigma` Prior mean and SD for δ_{1d} (higher-order discrimination parameters; log-normal). Defaults: 0, 0.5. The log-normal prior constrains $\delta_{1d} > 0$, reflecting the assumption that higher θ_j increases the probability of mastering each attribute. Larger values of δ_{1d} indicate stronger relationships between the continuous trait θ_j and attribute mastery.

`delta0.mu`, `delta0.sigma` Prior mean and SD for δ_{0d} (higher-order difficulty/threshold parameters; normal). Defaults: 0, 1. The probability of mastering attribute d at $\theta_j = 0$ is $\{1 + \exp(\delta_{1d}\delta_{0d})\}^{-1}$. Higher values of δ_{0d} indicate more difficult attributes.

`eta0.mu`, `eta0.sigma` Prior mean and SD for η_{0b} (truncated normal on (0, 0.5)). Defaults: 0.10, 0.10. The prior encodes the expectation that η_{0b} is small (rarely choose the lower-mastery statement).

`etaAB.mu`, `etaAB.sigma` Prior mean and SD for η_{ABb} (truncated normal on (0, 0.5)). Defaults: 0.30, 0.10. The prior encodes the expectation that η_{ABb} is larger (usually choose the higher-mastery statement).

`use.prior` Logical; if `TRUE` (default), the normal priors on the block response parameters are applied during iStEM estimation. Set to `FALSE` to use flat likelihood updates.

`par` Optional initial $B \times 2$ matrix for iStEM with columns `eta0` and `etaAB`.

`L` Theta grid size for numerical integration and the unidimensional iStEM theta Gibbs sampler. Default: 61.

`theta.lower`, `theta.upper` Bounds for the unidimensional theta grid used by marginal log-likelihood computation and iStEM block Gibbs sampling. Defaults: -6, 6.

`control.method` A named list of method-specific tuning parameters. Common entries (used by both Stan and iStEM):

`cores` Number of CPU cores for parallel chains (Stan) or ignored (iStEM). Default: the number of `chains`.

`vis` Logical; if `TRUE` (default), prints progress information to the console.

`seed` Random seed for reproducibility. Default: a random integer.

Stan-specific entries:

`chains` Number of MCMC chains (default: 2).

`iter` Total iterations per chain (default: 5000).

`warmup` Warmup/burn-in iterations per chain (default: `iter / 2`).

`thin` Thinning interval (default: 1).

`init` Initial values: `"random"` (default) for uniform(-2, 2) initialization, or a list of initial values per chain.

`algorithm` MCMC algorithm: `"HMC"` (default), `"HMC"`, or `"Fixed_param"`.

`adapt_delta` Target average acceptance probability (NUTS; default: 0.95). The discrete attribute-profile marginalization can create a challenging posterior; consider increasing to 0.9–0.95 if divergences occur.

`max_treedepth` Maximum tree depth (NUTS; default: 10). Increase if "max treedepth exceeded" warnings appear.

`stepsize` Initial step size for the leapfrog integrator (auto-tuned by Stan if not set).

`int_time` Total integration time for HMC trajectories (only when `algorithm = "HMC"`).

`metric` Mass matrix type: `"unit_e"`, `"diag_e"` (default), or `"dense_e"`.

`adapt_engaged` Logical; if `TRUE` (default), warmup adaptation is enabled.

`adapt_init_buffer`, `adapt_term_buffer`, `adapt_window` Warmup adaptation scheduling parameters (defaults: 25, 50, 25).

iStEM-specific entries:

`M` Number of burn-in batches retained for Geweke convergence diagnosis (default: 10; must be ≥ 2).

`B` Batch size: MCMC iterations per batch (default: 20).

`burnin.maxitr` Maximum burn-in batches (default: 100).

`maxitr` Maximum total batches (default: 2000).

`eps1` Geweke z-score convergence threshold for burn-in (default: 1.5).

`eps2` Monte Carlo error tolerance for the final chain (default: 0.4).

`frac1`, `frac2` Fractions for the Geweke diagnostic (defaults: 0.1, 0.5).

`optim.maxit` Maximum iterations for parameter optimization steps (default: 50). In FCDCM iStEM, the higher-order δ parameters and block η parameters are updated via closed-form or numerical optimization.

`fix.corr` Logical; in the FCDCM, the latent trait is unidimensional, so this parameter has no effect (default: `FALSE`).

`estimate.se` Logical; if `TRUE` (default), standard errors are computed from the final Monte Carlo chain.

Value

An object of class `"FCDCM"` containing:

`npar` Number of free parameters ($2D + 2B$).

`method` `"stan"` or `"iStEM"`.

`theta` List with `est`, `se`, `Rhat` ($N \times 1$); higher-order trait estimates.

`delta` List with `est`, `se`, `Rhat` ($D \times 2$); higher-order δ_1, δ_0 parameters.

`par` List with `est`, `se`, `Rhat` ($B \times 2$); block response parameters `eta0` and `etaAB`.

`alpha` List with `est`, `se`, `Rhat`, and `prob`; `est` is the binary classified attribute profile ($N \times D$) and `prob` contains posterior mastery probabilities.

`class.post` Posterior probabilities for all 2^D attribute profiles.

`alpha.patterns`, `zeta.patterns` Full enumeration of attribute mastery patterns and their condensation outputs.

`logLik` Marginal log-likelihood (class `"logLik"`).

Model Specification

Higher-order latent structure. A unidimensional continuous trait $\theta_j \sim N(0,1)$ governs the probability of mastering each attribute $d = 1, \ldots, D$:

$$P(\alpha_{jd} = 1 \mid \theta_j) = \frac{1}{1 + \exp\left[-\left(\delta_{1d}\,\theta_j - \delta_{1d}\,\delta_{0d}\right)\right]},$$

where $\delta_{1d} > 0$ is the discrimination and δ_{0d} is the difficulty (threshold) for attribute d. Attributes are conditionally independent given θ_j.

Statement-level condensation. For each statement $i = 1, \ldots, I$ with Q-vector $\mathbf{q}_i = (q_{i1}, \ldots, q_{iD})'$, the latent mastery status $\zeta_{ij} \in \{0, 1\}$ is determined by one of two condensation rules:

DINA (conjunctive):

$$\zeta_{ij} = \prod_{d:q_{id}=1} \alpha_{jd}.$$

All required attributes must be mastered.

DINO (disjunctive):

$$\zeta_{ij} = 1 - \prod_{d:q_{id}=1} (1 - \alpha_{jd}).$$

At least one required attribute must be mastered.

Forced-choice block response. Block b consists of two statements (A_b, B_b). The probability that the respondent selects statement A_b over B_b follows the original FC-DCM specification and fixes equal-condensation probabilities to chance:

$$P(\text{choose } A_b \mid \zeta_{A_b}, \zeta_{B_b}) = \begin{cases} \eta_{0b}, & \text{if } \zeta_{A_b} < \zeta_{B_b}, \\ 0.5, & \text{if } \zeta_{A_b} = \zeta_{B_b}, \\ 0.5 + \eta_{ABb}, & \text{if } \zeta_{A_b} > \zeta_{B_b}. \end{cases}$$

with $0 < \eta_{0b} < 0.5$ and $0 < \eta_{ABb} < 0.5$. Thus equal statement-level mastery produces chance choice, choosing the lower mastery statement has probability η_{0b}, and choosing the higher mastery statement has probability $0.5 + \eta_{ABb}$. The probability of choosing B_b is the complement.

The marginal likelihood integrates over all 2^D attribute patterns:

$$P(\mathbf{Y}_j \mid \theta_j) = \sum_{\boldsymbol{\alpha} \in \{0,1\}^D} P(\boldsymbol{\alpha} \mid \theta_j) \times \prod_{b=1}^{B} P(Y_{jb} \mid \zeta_{A_b}, \zeta_{B_b}).$$

Design and Identifiability

A block with $\zeta_{A_b} = \zeta_{B_b}$ contributes no direct information for distinguishing the two equal condensation states (both are fixed at 0.5). For single-attribute statements and $D = 2$, if all blocks compare the same ordered attribute pair, the response distribution distinguishes `10` from `01` but leaves `00` and `11` to be separated only by the higher-order structural model. Good global fit can therefore coexist with weak profile classification.

A forced-choice Q/block design should provide repeated cross-attribute comparisons, avoid pairing statements with identical Q-vectors, and keep each attribute represented in both statement positions. A practical numerical target, consistent with the original FC-DCM simulation logic, is that each attribute appears multiple times as the first/predominant statement and multiple times as the second/inferior statement, with comparisons distributed across different attribute pairs. The automatic simulator uses this balanced assembly rule when `block.items` is not supplied.

Estimation Methods

Stan (`method = "stan"`)**:** Full Bayesian inference via HMC. The discrete attribute profile is exactly marginalized in the Stan likelihood.

iStEM (`method = "iStEM"`)**:** Improved Stochastic EM. Person sampling considers the joint space of θ_j and $\boldsymbol{\alpha}_j$. Parameter updates use closed-form or numerical optimization steps.

References

de la Torre, J., & Douglas, J. A. (2004). Higher-order latent trait models for cognitive diagnosis. *Psychometrika*, 69(3), 333–353. doi:10.1007/BF02295640

Junker, B. W., & Sijtsma, K. (2001). Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, 25(3), 258–272.

See Also

`sim.data.FCDCM`, `get.fit.index.FCDCM`, `logLik.FCDCM`

Examples

```
sim <- sim.data.FCDCM(N.person = 20, N.block = 3, D = 2,
                      dcm.type = "DINA")
fit <- fit.FCDCM(sim$data, Q.matrix = sim$Q.matrix,
                 block.items = sim$block.items,
                 dcm.type = "DINA", method = "iStEM",
                 control.method = list(
                   seed = 123, vis = FALSE,
                   M = 2, B = 2, burnin.maxitr = 2,
                   maxitr = 3, eps1 = 10, eps2 = 10,
                   estimate.se = FALSE))
head(fit$alpha$est)               # binary attribute profiles (0/1)
head(fit$alpha$prob)              # marginal mastery probabilities
gof <- get.fit.index(fit)
summary(gof)
```

`fit.FCGDINA` *Fit the Forced-Choice GDINA Model*

Description

Fits a forced-choice cognitive diagnostic model (FCGDINA) to comparative response data. The statement-level cognitive diagnosis component can be DINA, DINO, ACDM, or GDINA, and the block-level response model transforms statement endorsement probabilities into forced-choice probabilities for full rankings (`"RANK"`), most-least choices (`"MOLE"`), or best-only choices (`"PICK"`).

Usage

```
fit.FCGDINA(
  data,
  Q.matrix,
  model = c("GDINA", "DINA", "DINO", "ACDM"),
  block.items = NULL,
  fc.type = NULL,
  method = c("EM", "stan", "iStEM"),
  control.model = NULL,
  control.method = NULL
)
```

Arguments

`data`	An $N \times B$ forced-choice data matrix.
`Q.matrix`	An $I \times D$ binary Q-matrix. Rows correspond to statements, columns correspond to attributes, and every row must contain at least one 1.
`model`	CDM item model: `"GDINA"`, `"DINA"`, `"DINO"`, or `"ACDM"`.
`block.items`	A list of length B. Each element gives the global statement indices in the corresponding forced-choice block. If `NULL`, the block structure is parsed from full-ranking character data when possible.
`fc.type`	Forced-choice response type: `"RANK"`, `"MOLE"`, or `"PICK"`. A scalar is recycled to all blocks. If `NULL`, the type is inferred from character data when possible and otherwise defaults to `"RANK"`.
`method`	Estimation method. `"EM"` (default) runs a deterministic posterior-weight EM algorithm, `"stan"` runs full Bayesian MCMC, and `"iStEM"` runs improved stochastic EM.
`control.model`	Optional list of model-level controls. For Stan, `delta.mu` and `delta.sigma` set the normal prior $\delta \sim N(\mu, \sigma)$ for the free Stan FCGDINA logit-scale delta effects (defaults: 0 and 1). `pi.prior.alpha` sets the Dirichlet concentration for the structural parameters π (default 1, uniform prior over the simplex). For EM and iStEM, Stan priors are ignored; use `delta.lower` and `delta.upper` in `control.method` instead.
`control.method`	Optional list of method-specific controls.

Value

An object of class `"FCGDINA"`, a list containing:

`npar` Number of free CDM delta parameters.

`method` Estimation method actually used: `"stan"`, `"iStEM"`, or `"EM"`.

`alpha` List with `est`, `se`, `Rhat`, and `prob`. `est` is an $N \times D$ matrix of posterior attribute mastery probabilities; `prob` contains posterior probabilities for all 2^D attribute profiles.

`delta` List with `est`, `se`, and `Rhat`. For Stan fits, each element is the probability-scale transform of the identified minimum-norm logit coefficient vector. Absolute statement endorsement probabilities are not identified by forced-choice data alone.

`delta.identification` Stan-only rank, nullity, blockwise rank diagnostics, and the identifying convention.

`delta.store` Stored posterior, iStEM, or bootstrap delta draws when available.

`Q.matrix`, `block.items`, `patterns`, `patterns.total`, `design.matrix.list`, `alpha.patterns`, `fc.type`, `response` Processed model and data structures used by prediction, fit indices, and S3 methods.

`stan.obj`, `MCMC.obj` Stan fit and extracted posterior draws for `method = "stan"`; otherwise `NULL`.

`iStEM` iStEM convergence diagnostics and chains for `method = "iStEM"`; otherwise `NULL`.

`EM` EM convergence diagnostics, log-likelihood trace, and bootstrap summary for `method = "EM"`; otherwise `NULL`.

`logLik` Marginal log-likelihood with class `"logLik"`.

`call`, `arguments` Matched call and effective fitting arguments.

Model Specification

Let $\boldsymbol{\alpha}_n \in \{0,1\}^D$ denote the latent attribute profile for person n. For statement i, the binary Q-vector $\mathbf{q}_i$ selects the required attributes and the reduced attribute pattern is mapped to a CDM design row $\mathbf{x}_{ic}$. For Stan estimation, $\boldsymbol{\delta}_i$ is represented directly as the item-specific CDM effect vector on the logit scale: intercept, attribute main effects, and interaction effects. The item-level endorsement probability for attribute class c is

$$p_{ic} = \text{logit}^{-1}(\mathbf{x}'_{ic}\boldsymbol{\delta}_i),$$

so main effects and interactions may be negative while probabilities remain in $(0,1)$. Forced-choice observations identify only within-block utility contrasts, not the absolute endorsement logits of individual statements. The Stan implementation therefore removes the exact null space of those contrasts and reports the unique minimum-norm coefficient representative. The design matrix is generated according to `model`: `"DINA"`, `"DINO"`, `"ACDM"`, or `"GDINA"`. EM and iStEM retain the existing probability-scale CDM parameterisation.

For each forced-choice block, the statement endorsement probabilities are converted to ranking-pattern probabilities by the same sequential Luce–Plackett transformation used by `fit.FCMIRT` and `fit.FCGGUM`. The likelihood marginalizes over all 2^D attribute profiles using a uniform latent-class prior:

$$P(\mathbf{Y}_n) = \sum_{c=1}^{2^D} \pi_c \prod_{b=1}^{B} P(Y_{nb} \mid \boldsymbol{\alpha}_c, \boldsymbol{\delta}, \text{block}_b).$$

Data Coding

`data` must be an $N \times B$ matrix, with one column per forced-choice block. Entries may be character rankings such as `"2>1>3"` or 1-based integer pattern indices. If integer pattern indices are supplied, `block.items` must be provided because item identities cannot be recovered from indices alone. For `"MOLE"` and `"PICK"` character data, supplying `block.items` is recommended and required whenever partial rankings do not identify all items in a block.

Estimation Controls

Common `control.method` entries are:

`seed` Integer random seed.

`cores` Number of cores used by Stan chains; ignored by the deterministic EM updates.

`vis` Logical progress flag.

`delta.lower`, `delta.upper` Lower and upper bounds for delta optimization in EM and iStEM. Defaults are -4 and 4.

Stan-specific entries include `chains`, `iter`, `warmup`, `thin`, `init`, `algorithm`, and the usual Stan `control` entries such as `adapt_delta`, `max_treedepth`, `stepsize`, and `metric`. Defaults are inherited from the common ForceChoice Stan helper: 4 chains, 2000 iterations, and half the iterations used as warmup. The default `init = 0` is valid for Stan FCGDINA because it implies $p_{ic} = 0.5$ under the logit link.

iStEM-specific entries include `M`, `B`, `burnin.maxitr`, `maxitr`, `eps1`, `eps2`, `frac1`, `frac2`, `optim.maxit`, and `estimate.se`. Each iStEM update samples discrete attribute classes from the current posterior and optimises δ parameters given the sampled class memberships.

EM-specific entries are:

`maxitr` Maximum EM iterations; default 500.

`minitr` Minimum EM iterations before convergence checks; default 2.

`tol` Absolute log-likelihood change tolerance; default `1e-6`.

`par.tol` Maximum absolute delta-parameter change tolerance; default `1e-4`.

`optim.maxit` Maximum L-BFGS-B iterations for blockwise delta updates; default 200.

`estimate.se` Logical; whether to estimate standard errors by nonparametric bootstrap. Default `FALSE`.

`bootstrap` Number of bootstrap samples when `estimate.se = TRUE`. Default 100; must be at least 2.

References

de la Torre, J. (2011). The generalized DINA model framework. *Psychometrika*, 76(2), 179–199. doi:10.1007/s11336-011-9207-7

Templin, J., & Henson, R. A. (2006). Measurement of psychological disorders using cognitive diagnosis models. *Psychological Methods*, 11(3), 287–305. doi:10.1037/1082-989X.11.3.287

See Also

`sim.data.FCGDINA`, `model.FCGDINA`, `get.fit.index.FCGDINA`, `fit.FCDCM`, `fit.FCMIRT`, `fit.FCGGUM`

Examples

```
sim <- sim.data.FCGDINA(N.person = 20, N.block = 2, I.block = 2,
                        D = 2, model = "DINA", fc.type = "RANK")

fit <- fit.FCGDINA(
  sim$data,
  Q.matrix = sim$Q.matrix,
  block.items = sim$block.items,
  model = "DINA",
  fc.type = sim$fc.type,
```

```
  method = "EM",
  control.method = list(seed = 123, vis = FALSE,
                        maxitr = 2, estimate.se = FALSE)
)

coef(fit, type = "par")
head(fit$alpha$est)
logLik(fit)


# stan code, long time
fit.stan <- fit.FCGDINA(
  sim$data,
  Q.matrix = sim$Q.matrix,
  block.items = sim$block.items,
  model = "GDINA",
  method = "stan",
  control.method = list(seed = 123, cores = 2,
                        chains = 1, iter = 200, warmup = 100)
)
```

`fit.FCGGUM` *Fit the Forced-Choice Generalized Graded Unfolding Model (FCGGUM)*

Description

Fits a forced-choice variant of the multidimensional generalized graded unfolding model (GGUM) to comparative ranking data. Unlike dominance-based FC models, the FCGGUM uses an ideal-point (unfolding) item response process: endorsement probability peaks when the person and item locations coincide, and the block-level ranking follows the Luce–Plackett model.

Usage

```
fit.FCGGUM(
  data,
  Q.matrix = NULL,
  block.items = NULL,
  D = NULL,
  fc.type = NULL,
  method = c("iStEM", "stan"),
  control.model = NULL,
  control.method = NULL
)
```

Arguments

`data` An $N \times B$ forced-choice data matrix with ranking strings or 1-based pattern indices.

`Q.matrix` An optional $I \times D$ matrix with entries $-1, 0, 1$. See `fit.MGGUM` for the sign convention.

`block.items` A list of length B giving global item indices in each block. Parsed from `data` if `NULL` and data contains ranking strings.

`D` Integer; number of latent dimensions. If `NULL`, inferred from `Q.matrix` when available, otherwise from the block structure.

`fc.type` Forced-choice response type: `"RANK"` (default), `"MOLE"`, or `"PICK"`.

`method` Estimation method: `"iStEM"` (default) or `"stan"`.

`control.model` A named list of model-level hyperparameters for the GGUM unfolding model in a forced-choice framework. Supported entries:

`a.mu`, `a.sigma` Prior location and scale for $\log a_{id}$ (log-normal). Defaults: 0, 0.5. The log-normal prior ensures positivity of discrimination parameters in the unfolding model. The zero mean-log encourages slopes near 1 unless the data strongly indicate otherwise.

`delta.pos.mu`, `delta.pos.sigma` Prior mean and SD for δ_{id} when $q_{id} = 1$ (positive-side item locations; normal). Defaults: 1, 0.5. In the FC context, these priors regularize the location of statements on the positive side of each latent dimension.

`delta.neg.mu`, `delta.neg.sigma` Prior mean and SD for δ_{id} when $q_{id} = -1$ (negative-side item locations; normal). Defaults: -1, 0.5. These priors regularize statement locations on the negative side of each dimension.

`tau.mu`, `tau.sigma` Location and scale for the log-normal prior on the positive threshold magnitude $-\tau_{i1}$. Defaults: 0, 0.5. In the binary FCGGUM, each statement has a single free threshold $\tau_{i1} < 0$.

`theta.mu` Prior mean vector for $\boldsymbol{\theta}_j$. Default: `rep(0, D)`.

`L` Theta grid size per dimension for marginal log-likelihood computation and iStEM block Gibbs sampling. Default adapts to D.

`theta.lower`, `theta.upper` Bounds for the theta grid used by marginal log-likelihood computation and iStEM block Gibbs sampling. Defaults: -6, 6.

`control.method` A named list of method-specific tuning parameters. Common entries (used by both Stan and iStEM):

`cores` Number of CPU cores for parallel chains (Stan) or ignored (iStEM). Default: the number of `chains`.

`vis` Logical; if `TRUE` (default), prints progress information to the console.

`seed` Random seed for reproducibility. Default: a random integer.

Stan-specific entries:

`chains` Number of MCMC chains (default: 2).

`iter` Total iterations per chain (default: 5000).

`warmup` Warmup/burn-in iterations per chain (default: `iter / 2`).

`thin` Thinning interval (default: 1).

`init` Initial values: `"random"` (default) for uniform(-2, 2) initialization, or a list of initial values per chain.

`algorithm` MCMC algorithm: `"HMC"` (default), `"HMC"`, or `"Fixed_param"`.

`adapt_delta` Target average acceptance probability (NUTS; default: 0.95). The GGUM likelihood involves summation terms that can create challenging posterior geometries; consider increasing to 0.9–0.95 if divergences occur.

`max_treedepth` Maximum tree depth (NUTS; default: 10). Increase if "max treedepth exceeded" warnings appear.

`stepsize` Initial step size for the leapfrog integrator (auto-tuned by Stan if not set).

`int_time` Total integration time for HMC trajectories (only when `algorithm = "HMC"`).

`metric` Mass matrix type: `"unit_e"`, `"diag_e"` (default), or `"dense_e"`.

`adapt_engaged` Logical; if `TRUE` (default), warmup adaptation is enabled.

`adapt_init_buffer`, `adapt_term_buffer`, `adapt_window` Warmup adaptation scheduling parameters (defaults: 25, 50, 25).

iStEM-specific entries:

`M` Number of burn-in batches retained for Geweke convergence diagnosis (default: 10; must be ≥ 2).

`B` Batch size: MCMC iterations per batch (default: 20).

`burnin.maxitr` Maximum burn-in batches (default: 100).

`maxitr` Maximum total batches (default: 2000).

`eps1` Geweke z-score convergence threshold for burn-in (default: 1.5).

`eps2` Monte Carlo error tolerance for the final chain (default: 0.4).

`frac1`, `frac2` Fractions for the Geweke diagnostic (defaults: 0.1, 0.5).

`corr.optim.maxit` Maximum L-BFGS-B iterations for the constrained unit-diagonal correlation update (default: 50).

`optim.maxit` Maximum L-BFGS-B iterations per item (default: 50). GGUM item parameters (a, δ, τ) are optimized jointly under monotonicity and sign constraints. Increase if optimization warnings appear.

`fix.corr` Logical; if `TRUE`, the inter-trait correlation matrix is fixed to the identity (default: `FALSE`).

`estimate.se` Logical; if `TRUE` (default), standard errors are computed from the final Monte Carlo chain.

`delta.lower` Lower bound on $|\delta_{id}|$ (default: `1e-4`). Ensures item locations are bounded away from zero for active dimensions.

`tau.gap.lower`, `tau.gap.upper` Lower and upper bounds on the positive gap used to construct $\tau_{i1} < 0$. Defaults are 1e-4 and 6.

Value

An object of class `"FCGGUM"` containing:

`npar` Number of free parameters.

`method` `"stan"` or `"iStEM"`.

`theta` List with `est`, `se`, `Rhat` ($N \times D$).

`par` List with `est`, `se`, `Rhat`, `free` ($I \times (2D + 2)$).

`Corr` List with `est`, `se`, `Rhat` ($D \times D$).

`block.items`, `patterns`, `patterns.total`, `response` Block structure.

`logLik` Marginal log-likelihood (class `"logLik"`).

Model Specification

Item-level unfolding process. For statement i with Q-vector $\mathbf{q}_i \in \{-1, 0, 1\}^D$, the category response function is the GGUM (see `fit.MGGUM` for the full polytomous specification). For the forced-choice context, each item is binary ($K_i = 2$). Define

$$r_{ij} = \left[\sum_{d=1}^{D} a_{id}^2 \left(\theta_{jd} - \delta_{id}\right)^2\right]^{1/2}, \qquad S_i = \sum_{d=1}^{D} a_{id}, \qquad \psi_{i1} = \tau_{i1} S_i.$$

The binary GGUM endorsement probability used by the code is:

$$P_i(\boldsymbol{\theta}_j) = \frac{\exp\{r_{ij} - \psi_{i1}\} + \exp\{2r_{ij} - \psi_{i1}\}}{1 + \exp\{3r_{ij}\} + \exp\{r_{ij} - \psi_{i1}\} + \exp\{2r_{ij} - \psi_{i1}\}},$$

where $\tau_{i1} < 0$ is the first (and only free) threshold.

Block-level forced-choice ranking. Within each block b, the probability of the observed ranking is computed by applying the sequential Luce/Plackett rule to $u_i(\boldsymbol{\theta}_j) = \text{logit}\{P_i(\boldsymbol{\theta}_j)\}$:

$$P(\text{ranking} \mid \boldsymbol{\theta}_j) = \prod_{m=1}^{K_b - 1} \frac{\exp\{u_{i_{(m)}}(\boldsymbol{\theta}_j)\}}{\sum_{r=m}^{K_b} \exp\{u_{i_{(r)}}(\boldsymbol{\theta}_j)\}}.$$

MOLE and PICK probabilities are sums over compatible full rankings, normalized over the observed block patterns by the C++ probability helper.

Q-matrix for unfolding. The Q-matrix serves a dual role: entries of ± 1 activate both the slope a_{id} and the location δ_{id}; the sign of q_{id} determines the sign of δ_{id} (positive vs. negative side of dimension d).

Estimation Methods

Stan (`method = "stan"`): Full Bayesian inference via HMC. The GGUM likelihood involves summation terms that are handled in the Stan model block.

iStEM (`method = "iStEM"`): Improved Stochastic EM. Person sampling uses finite-grid block Gibbs; GGUM item parameters updated via constrained L-BFGS-B with monotonicity constraints on τ and sign constraints on δ.

References

Roberts, J. S., Donoghue, J. R., & Laughlin, J. E. (2000). A general item response theory model for unfolding unidimensional polytomous responses. *Applied Psychological Measurement*, 24(1), 3–32.

See Also

sim.data.FCGGUM, get.fit.index.FCGGUM, logLik.FCGGUM, fit.MGGUM

Examples

```
sim <- sim.data.FCGGUM(N.person = 20, N.block = 3, I.block = 2,
                       D = 2, fc.type = "RANK")
fit <- fit.FCGGUM(sim$data, Q.matrix = sim$Q.matrix,
                  block.items = sim$block.items,
                  D = 2, fc.type = "RANK", method = "iStEM",
                  control.method = list(
                    vis = FALSE, seed = 123,
                    M = 2, B = 2, burnin.maxitr = 2,
                    maxitr = 3, eps1 = 10, eps2 = 10,
                    estimate.se = FALSE))
head(fit$theta$est)
gof <- get.fit.index(fit)
summary(gof)
```

---

fit.FCMIRT *Fit the Forced-Choice Multidimensional IRT (FCMIRT) Model*

Description

Fits a forced-choice variant of the multidimensional item response theory model to comparative ranking data. In forced-choice (FC) formats, respondents compare items within blocks rather than rating each item in isolation. The FCMIRT model couples an item-level MIRT endorsement model with a block-level ranking mechanism to recover latent trait estimates from comparative responses.

Usage

```
fit.FCMIRT(
  data,
  model = "2PL",
  Q.matrix = NULL,
  block.items = NULL,
  D = NULL,
  fc.type = NULL,
  method = c("iStEM", "stan"),
  control.model = NULL,
  control.method = NULL
)
```

Arguments

`data`	An $N \times B$ forced-choice data object. Each entry is either a ranking string (e.g., `"2>1>3"`) or a 1-based pattern index into the block's ranking patterns. If pattern indices are used, `block.items` must be supplied.
`model`	MIRT model type: `"m1pl"`, `"m2pl"` (default), `"m3pl"`, or `"m4pl"`.
`Q.matrix`	An optional $I \times D$ binary (0/1) matrix indicating active item loadings.
`block.items`	A list of length B where each element is an integer vector of global item indices belonging to that block. If `NULL` and `data` contains ranking strings, automatically parsed from the data.
`D`	Integer; number of latent dimensions. If `NULL`, inferred from `Q.matrix`; if both are `NULL`, an error is raised.
`fc.type`	Character vector (length 1 or B): `"RANK"` (full ranking, default), `"MOLE"` (most-least), or `"PICK"` (best only). Recycled to all blocks if scalar.
`method`	Estimation method: `"iStEM"` (default) or `"stan"`.
`control.model`	A named list of model-level hyperparameters. Supported entries:
	`a.mu`, `a.sigma` Prior location and scale for $\log a_{id}$ (log-normal). Defaults: 0, 1.0. **Important:** The Stan default for `a.sigma` is deliberately set to 1.0 (wider than the MIRT default of 0.25) because forced-choice blocks identify relative utilities; tight common slope priors can spuriously shrink within-block slopes toward equality, degrading trait recovery. For iStEM, the prior defaults match the standard MIRT values (`a.mu` = `0.25`, `a.sigma` = `0.25`) unless overridden.
	`b.mu`, `b.sigma` Prior mean and SD for b_i (normal). Defaults: 0, 1. Under the block zero-sum constraint $\sum_{i \in \text{block } b} b_i = 0$, these priors regularize the free difficulty parameters.
	`c.mu`, `c.sigma` Uniform support bounds $[c_{\min}, c_{\max}]$ for lower-asymptote parameters c_i. Defaults: 0, 0.35. Only used for 3PL/4PL models.
	`d.mu`, `d.sigma` Uniform support bounds $[d_{\min}, d_{\max}]$ for upper-asymptote parameters d_i. Defaults: 0.65, 1. Only used for 4PL models.
	`theta.mu` Prior mean vector for $\boldsymbol{\theta}_j$. Default: `rep(0, D)`.
	`L` Theta grid size per dimension for marginal log-likelihood computation and iStEM block Gibbs sampling. Default adapts to D (e.g., 61 for D = 1, 31 for D = 2, 15 for D = 3).
	`theta.lower`, `theta.upper` Bounds for the theta grid used by marginal log-likelihood computation and iStEM block Gibbs sampling. Defaults: -6, 6.
	`use.prior` Logical (iStEM only). If `FALSE`, the item-prior penalty term is omitted from the item update, recovering an approximate maximum-likelihood item step. Default is `TRUE`.
`control.method`	A named list of method-specific tuning parameters. Common entries (used by both Stan and iStEM):
	`cores` Number of CPU cores for parallel chains (Stan) or ignored (iStEM). Default: the number of `chains`.
	`vis` Logical; if `TRUE` (default), prints progress information to the console.
	`seed` Random seed for reproducibility. Default: a random integer.

Stan-specific entries:

`chains` Number of MCMC chains (default: 2).

`iter` Total iterations per chain (default: 5000).

`warmup` Warmup/burn-in iterations per chain (default: `iter / 2`).

`thin` Thinning interval (default: 1).

`init` Initial values: `"random"` (default) for uniform(-2, 2) initialization, or a list of initial values per chain.

`algorithm` MCMC algorithm: `"HMC"` (default), `"HMC"`, or `"Fixed_param"`.

`adapt_delta` Target average acceptance probability (NUTS; default: 0.95). Forced-choice models often have challenging posterior geometries; consider increasing to 0.9 or higher if divergent transitions appear.

`max_treedepth` Maximum tree depth (NUTS; default: 10). Increase if "max treedepth exceeded" warnings appear.

`stepsize` Initial step size for the leapfrog integrator (auto-tuned by Stan if not set).

`int_time` Total integration time for HMC trajectories (only when `algorithm = "HMC"`).

`metric` Mass matrix type: `"unit_e"`, `"diag_e"` (default), or `"dense_e"`.

`adapt_engaged` Logical; if `TRUE` (default), warmup adaptation is enabled.

`adapt_init_buffer`, `adapt_term_buffer`, `adapt_window` Warmup adaptation scheduling parameters (defaults: 25, 50, 25).

iStEM-specific entries:

`M` Number of burn-in batches retained for Geweke convergence diagnosis (default: 10; must be ≥ 2).

`B` Batch size: MCMC iterations per batch (default: 20).

`burnin.maxitr` Maximum burn-in batches (default: 100).

`maxitr` Maximum total batches (default: 2000).

`eps1` Geweke z-score convergence threshold for burn-in (default: 1.5).

`eps2` Monte Carlo error tolerance for the final chain (default: 0.4).

`frac1`, `frac2` Fractions for the Geweke diagnostic (defaults: 0.1, 0.5).

`corr.optim.maxit` Maximum L-BFGS-B iterations for the constrained unit-diagonal correlation update (default: 50).

`optim.maxit` Maximum L-BFGS-B iterations per item (default: 50).

`fix.corr` Logical; fix correlations to identity (default: `FALSE`).

`estimate.se` Logical; compute standard errors from final MC chain (default: `TRUE`).

`a.lower`, `a.upper` Bounds on a_{id} during optimization (defaults: 1e-4, 6).

`b.lower`, `b.upper` Bounds on b_i during optimization (defaults: -6, 6).

`c.lower`, `c.upper` Bounds on c_i for 3PL/4PL models. Defaults from prior support.

`d.lower`, `d.upper` Bounds on d_i for 4PL models. Defaults from prior support.

Value

An object of class `"FCMIRT"` containing:

`npar` Number of free parameters.

`method` `"stan"` or `"iStEM"`.

`theta` List with `est`, `se`, `Rhat` ($N \times D$); person trait estimates.

`par` List with `est`, `se`, `Rhat`, `free` ($I \times (D+3)$); item parameters `a1..aD`, `b`, `c`, `d`.

`Corr` List with `est`, `se`, `Rhat` ($D \times D$).

`block.items`, `patterns`, `patterns.total`, `response` Block structure and response data.

`logLik` Marginal log-likelihood (class `"logLik"`).

Model Specification

Item-level endorsement. Each statement $i = 1, \ldots, I$ is governed by a MIRT model (1PL through 4PL). For a person with trait $\boldsymbol{\theta}_j$, the probability of endorsing statement i in isolation is:

$$P_i(\boldsymbol{\theta}_j) = c_i + (d_i - c_i) \times \frac{1}{1 + \exp\left[-\left(\sum_{d=1}^{D} a_{id}\,\theta_{jd} - b_i\right)\right]}.$$

Block-level ranking. Let block $b = 1, \ldots, B$ consist of items $\{i_1, i_2, \ldots, i_{K_b}\}$ and define the implemented statement utility as the logit of the item-level endorsement probability,

$$u_i(\boldsymbol{\theta}_j) = \log\left[\frac{P_i(\boldsymbol{\theta}_j)}{1 - P_i(\boldsymbol{\theta}_j)}\right].$$

The probability that an examinee produces the ranking $i_{(1)} \succ i_{(2)} \succ \cdots \succ i_{(K_b)}$ (read as "$i_{(1)}$ is preferred over $i_{(2)}$ over ...") is:

$$P\big(i_{(1)} \succ \cdots \succ i_{(K_b)} \mid \boldsymbol{\theta}_j\big) = \prod_{m=1}^{K_b - 1} \frac{\exp\{u_{i_{(m)}}(\boldsymbol{\theta}_j)\}}{\sum_{r=m}^{K_b} \exp\{u_{i_{(r)}}(\boldsymbol{\theta}_j)\}}.$$

The Stan code evaluates the same choice kernel as $\log P_i + \sum_{h \neq i} \log(1 - P_h)$ within each remaining set, which is algebraically equivalent to a softmax over $\operatorname{logit}(P_i)$.

Partial rankings (MOLE / PICK). For MOLE (most-least) data, only the best and worst items in each block are identified; for PICK data, only the best item. The probability of a partial ranking is obtained by marginalizing (summing) the full-ranking probabilities over all completions consistent with the partial constraint; the C++ probability helper returns probabilities normalized over the observed RANK/MOLE/PICK patterns for each block.

Identification constraint (2PLM-RANK). Block intercepts b_i are constrained to sum to zero within each block:

$$\sum_{i \in \text{block } b} b_i = 0, \qquad b = 1, \ldots, B,$$

so that only $K_b - 1$ difficulties are freely estimated per block.

Estimation Methods

Stan (`method = "stan"`)**:** Full Bayesian inference via HMC. The forced-choice likelihood is evaluated over the ranking pattern probabilities. By default, `a.sigma` is set to `1.0` (wider prior) to avoid spurious shrinkage of within-block slopes toward equality.

iStEM (`method = "iStEM"`)**:** Improved Stochastic EM with finite-grid block Gibbs person-sampling and item optimization (L-BFGS-B). Block-level identification constraints are enforced during each item-parameter update.

References

Brown, A., & Maydeu-Olivares, A. (2011). Item response modeling of forced-choice questionnaires. *Educational and Psychological Measurement*, 71(3), 460–502. doi:10.1177/0013164410375112

Luce, R. D. (1959). *Individual choice behavior: A theoretical analysis*. Wiley.

Plackett, R. L. (1975). The analysis of permutations. *Journal of the Royal Statistical Society: Series C*, 24(2), 193–202.

See Also

`sim.data.FCMIRT`, `get.fit.index.FCMIRT`, `logLik.FCMIRT`, `fit.MIRT`

Examples

```
# Simulate forced-choice data
sim <- sim.data.FCMIRT(N.person = 20, N.block = 3, I.block = 2,
                       D = 2, model = "m2pl", fc.type = "RANK")

# Fit via iStEM
fit <- fit.FCMIRT(sim$data, model = "m2pl",
                  Q.matrix = sim$Q.matrix,
                  block.items = sim$block.items,
                  D = 2, fc.type = "RANK",
                  method = "iStEM",
                  control.method = list(
                    vis = FALSE, seed = 123,
                    M = 2, B = 2, burnin.maxitr = 2,
                    maxitr = 3, eps1 = 10, eps2 = 10,
                    estimate.se = FALSE))

# Examine trait recovery
cor(fit$theta$est, sim$theta)

# Fit indices (nominal binary expansion)
gof <- get.fit.index(fit)
summary(gof)
```

`fit.MGGUM` *Fit the Multidimensional Generalized Graded Unfolding Model (MGGUM)*

Description

Fits a multidimensional generalization of the generalized graded unfolding model (GGUM) to polytomous response data. Unlike dominance (cumulative) IRT models, the MGGUM belongs to the ideal-point (unfolding) model family where the probability of endorsement is highest when the person and item locations match, decreasing as the person moves away in either direction. Two estimation backends are provided: Bayesian MCMC (Stan) and iStEM.

Usage

```
fit.MGGUM(
  data,
  D = NULL,
  Q.matrix = NULL,
  length.poly = NULL,
  method = c("iStEM", "stan"),
  control.model = NULL,
  control.method = NULL
)
```

Arguments

`data`	An $N \times I$ matrix of integer responses coded $0, 1, \ldots, K_i - 1$.
`D`	Integer; number of latent dimensions ($D \geq 1$). Default is `2`.
`Q.matrix`	An optional $I \times D$ matrix with entries $-1, 0, 1$. See section **Q-matrix sign convention**. Default generates a random signed matrix.
`length.poly`	Integer vector (length I) or scalar giving the number of categories per item. If `NULL`, inferred from the observed response maxima.
`method`	Estimation method: `"iStEM"` (default) or `"stan"`.
`control.model`	A named list of model-level hyperparameters for the GGUM unfolding model. Supported entries:
	`a.mu`, `a.sigma` Prior location and scale for $\log a_{id}$ (log-normal). Defaults: 0, 0.5. The log-normal prior ensures positivity of discrimination parameters. The zero mean-log encourages slopes near 1 unless the data strongly indicate otherwise.
	`delta.pos.mu`, `delta.pos.sigma` Prior mean and SD for δ_{id} when $q_{id} = 1$ (positive-side item locations; normal). Defaults: 1, 0.5. These priors regularize item locations on the positive side of each dimension.
	`delta.neg.mu`, `delta.neg.sigma` Prior mean and SD for δ_{id} when $q_{id} = -1$ (negative-side item locations; normal). Defaults: -1, 0.5. These priors regularize item locations on the negative side of each dimension.

`tau.mu`, `tau.sigma` Location and scale for the log-normal prior on positive threshold magnitudes $-\tau_{ik}$ for $k \geq 1$. Defaults: 0, 0.5. The thresholds satisfy $\tau_{i1} < \tau_{i2} < \cdots < \tau_{i,K_i-1} < 0$ with $\tau_{i0} = 0$ fixed for identification.

`theta.mu` Prior mean vector for $\boldsymbol{\theta}_j$. Default: `rep(0, D)`.

`L` Theta grid size per dimension for marginal log-likelihood computation and iStEM block Gibbs sampling. Default adapts to D.

`theta.lower`, `theta.upper` Bounds for the theta grid used by marginal log-likelihood computation and iStEM block Gibbs sampling. Defaults: -6, 6.

`control.method` A named list of method-specific tuning parameters. Common entries (used by both Stan and iStEM):

`cores` Number of CPU cores for parallel chains (Stan) or ignored (iStEM). Default: the number of `chains`.

`vis` Logical; if `TRUE` (default), prints progress information to the console.

`seed` Random seed for reproducibility. Default: a random integer.

Stan-specific entries:

`chains` Number of MCMC chains (default: 2).

`iter` Total iterations per chain (default: 5000).

`warmup` Warmup/burn-in iterations per chain (default: `iter / 2`).

`thin` Thinning interval (default: 1).

`init` Initial values: `"random"` (default) for uniform(-2, 2) initialization, or a list of initial values per chain.

`algorithm` MCMC algorithm: `"HMC"` (default), `"HMC"`, or `"Fixed_param"`.

`adapt_delta` Target average acceptance probability (NUTS; default: 0.95). Values closer to 1 reduce step size and improve sampling for difficult posteriors.

`max_treedepth` Maximum tree depth (NUTS; default: 10). Increase if "max treedepth exceeded" warnings appear.

`stepsize` Initial step size for the leapfrog integrator (auto-tuned by Stan if not set).

`int_time` Total integration time for HMC trajectories (only when `algorithm = "HMC"`).

`metric` Mass matrix type: `"unit_e"`, `"diag_e"` (default), or `"dense_e"`.

`adapt_engaged` Logical; if `TRUE` (default), warmup adaptation is enabled.

`adapt_init_buffer`, `adapt_term_buffer`, `adapt_window` Warmup adaptation scheduling parameters (defaults: 25, 50, 25).

iStEM-specific entries:

`M` Number of burn-in batches retained for Geweke convergence diagnosis (default: 10; must be ≥ 2).

`B` Batch size: MCMC iterations per batch (default: 20).

`burnin.maxitr` Maximum burn-in batches (default: 100).

`maxitr` Maximum total batches (default: 2000).

`eps1` Geweke z-score convergence threshold (default: 1.5). The burn-in phase ends when $\sum z^2/K < \epsilon_1$ or the MC error criterion is satisfied.

`eps2` Monte Carlo error tolerance (default: 0.4). The algorithm continues until $max(d_k \cdot N) < \epsilon_2$.

`frac1`, `frac2` Fractions for the Geweke diagnostic (defaults: 0.1, 0.5).

`corr.optim.maxit` Maximum L-BFGS-B iterations for the constrained unit-diagonal correlation update (default: 50).

`optim.maxit` Maximum L-BFGS-B iterations per item during the item-parameter update (default: 50). Increase if item optimization does not converge within this limit.

`fix.corr` Logical; if `TRUE`, the inter-trait correlation matrix is fixed to the identity (default: `FALSE`).

`estimate.se` Logical; if `TRUE` (default), standard errors are computed from the final Monte Carlo chain.

`delta.lower` Lower bound on $|\delta_{id}|$ (default: `1e-4`). Ensures item locations are bounded away from zero for active dimensions.

`tau.gap.lower`, `tau.gap.upper` Lower and upper bounds on the positive gaps used to construct ordered thresholds. Defaults are 1e-4 and 6.

Value

An object of class `"MGGUM"` containing:

`npar` Number of free parameters.

`method` `"stan"` or `"iStEM"`.

`theta` List with `est`, `se`, `Rhat` ($N \times D$).

`par` List with `est`, `se`, `Rhat`, `free` ($I \times (2D + K_{max})$). Columns: `a1..aD`, `delta1..deltaD`, `tau0..tau_{K_{max}-1}`.

`Corr` List with `est`, `se`, `Rhat` ($D \times D$).

`length.poly` Per-item category counts.

`logLik` Marginal log-likelihood (class `"logLik"`).

Model Specification

Let $Y_{ij} \in \{0, 1, \ldots, K_i - 1\}$ be the categorical response of person j to item i, with $K_i \geq 2$ categories. Let $\boldsymbol{\theta}_j$ denote the D-dimensional latent trait vector.

The category response probability implemented by `model.MGGUM`, iStEM, and Stan is the distance-based MGGUM form. Define

$$r_{ij} = \left[\sum_{d=1}^{D} a_{id}^2 \left(\theta_{jd} - \delta_{id} \right)^2 \right]^{1/2}, \qquad S_i = \sum_{d=1}^{D} a_{id}.$$

Let $\tau_{i0} = 0$ and $\psi_{ik} = S_i \sum_{v=0}^{k} \tau_{iv}$. With $M_i = 2K_i - 1$, the probability of category $k = 0, \ldots, K_i - 1$ is

$$P(Y_{ij} = k \mid \boldsymbol{\theta}_j) = \frac{\exp\{k r_{ij} - \psi_{ik}\} + \exp\{(M_i - k) r_{ij} - \psi_{ik}\}}{\sum_{r=0}^{K_i - 1} \left[\exp\{r r_{ij} - \psi_{ir}\} + \exp\{(M_i - r) r_{ij} - \psi_{ir}\} \right]}.$$

Here $a_{id} \geq 0$ are discrimination parameters, δ_{id} are item-location parameters (signed positive when $q_{id} = 1$, negative when $q_{id} = -1$), and threshold parameters τ_{ik} satisfying $\tau_{i0} = 0$ and $\tau_{i1} < \tau_{i2} < \cdots < \tau_{i,K_i-1} < 0$.

Q-matrix sign convention:

$q_{id} = 1$ Active dimension; $\delta_{id} > 0$ (item located on the positive side of dimension d).

$q_{id} = -1$ Active dimension; $\delta_{id} < 0$ (item located on the negative side of dimension d).

$q_{id} = 0$ Inactive dimension; $a_{id} = \delta_{id} = 0$.

The sign of q_{id} controls the item-location side, not the sign of the slope. Both a_{id} and $|\delta_{id}|$ reflect the item's relevance to dimension d.

Estimation Methods

Stan (`method = "stan"`)**:** Full Bayesian inference via HMC. All parameters are jointly sampled from the posterior distribution.

iStEM (`method = "iStEM"`)**:** Improved Stochastic EM with finite-grid block Gibbs person-sampling and item optimization via L-BFGS-B. The threshold parameters τ_{ik} are constrained to maintain monotonicity during optimization.

References

Roberts, J. S., Donoghue, J. R., & Laughlin, J. E. (2000). A general item response theory model for unfolding unidimensional polytomous responses. *Applied Psychological Measurement*, 24(1), 3–32. doi:10.1177/01466216000241001

Usami, S. (2011). Generalized graded unfolding model with structural equation for subject parameters. *Japanese Psychological Research*, 53(3), 221–232.

See Also

`sim.data.MGGUM`, `get.fit.index.MGGUM`, `logLik.MGGUM`

Examples

```
sim <- sim.data.MGGUM(N = 20, I = 6, D = 2, length.poly = 4)
fit <- fit.MGGUM(sim$response, D = 2, method = "iStEM",
                 control.method = list(
                   vis = FALSE, seed = 123,
                   M = 2, B = 2, burnin.maxitr = 2,
                   maxitr = 3, eps1 = 10, eps2 = 10,
                   estimate.se = FALSE))
head(fit$theta$est)
gof <- get.fit.index(fit)
summary(gof)
```

`fit.MGPCM` *Fit the Multidimensional Generalized Partial Credit Model (MGPCM)*

Description

Fits a multidimensional generalization of the partial credit model to polytomous (multi-category) response data. Two estimation backends are provided: full Bayesian inference via Hamiltonian Monte Carlo (Stan) and a fast stochastic-EM algorithm (iStEM).

Usage

```
fit.MGPCM(
  data,
  D = NULL,
  Q.matrix = NULL,
  length.poly = NULL,
  method = c("iStEM", "stan"),
  control.model = NULL,
  control.method = NULL
)
```

Arguments

`data`	An $N \times I$ matrix of integer responses coded $0, 1, \ldots, K_i - 1$. Rows index persons, columns index items.
`D`	Integer; number of latent dimensions ($D \geq 1$). Default is `2`.
`Q.matrix`	An optional $I \times D$ binary matrix. Entry $q_{id} = 1$ frees a_{id}; $q_{id} = 0$ fixes it to 0. Default uses a triangular identification structure.
`length.poly`	Optional integer scalar or length-I vector giving the number of categories per item. If `NULL`, counts are inferred from observed maxima; supply this argument when a valid category is unobserved in the sample.
`method`	Estimation method: `"iStEM"` (default) or `"stan"`.
`control.model`	A named list of model-level hyperparameters. Supported entries:

`a.mu`, `a.sigma` Prior location and scale for $\log a_{id}$ (log-normal). Defaults: 0.25, 0.25. Controls the prior mean and spread of the discrimination parameters across dimensions.

`d.mu`, `d.sigma` Prior mean and SD for free category intercepts d_{ik} ($k \geq 1$; normal). Defaults: 0, 1. The first category intercept $d_{i0} = 0$ is fixed for identification.

`theta.mu` Prior mean vector for $\boldsymbol{\theta}_j$. Default: `rep(0, D)`. A vector of length D specifying the prior mean for each latent dimension.

`L` Theta grid size per dimension for marginal log-likelihood computation and iStEM block Gibbs sampling. Default adapts to D (e.g., 61 for D = 1, 31 for D = 2, 15 for D = 3). Larger grids increase numerical precision at the cost of exponential growth in computation (L^D total nodes).

`theta.lower`, `theta.upper` Bounds for the theta grid used by marginal log-likelihood computation and iStEM block Gibbs sampling. Defaults: -6, 6.

`control.method` A named list of method-specific tuning parameters. Common entries (used by both Stan and iStEM):

`cores` Number of CPU cores for parallel chains (Stan) or ignored (iStEM). Default: the number of `chains`.

`vis` Logical; if `TRUE` (default), prints progress information to the console.

`seed` Random seed for reproducibility. Default: a random integer.

Stan-specific entries:

`chains` Number of MCMC chains (default: 2).

`iter` Total iterations per chain (default: 5000).

`warmup` Warmup/burn-in iterations per chain (default: `iter / 2`).

`thin` Thinning interval (default: 1).

`init` Initial values: `"random"` (default) for uniform(-2, 2) initialization, or a list of initial values per chain.

`algorithm` MCMC algorithm: `"HMC"` (default), `"HMC"`, or `"Fixed_param"`.

`adapt_delta` Target average acceptance probability (NUTS; default: 0.95). Values closer to 1 reduce step size and improve sampling for difficult posteriors.

`max_treedepth` Maximum tree depth (NUTS; default: 10). Increase if "max treedepth exceeded" warnings appear.

`stepsize` Initial step size for the leapfrog integrator (auto-tuned by Stan if not set).

`int_time` Total integration time for HMC trajectories (only when `algorithm = "HMC"`).

`metric` Mass matrix type: `"unit_e"`, `"diag_e"` (default), or `"dense_e"`.

`adapt_engaged` Logical; if `TRUE` (default), warmup adaptation is enabled.

`adapt_init_buffer`, `adapt_term_buffer`, `adapt_window` Warmup adaptation scheduling parameters (defaults: 25, 50, 25).

iStEM-specific entries:

`M` Number of burn-in batches retained for Geweke convergence diagnosis (default: 10; must be ≥ 2).

`B` Batch size: MCMC iterations per batch (default: 20).

`burnin.maxitr` Maximum burn-in batches (default: 100).

`maxitr` Maximum total batches (default: 2000).

`eps1` Geweke z-score convergence threshold (default: 1.5).

`eps2` Monte Carlo error tolerance (default: 0.4).

`frac1`, `frac2` Fractions for the Geweke diagnostic (defaults: 0.1, 0.5).

`corr.optim.maxit` Maximum L-BFGS-B iterations for the constrained unit-diagonal correlation update (default: 50).

`optim.maxit` Maximum L-BFGS-B iterations per item (default: 50).

`fix.corr` Logical; fix correlations to identity (default: `FALSE`).

`estimate.se` Logical; compute standard errors from final MC chain (default: `TRUE`).

`a.lower`, `a.upper` Bounds on a_{id} (defaults: 1e-4, 6).

`d.lower`, `d.upper` Bounds on free category intercepts d_{ik} for $k \geq 1$. Defaults are -8 and 8 in the iStEM backend.

Value

An object of class `"MGPCM"` with components:

`npar` Number of free parameters.

`method` `"stan"` or `"iStEM"`.

`theta` List with `est`, `se`, `Rhat` ($N \times D$).

`par` List with `est`, `se`, `Rhat`, `free` ($I \times (D + K_{max})$). Columns are `a1..aD`, `d0`, `d1`, `...`, `d_{K_{max}-1}`.

`Corr` List with `est`, `se`, `Rhat` ($D \times D$).

`length.poly` Integer vector of per-item category counts.

`logLik` Marginal log-likelihood (class `"logLik"`).

`call`, `arguments` Call and argument records.

Model Specification

Let $Y_{ij} \in \{0, 1, \ldots, K_i - 1\}$ denote the categorical response of person $j = 1, \ldots, N$ to item $i = 1, \ldots, I$, where $K_i \geq 2$ is the number of response categories for item i. Let $\boldsymbol{\theta}_j$ be the D-dimensional latent trait vector.

The category response probability implemented in both the C++ and Stan backends is a softmax over category scores:

$$P(Y_{ij} = k \mid \boldsymbol{\theta}_j) = \frac{\exp\{k\,\eta_{ij} + d_{ik}\}}{\sum_{r=0}^{K_i-1} \exp\{r\,\eta_{ij} + d_{ir}\}}, \qquad k = 0, 1, \ldots, K_i - 1,$$

where

$$\eta_{ij} = \sum_{d=1}^{D} a_{id}\,\theta_{jd}.$$

The discrimination parameters satisfy $a_{id} > 0$ when $q_{id} = 1$ and $a_{id} = 0$ when $q_{id} = 0$. The d_{ik} values are category intercepts, not cumulative step difficulties; $d_{i0} = 0$ is fixed for identification and $d_{i1}, \ldots, d_{i,K_i-1}$ are free.

This formulation nests the standard (unidimensional) generalized partial credit model (Muraki, 1992) when $D = 1$ and all $q_{i1} = 1$.

Prior distributions:

a_{id} **(free)** Log-normal: $\log a_{id} \sim N(\mu_a, \sigma_a^2)$

d_{ik} **(free)** Normal: $d_{ik} \sim N(\mu_d, \sigma_d^2)$ for $k \geq 1$; $d_{i0} = 0$ is fixed. The same normal penalty is used by Stan and by the iStEM item update when `use.prior = TRUE`.

$\boldsymbol{\theta}_j$ Multivariate normal: $\boldsymbol{\theta}_j \sim N_D(\mathbf{0}, \boldsymbol{\Sigma})$

Estimation Methods

Stan (`method = "stan"`)**:** Full Bayesian inference via HMC. The joint posterior is sampled with multiple chains, providing posterior means, standard deviations, and $\hat{R}$ diagnostics.

iStEM (`method = "iStEM"`)**:** Improved Stochastic EM alternating finite-grid block Gibbs person-sampling and item-parameter optimization (L-BFGS-B). Convergence monitored via Geweke diagnostics and batch-means MC error.

References

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176. doi:10.1177/014662169201600206

Yao, L., & Schwarz, R. D. (2006). A multidimensional partial credit model for polytomous data. *Applied Psychological Measurement*, 30(4), 295–318.

See Also

`sim.data.MGPCM`, `get.fit.index.MGPCM`, `logLik.MGPCM`, `rotate`

Examples

```
sim <- sim.data.MGPCM(N = 20, I = 6, D = 2, length.poly = 4)
fit <- fit.MGPCM(sim$response, D = 2, method = "iStEM",
                 control.method = list(
                   vis = FALSE, seed = 123,
                   M = 2, B = 2, burnin.maxitr = 2,
                   maxitr = 3, eps1 = 10, eps2 = 10,
                   estimate.se = FALSE))
head(fit$theta$est)
fit$par$est[1:5, ]
gof <- get.fit.index(fit)
summary(gof)
```

`fit.MIRT` *Fit the Multidimensional Item Response Theory (MIRT) Model*

Description

Fits a multidimensional extension of the 1PL, 2PL, 3PL, or 4PL item response model to binary response data. Two estimation backends are provided: full Bayesian inference via Hamiltonian Monte Carlo (Stan) and a fast stochastic-EM algorithm (iStEM) that scales to large data sets.

Usage

```
fit.MIRT(
  data,
  model = "2PL",
  D = NULL,
  Q.matrix = NULL,
  method = c("iStEM", "stan"),
  control.model = NULL,
  control.method = NULL
)
```

Arguments

`data`	An $N \times I$ matrix of binary responses coded as `0` (incorrect) and `1` (correct). Rows index persons, columns index items. Missing values are not allowed.
`model`	Character string specifying the model type. Accepts both internal codes (`"m1pl"`, `"m2pl"`, `"m3pl"`, `"m4pl"`) and user-friendly aliases (`"Rasch"`, `"1PL"`, `"2PL"`, `"3PL"`, `"4PL"`). Case-insensitive. Default is `"2PL"`.
`D`	Integer; number of latent dimensions ($D \geq 1$). If `NULL` (default), `D` is inferred from `Q.matrix`; if both are `NULL`, an error is raised. When `D = 1`, the model reduces to unidimensional IRT.
`Q.matrix`	An optional $I \times D$ binary matrix. Entry $q_{id} = 1$ indicates that the d-th dimension loads on item i. For 2PL–4PL models, $q_{id} = 1$ frees a_{id} and $q_{id} = 0$ fixes $a_{id} = 0$. For the current M1PL backend, slopes are fixed to 1 and the Q-matrix does not zero out slope entries. If `NULL` (default), a triangular identification pattern is used: all items load on all dimensions except the last D items, which follow a lower-triangular structure for rotational invariance resolution.
`method`	Estimation method: `"iStEM"` (fast stochastic EM for large data; default) or `"stan"` (full Bayesian HMC).
`control.model`	A named list of model-level hyperparameters. Supported entries:
	`a.mu`, `a.sigma` Prior location and scale for $\log a_{id}$ (log-normal). Defaults: 0.25, 0.25.
	`b.mu`, `b.sigma` Prior mean and SD for b_i (normal). Defaults: 0, 1.
	`c.mu`, `c.sigma` Uniform support bounds $[c_{\min}, c_{\max}]$ for c_i. Defaults: 0, 0.35.
	`d.mu`, `d.sigma` Uniform support bounds $[d_{\min}, d_{\max}]$ for d_i. Defaults: 0.65, 1.
	`theta.mu` Prior mean vector for $\boldsymbol{\theta}_j$. Default: `rep(0, D)`.
	`L` Theta grid size per dimension for marginal log-likelihood computation and iStEM block Gibbs sampling. Default adapts to D (e.g., 61 for D = 1, 31 for D = 2, 15 for D = 3).
	`theta.lower`, `theta.upper` Bounds for the theta grid used by marginal log-likelihood computation and iStEM block Gibbs sampling. Defaults: -6, 6.
	`use.prior` Logical (iStEM only). If `FALSE`, the item-prior penalty term is omitted from the item update, recovering an approximate maximum-likelihood item step. Default is `TRUE`.
`control.method`	A named list of method-specific tuning parameters. Common entries (used by both Stan and iStEM):

`cores` Number of CPU cores for parallel chains (Stan) or ignored (iStEM). Default: the number of `chains`.

`vis` Logical; if `TRUE` (default), prints progress information to the console.

`seed` Random seed for reproducibility. Default: a random integer.

Stan-specific entries:

`chains` Number of MCMC chains (default: 2).

`iter` Total iterations per chain (default: 5000).

`warmup` Warmup/burn-in iterations per chain (default: `iter / 2`).

`thin` Thinning interval (default: 1).

`init` Initial values: `"random"` (default) for uniform(-2, 2) initialization, or a list of initial values per chain.

`algorithm` MCMC algorithm: `"HMC"` (default), `"HMC"`, or `"Fixed_param"`.

`adapt_delta` Target average acceptance probability (NUTS; default: 0.95).

`max_treedepth` Maximum tree depth (NUTS; default: 10).

`stepsize` Initial step size for the leapfrog integrator. If not set, Stan determines an appropriate value automatically during warmup. Only applicable when `algorithm = "HMC"` or `"NUTS"`.

`int_time` Total integration time for each HMC leapfrog trajectory. The number of steps is `int_time / stepsize`. Only applicable when `algorithm = "HMC"`.

`metric` The mass matrix for HMC sampling. Can be a unit vector (`"unit_e"`), a diagonal matrix (`"diag_e"`), or a dense matrix (`"dense_e"`). Stan defaults to `"diag_e"`.

`adapt_engaged` Logical; if `TRUE` (default), the warmup adaptation is enabled. Set to `FALSE` to disable step-size and mass-matrix adaptation during warmup.

`adapt_init_buffer` Number of initial warmup iterations used for pure exploration before adaptation begins (default: 25).

`adapt_term_buffer` Number of final warmup iterations where adaptation is frozen so the sampler can converge to the stationary distribution (default: 50).

`adapt_window` Number of warmup iterations between adaptation updates. Larger values reduce the frequency of adaptation (default: 25).

iStEM-specific entries:

`M` Number of burn-in batches retained for convergence diagnosis (default: 10). Must be at least 2. Larger values improve the Geweke diagnostic stability but increase computation during burn-in.

`B` Batch size: number of stochastic EM iterations per batch (default: 20). Each iteration samples all persons' $\boldsymbol{\theta}_j$ blocks and updates all items.

`burnin.maxitr` Maximum number of burn-in batches (default: 100). If convergence criteria are not met before this limit, the algorithm proceeds with a warning. Increase this value if convergence warnings appear consistently.

`maxitr` Maximum number of total batches including post-burn-in iterations (default: 2000). The algorithm stops when either the MC error criterion is met or this limit is reached.

`eps1` Geweke z-score convergence threshold for the burn-in phase (default: 1.5). The burn-in phase ends when $\sum z^2/K < \epsilon_1$ or the MC error criterion is satisfied. Lower values enforce stricter convergence but prolong burn-in.

`eps2` Monte Carlo error tolerance for the final chain (default: 0.4). The algorithm continues sampling until $\max(d_k \cdot N) < \epsilon_2$, where d_k is the batch-means variance for each parameter. Lower values yield more precise estimates.

`frac1`, `frac2` Fractions used in the Geweke convergence diagnostic (defaults: 0.1, 0.5). `frac1` defines the proportion of the chain used for the early segment; `frac2` defines the proportion for the late segment. These follow standard practice from the **coda** package.

`corr.optim.maxit` Maximum L-BFGS-B iterations for the constrained unit-diagonal correlation update (default: 50).

`optim.maxit` Maximum L-BFGS-B iterations per item during the item-parameter update step (default: 50). Increase if item-level optimization warnings appear.

`fix.corr` Logical; if `TRUE`, the inter-trait correlation matrix is fixed to the identity during estimation (default: `FALSE`). Setting to `TRUE` enforces orthogonal latent dimensions.

`estimate.se` Logical; if `TRUE` (default), standard errors are computed as the batch-means-based standard deviation of the final Monte Carlo chain. Set to `FALSE` to skip SE computation (slightly faster).

`a.lower`, `a.upper` Bounds on each a_{id} (defaults: 1e-4, 6). For 2PL–4PL models, discrimination parameters are constrained to this interval during L-BFGS-B optimization.

`b.lower`, `b.upper` Bounds on b_i (defaults: -6, 6). Item difficulty/intercept parameters are constrained to this interval during optimization.

`c.lower`, `c.upper` Bounds on c_i for 3PL/4PL models. Defaults are inherited from the prior support bounds (`control.model$c.mu` = 0 and `control.model$c.sigma` = 0.35, respectively). Override these to impose tighter or wider bounds on the lower-asymptote parameters during optimization.

`d.lower`, `d.upper` Bounds on d_i for 4PL models. Defaults are inherited from the prior support bounds (`control.model$d.mu` = 0.65 and `control.model$d.sigma` = 1, respectively). Override these to constrain the upper-asymptote parameters during optimization.

Value

An object of class `"MIRT"` containing the following components:

`npar` Integer; number of free parameters (= freely estimated item parameters + free correlation elements).

`method` Character; `"stan"` or `"iStEM"`.

`theta` List with matrices `est`, `se`, `Rhat` ($N \times D$); person parameter estimates.

`par` List with matrices `est`, `se`, `Rhat`, `free` ($I \times (D + 3)$); item parameter arrays. Columns are `a1..aD`, `b`, `c`, `d`.

`Corr` List with matrices `est`, `se`, `Rhat` ($D \times D$); inter-trait correlation matrix.

`Q.matrix` The $I \times D$ Q-matrix used.

`stan.obj` The `stanfit` object (Stan only; `NULL` for iStEM).

`MCMC.obj` The list returned by `rstan::extract()` (Stan only; `NULL` for iStEM).

`logLik` The marginal log-likelihood computed via Gauss–Hermite-type quadrature (class `"logLik"`).

`call` The matched call.

`arguments` List of arguments used in fitting.

`iStEM` List of iStEM diagnostic quantities (iStEM only), including burn-in size, convergence flags, and theta grid length.

Model Specification

Let $Y_{ij} \in \{0, 1\}$ denote the binary response of person $j = 1, \ldots, N$ to item $i = 1, \ldots, I$, and let $\boldsymbol{\theta}_j = (\theta_{j1}, \ldots, \theta_{jD})'$ denote the D-dimensional latent trait vector. The item response function (IRF) for the four model variants is:

M1PL (Rasch / one-parameter logistic):

$$P(Y_{ij} = 1 \mid \boldsymbol{\theta}_j) = \frac{1}{1 + \exp\left[-\left(\sum_{d=1}^{D} a_{id}\,\theta_{jd} - b_i\right)\right]}, \qquad a_{id} = 1$$

The M1PL implementation fixes the slope to 1 and is restricted to $D = 1$; fixed unit slopes do not identify separate dimensions.

M2PL (two-parameter logistic):

$$P(Y_{ij} = 1 \mid \boldsymbol{\theta}_j) = \frac{1}{1 + \exp\left[-\left(\sum_{d=1}^{D} a_{id}\,\theta_{jd} - b_i\right)\right]}, \qquad a_{id} > 0 \text{ if } q_{id} = 1,\ a_{id} = 0 \text{ otherwise}$$

The discrimination (slope) parameters a_{id} are freely estimated subject to the Q-matrix pattern and a log-normal prior.

M3PL (three-parameter logistic):

$$P(Y_{ij} = 1 \mid \boldsymbol{\theta}_j) = c_i + (1 - c_i) \times \frac{1}{1 + \exp\left[-\left(\sum_{d=1}^{D} a_{id}\,\theta_{jd} - b_i\right)\right]}$$

where $c_i \in [0, 1)$ is the lower-asymptote (pseudo-guessing) parameter.

M4PL (four-parameter logistic):

$$P(Y_{ij} = 1 \mid \boldsymbol{\theta}_j) = c_i + (d_i - c_i) \times \frac{1}{1 + \exp\left[-\left(\sum_{d=1}^{D} a_{id}\,\theta_{jd} - b_i\right)\right]}$$

where $c_i \in [0, 1)$ is the lower asymptote and $d_i \in (0, 1]$ is the upper asymptote (1 - slippage).

In the unidimensional case ($D = 1$), these reduce to the standard 1PL–4PL models. For $D > 1$, the Q-matrix governs which dimensions load on each item, enabling both exploratory (default triangular identification) and confirmatory (user-specified Q-matrix) structures.

Prior distributions (Bayesian / MAP):

a_{id} **(free)** Log-normal: $\log a_{id} \sim N(\mu_a, \sigma_a^2)$. Defaults: $\mu_a = 0.25$, $\sigma_a = 0.25$.

b_i Normal: $b_i \sim N(\mu_b, \sigma_b^2)$. Defaults: $\mu_b = 0$, $\sigma_b = 1$.

c_i Uniform: $c_i \sim U(c_{\min}, c_{\max})$. Defaults: $c_{\min} = 0$, $c_{\max} = 0.35$.

d_i Uniform: $d_i \sim U(d_{\min}, d_{\max})$. Defaults: $d_{\min} = 0.65$, $d_{\max} = 1$.

$\boldsymbol{\theta}_j$ Multivariate normal: $\boldsymbol{\theta}_j \sim N_D(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma})$, with $\boldsymbol{\mu}_\theta = \mathbf{0}$ and $\boldsymbol{\Sigma}$ a correlation matrix.

Estimation Methods

Stan (`method = "stan"`): Full Bayesian inference via Hamiltonian Monte Carlo (NUTS/HMC). The joint posterior of all parameters is explored using multiple Markov chains. Output includes posterior means, standard deviations, and $\hat{R}$ convergence diagnostics for all parameters. Marginal log-likelihood is approximated via Gauss–Hermite-type quadrature over the latent space.

iStEM (`method = "iStEM"`): An improved Stochastic EM (iStEM) algorithm that alternates between (a) sampling $\boldsymbol{\theta}_j$ from its finite-grid full conditional as one D-dimensional block using the person's complete response likelihood and (b) maximizing the complete-data posterior for item parameters via L-BFGS-B. The inter-trait correlation matrix $\boldsymbol{\Sigma}$ is estimated by constrained unit-diagonal normal-likelihood optimization. Convergence is monitored using Geweke (1992) convergence diagnostics and batch-means Monte Carlo error estimates. Standard errors are obtained from the final Monte Carlo chain.

References

Reckase, M. D. (2009). *Multidimensional Item Response Theory*. Springer. doi:10.1007/9780387-899763

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord & M. R. Novick, *Statistical theories of mental test scores* (pp. 397–479). Addison-Wesley.

Barton, M. A., & Lord, F. M. (1981). An upper asymptote for the three-parameter logistic item-response model. *ETS Research Report Series*, 1981(1), i–21.

Geweke, J. (1992). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In J. M. Bernardo et al. (Eds.), *Bayesian Statistics 4* (pp. 169–193). Oxford University Press.

See Also

`good.of.fit` for goodness-of-fit evaluation, `get.fit.index.MIRT` for MIRT-specific fit indices, `sim.data.MIRT` for simulating data from this model, `rotate` for post-hoc rotation of MIRT solutions, `logLik.MIRT` for marginal log-likelihood extraction.

Examples

```
# Simulate data from a 2-dimensional 2PL model
sim <- sim.data.MIRT(N = 20, I = 6, D = 2, model = "m2pl")

# Fit via iStEM (fast, large-data friendly)
fit_istem <- fit.MIRT(sim$response, model = "m2pl", D = 2,
                      method = "iStEM",
                      control.method = list(
                        vis = FALSE, seed = 123,
```

```
                        M = 2, B = 2, burnin.maxitr = 2,
                        maxitr = 3, eps1 = 10, eps2 = 10,
                        estimate.se = FALSE))


# stan code, long time
# Fit via Stan (full Bayesian inference, computationally heavier)
fit_stan <- fit.MIRT(sim$response, model = "m2pl", D = 2,
                     method = "stan",
                     control.method = list(
                       chains = 1, iter = 200, warmup = 100,
                       cores = 1, seed = 123))


# Extract results
print(fit_istem$par$est)       # item parameter estimates
print(fit_istem$theta$est)     # person trait estimates
print(fit_istem$Corr$est)      # factor correlation matrix

# Compute goodness-of-fit indices
gof <- get.fit.index(fit_istem)
summary(gof)
```

`fit.TIRT` *Fit the Thurstonian Item Response Theory (TIRT) Model*

Description

Fits the Thurstonian IRT model for forced-choice data. Unlike the FCMIRT/FCGGUM models that operate on the item-level endorsement probabilities, the TIRT model works directly at the level of paired comparisons: each pairwise comparison is treated as a binary outcome governed by a latent difference process with a probit link.

Usage

```
fit.TIRT(
  data,
  Q.matrix,
  block.items = NULL,
  fc.type = NULL,
  method = c("iStEM", "stan"),
  control.model = NULL,
  control.method = NULL
)
```

Arguments

`data`	An $N \times B$ forced-choice data matrix with ranking strings (e.g., `"2>1>3"`).
`Q.matrix`	An $I \times D$ matrix with entries $-1, 0, 1$. Each row must have exactly one non-zero entry that assigns the statement to a specific dimension with a sign indicating the scoring direction.
`block.items`	A list of length B giving item indices in each block. Parsed from `data` if `NULL`.
`fc.type`	Forced-choice response type: `"RANK"` (default), `"MOLE"`, or `"PICK"`.
`method`	Estimation method: `"iStEM"` (default) or `"stan"`.
`control.model`	A named list of model-level hyperparameters for the Thurstonian IRT model. Supported entries:
	`lambda.alpha`, `lambda.beta` Lower and upper bounds for free loading magnitudes λ_i. Defaults: 0.40, 0.90. Stan uses $\lambda_i \sim U(\lambda_{\min}, \lambda_{\max})$, where these two controls supply $\lambda_{\min}$ and $\lambda_{\max}$; iStEM uses the same values as optimization bounds unless `lambda.lower` or `lambda.upper` is supplied in `control.method`.
	`psi.mu`, `psi.sigma` Prior mean and SD for uniqueness standard deviations ψ_i (normal, truncated to positive values). Defaults: 0.80, 0.30. The prior $\psi_i \sim N_+(0.80, 0.30^2)$ reflects the expectation that uniqueness SDs are moderate in size. Note that the model parameterizes the uniqueness variance ψ_i^2, so the prior on the SD scale is transformed accordingly.
	`gamma.mu`, `gamma.sigma` Prior mean and SD for pairwise intercepts γ_{ik} (normal). Defaults: 0, 0.8. The skew-symmetry constraint $\gamma_{ik} = -\gamma_{ki}$ is automatically enforced. Unobserved pairwise comparisons (in MOLE/PICK designs) have their γ parameters fixed to `gamma.mu`.
	`theta.mu` Mean vector for the iStEM normal kernel for $\boldsymbol{\theta}_j$. Default: `rep(0, D)`. The current Stan model accepts this value in its data list but centers $\boldsymbol{\theta}_j$ at zero.
	`L` Theta grid size per dimension for marginal log-likelihood computation and iStEM block Gibbs sampling. Default adapts to D.
	`theta.lower`, `theta.upper` Bounds for the theta grid used by marginal log-likelihood computation and iStEM block Gibbs sampling. Defaults: -6, 6.
`control.method`	A named list of method-specific tuning parameters. Common entries (used by both Stan and iStEM):
	`cores` Number of CPU cores for parallel chains (Stan) or ignored (iStEM). Default: the number of `chains`.
	`vis` Logical; if `TRUE` (default), prints progress information to the console.
	`seed` Random seed for reproducibility. Default: a random integer.
	Stan-specific entries:
	`chains` Number of MCMC chains (default: 2).
	`iter` Total iterations per chain (default: 5000).
	`warmup` Warmup/burn-in iterations per chain (default: `iter / 2`).
	`thin` Thinning interval (default: 1).
	`init` Initial values: `"random"` (default) for uniform(-2, 2) initialization, or a list of initial values per chain.

`algorithm` MCMC algorithm: `"HMC"` (default), `"HMC"`, or `"Fixed_param"`.

`adapt_delta` Target average acceptance probability (NUTS; default: 0.95). The TIRT model's pairwise probit likelihood typically mixes well; however, for high-dimensional models ($D > 5$) or models with many fixed parameters, consider increasing to 0.9.

`max_treedepth` Maximum tree depth (NUTS; default: 10). Increase if "max treedepth exceeded" warnings appear.

`stepsize` Initial step size for the leapfrog integrator (auto-tuned by Stan if not set).

`int_time` Total integration time for HMC trajectories (only when `algorithm = "HMC"`).

`metric` Mass matrix type: `"unit_e"`, `"diag_e"` (default), or `"dense_e"`.

`adapt_engaged` Logical; if `TRUE` (default), warmup adaptation is enabled.

`adapt_init_buffer`, `adapt_term_buffer`, `adapt_window` Warmup adaptation scheduling parameters (defaults: 25, 50, 25).

iStEM-specific entries:

`M` Number of burn-in batches retained for Geweke convergence diagnosis (default: 10; must be ≥ 2).

`B` Batch size: MCMC iterations per batch (default: 20).

`burnin.maxitr` Maximum burn-in batches (default: 100).

`maxitr` Maximum total batches (default: 2000).

`eps1` Geweke z-score convergence threshold for burn-in (default: 1.5).

`eps2` Monte Carlo error tolerance for the final chain (default: 0.4).

`frac1`, `frac2` Fractions for the Geweke diagnostic (defaults: 0.1, 0.5).

`corr.optim.maxit` Maximum L-BFGS-B iterations for the constrained unit-diagonal correlation update (default: 50).

`optim.maxit` Maximum L-BFGS-B iterations per parameter block during structural parameter updates (default: 50). Applies to the optimization of λ, ψ^2, and γ parameters.

`fix.corr` Logical; if `TRUE`, the inter-trait correlation matrix is fixed to the identity (default: `FALSE`). Useful when orthogonal dimensions are theoretically expected.

`estimate.se` Logical; if `TRUE` (default), standard errors are computed from the final Monte Carlo chain.

`lambda.lower`, `lambda.upper` Bounds on free loading magnitudes during iStEM optimization. Defaults are `lambda.alpha` and `lambda.beta` from `control.model`.

`psi.lower`, `psi.upper` Bounds on uniqueness standard deviations ψ_i during iStEM optimization (defaults: 1e-4 and 5). The reported `par` matrix stores ψ_i^2.

`gamma.lower`, `gamma.upper` Bounds on pairwise intercepts γ_{ik} during iStEM optimization (defaults: -6 and 6).

Value

An object of class `"TIRT"` containing:

`npar` Number of free parameters.

`method` `"stan"` or `"iStEM"`.

`theta` List with `est`, `se`, `Rhat` ($N \times D$).

`par` List with `est`, `se`, `Rhat`, `free` ($I \times (D+1)$). Columns include the loading λ_i on the design dimension and ψ_i^2.

`Corr` List with `est`, `se`, `Rhat` ($D \times D$).

`gamma.matrix` List with `est`, `se`, `Rhat` ($I \times I$); pairwise intercept matrix.

`pairs.matrix`, `pairs.value`, `response`, `response.total` Pairwise and response data structures.

`logLik` Marginal log-likelihood (class `"logLik"`).

Model Specification

Pairwise utility difference. For a pair of statements (i, k) in block b, the latent comparative judgment is:

$$y^*_{j,ik} = -\gamma_{ik} + \lambda_i \sum_{d=1}^{D} q_{id}\,\theta_{jd} - \lambda_k \sum_{d=1}^{D} q_{kd}\,\theta_{jd} + \varepsilon_{j,i} - \varepsilon_{j,k},$$

where:

- γ_{ik} is the pairwise intercept (unfolding threshold parameter). Skew-symmetry requires $\gamma_{ik} = -\gamma_{ki}$.
- $\lambda_i > 0$ is the loading magnitude of statement i; the sign of the statement direction is carried by q_{id}.
- $q_{id} \in \{-1, 0, 1\}$ is the Q-matrix entry; each row has exactly one non-zero entry.
- $\varepsilon_{j,i} \sim N(0, \psi_i^2)$ is the uniqueness (error) of statement i.

Binary pairwise response. The observed response is

$$Y_{j,ik} = \begin{cases} 1, & \text{if } y^*_{j,ik} \geq 0 \quad (\text{prefer } i \text{ over } k), \\ 0, & \text{otherwise.} \end{cases}$$

Under the normality of errors, the probability is:

$$P(Y_{j,ik} = 1 \mid \boldsymbol{\theta}_j) = \Phi\left(\frac{-\gamma_{ik} + \lambda_i\, \mathbf{q}_i' \boldsymbol{\theta}_j - \lambda_k\, \mathbf{q}_k' \boldsymbol{\theta}_j}{\sqrt{\psi_i^2 + \psi_k^2}} \right),$$

where $\Phi(\cdot)$ is the standard normal CDF.

Identification constraints. Several standard constraints are applied:

- **Case A** ($I_{\text{block}} = 2, D > 2$): All $\psi_i^2 = 0.5$ are fixed.
- **Case B** ($D = 2, I_{\text{block}} = 2$): The first block's λ_i are fixed to 0.80, all $\psi_i^2 = 0.5$ are fixed.
- **General case**: The last item in each block has $\psi_i^2 = 1.0$ fixed. Pairwise γ_{ik} parameters for comparisons that are never observed under the supplied RANK/MOLE/PICK design are fixed to `gamma.mu`.

Correlation structure. The inter-trait correlation matrix $\boldsymbol{\Sigma}$ is estimated as in the MIRT model. In the Stan file, $\boldsymbol{\theta}_j$ is centered at zero; the iStEM backend uses `control.model$theta.mu` as the mean in the normal kernel.

Estimation Methods

Stan (`method = "stan"`)**:** Full Bayesian inference via HMC. The probit likelihood is evaluated over all pairwise comparisons.

iStEM (`method = "iStEM"`)**:** Improved Stochastic EM. Person sampling uses finite-grid block Gibbs; structural parameters (λ, ψ^2, γ) updated via optimization or closed-form steps.

References

Brown, A., & Maydeu-Olivares, A. (2011). Item response modeling of forced-choice questionnaires. *Educational and Psychological Measurement*, 71(3), 460–502. doi:10.1177/0013164410375112

Maydeu-Olivares, A., & Brown, A. (2010). Item response modeling of paired comparison and ranking data. *Multivariate Behavioral Research*, 45(6), 935–974.

Thurstone, L. L. (1927). A law of comparative judgment. *Psychological Review*, 34(4), 273–286.

See Also

`sim.data.TIRT`, `get.fit.index.TIRT`, `logLik.TIRT`

Examples

```
sim <- sim.data.TIRT(N.person = 20, N.block = 3, I.block = 2,
                     D = 2, fc.type = "RANK")
fit <- fit.TIRT(sim$data, Q.matrix = sim$Q.matrix,
                block.items = sim$block.items,
                fc.type = "RANK", method = "iStEM",
                control.method = list(
                  vis = FALSE, seed = 123,
                  M = 2, B = 2, burnin.maxitr = 2,
                  maxitr = 3, eps1 = 10, eps2 = 10,
                  estimate.se = FALSE))
head(fit$theta$est)
cor(fit$theta$est, sim$theta)
gof <- get.fit.index(fit)
summary(gof)
```

`fitted` *S3 Methods: fitted*

Description

Extract model-implied response probabilities from fitted **ForceChoice** model objects. Returns the conditional probability of each observed response given each person's estimated latent trait $\hat{\boldsymbol{\theta}}_j$, evaluated via the model's response function.

Usage

```
## S3 method for class 'MIRT'
fitted(object, ...)

## S3 method for class 'MGPCM'
fitted(object, ...)

## S3 method for class 'MGGUM'
fitted(object, ...)

## S3 method for class 'FCMIRT'
fitted(object, ...)

## S3 method for class 'FCDCM'
fitted(object, ...)

## S3 method for class 'FCGDINA'
fitted(object, ...)

## S3 method for class 'FCGGUM'
fitted(object, ...)

## S3 method for class 'TIRT'
fitted(object, ...)
```

Arguments

`object`	A fitted model object of class `"MIRT"`, `"MGPCM"`, `"MGGUM"`, `"FCMIRT"`, `"FCDCM"`, `"FCGDINA"`, `"FCGGUM"`, or `"TIRT"`.
`...`	Additional arguments (currently ignored).

Details

For binary-response models, each entry is the endorsement probability $P(Y_{ij} = 1 \mid \hat{\boldsymbol{\theta}}_j)$. For polytomous models, probabilities are returned in stacked format (one column per category). For forced-choice models, each entry is the probability of the observed ranking pattern in the given block.

Value

A numeric matrix of model-implied probabilities. Dimensions:

- Binary models (MIRT, TIRT): $N \times I$
- Polytomous models (MGPCM, MGGUM): $N \times \sum K_i$ (stacked category probabilities)
- Forced-choice models (FCMIRT, FCGDINA, FCGGUM): $N \times \sum K_b$ (block-pattern probabilities)
- FCDCM: $N \times B$ (marginal block-choice probabilities after integrating over 2^D attribute patterns)

Methods (by class)

- `fitted(MIRT)`: Fitted probabilities for `MIRT` objects. Returns $N \times I$ matrix of endorsement probabilities.
- `fitted(MGPCM)`: Fitted probabilities for `MGPCM` objects. Returns stacked category probability matrix.
- `fitted(MGGUM)`: Fitted probabilities for `MGGUM` objects.
- `fitted(FCMIRT)`: Fitted probabilities for `FCMIRT` objects. Returns block-pattern probability matrix.
- `fitted(FCDCM)`: Fitted probabilities for `FCDCM` objects. Returns $N \times B$ matrix of marginal block-choice probabilities after integrating over 2^D attribute patterns.
- `fitted(FCGDINA)`: Fitted probabilities for `FCGDINA` objects. Returns block-pattern probability matrix after integrating over posterior attribute classes.
- `fitted(FCGGUM)`: Fitted probabilities for `FCGGUM` objects.
- `fitted(TIRT)`: Fitted probabilities for `TIRT` objects. Returns $N \times I_{pairs}$ matrix of pairwise comparison probabilities.

`forced_choice_data_conversion`

Forced-Choice Data Conversion Formats

Description

Documents the three data objects used by the forced-choice conversion helpers: human-readable `data`, block definitions `block.items`, and model-ready `response` matrices. This overview is intended as the starting point before using `get.response.from.data`, `get.data.from.response`, the TIRT-specific converters, or the FCDCM-specific converters.

Details

The package uses two layers of forced-choice representation:

`data` An $N \times B$ matrix or data frame that is easy to read and edit. Rows are persons and columns are forced-choice blocks. A cell records the observed ordering with item labels separated by `">"`. For example, `"3>4>5"` means item 3 was ranked above item 4, which was ranked above item 5.

`block.items` A list of length B. Element b gives the item labels appearing in column b of `data`. This object is the bridge between column order in the data file and item rows in model parameter matrices such as `Q.matrix`.

`response` The compact numeric representation consumed by model fitting code. Its meaning depends on the model family. For FCGGUM, FCMIRT, and FCGDINA, `response` is an $N \times B$ matrix of 1-based pattern indices. For TIRT, `response` is an $N \times P$ matrix of pairwise binary outcomes. For FCDCM, `response` is an $N \times B$ binary matrix for two-item blocks.

Choosing a Converter

FCGGUM, FCMIRT, and FCGDINA Use `get.response.from.data` to convert character strings to pattern indices, and `get.data.from.response` for the inverse conversion.

TIRT Use `get.response.from.data.TIRT` and `get.data.from.response.TIRT`. TIRT expands each block to pairwise comparisons, so its `response` columns are pairs rather than block-level pattern numbers.

FCDCM Use `get.response.from.data.FCDCM` and `get.data.from.response.FCDCM`. Each block has exactly two statements, so `response = 1` means the first statement in `block.items[[b]]` was preferred and `response = 0` means the second statement was preferred.

Recovering `block.items` Use `get.block.items.from.data` for generic multi-item forced-choice data and `get.block.items.from.data.FCDCM` for FCDCM pair data. Recovery is only possible for item labels that actually appear in the observed strings.

Core Conventions

- The *b*th element of `block.items` always corresponds to the *b*th column of `data` or `response`.
- `fc.type = "RANK"` expects a complete ranking such as `"3>4>5"` for a three-item block.
- `fc.type = "MOLE"` expects a most-least string such as `"3>5"`: item 3 is most preferred and item 5 is least preferred.
- `fc.type = "PICK"` expects a single most-preferred item such as `"3"`.
- Generic FC pattern-index responses are 1-based. They are row numbers in the deterministic pattern matrix generated internally; see `generate_fc_permutation_patterns` for the public pattern generator.
- TIRT data should use consecutive item labels across blocks, for example `list(1:2, 3:5, 6:8)`. This is the format returned by `sim.data.TIRT` and expected by the TIRT conversion routines.

See Also

`get.block.items.from.data`, `get.response.from.data`, `get.data.from.response`, `get.response.from.data.TIRT`, `get.data.from.response.TIRT`, `get.response.from.data.FCDCM`, `get.data.from.response.FCDCM`

Examples

```
# Generic FC conversion: response is a block-level pattern index.
block.items <- list(1:2, 3:5)
dat <- data.frame(
  block.1 = c("1>2", "2>1"),
  block.2 = c("3>4>5", "5>4>3"),
  stringsAsFactors = FALSE
)
resp <- get.response.from.data(dat, block.items, fc.type = "RANK")
resp
get.data.from.response(resp, block.items, fc.type = "RANK")

# TIRT conversion: response is pairwise 0/1 data.
tirt <- get.response.from.data.TIRT(dat, block.items, fc.type = "RANK")
```

```
tirt$response
get.data.from.response.TIRT(
  tirt$response, block.items, fc.type = "RANK",
  pairs.value = tirt$pairs.value
)

# FCDCM conversion: response is binary within each two-item block.
fcdcm.items <- list(c(1L, 3L), c(2L, 4L))
fcdcm.dat <- data.frame(
  block.1 = c("1>3", "3>1"),
  block.2 = c("2>4", "4>2"),
  stringsAsFactors = FALSE
)
fcdcm.resp <- get.response.from.data.FCDCM(fcdcm.dat, fcdcm.items)
fcdcm.resp$response
get.data.from.response.FCDCM(fcdcm.resp$response, fcdcm.items)
```

`get.block.items.from.data` *Extract Block Items from Forced-Choice Data*

Description

Infers the item composition of each forced-choice block from a character data matrix. Each cell should encode the within-block item ordering (e.g. `"2>1>3"` for a rank-3 block). The function parses all non-missing cells per block and returns sorted, unique item indices.

This function is the recommended way to recover `block.items` when only character-rank data are available. It works for all forced-choice types (`"RANK"`, `"MOLE"`, `"PICK"`) as long as every item appears at least once across persons.

Usage

```
get.block.items.from.data(data)
```

Arguments

`data`	An $N \times B$ character matrix (or data frame) of forced-choice responses. Each entry encodes an item ordering using integer item indices separated by `">"` (e.g. `"3>1>2"`). Missing values (`NA`) are permitted and skipped.

Details

The returned `block.items` list defines the block layout used by the other converters. If `data` has B columns, the result has B elements, and result element b belongs to data column b. Item labels are treated as global item indices; they are not recoded to local positions.

The function scans observed strings only. Therefore, for partial-response formats, it can only recover items that appear in the observed partial strings:

- For "RANK", a complete ranking normally contains every item in the block, so one valid row can be sufficient.
- For "MOLE", only most- and least-preferred items appear in each cell. An item that is never chosen as most or least cannot be recovered from data alone.
- For "PICK", only picked items appear. Items that are never picked cannot be recovered from data alone.

In those partial-response cases, pass a complete user-defined block.items list to the fitting or conversion function when the raw data do not mention every item.

Value

A list of length B. Each element is an integer vector of global item indices appearing in that block, sorted in ascending order.

See Also

get.block.items.from.data.FCDCM for the FCDCM (binary-pair) variant.

get.response.from.data to convert character data to an integer pattern-index matrix.

Examples

```
# 3 persons, 2 blocks.
# Block 1 contains items 1 and 2; block 2 contains items 3, 4, and 5.
dat <- data.frame(
  block.1 = c("1>2", "2>1", "1>2"),
  block.2 = c("3>4>5", "5>4>3", "4>3>5"),
  stringsAsFactors = FALSE
)
get.block.items.from.data(dat)
```

---

get.block.items.from.data.FCDCM

Extract Block Items from FCDCM Data

Description

Recovers the two-item composition of each FCDCM block from a character data matrix of "a>b" / "b>a" preference strings. Because every FCDCM block always compares exactly two statements, this function only needs to inspect the first non-missing row of each column.

Usage

```
get.block.items.from.data.FCDCM(data)
```

Arguments

`data`	An $N \times B$ character matrix (or data frame) of binary-preference strings. Each cell must be of the form `"a>b"` or `"b>a"` where `a` and `b` are global statement indices. Missing values are skipped.

Details

The returned item pair is sorted in ascending order. For example, both `"1>3"` and `"3>1"` imply `c(1L, 3L)` for that block. This sorted pair is then the default coding direction used by `get.response.from.data.FCDCM`: response value 1 means the first element of the pair was preferred, and response value 0 means the second element was preferred.

This helper assumes the two compared statements are fixed within each column. If a column accidentally mixes different item pairs, only the first parseable non-missing cell determines the returned pair, so such data should be cleaned before conversion.

Value

A list of length B. Each element is a two-element integer vector `c(a, b)` with `a < b`.

See Also

`get.block.items.from.data` for the generic multi-item variant used by FCGGUM, FCMIRT, and TIRT.

`get.response.from.data.FCDCM` to convert FCDCM data strings to a 0/1 response matrix.

Examples

```
dat <- data.frame(
  b1 = c("1>3", "3>1", "1>3"),
  b2 = c("2>4", "2>4", "4>2"),
  stringsAsFactors = FALSE
)
get.block.items.from.data.FCDCM(dat)
```

`get.data.from.response` *Convert an Integer Response (Pattern-Index) Matrix to FC Data Strings*

Description

Converts an integer matrix of pattern indices back to human-readable forced-choice data strings. This is the generic converter used by **FCGGUM**, **FCMIRT**, and **FCGDINA** where each column of the response matrix is an index into the block's observed permutation-pattern list.

Usage

```
get.data.from.response(response, block.items, fc.type = "RANK")
```

Arguments

`response`	An $N \times B$ integer matrix of pattern indices. Each entry is an integer between 1 and the number of observed patterns for that block.
`block.items`	A list of length B giving the global item indices in each forced-choice block.
`fc.type`	Character vector of length B (or a scalar recycled to all blocks): `"RANK"`, `"MOLE"`, or `"PICK"`.

Value

A data frame with N rows and B columns. Column names are `"block.1"`, ..., `"block.B"`. Cell values are character strings: full rankings (`"2>5>8"`) for `"RANK"`, `"best>worst"` pairs for `"MOLE"`, and single integers for `"PICK"`.

Pattern Generation and Numbering

For a block with K items, the set of possible response patterns depends on the forced-choice type:

RANK All $K!$ full permutations of the K items are generated. The patterns are listed in lexicographic order of the within-block item positions. For example, a 3-item block with items `c(2,5,8)` produces $3! = 6$ patterns: `(2,5,8)`, `(2,8,5)`, `(5,2,8)`, `(5,8,2)`, `(8,2,5)`, `(8,5,2)`. Pattern index 1 means the first pattern was observed.

MOLE For a K-item block, $K(K-1)$ most-least observation patterns are possible: K choices for the most preferred item times $K-1$ choices for the least preferred item. Pattern (a, b) means item a was most preferred and item b was least preferred.

PICK For a K-item block, exactly K patterns exist: one for each possible most-preferred item.

In all cases the pattern index (an integer from 1 to the number of observed patterns) is the row number in the matrix returned by the internal permutation helper for that block. The same deterministic pattern construction is exposed by `generate_fc_permutation_patterns`.

This format is a compact block-level encoding. One response column equals one forced-choice block, even when the block contains more than two items. This is the representation expected by the generic forced-choice model families FCGGUM, FCMIRT, and FCGDINA. It should not be confused with the TIRT pairwise format, where one block usually expands to several pair columns.

See Also

`get.response.from.data` for the reverse conversion.

`generate_fc_permutation_patterns` for pattern generation.

`get.data.from.response.TIRT` for the TIRT-specific converter.

Examples

```
# 2 persons, 2 RANK blocks.
items <- list(1:2, 3:5)
resp <- cbind(c(1, 2), c(1, 6))
get.data.from.response(resp, items, fc.type = "RANK")

# Mixed block formats: first block is most-least, second block is pick.
```

```
mixed.items <- list(1:3, 4:6)
mixed.resp <- cbind(c(1, 6), c(1, 3))
get.data.from.response(
  mixed.resp, mixed.items, fc.type = c("MOLE", "PICK")
)
```

`get.data.from.response.FCDCM` *Convert an FCDCM Response Matrix to FC Data Strings*

Description

Converts a binary FCDCM response matrix (0/1 or 1/2 coding) back to human-readable `"a>b"` / `"b>a"` preference strings. Each block compares exactly two items; the response indicates which item was selected.

Usage

```
get.data.from.response.FCDCM(response, block.items)
```

Arguments

`response`	An $N \times B$ integer matrix coded as 0/1 or 1/2, where B is the number of FCDCM blocks.
`block.items`	A list of length B. Each element is an integer vector of length 2 giving the two global statement indices compared in that block.

Value

A data frame with N rows and B columns. Each cell is a character string `"a>b"` or `"b>a"`.

Response Coding

Two coding conventions are accepted:

- **0/1 coding**: 1 means the first-named item in `block.items` was preferred; 0 means the second-named item was preferred.
- **1/2 coding**: 1 means the first-named item; 2 means the second-named item. This is auto-detected and converted.

The output always uses `"a>b"` format where `a` and `b` are the two item indices in the comparison order.

The direction is controlled by `block.items`. If `block.items[[1]] = c(1L, 3L)`, then response value 1 becomes `"1>3"` and response value 0 becomes `"3>1"`. With 1/2 coding, value 1 has the same meaning as above and value 2 means the second item. A response matrix must use one coding convention consistently; mixed `0/1/2` values are rejected.

See Also

`get.response.from.data.FCDCM` for the reverse conversion.

`get.data.from.response` for the generic multi-item converter.

Examples

```
items <- list(c(1L, 3L), c(2L, 4L))
resp <- cbind(c(1, 0, 1), c(0, 1, 1))
dat <- get.data.from.response.FCDCM(resp, items)
dat
get.response.from.data.FCDCM(dat, items)$response

# The same preferences can also be supplied as 1/2 responses.
resp12 <- cbind(c(1, 2, 1), c(2, 1, 1))
get.data.from.response.FCDCM(resp12, items)
```

---

get.data.from.response.TIRT

Convert TIRT Pairwise-Response Matrix to Forced-Choice Data Strings

---

Description

Converts a Thurstonian IRT pairwise-response matrix back to the human-readable forced-choice strings used in `data`. This is the inverse of `get.response.from.data.TIRT`.

Usage

```
get.data.from.response.TIRT(
  response,
  block.items,
  fc.type = NULL,
  pairs.value = NULL
)
```

Arguments

`response`	An $N \times P$ integer matrix of pairwise binary responses. Entries must be 1 (first item in the pair preferred) or 0 (second item in the pair preferred).
`block.items`	A list of length B giving the consecutive item labels in each forced-choice block.
`fc.type`	Character vector of length B (or a scalar recycled to all blocks): `"RANK"`, `"MOLE"`, or `"PICK"`.
`pairs.value`	A list of length N, where each element is itself a list of length B giving the pair-definition matrix for each block. Required when any block uses `"MOLE"` or `"PICK"`. For all-`"RANK"` data this argument may be omitted because the pair definitions are fixed by `block.items`.

Details

TIRT differs from the generic FC converters because the model is expressed through pairwise comparisons. The input `response` is therefore not a block-level pattern-index matrix. It is an $N \times P$ binary matrix, where each column is one observed item pair and each entry is coded:

- 1: the first item in that pair was preferred;
- 0: the second item in that pair was preferred.

The number of response columns contributed by a block with K items is determined by `fc.type`:

RANK All $K(K-1)/2$ unordered pairs are observed. Pair columns are generated in the order $(1,2),(1,3),\ldots,(K-1,K)$ using the item labels in `block.items[[b]]`.

MOLE Only the pairs needed to identify the most- and least-preferred items are retained. The block contributes $2K-3$ response columns per person.

PICK Only the pairs involving the picked item are retained. The block contributes $K-1$ response columns per person.

For `"MOLE"` and `"PICK"`, the identities of the retained pairs can vary by person. The `pairs.value` object records those person-specific pair definitions and is required to reconstruct data strings without ambiguity. It is normally copied directly from the `pairs.value` component returned by `get.response.from.data.TIRT`.

TIRT conversion assumes consecutive item labels across blocks, for example `list(1:2, 3:5)`. This is the layout returned by `sim.data.TIRT`.

Value

A data frame with N rows and B columns. Column names are `"block.1"`, `"block.2"`, etc. Cell values are character strings encoding the within-block ordering: full rankings for `"RANK"`, `"best>worst"` for `"MOLE"`, and single integers for `"PICK"`.

See Also

`get.response.from.data.TIRT` for the reverse conversion.

`get.data.from.response` for the generic (non-TIRT) converter used by FCGGUM and FCMIRT.

Examples

```
# 2 RANK blocks: block 1 has items 1,2; block 2 has items 3,4,5.
items <- list(1:2, 3:5)
resp <- cbind(
  c(1, 0),            # block 1 pair: 1 vs 2
  c(1, 0),            # block 2 pair: 3 vs 4
  c(1, 0),            # block 2 pair: 3 vs 5
  c(1, 0)             # block 2 pair: 4 vs 5
)
get.data.from.response.TIRT(resp, items, fc.type = "RANK")
```

get.fit.index.FCDCM *Goodness-of-Fit Indices for Fitted Models*

Description

Generic function to compute a comprehensive suite of model-fit diagnostics for fitted ForceChoice model objects. The function dispatches on the class of the fitted model and returns an object of class "good.of.fit" containing likelihood-based, limited-information, and residual-based diagnostics.

Usage

```
## S3 method for class 'FCDCM'
get.fit.index(object, ...)

## S3 method for class 'FCGDINA'
get.fit.index(object, ...)

## S3 method for class 'FCGGUM'
get.fit.index(object, ...)

## S3 method for class 'FCMIRT'
get.fit.index(object, ...)

## S3 method for class 'MGGUM'
get.fit.index(object, ...)

## S3 method for class 'MGPCM'
get.fit.index(object, ...)

## S3 method for class 'MIRT'
get.fit.index(object, ...)

## S3 method for class 'TIRT'
get.fit.index(object, ...)
```

Arguments

object	A fitted model object (class "MIRT", "MGPCM", "MGGUM", "FCMIRT", "FCDCM", "FCGGUM", or "TIRT").
...	Additional arguments passed to good.of.fit. Common options include alpha (tail probability for RMSEA CI; default 0.05 gives a 90% CI), bivariate.groups (exclude pairs within the same group from bivariate moments), and n.boot (bootstrap replicates).

Value

An object of class "good.of.fit". Use summary.good.of.fit and print.good.of.fit for formatted output. Key components:

- LogLik, deviance, AIC, AICc, BIC, CAIC, SABIC, HQIC
- M2.value, df, p.value
- RMSEA2, RMSEA.lower, RMSEA.upper, RMSEA.p.close
- SRMSR
- CFI, TLI, IFI
- McFadden.R2, CoxSnell.R2, Nagelkerke.R2
- q3.mean, q3.abs.max, q3.adjusted.abs.max
- posterior.entropy, posterior.entropy.normalized

Methods (by class)

- get.fit.index(FCDCM): FCDCM model: binary block responses with quadrature over higher-order θ and exact marginalization over 2^D attribute patterns.
- get.fit.index(FCGDINA): FCGDINA model: forced-choice CDM with nominal binary expansion; within-block pairs excluded.
- get.fit.index(FCGGUM): FCGGUM model: forced-choice unfolding with nominal binary expansion; within-block pairs excluded.
- get.fit.index(FCMIRT): FCMIRT model: forced-choice ranking data expanded to nominal binary indicators; within-block pairs excluded from bivariate moments.
- get.fit.index(MGGUM): MGGUM model: polytomous unfolding responses with M2* category-collapsed construction.
- get.fit.index(MGPCM): MGPCM model: polytomous responses with M2* category-collapsed construction.
- get.fit.index(MIRT): MIRT model: binary responses with Gauss–Hermite quadrature over D-dimensional latent space.
- get.fit.index(TIRT): TIRT model: pairwise binary responses with within-block pairs excluded via bivariate.groups.

Limited-Information M2 Framework

Following Maydeu-Olivares and Joe (2005, 2006), the test statistic uses first- and second-order marginal moments rather than the full contingency table, making it suitable for sparse high-dimensional responses.

Let $\boldsymbol{\pi}(\boldsymbol{\psi})$ be the vector of M model-implied univariate and bivariate moments, $\mathbf{p}$ the observed moments, and $\boldsymbol{\Delta} = \partial\boldsymbol{\pi}/\partial\boldsymbol{\psi}'$ the Jacobian. The M2 statistic is:

$$M_2 = N\,\mathbf{e}'\hat{\mathbf{C}}_2\mathbf{e}, \qquad \mathbf{e} = \mathbf{p} - \boldsymbol{\pi}(\hat{\boldsymbol{\psi}}),$$

where $\hat{\mathbf{C}}_2 = \boldsymbol{\Delta}_c(\boldsymbol{\Delta}_c'\hat{\boldsymbol{\Xi}}_2\boldsymbol{\Delta}_c)^{-1}\boldsymbol{\Delta}_c'$ with $\boldsymbol{\Delta}_c$ the orthogonal complement of $\boldsymbol{\Delta}$. Under the null, $M_2 \sim \chi^2_{M-p}$ asymptotically.

Fit Index Formulas

Information criteria (Akaike 1974; Schwarz 1978; Bozdogan 1987; Sclove 1987; Hannan & Quinn 1979):

$$\begin{aligned}
\text{AIC} &= -2\ell + 2p, \\
\text{AICc} &= \text{AIC} + \frac{2p(p+1)}{N-p-1}, \\
\text{BIC} &= -2\ell + p\log N, \\
\text{CAIC} &= -2\ell + p(\log N + 1), \\
\text{SABIC} &= -2\ell + p\log\!\Big(\frac{N+2}{24}\Big), \\
\text{HQIC} &= -2\ell + 2p\log(\log N).
\end{aligned}$$

RMSEA (Browne & Cudeck 1993):

$$\text{RMSEA} = \sqrt{\max\!\left(0,\ \frac{M_2 - df}{N \cdot df}\right)}, \qquad df = M - p.$$

Confidence intervals via non-central χ^2 inversion; close-fit test $H_0 : \text{RMSEA} \le 0.05$.

SRMSR:

$$\text{SRMSR} = \sqrt{\frac{1}{J}\sum_{i<j}(r_{ij}^{\text{obs}} - r_{ij}^{\text{model}})^2}.$$

CFI / TLI / IFI (Bentler 1990; Tucker & Lewis 1973; Bollen 1989):

$$\begin{aligned}
\text{CFI} &= 1 - \frac{\max(M_2 - df, 0)}{\max(M_2^{\text{null}} - df^{\text{null}},\ M_2 - df,\ 0)}, \\
\text{TLI} &= \frac{M_2^{\text{null}}/df^{\text{null}} - M_2/df}{M_2^{\text{null}}/df^{\text{null}} - 1}, \\
\text{IFI} &= \frac{M_2^{\text{null}} - M_2}{M_2^{\text{null}} - df}.
\end{aligned}$$

Pseudo-R^2:

$$\begin{aligned}
R^2_{\text{McF}} &= 1 - \ell/\ell_0, \\
R^2_{\text{CS}} &= 1 - \exp\!\big(\tfrac{2}{N}(\ell_0 - \ell)\big), \\
R^2_{\text{N}} &= R^2_{\text{CS}}/\big(1 - \exp(2\ell_0/N)\big), \\
R^2_{\text{AN}} &= G^2/(G^2 + N), \quad G^2 = D_0 - D.
\end{aligned}$$

Yen's Q3 (Yen 1984):

$$Q_{3,ik} = \text{Corr}(r_{ji}, r_{jk}), \qquad r_{ji} = Y_{ji} - E[Y_{ji} \mid \hat{\boldsymbol{\theta}}_j].$$

Posterior diagnostics (Celeux & Soromenho 1996):

$$E = -\frac{1}{N}\sum_{j=1}^{N}\sum_{c=1}^{Q}\hat{\tau}_{jc}\log\hat{\tau}_{jc}, \qquad E_{\text{norm}} = 1 - E/\log Q.$$

Heuristic Guidelines

- RMSEA: ≤ 0.05 close, ≤ 0.08 reasonable
- SRMSR: ≤ 0.08 acceptable (Hu & Bentler 1999)
- CFI / TLI: ≥ 0.95 good, ≥ 0.90 acceptable
- Normalized entropy: ≥ 0.70 acceptable separation

References

Akaike, H. (1974). *IEEE Trans. Autom. Control*, 19, 716–723. Bozdogan, H. (1987). *Psychometrika*, 52, 345–370. Browne, M. W., & Cudeck, R. (1993). In Bollen & Long (Eds.), *Testing structural equation models* (pp. 136–162). Sage. Celeux, G., & Soromenho, G. (1996). *J. Classification*, 13, 195–212. Hu, L. T., & Bentler, P. M. (1999). *Struct. Equ. Modeling*, 6, 1–55. Maydeu-Olivares, A., & Joe, H. (2005). *JASA*, 100, 1009–1020. Maydeu-Olivares, A., & Joe, H. (2006). *Psychometrika*, 71, 713–732. Schwarz, G. (1978). *Ann. Statist.*, 6, 461–464. Yen, W. M. (1984). *Appl. Psychol. Meas.*, 8, 125–145.

See Also

`good.of.fit` for the underlying engine, `summary.good.of.fit` for formatted summary tables.

Examples

```
sim <- sim.data.MIRT(N = 20, I = 6, D = 2, model = "m2pl")
fit <- fit.MIRT(
  sim$response, model = "m2pl", D = 2, method = "iStEM",
  control.method = list(
    vis = FALSE, seed = 123,
    M = 2, B = 2, burnin.maxitr = 2,
    maxitr = 3, eps1 = 10, eps2 = 10,
    estimate.se = FALSE)
)
gof <- get.fit.index(fit)
summary(gof)

# Custom RMSEA confidence level (99%)
gof99 <- get.fit.index(fit, alpha = 0.005)
summary(gof99)
```

`get.log_lik` *Extract pointwise log-likelihood matrix*

Description

For Stan-fitted models, returns the posterior draws of pointwise log-likelihood (an $S \times N_{\text{obs}}$ matrix for most models, or $S \times N$ for FCDCM and FCGDINA). For EM and iStEM fits, returns `NULL`.

Usage

```
get.log_lik(object, ...)

## Default S3 method:
get.log_lik(object, ...)
```

Arguments

`object`	A fitted ForceChoice model object.
`...`	Additional arguments (currently ignored).

Details

The returned matrix can be passed directly to `loo` or `waic` for model comparison.

Value

An $S \times N$ matrix of pointwise log-likelihood draws (Stan), or `NULL` (EM / iStEM).

Methods (by class)

- `get.log_lik(default)`: Default method; returns `object$log_lik` when present and `NULL` otherwise.

Examples

```
# stan code, long time
sim <- sim.data.MIRT(N = 20, I = 6, D = 2, model = "2PL")
fit <- fit.MIRT(sim$response, model = "2PL", D = 2, method = "stan")
log_lik <- get.log_lik(fit)
loo::loo(log_lik)
loo::waic(log_lik)
```

`get.response.from.data` *Convert FC Data Strings to an Integer Response (Pattern-Index) Matrix*

Description

Parses human-readable forced-choice data strings into integer pattern indices. This is the generic converter used by **FCGGUM**, **FCMIRT**, and **FCGDINA**. Each data cell is matched against the full set of permutation/observation patterns for that block, and the resulting row index is stored in the response matrix.

Usage

```
get.response.from.data(data, block.items, fc.type = "RANK")
```

Arguments

`data`	An $N \times B$ character matrix or data frame. Cells encode within-block item orderings as global indices separated by `">"` (e.g. `"3>1>2"` for RANK, `"3>2"` for MOLE, `"3"` for PICK).
`block.items`	A list of length B giving the global item indices in each forced-choice block.
`fc.type`	Character vector of length B (or a scalar recycled to all blocks): `"RANK"`, `"MOLE"`, or `"PICK"`.

Value

An $N \times B$ integer matrix. Each entry is a 1-based pattern index (row number in the block's pattern matrix). Column names are `"block.1"`, ..., `"block.B"`.

Pattern Matching

The conversion first generates all possible observation patterns for each block via the internal permutation helper:

- **RANK** (K items): all $K!$ full rankings, ordered lexicographically by within-block position.
- **MOLE**: $K(K-1)$ most-least observation patterns.
- **PICK**: K best-only patterns.

Each data cell string is parsed into an integer vector of global item indices, which is then matched (as an ordered sequence) against the pattern matrix. The 1-based row index of the match becomes the response value. Pattern ordering is deterministic and reproducible.

A non-missing cell that cannot be matched to a valid pattern is returned as `NA_integer_`. This is useful for data-screening workflows: after conversion, check `anyNA(response)` before passing the response matrix to a fitting function.

See Also

`get.data.from.response` for the reverse conversion.

`generate_fc_permutation_patterns` for pattern generation details.

`get.response.from.data.TIRT` for the TIRT-specific converter.

Examples

```
# 2 persons, 2 RANK blocks.
items <- list(1:2, 3:5)
dat <- data.frame(
  block.1 = c("1>2", "2>1"),
  block.2 = c("3>4>5", "5>4>3"),
  stringsAsFactors = FALSE
)
```

```
resp <- get.response.from.data(dat, items, fc.type = "RANK")
resp
get.data.from.response(resp, items, fc.type = "RANK")

# Mixed block formats: most-least plus pick.
dat.mixed <- data.frame(
  mole = c("1>3", "3>2"),
  pick = c("4", "6"),
  stringsAsFactors = FALSE
)
mixed.items <- list(1:3, 4:6)
get.response.from.data(
  dat.mixed, mixed.items, fc.type = c("MOLE", "PICK")
)
```

`get.response.from.data.FCDCM` *Convert FCDCM Data Strings to a Binary Response Matrix*

Description

Parses FCDCM preference strings (`"a>b"` / `"b>a"`) into a 0/1 binary response matrix. Each block compares exactly two items.

Usage

```
get.response.from.data.FCDCM(data, block.items)
```

Arguments

`data`	An $N \times B$ character matrix or data frame. Each cell must be of the form `"a>b"` or `"b>a"` where `a` and `b` are the two item indices for that block.
`block.items`	A list of length B. Each element is an integer vector of length 2 giving the two global statement indices compared in that block, in the order used for the 0/1 coding.

Details

For each block, only two strings are valid: `"a>b"` and `"b>a"`, where `a = block.items[[b]][1]` and `b = block.items[[b]][2]`. Leading and trailing whitespace is ignored. Missing values or strings that do not match the block-specific item pair produce an error, because FCDCM fitting requires a complete binary response matrix.

If `block.items` was obtained from `get.block.items.from.data.FCDCM`, the pair is sorted in ascending order. In that common workflow, response value 1 means the smaller item index was preferred and response value 0 means the larger item index was preferred.

Value

A named list with component `response`, an $N \times B$ integer matrix coded 0/1.

Response Coding

The output uses 0/1 coding: 1 indicates preference for the first-named item in `block.items[[b]]`; 0 indicates preference for the second-named item. This coding is the native format expected by the FCDCM model estimation functions.

See Also

`get.data.from.response.FCDCM` for the reverse conversion.

`get.response.from.data` for the generic multi-item converter.

Examples

```
items <- list(c(1L, 3L), c(2L, 4L))
dat <- data.frame(
  b1 = c("1>3", "3>1", "1>3"),
  b2 = c("2>4", "2>4", "4>2"),
  stringsAsFactors = FALSE
)
get.response.from.data.FCDCM(dat, items)
get.data.from.response.FCDCM(
  get.response.from.data.FCDCM(dat, items)$response, items
)
```

`get.response.from.data.TIRT` *Convert TIRT Forced-Choice Data to Pairwise-Response Matrix*

Description

Parses human-readable forced-choice data strings (full rankings, most-least choices, or best-only picks) into the Thurstonian IRT pairwise binary response format. Each unordered item pair within a block receives a 1/0 outcome indicating which item was preferred.

Usage

```
get.response.from.data.TIRT(data, block.items, fc.type = NULL)
```

Arguments

`data`	An $N \times B$ character matrix or data frame of forced-choice responses. Each cell encodes the within-block item ordering using item labels from `block.items` separated by `">"` (e.g. `"3>1>2"` for RANK, `"3>2"` for MOLE, `"3"` for PICK).
`block.items`	A list of length B giving the unique item labels in each forced-choice block.
`fc.type`	Character vector of length B (or a scalar recycled to all blocks): `"RANK"`, `"MOLE"`, or `"PICK"`.

Details

This function expands each block column in `data` into one or more pairwise response columns. Pair columns are grouped by block in the same order as `block.items`. Within a pair column, the coding is always 1 if the first item in the pair is preferred, and 0 if the second item is preferred.

Item labels must match the values supplied in `block.items`; they do not need to be consecutive. For example, if `block.items = list(c(2, 7), c(10, 20, 30))`, the second block can contain `"10>20>30"`, `"30>10"`, or `"20"`, depending on `fc.type`.

Value

A list with two components:

response An $N \times P$ integer matrix of pairwise binary outcomes (1/0). For block b, the number of contributed columns is $K_b(K_b-1)/2$ for RANK, $2K_b-3$ for MOLE, and K_b-1 for PICK.

pairs.value A list of length N, each element being a list of length B of integer pair-definition matrices. Required as input for `get.data.from.response.TIRT`.

Pattern Naming and Pair Encoding

For a block with K items, the number of pair columns depends on `fc.type`: $K(K-1)/2$ for `"RANK"`, $2K-3$ for `"MOLE"`, and $K-1$ for `"PICK"`. For each retained pair:

- **RANK**: the item ranked earlier in the ordering is preferred. For `block.items[[b]] = 3:5`, pair order is `(3,4)`, `(3,5)`, `(4,5)`. The string `"3>4>5"` therefore produces pairwise responses `1, 1, 1`.
- **MOLE**: only the most- and least-preferred items are identified (e.g. `"3>2"` in a 3-item block). The remaining items are treated as tied in the middle, producing preference relations consistent with the partial ordering.
- **PICK**: only the most-preferred item is identified (e.g. `"3"` in a 3-item block). All other items are treated as equally less preferred.

The `pairs.value` component records the actual pair matrices used for each person and block. It is essential for converting MOLE/PICK responses back to `data`, because the retained pair set can differ across persons.

See Also

`get.data.from.response.TIRT` for the reverse conversion.

`get.response.from.data` for the generic (non-TIRT) converter used by FCGGUM and FCMIRT.

Examples

```
# 2 persons, 2 RANK blocks.
dat <- data.frame(
  block.1 = c("1>2", "2>1"),
  block.2 = c("3>4>5", "5>4>3"),
  stringsAsFactors = FALSE
)
items <- list(1:2, 3:5)
```

```
result <- get.response.from.data.TIRT(dat, items, fc.type = "RANK")
result$response
result$pairs.value[[1]]
get.data.from.response.TIRT(
  result$response, items, fc.type = "RANK",
  pairs.value = result$pairs.value
)

# Mixed partial-response blocks.
dat.partial <- data.frame(
  mole = c("1>3", "2>1"),
  pick = c("4", "5"),
  stringsAsFactors = FALSE
)
items.partial <- list(1:3, 4:6)
partial <- get.response.from.data.TIRT(
  dat.partial, items.partial, fc.type = c("MOLE", "PICK")
)
partial$response
get.data.from.response.TIRT(
  partial$response, items.partial, fc.type = c("MOLE", "PICK"),
  pairs.value = partial$pairs.value
)
```

`good.of.fit` *Goodness-of-Fit Indices for Latent Probability Models*

Description

Computes a comprehensive suite of model-fit diagnostics for probability models used in IRT, CDM, latent class, and forced-choice applications. Indices include likelihood-based information criteria, limited-information goodness-of-fit statistics (M2, RMSEA, SRMSR), incremental/comparative fit indices (CFI, TLI), pseudo-R^2 measures, residual diagnostics, local-dependence statistics (Yen's Q3), and posterior classification diagnostics.

Usage

```
good.of.fit(
  par.vec,
  loglik.fun,
  prob.fun,
  response,
  npar,
  pi,
  pi.fun = NULL,
  alpha = 0.05,
  response.type = c("auto", "binary", "polytomous", "nominal", "ordinary"),
  response.K = NULL,
  bivariate.groups = NULL,
```

```
  nominal.groups = NULL,
  exclusive.groups = NULL,
  n.boot = 1000L,
  ...
)
```

Arguments

par.vec	Numeric vector of free model parameters.
loglik.fun	Function returning the fitted model log-likelihood.
prob.fun	Function returning conditional response probabilities at the latent support points.
response	An $N \times I$ response matrix (binary, polytomous, or nominal coding depending on response.type).
npar	Integer; number of free parameters in the model.
pi	Numeric vector of latent support weights (summing to 1).
pi.fun	Optional function returning latent support weights given par.vec and .... If NULL, pi is used as-is.
alpha	Tail probability for the RMSEA confidence interval. Default 0.05 yields a 90% CI (two-sided 2α). Use 0.025 for 95% or 0.005 for 99%.
response.type	Character string specifying the response coding: "auto" (default; detects from probability dimensions), "binary" (each column is a binary indicator), "polytomous" (stacked category probabilities, requires response.K), or "nominal" (expanded binary indicators with mutually exclusive categories within groups, requires nominal.groups).
response.K	Integer vector of length I giving the number of categories for each response column. Required when response.type = "polytomous".
bivariate.groups	Optional integer vector of length I; response column pairs within the same group are excluded from bivariate limited-information moments. Useful for forced-choice or testlet designs where within-block comparisons are structural zeros.
nominal.groups	Integer vector of length matching the number of expanded binary indicators for response.type = "nominal". Indicators sharing the same group value belong to the same original nominal item/block.
exclusive.groups	Deprecated alias for nominal.groups.
n.boot	Integer; number of bootstrap replicates for bootstrapped diagnostics. Default 1000; set to 0 to skip.
...	Additional arguments passed to loglik.fun, prob.fun, and pi.fun.

Value

An object of class "good.of.fit" (a list). Key components:

LogLik, deviance Log-likelihood and deviance.

npar, N, nitems Parameter count, sample size, number of items/indicators.

`AIC`, `AICc`, `BIC`, `CAIC`, `SABIC`, `HQIC` Information criteria.

`M2.value`, `df`, `p.value` M2 statistic, degrees of freedom, asymptotic p-value.

`RMSEA2`, `RMSEA.lower`, `RMSEA.upper`, `RMSEA.CI.level`, `RMSEA.p.close` RMSEA estimate, confidence interval, interval level, close-fit test p-value.

`SRMSR`, `RMSR` Standardized and raw root mean square residuals.

`CFI`, `TLI`, `IFI` Incremental fit indices.

`McFadden.R2`, `Nagelkerke.R2`, `CoxSnell.R2`, `AldrichNelson.R2`, `VeallZimmermann.R2` Pseudo-R^2 measures.

`residuals`, `residuals.standardized`, `correlation.residuals` Residual vectors.

`moments.observed`, `moments.predicted` First- and second-order marginal moments.

`q3.mean`, `q3.abs.max`, `q3.adjusted.abs.max`, `q3.p95` Yen's Q3 local dependence statistics.

`mean.max.posterior`, `median.max.posterior`, `min.max.posterior` Posterior classification diagnostics.

`null.LogLik`, `null.M2.value`, `null.df` Null (independence) model diagnostics.

`LRT`, `LRT.df`, `LRT.p` Likelihood ratio test against the null model.

Likelihood and Information Criteria

Let $\ell = \log L(\hat{\psi})$ be the maximized log-likelihood, N the sample size, and p the number of free parameters (`npar`).

Deviance:

$$D = -2\ell$$

AIC (Akaike, 1974):

$$\text{AIC} = D + 2p$$

AICc (Hurvich & Tsai, 1989), small-sample correction:

$$\text{AICc} = \text{AIC} + \frac{2p(p+1)}{N-p-1}, \quad N > p+1$$

BIC (Schwarz, 1978):

$$\text{BIC} = D + p \log N$$

CAIC (Bozdogan, 1987), consistent AIC:

$$\text{CAIC} = D + p(\log N + 1)$$

SABIC (Sclove, 1987), sample-size adjusted BIC:

$$\text{SABIC} = D + p \log\left(\frac{N+2}{24}\right)$$

HQIC (Hannan & Quinn, 1979):

$$\text{HQIC} = D + 2p \log(\log N), \quad N > e$$

Limited-Information M2 Family

The limited-information framework (Maydeu-Olivares & Joe, 2005, 2006) uses first- and second-order marginal moments rather than the full I-way contingency table. This is essential for sparse high-dimensional categorical responses where full-information tests break down.

Let $\mathbf{m}(\boldsymbol{\psi})$ be the vector of model-implied univariate and bivariate moments (dimension M), and let $\hat{\boldsymbol{\Xi}}_2$ be their asymptotic covariance matrix under the model.

M2 statistic:

$$M_2 = N\,\mathbf{e}'\hat{\mathbf{C}}_2\mathbf{e},$$

where $\mathbf{e} = \mathbf{p} - \boldsymbol{\pi}(\hat{\boldsymbol{\psi}})$ is the vector of marginal residuals, and $\hat{\mathbf{C}}_2 = \boldsymbol{\Delta}_c(\boldsymbol{\Delta}_c'\hat{\boldsymbol{\Xi}}_2\boldsymbol{\Delta}_c)^{-1}\boldsymbol{\Delta}_c'$ with $\boldsymbol{\Delta}_c$ the orthogonal complement of the Jacobian matrix $\boldsymbol{\Delta} = \partial\boldsymbol{\pi}/\partial\boldsymbol{\psi}'$.

Under correct model specification, $M_2 \sim \chi^2_{M-p}$ asymptotically.

RMSEA (Root Mean Square Error of Approximation):

$$\text{RMSEA} = \sqrt{\max\left(0,\ \frac{M_2 - df}{N \cdot df}\right)}, \qquad df = M - p$$

Confidence intervals are obtained via non-central χ^2 inversion. The close-fit test evaluates H_0 : RMSEA ≤ 0.05.

SRMSR (Standardized Root Mean Square Residual):

$$\text{SRMSR} = \sqrt{\frac{1}{J}\sum_{i<j}(r_{ij}^{\text{obs}} - r_{ij}^{\text{model}})^2},$$

where r_{ij} are the observed and model-implied correlation residuals, summed over J unique off-diagonal pairs.

McDonald's NCI (Non-Centrality Index):

$$\text{NCI} = \exp\left(-\frac{\max(M_2 - df, 0)}{2N}\right)$$

Incremental / Comparative Fit Indices

CFI (Comparative Fit Index):

$$\text{CFI} = 1 - \frac{\max(M_2 - df, 0)}{\max(M_2^{\text{null}} - df^{\text{null}}, M_2 - df, 0)},$$

where the null (independence) model has M_2^{null} with df^{null} degrees of freedom.

TLI (Tucker–Lewis Index, a.k.a. NNFI):

$$\text{TLI} = \frac{M_2^{\text{null}}/df^{\text{null}} - M_2/df}{M_2^{\text{null}}/df^{\text{null}} - 1}$$

IFI (Incremental Fit Index, a.k.a. BFI):

$$\text{IFI} = \frac{M_2^{\text{null}} - M_2}{M_2^{\text{null}} - df}$$

Pseudo-R^2 Measures

McFadden's R^2:

$$R^2_{\text{McF}} = 1 - \frac{\ell}{\ell_0}$$

McFadden's adjusted R^2:

$$R^2_{\text{McF,adj}} = 1 - \frac{\ell - p}{\ell_0}$$

Cox–Snell R^2:

$$R^2_{\text{CS}} = 1 - \exp\left(\frac{2}{N}(\ell_0 - \ell)\right)$$

Nagelkerke R^2:

$$R^2_{\text{N}} = \frac{R^2_{\text{CS}}}{1 - \exp(2\ell_0/N)}$$

Aldrich–Nelson R^2:

$$R^2_{\text{AN}} = \frac{G^2}{G^2 + N}, \quad G^2 = D_0 - D$$

Veall–Zimmermann R^2:

$$R^2_{\text{VZ}} = R^2_{\text{AN}} \times \frac{-2\ell_0 + N}{-2\ell_0}$$

where ℓ_0 is the null-model (independence) log-likelihood and D_0 the corresponding deviance.

Yen's Q3 Local Dependence

For each pair of items (i, k), the residual correlation is

$$Q_{3,ik} = \text{Corr}(r_{ji}, r_{jk}), \qquad r_{ji} = Y_{ji} - E[Y_{ji} \mid \hat{\boldsymbol{\theta}}_j],$$

where residuals are computed from posterior expected scores. Adjusted Q3 values subtract the mean Q3 across all pairs to center the distribution (Christensen, Makransky, & Horton, 2017).

Heuristic Guidelines

The `print.summary.good.of.fit()` method reports descriptive recommended ranges based on common conventions. These are heuristics, not decision rules; fit cutoffs depend on model family, dimensionality, category sparsity, local dependence, and sample size.

- **RMSEA**: ≤ 0.05 close fit; ≤ 0.08 reasonable; ≤ 0.10 mediocre (Browne & Cudeck, 1993)
- **SRMSR**: ≤ 0.08 acceptable (Hu & Bentler, 1999)
- **CFI / TLI**: ≥ 0.95 good; ≥ 0.90 acceptable

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, 716–723.

Bozdogan, H. (1987). Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52, 345–370. doi:10.1007/BF02294361

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.

Christensen, K. B., Makransky, G., & Horton, M. C. (2017). Critical values for Yen's Q3. *Applied Psychological Measurement*, 41, 178–194. doi:10.1177/0146621616677520

Hannan, E. J., & Quinn, B. G. (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society: Series B*, 41, 190–195.

Hu, L. T., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis. *Structural Equation Modeling*, 6, 1–55. doi:10.1080/10705519909540118

Hurvich, C. M., & Tsai, C. L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76, 297–307. doi:10.1093/biomet/76.2.297

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods*, 1, 130–149.

Maydeu-Olivares, A., & Joe, H. (2005). Limited- and full-information estimation and goodness-of-fit testing in 2^n contingency tables: A unified framework. *Journal of the American Statistical Association*, 100, 1009–1020.

Maydeu-Olivares, A., & Joe, H. (2006). Limited information goodness-of-fit testing in multidimensional contingency tables. *Psychometrika*, 71, 713–732. doi:10.1007/s1133600512959

McFadden, D. (1974). Conditional logit analysis of qualitative choice behavior. In P. Zarembka (Ed.), *Frontiers in econometrics* (pp. 105–142). Academic Press.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6, 461–464. doi:10.1214/aos/1176344136

Sclove, S. L. (1987). Application of model-selection criteria to some problems in multivariate analysis. *Psychometrika*, 52, 333–343.

Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three-parameter logistic model. *Applied Psychological Measurement*, 8, 125–145. doi:10.1177/014662168400800201

See Also

`get.fit.index` for model-specific wrappers, `summary.good.of.fit` for formatted fit tables, `print.good.of.fit` for the print method.

nobs *S3 Methods: nobs*

Description

Extracts the number of observations (persons) from a fitted **ForceChoice** model object. Returns the number of rows in the response matrix used for model fitting.

Usage

```
## S3 method for class 'MIRT'
nobs(object, ...)

## S3 method for class 'MGPCM'
nobs(object, ...)

## S3 method for class 'MGGUM'
nobs(object, ...)

## S3 method for class 'FCMIRT'
nobs(object, ...)

## S3 method for class 'FCDCM'
nobs(object, ...)

## S3 method for class 'FCGDINA'
nobs(object, ...)

## S3 method for class 'FCGGUM'
nobs(object, ...)

## S3 method for class 'TIRT'
nobs(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Additional arguments (currently ignored).

Value

An integer: the sample size N.

Methods (by class)

- `nobs(MIRT)`: Number of observations (persons) for `MIRT` objects.
- `nobs(MGPCM)`: Number of observations for `MGPCM` objects.

- nobs(MGGUM): Number of observations for MGGUM objects.
- nobs(FCMIRT): Number of observations for FCMIRT objects.
- nobs(FCDCM): Number of observations for FCDCM objects.
- nobs(FCGDINA): Number of observations for FCGDINA objects.
- nobs(FCGGUM): Number of observations for FCGGUM objects.
- nobs(TIRT): Number of observations for TIRT objects.

---

plot *S3 Methods: plot*

Description

Draws log-likelihood trace plots for fitted **ForceChoice** model objects. Stan, iStEM, and EM fits are visualised through the same convergence target: the total log-likelihood recorded across iterations of the corresponding estimation algorithm.

Usage

```
## S3 method for class 'MIRT'
plot(x, y = NULL, ...)

## S3 method for class 'MGPCM'
plot(x, y = NULL, ...)

## S3 method for class 'MGGUM'
plot(x, y = NULL, ...)

## S3 method for class 'FCMIRT'
plot(x, y = NULL, ...)

## S3 method for class 'FCDCM'
plot(x, y = NULL, ...)

## S3 method for class 'FCGGUM'
plot(x, y = NULL, ...)

## S3 method for class 'FCGDINA'
plot(x, y = NULL, ...)

## S3 method for class 'TIRT'
plot(x, y = NULL, ...)
```

Arguments

x	A fitted model object of class "MIRT", "MGPCM", "MGGUM", "FCMIRT", "FCDCM", "FCGGUM", "FCGDINA", or "TIRT".
y	Ignored.
...	Ignored.

Details

The horizontal axis is the saved Stan iteration, iStEM iteration, or EM iteration. The vertical axis is the summed log-likelihood. Stan plots use log_lik or log_lik_group from generated quantities and draw all chains in one panel. iStEM plots use the iteration-level iStEM$logLik.trace, and the stable chain is the final M * B iStEM iterations. EM plots use EM$logLik.trace. For Stan and iStEM fits, a vertical dashed line marks the stable-chain boundary. EM fits do not have a stable-chain boundary.

Value

Invisibly returns NULL. Plots are produced as a side effect on the current graphics device.

Methods (by class)

- plot(MIRT): Log-likelihood trace plot for MIRT objects.
- plot(MGPCM): Log-likelihood trace plot for MGPCM objects.
- plot(MGGUM): Log-likelihood trace plot for MGGUM objects.
- plot(FCMIRT): Log-likelihood trace plot for FCMIRT objects.
- plot(FCDCM): Log-likelihood trace plot for FCDCM objects.
- plot(FCGGUM): Log-likelihood trace plot for FCGGUM objects.
- plot(FCGDINA): Log-likelihood trace plot for FCGDINA objects.
- plot(TIRT): Log-likelihood trace plot for TIRT objects.

---

predict *S3 Methods: predict*

Description

Predict response probabilities from fitted **ForceChoice** model objects. Given a fitted model and optionally new person parameter (latent trait) values, returns the model-implied response probabilities. When newdata is not provided, predictions are made at the estimated latent trait values for each person.

Usage

```
## S3 method for class 'MIRT'
predict(object, newdata = NULL, type = c("probability", "response"), ...)

## S3 method for class 'MGPCM'
predict(object, newdata = NULL, type = c("probability", "response"), ...)

## S3 method for class 'MGGUM'
predict(object, newdata = NULL, type = c("probability", "response"), ...)

## S3 method for class 'FCMIRT'
predict(object, newdata = NULL, type = c("probability", "response"), ...)

## S3 method for class 'FCDCM'
predict(object, newdata = NULL, type = c("probability", "response"), ...)

## S3 method for class 'FCGDINA'
predict(object, newdata = NULL, type = c("probability", "response"), ...)

## S3 method for class 'FCGGUM'
predict(object, newdata = NULL, type = c("probability", "response"), ...)

## S3 method for class 'TIRT'
predict(object, newdata = NULL, type = c("probability", "response"), ...)
```

Arguments

`object`	A fitted model object of class `"MIRT"`, `"MGPCM"`, `"MGGUM"`, `"FCMIRT"`, `"FCDCM"`, `"FCGDINA"`, `"FCGGUM"`, or `"TIRT"`.
`newdata`	An optional $M \times D$ matrix of latent trait values at which to compute predictions. If `NULL` (default), predictions use the estimated `object$theta$est`.
`type`	Character; `"probability"` (default) returns response probabilities, `"response"` returns expected category or simulated binary responses (random draws).
`...`	Additional arguments (currently ignored).

Value

`type = "probability"` For binary models (MIRT, TIRT): an $M \times I$ matrix of endorsement probabilities. For polytomous models (MGPCM, MGGUM): an $M \times \sum K_i$ matrix of stacked category probabilities. For forced-choice models (FCMIRT, FCGGUM): an $M \times \sum K_b$ matrix of block-pattern probabilities. For FCDCM: an $M \times B$ matrix of marginal block-choice probabilities.

`type = "response"` For binary models: an $M \times I$ matrix of simulated 0/1 responses. For polytomous models: an $M \times I$ matrix of category indices. For forced-choice models: an $M \times B$ matrix of chosen pattern indices.

Methods (by class)

- `predict(MIRT)`: Predict response probabilities or simulated responses from `MIRT` objects.
- `predict(MGPCM)`: Predict response probabilities from `MGPCM` objects.
- `predict(MGGUM)`: Predict response probabilities from `MGGUM` objects.
- `predict(FCMIRT)`: Predict response probabilities from `FCMIRT` objects.
- `predict(FCDCM)`: Predict response probabilities from `FCDCM` objects.
- `predict(FCGDINA)`: Predict response probabilities from `FCGDINA` objects. For `newdata`, provide either binary alpha profiles or class probability rows.
- `predict(FCGGUM)`: Predict response probabilities from `FCGGUM` objects.
- `predict(TIRT)`: Predict response probabilities from `TIRT` objects.

`print` *S3 Methods: print*

Description

Provides user-friendly, formatted console output for objects generated by the **ForceChoice** package. This generic function dispatches to class-specific methods that display concise summaries of fitted models, simulation datasets, goodness-of-fit indices, and rotated solutions. Designed for interactive use and quick diagnostic inspection.

Usage

```
## S3 method for class 'good.of.fit'
print(x, digits = 4, ...)

## S3 method for class 'summary.good.of.fit'
print(x, ...)

## S3 method for class 'MIRT'
print(x, ...)

## S3 method for class 'summary.MIRT'
print(x, ...)

## S3 method for class 'MGPCM'
print(x, ...)

## S3 method for class 'summary.MGPCM'
print(x, ...)

## S3 method for class 'MGGUM'
print(x, ...)
```

```
## S3 method for class 'summary.MGGUM'
print(x, ...)

## S3 method for class 'FCMIRT'
print(x, ...)

## S3 method for class 'summary.FCMIRT'
print(x, ...)

## S3 method for class 'FCDCM'
print(x, ...)

## S3 method for class 'summary.FCDCM'
print(x, ...)

## S3 method for class 'FCGDINA'
print(x, ...)

## S3 method for class 'summary.FCGDINA'
print(x, ...)

## S3 method for class 'FCGGUM'
print(x, ...)

## S3 method for class 'summary.FCGGUM'
print(x, ...)

## S3 method for class 'TIRT'
print(x, ...)

## S3 method for class 'summary.TIRT'
print(x, ...)

## S3 method for class 'data.MIRT'
print(x, ...)

## S3 method for class 'data.MGPCM'
print(x, ...)

## S3 method for class 'data.MGGUM'
print(x, ...)

## S3 method for class 'data.FCMIRT'
print(x, ...)

## S3 method for class 'data.FCDCM'
print(x, ...)
```

```
## S3 method for class 'data.FCGDINA'
print(x, ...)

## S3 method for class 'data.FCGGUM'
print(x, ...)

## S3 method for class 'data.TIRT'
print(x, ...)
```

Arguments

x	An object of one of the following classes: • Fitted model objects: fit.MIRT ("MIRT"), fit.MGPCM ("MGPCM"), fit.MGGUM ("MGGUM"), fit.FCMIRT ("FCMIRT"), fit.FCDCM ("FCDCM"), fit.FCGGUM ("FCGGUM"), fit.TIRT ("TIRT") • Goodness-of-fit objects: good.of.fit ("good.of.fit") • Summary objects: summary.MIRT, summary.MGPCM, summary.MGGUM, summary.FCMIRT, summary.FCDCM, summary.FCGGUM, summary.TIRT, summary.good.of.fit
digits	Number of decimal places for numeric output (default: 4). Used by print.good.of.fit.
...	Additional arguments passed to methods (currently ignored in most cases).

Details

Each method produces a structured, human-readable summary optimized for its object type:

Fitted Model Objects Invokes summary() internally and prints comprehensive output including:

- Model call and configuration (family, method, dimensions)
- Data characteristics (N, I or B, response type)
- Fit statistics (LogLik, npar, AIC, BIC)
- Factor correlation matrix
- Person parameter summary (N x D)
- Item / statement / block parameter summary
- For FCDCM: higher-order delta parameters and classification diagnostics
- For TIRT: loading (lambda) parameter summary
- Convergence diagnostics (algorithm, batches, latent grid length L)

Goodness-of-Fit Objects (good.of.fit) Delegates to summary.good.of.fit and prints:

- Overview table (sample size, items, parameters, moments)
- Limited-information absolute fit (M2, RMSEA, SRMSR)
- Incremental / comparative fit (CFI, TLI, IFI)
- Likelihood and information criteria (AIC, BIC, SABIC, CAIC)
- Pseudo-R^2 measures

Value

Invisibly returns the input object x. No data is modified.

Methods (by class)

- `print(good.of.fit)`: Print method for `good.of.fit` objects. Delegates to `summary.good.of.fit` and prints formatted tables.
- `print(summary.good.of.fit)`: Print method for `summary.good.of.fit` objects. Displays overview, absolute-fit, comparative-fit, likelihood/IC, and pseudo-R^2 tables.
- `print(MIRT)`: Print method for `MIRT` objects. Multidimensional IRT (1PL–4PL) model summary.
- `print(summary.MIRT)`: Print method for `summary.MIRT` objects. Shows model config, data info, fit stats, factor correlations, person/item parameter summaries, and convergence.
- `print(MGPCM)`: Print method for `MGPCM` objects. Multidimensional Generalized Partial Credit Model summary.
- `print(summary.MGPCM)`: Print method for `summary.MGPCM` objects. Shows polytomous model config, fit stats, factor correlations, person/item parameter summaries, and convergence.
- `print(MGGUM)`: Print method for `MGGUM` objects. Multidimensional Generalized Graded Unfolding Model summary.
- `print(summary.MGGUM)`: Print method for `summary.MGGUM` objects. Shows unfolding model config, fit stats, factor correlations, person/item parameter summaries, and convergence.
- `print(FCMIRT)`: Print method for `FCMIRT` objects. Forced-Choice Multidimensional IRT model summary.
- `print(summary.FCMIRT)`: Print method for `summary.FCMIRT` objects. Shows forced-choice model config with block info, fit stats, factor correlations, person/statement parameter summaries.
- `print(FCDCM)`: Print method for `FCDCM` objects. Forced-Choice Diagnostic Classification Model summary with higher-order trait, attribute profiles, and classification diagnostics.
- `print(summary.FCDCM)`: Print method for `summary.FCDCM` objects. Shows higher-order delta parameters, block eta parameters, attribute classification summary, and convergence.
- `print(FCGDINA)`: Print method for `FCGDINA` objects. Forced-Choice GDINA model summary.
- `print(summary.FCGDINA)`: Print method for `summary.FCGDINA` objects. Shows CDM type, forced-choice block info, fit stats, delta summaries, posterior attribute probabilities, and convergence.
- `print(FCGGUM)`: Print method for `FCGGUM` objects. Forced-Choice Generalized Graded Unfolding Model summary.
- `print(summary.FCGGUM)`: Print method for `summary.FCGGUM` objects. Shows forced-choice unfolding model config, fit stats, factor correlations, person/statement parameter summaries.
- `print(TIRT)`: Print method for `TIRT` objects. Thurstonian IRT for Forced-Choice model summary.
- `print(summary.TIRT)`: Print method for `summary.TIRT` objects. Shows pairwise-comparison model config, fit stats, factor correlations, loading/person parameter summaries.
- `print(data.MIRT)`: Print method for `data.MIRT` objects.
- `print(data.MGPCM)`: Print method for `data.MGPCM` objects.
- `print(data.MGGUM)`: Print method for `data.MGGUM` objects.
- `print(data.FCMIRT)`: Print method for `data.FCMIRT` objects.

- `print(data.FCDCM)`: Print method for `data.FCDCM` objects.
- `print(data.FCGDINA)`: Print method for `data.FCGDINA` objects.
- `print(data.FCGGUM)`: Print method for `data.FCGGUM` objects.
- `print(data.TIRT)`: Print method for `data.TIRT` objects.

Output Conventions

- Numeric values are rounded to the precision specified by `digits` (default 4).
- Parameter summary tables show Min, Q1, Median, Mean, Q3, Max across items/statements for each parameter type.
- Convergence diagnostics adapt to the estimation backend (iStEM batch diagnostics vs. Stan HMC notes).

`residuals` *S3 Methods: residuals*

Description

Compute residuals from fitted **ForceChoice** model objects. Supported types are raw (response), Pearson, and deviance residuals.

Usage

```
## S3 method for class 'MIRT'
residuals(object, type = c("raw", "pearson", "deviance"), ...)

## S3 method for class 'MGPCM'
residuals(object, type = c("raw", "pearson", "deviance"), ...)

## S3 method for class 'MGGUM'
residuals(object, type = c("raw", "pearson", "deviance"), ...)

## S3 method for class 'FCMIRT'
residuals(object, type = c("raw", "pearson", "deviance"), ...)

## S3 method for class 'FCDCM'
residuals(object, type = c("raw", "pearson", "deviance"), ...)

## S3 method for class 'FCGDINA'
residuals(object, type = c("raw", "pearson", "deviance"), ...)

## S3 method for class 'FCGGUM'
residuals(object, type = c("raw", "pearson", "deviance"), ...)

## S3 method for class 'TIRT'
residuals(object, type = c("raw", "pearson", "deviance"), ...)
```

Arguments

`object`	A fitted model object.
`type`	Character; `"raw"` (default), `"pearson"`, or `"deviance"`.
`...`	Additional arguments (currently ignored).

Details

Residual types follow standard GLM conventions for binary responses:

- **raw**: $e_{ij} = y_{ij} - \hat{p}_{ij}$
- **Pearson**: $r_{ij} = e_{ij} / \sqrt{\hat{p}_{ij}(1 - \hat{p}_{ij})}$
- **deviance**: $d_{ij} = \text{sign}(e_{ij}) \sqrt{2 \left[y_{ij} \log \frac{y_{ij}}{\hat{p}_{ij}} + (1 - y_{ij}) \log \frac{1 - y_{ij}}{1 - \hat{p}_{ij}} \right]}$

For polytomous and forced-choice models, residuals are computed on a category-indicator matrix. This treats ordered categories and nominal forced-choice response patterns as multinomial outcomes rather than as numeric scores.

Value

A numeric matrix with the same dimensions as the corresponding `fitted` output.

Methods (by class)

- `residuals(MIRT)`: Residuals for `MIRT` objects.
- `residuals(MGPCM)`: Residuals for `MGPCM` objects.
- `residuals(MGGUM)`: Residuals for `MGGUM` objects.
- `residuals(FCMIRT)`: Residuals for `FCMIRT` objects.
- `residuals(FCDCM)`: Residuals for `FCDCM` objects.
- `residuals(FCGDINA)`: Residuals for `FCGDINA` objects.
- `residuals(FCGGUM)`: Residuals for `FCGGUM` objects.
- `residuals(TIRT)`: Residuals for `TIRT` objects.

`rotate` *S3 Methods: rotate*

Description

Applies factor rotation to the posterior mean parameter estimates of fitted exploratory multidimensional models from the **ForceChoice** package. Rotation reparameterises the latent space to improve interpretability of factor loadings while preserving the model-implied covariance structure. Standard errors and other posterior uncertainty summaries are carried over from the original fit object without recomputation. No per-iteration rotation or cross-iteration alignment is performed.

For GPArotation methods, theta is re-standardized to a standard normal scale after rotation. For Promax rotation, no post-rotation standardization is applied. Loadings and the factor correlation matrix are updated accordingly to preserve model-implied covariances.

Rotation is not permitted when the fitted `Q.matrix` contains zeros because that confirmatory structure would not generally be preserved under rotation.

Usage

```
## S3 method for class 'MGPCM'
rotate(
  object,
  method,
  vis = TRUE,
  target.a = NULL,
  target.theta = NULL,
  pst.W = NULL,
  lp.p = 1,
  lp.gpaiter = 5,
  promax_m = 4,
  ...
)

## S3 method for class 'MIRT'
rotate(
  object,
  method,
  vis = TRUE,
  target.a = NULL,
  target.theta = NULL,
  pst.W = NULL,
  lp.p = 1,
  lp.gpaiter = 5,
  promax_m = 4,
  ...
)
```

Arguments

`object`	A fitted model object of class `"MIRT"` (returned by `fit.MIRT`) or `"MGPCM"` (returned by `fit.MGPCM`).
`method`	Character string specifying the rotation method. Supported methods include `"promax"`, `"Varimax"`, `"quartimax"`, `"oblimin"`, `"quartimin"`, `"geominT"`,

	`"geominQ"`, `"targetT"`, `"targetQ"`, `"pstT"`, `"pstQ"`, and others provided by the **GPArotation** package.
`vis`	Logical; if `TRUE` (default), displays informational messages during rotation.
`target.a`	Numeric matrix of target loadings for target rotations (`"targetT"`, `"targetQ"`, `"pstT"`, `"pstQ"`).
`target.theta`	Numeric matrix of target latent traits for target rotations (`"targetT"`, `"targetQ"` only).
`pst.W`	Numeric weight matrix for PST rotations (`"pstT"`, `"pstQ"`).
`lp.p`	Numeric; p parameter for Lp rotation (default = 1).
`lp.gpaiter`	Integer; max GPA iterations for Lp rotation (default = 5).
`promax_m`	Numeric; power parameter for Promax rotation (default = 4).
`...`	Additional arguments passed to the **GPArotation** rotation function.

Details

Only posterior means are rotated. For `MGPCM` objects, the first D columns of `object$par$est` are treated as discrimination parameters; all category intercepts are unchanged.

Value

`rotate.MIRT` An object of class `"RotateMIRT"` containing the rotated `"MIRT"` fit object (`object`), the matched call (`call`), and the rotation arguments (`arguments`). Standard errors in the fitted object are preserved without recomputation.

`rotate.MGPCM` An object of class `"RotateMGPCM"` containing the rotated `"MGPCM"` fit object (`object`), the matched call (`call`), and the rotation arguments (`arguments`). Category intercepts are retained from the original fit.

Methods (by class)

- `rotate(MGPCM)`: Applies a factor rotation to the posterior mean discrimination parameters of `MGPCM` objects. The latent trait estimates and factor correlation matrix are transformed consistently. Category intercepts and posterior uncertainty summaries are retained from the original fit without recomputation.
- `rotate(MIRT)`: Applies rotation to the posterior mean of loadings only from `MIRT` objects. Loadings (a) and factor correlation matrix (Corr) are updated to preserve model-implied covariances. For GPArotation methods, theta is re-standardized to a standard normal scale after rotation; for Promax, no post-rotation standardization is applied.

See Also

`fit.MIRT`, `fit.MGPCM`

`sim.data.FCDCM` *Simulate Data from the Forced-Choice DCM Model*

Description

Generates forced-choice paired-comparison data from a higher-order cognitive diagnostic model (DINA/DINO). The continuous higher-order trait governs attribute mastery probabilities via a logistic link; two-statement blocks produce binary choices based on attribute mastery patterns. In each block, selecting the first statement has probability η_0 when its condensed mastery is lower than the second statement's, 0.5 when they are equal, and $0.5 + \eta_{AB}$ when it is higher.

Usage

```
sim.data.FCDCM(
  N.person = 1000,
  N.block = 10,
  D = 3,
  dcm.type = "DINA",
  control = NULL
)
```

Arguments

`N.person`	Integer; number of persons (default: 1000).
`N.block`	Integer; number of forced-choice block pairs (default: 10).
`D`	Integer; number of attributes (default: 3, must be >= 2).
`dcm.type`	Character; `"DINA"` (default) or `"DINO"`.
`control`	Optional list with entries `Q.matrix`, `block.items`, `single`, `delta1`, `delta0`, `theta`, `alpha`, `par`. If `block.items` is omitted, statements are assembled into cross-Q-vector forced-choice blocks with balanced first/second positions and diversified attribute pairs. `par` is a $B \times 2$ matrix with columns `eta0` and `etaAB`.

Value

An object of class `"data.FCDCM"` containing `data`, `response`, `theta`, `alpha`, `zeta`, `par`, `Q.matrix`, `block.items`, `prob`, and data-generating arguments.

See Also

`fit.FCDCM`, `model.FCDCM`

Examples

```
set.seed(123)
sim <- sim.data.FCDCM(N.person = 20, N.block = 3, D = 2,
                      dcm.type = "DINA")
str(sim$data)
```

```
head(sim$response)
head(sim$alpha)
```

`sim.data.FCGDINA` *Simulate Forced-Choice GDINA Data*

Description

Generates forced-choice response data from a cognitive diagnostic item model. The item-level model may be DINA, DINO, ACDM, or GDINA. The resulting statement endorsement probabilities are transformed into forced-choice response probabilities for `"RANK"`, `"MOLE"`, or `"PICK"` blocks using the same sequential Luce–Plackett mechanism used by `sim.data.FCMIRT` and `sim.data.FCGGUM`.

Usage

```
sim.data.FCGDINA(
  N.person = 1000,
  N.block = 10,
  I.block = 2,
  D = 3,
  model = "GDINA",
  fc.type = "RANK",
  control = NULL
)
```

Arguments

`N.person`	Number of persons.
`N.block`	Number of forced-choice blocks.
`I.block`	Number of items per block when `control$block.items` is not supplied. Must be at least 2.
`D`	Number of latent attributes. If `control$Q.matrix` is supplied, `D` is reset to `ncol(control$Q.matrix)`.
`model`	CDM item model: `"DINA"`, `"DINO"`, `"ACDM"`, or `"GDINA"`.
`fc.type`	Forced-choice response type: `"RANK"`, `"MOLE"`, or `"PICK"`. A scalar value is recycled to all blocks; a vector of length B may be supplied for mixed block formats.
`control`	Optional named list controlling the data-generating process. Supported entries are described below.

Value

An object of class "data.FCGDINA", a list containing:

data $N \times B$ character matrix of forced-choice responses, such as "1>3>2" for ranking blocks.

response $N \times B$ integer matrix of 1-based pattern indices matching patterns.

alpha $N \times D$ binary matrix of true latent attribute profiles.

gs $I \times 2$ matrix of item probability bounds P0 and P1.

delta.list List of true CDM delta vectors, one per statement.

design.matrix.list List of CDM design matrices used to map delta parameters to item endorsement probabilities.

Q.matrix $I \times D$ Q-matrix.

block.items List of statement indices per forced-choice block.

model CDM item model used for simulation.

patterns List of observed response-pattern matrices per block.

patterns.total List of full-ranking pattern matrices per block.

prob.item $N \times I$ matrix of item-level endorsement probabilities for sampled persons.

prob.states $I \times 2^D$ matrix of item endorsement probabilities for every attribute pattern.

prob $N \times \sum_b P_b$ matrix of forced-choice response-pattern probabilities.

N.person, N.block, I.block, D, I.states, fc.type Final data-generating dimensions and response type.

call, arguments Matched call and effective simulation arguments.

Data-Generating Process

The simulation proceeds in four steps:

1. Generate or validate the Q-matrix and forced-choice block structure.
2. Generate latent attribute profiles $\boldsymbol{\alpha}_n$.
3. Generate item-level CDM endorsement probabilities using guessing and slipping-style bounds P0 and P1; then recover the corresponding CDM delta parameters from the design matrix.
4. Convert item endorsement probabilities into block-level forced-choice pattern probabilities and sample observed responses.

Attribute profiles can be generated from a higher-order latent trait model, a thresholded multivariate normal model, a uniform distribution over all attribute patterns, or supplied directly through control$alpha.

Control List

Q.matrix Optional $I \times D$ binary Q-matrix. If omitted, a Q-matrix is generated by sim.data.Q.CDM().

block.items Optional list of item indices per block. If omitted, blocks are formed sequentially using N.block and I.block.

single Logical passed to sim.data.Q.CDM() when Q.matrix is generated internally. Default TRUE.

`alpha` Optional $N \times D$ binary matrix of true attribute profiles. If supplied, latent attributes are not generated.

`distribution` Latent attribute distribution when `alpha` is not supplied. Options are `"horder"` (default; alias `"higher.order"`), `"mvnorm"`, and `"uniform"`.

`delta1`, `delta0`, `theta` Higher-order discrimination, threshold, and person trait values used when `distribution = "horder"`. Defaults are generated from log-normal, normal, and standard normal distributions.

`Corr`, `threshold` Correlation matrix and thresholds used when `distribution = "mvnorm"`.

`gs` Optional $I \times 2$ matrix with columns `P0` and `P1`; `P0` is the lower endorsement probability and `P1` is the upper endorsement probability for each statement. Defaults are sampled from `U(0, .2)` and `U(.8, 1)`.

`mono.constraint` Logical; whether to enforce monotonic item probabilities for GDINA item generation. Default `TRUE`.

References

de la Torre, J. (2011). The generalized DINA model framework. *Psychometrika*, 76(2), 179–199. doi:10.1007/s1133601192077

See Also

`fit.FCGDINA`, `model.FCGDINA`, `sim.data.FCMIRT`, `sim.data.FCGGUM`, `sim.data.FCDCM`

Examples

```
set.seed(123)
sim <- sim.data.FCGDINA(N.person = 20, N.block = 3, I.block = 2,
                        D = 2, model = "DINA", fc.type = "RANK")

str(sim$data)
head(sim$response)
head(sim$alpha)
sim$Q.matrix

# Mixed response formats across blocks.
sim.mix <- sim.data.FCGDINA(
  N.person = 12, N.block = 3, I.block = 3, D = 2,
  model = "GDINA", fc.type = c("RANK", "MOLE", "PICK")
)
vapply(sim.mix$patterns, nrow, integer(1))
```

`sim.data.FCGGUM` *Simulate Data from the Forced-Choice GGUM Model*

Description

Generates forced-choice ranking data using the GGUM ideal-point model as the item-level endorsement process. Within-block rankings follow the same sequential Luce/Plackett mechanism as `model.FCGGUM`: the utility for each statement is the logit of its binary GGUM endorsement probability.

Usage

```
sim.data.FCGGUM(
  N.person = 1000,
  N.block = 10,
  I.block = 2,
  D = 3,
  fc.type = "RANK",
  control = NULL
)
```

Arguments

`N.person`	Integer; number of persons (default: 1000).
`N.block`	Integer; number of forced-choice blocks (default: 10).
`I.block`	Integer; items per block (default: 2).
`D`	Integer; number of latent dimensions (default: 3, must be >= 2).
`fc.type`	Character; `"RANK"` (default), `"MOLE"`, or `"PICK"`.
`control`	Optional list with entries `Q.matrix`, `block.items`, and `Corr`.

Value

An object of class `"data.FCGGUM"` containing `data`, `response`, `theta`, `par`, `Q.matrix`, `block.items`, `Corr`, `patterns`, `patterns.total`, `prob`, `mask`, and data-generating arguments.

See Also

`fit.FCGGUM`, `model.FCGGUM`

Examples

```
set.seed(123)
sim <- sim.data.FCGGUM(N.person = 20, N.block = 3, I.block = 2,
                       D = 2, fc.type = "RANK")
str(sim$data)
head(sim$response)
dim(sim$prob)
```

`sim.data.FCMIRT` *Simulate FCMIRT Data*

Description

Simulates forced-choice ranking data using a MIRT item response model (`"m1pl"` through `"m4pl"`) as the item-level endorsement model. Forced-choice responses are generated by applying the sequential Luce/Plackett choice rule to the logits of the item-level endorsement probabilities, with RANK, MOLE, and PICK patterns handled by `model.FCMIRT`. For identification, item difficulty/intercept parameters `b` are centered within each forced-choice block so that their block sum is zero.

Usage

```
sim.data.FCMIRT(
  N.person = 1000,
  N.block = 10,
  I.block = 2,
  D = 3,
  model = "m2pl",
  fc.type = "RANK",
  control = NULL
)
```

Arguments

`N.person`	Number of persons.
`N.block`	Number of forced-choice blocks.
`I.block`	Number of items per block when `control$block.items` is not supplied.
`D`	Number of latent dimensions. If `control$Q.matrix` is supplied, `D` is reset to `ncol(control$Q.matrix)`.
`model`	MIRT model type: `"m1pl"`, `"m2pl"`, `"m3pl"`, or `"m4pl"`.
`fc.type`	Forced-choice response type: `"RANK"`, `"MOLE"`, or `"PICK"`. A scalar value is recycled to all blocks.
`control`	Optional list. Supported entries are `Q.matrix`, `block.items`, and `Corr`.

Value

An object of class `"data.FCMIRT"`, a list containing `data`, `response`, `theta`, `par`, `Q.matrix`, `block.items`, `Corr`, `model`, `patterns`, `patterns.total`, `probability`, `prob`, and data-generating arguments.

See Also

`fit.FCMIRT`, `model.FCMIRT`

Examples

```
set.seed(123)
sim <- sim.data.FCMIRT(N.person = 20, N.block = 3, I.block = 2,
                       D = 2, model = "m2pl", fc.type = "RANK")
str(sim$data)
head(sim$response)
dim(sim$prob)
```

`sim.data.MGGUM` *Simulate Data from the Multidimensional Generalized Graded Unfolding Model*

Description

Generates polytomous responses from the same distance-based MGGUM probability function used by `model.MGGUM` and `fit.MGGUM`. For each item, active discriminations are drawn from $U(0.5, 2)$, active locations follow the sign of the Q-matrix, $\tau_{i0} = 0$, and the remaining τ values are ordered negative thresholds.

Usage

```
sim.data.MGGUM(
  N = 500,
  I = 20,
  D = 2,
  length.poly = 5,
  Q.matrix = NULL,
  Corr = NULL
)
```

Arguments

`N`	Integer; number of examinees (default: 500).
`I`	Integer; number of items (default: 20).
`D`	Integer; number of latent dimensions (default: 2).
`length.poly`	Integer vector or scalar indicating the number of categories for each item. A scalar is recycled to length I.
`Q.matrix`	Optional $I \times D$ matrix with values -1, 0, or 1. • `1`: active dimension with positive-side `delta` • `-1`: active dimension with negative-side `delta` • `0`: `a` and `delta` fixed to 0 (inactive dimension) The sign controls the item-location side, not the sign of discrimination. If `NULL`, a default matrix of randomly signed active entries is used.
`Corr`	Optional $D \times D$ correlation matrix. If `NULL`, the identity matrix is used.

Value

An object of class "data.MGGUM", a list containing:

data, response $N \times I$ integer response matrices with categories coded from 0 to $K_i - 1$.

theta $N \times D$ matrix of true latent traits.

par $I \times (2D + \max_i K_i)$ matrix of GGUM item parameters: discrimination columns, location columns, and stacked threshold columns.

probability $N \times \sum_i K_i$ matrix of stacked category probabilities.

Q.matrix, length.poly, Corr Design matrix, category counts, and latent correlation matrix used to generate the data.

N, I, D, call Data-generating metadata.

Data Generation Process

1. Draw latent traits from $N_D(\mathbf{0}, \boldsymbol{\Sigma})$.
2. Generate discrimination, location, and threshold parameters under the sign constraints encoded by Q.matrix.
3. Compute stacked category probabilities with model.MGGUM.
4. Sample one ordinal response per person and item. A category can be absent in a finite sample; length.poly retains the intended support.

See Also

fit.MGGUM, model.MGGUM

Examples

```
set.seed(123)
sim <- sim.data.MGGUM(N = 20, I = 5, D = 2, length.poly = 4)
str(sim$response)
dim(sim$par)
table(sim$response[, 1])
```

---

sim.data.MGPCM *Simulate Data from the Multidimensional Generalized Partial Credit Model*

Description

Generates polytomous (multi-category) response data from the MGPCM. Category probabilities use the same softmax formulation as model.MGPCM: $\log P(Y = k)$ is proportional to $k\,\mathbf{a}_i'\boldsymbol{\theta}_j + d_{ik}$. The requested category counts are retained even when a finite sample does not contain every possible category.

Usage

```
sim.data.MGPCM(
  N = 500,
  I = 20,
  D = 2,
  length.poly = 5,
  Q.matrix = NULL,
  Corr = NULL,
  rotate = NULL,
  promax_m = 4
)
```

Arguments

`N`	Integer; number of examinees (default: 500).
`I`	Integer; number of items (default: 20).
`D`	Integer; number of latent dimensions (default: 2).
`length.poly`	Integer vector or scalar; number of categories per item. Recycled to length I if scalar (default: 5).
`Q.matrix`	Optional $I \times D$ 0/1 matrix. `NULL` uses a triangular identification pattern.
`Corr`	Optional $D \times D$ correlation matrix.
`rotate`	Optional rotation method (**GPArotation** or `"promax"`).
`promax_m`	Power parameter for Promax (default: 4).

Value

An object of class `"data.MGPCM"`, a list containing:

`data`, `response` $N \times I$ integer response matrices with categories coded from 0 to $K_i - 1$.

`theta` $N \times D$ matrix of true latent traits.

`par` $I \times (D + \max_i K_i)$ matrix of item parameters; discrimination columns are followed by category intercept columns.

`probability` $N \times \sum_i K_i$ matrix of stacked category probabilities.

`Q.matrix`, `length.poly`, `Corr` Design matrix, category counts, and latent correlation matrix used to generate the data.

`model`, `N`, `I`, `D`, `call` Data-generating metadata.

Data Generation Process

1. **Latent traits:** $\boldsymbol{\theta}_j \sim N_D(\mathbf{0}, \boldsymbol{\Sigma})$.
2. **Item parameters:** $a_{id} \sim \text{Lognormal}(0.25, 0.25)$ (active dimensions), $d_{i0} = 0$, and $d_{i1}, \ldots, d_{i,K_i-1}$ are sorted draws from $N(0, 1)$ used as category intercepts.
3. **Responses:** $Y_{ij} \sim \text{Categorical}(P(Y_{ij} = k \mid \boldsymbol{\theta}_j))$ for $k = 0, \ldots, K_i - 1$.

See Also

fit.MGPCM, model.MGPCM

Examples

```
set.seed(123)
sim <- sim.data.MGPCM(N = 20, I = 5, D = 2, length.poly = 4)
str(sim$response)
dim(sim$probability)
sim$length.poly
```

---

sim.data.MIRT *Simulate Data from the Multidimensional IRT (MIRT) Model*

Description

Generates binary response data, latent trait vectors, and item parameters from the multidimensional extension of the 1PL–4PL item response models. The latent traits are drawn from a multivariate normal distribution with a user-specified correlation matrix. Discrimination parameters respect a Q-matrix structure and may be optionally rotated.

Usage

```
sim.data.MIRT(
  N = 500,
  I = 20,
  D = 2,
  model = "m2pl",
  Q.matrix = NULL,
  Corr = NULL,
  rotate = NULL,
  promax_m = 4
)
```

Arguments

N	Integer; number of examinees (default: 500).
I	Integer; number of items (default: 20).
D	Integer; number of latent dimensions (default: 2).
model	Character; one of "m1pl", "m2pl" (default), "m3pl", "m4pl".
Q.matrix	An optional $I \times D$ 0/1 matrix. If NULL, a default structure with triangular identification pattern is used.
Corr	An optional $D \times D$ correlation matrix. If NULL, the identity matrix is used (orthogonal factors).
rotate	Optional rotation method name (passed to **GPArotation** or stats::promax). Applicable only when $D > 1$ and all Q-matrix entries are non-zero.
promax_m	Integer; power parameter for "promax" rotation (default: 4).

Value

An object of class `"data.MIRT"`, a list containing:

`data`, `response` $N \times I$ binary response matrix (identical).

`theta` $N \times D$ matrix of true latent trait values.

`par` $I \times (D + 3)$ matrix of true item parameters (columns `a1..aD`, `b`, `c`, `d`).

`probability` $N \times I$ matrix of response probabilities.

`Q.matrix` $I \times D$ Q-matrix used.

`Corr` $D \times D$ correlation matrix.

`model`, `N`, `I`, `D` Data-generating parameters.

Data Generation Process

1. **Latent traits:** $\boldsymbol{\theta}_j \sim N_D(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is the $D \times D$ correlation matrix `Corr`.
2. **Item parameters:**
   - $a_{id} \sim$ Lognormal$(0.25, 0.25)$ for $q_{id} = 1$ (M2PL/M3PL/M4PL); $a_{id} = 1$ for all items and dimensions in M1PL.
   - $b_i \sim N(0, 1)$ (difficulty).
   - $c_i \sim U(0, 0.35)$ (lower asymptote; M3PL/M4PL only).
   - $d_i \sim U(0.65, 1)$ (upper asymptote; M4PL only).
3. **Response probabilities:** $P_{ij} = P(Y_{ij} = 1 \mid \boldsymbol{\theta}_j)$ per the specified MIRT model (see `fit.MIRT` for the IRF equations).
4. **Binary responses:** $Y_{ij} \sim$ Bernoulli(P_{ij}).

Optional rotation of the loading matrix `a` and corresponding transformation of `theta` and `Corr` is applied before response generation.

See Also

`fit.MIRT` for fitting the MIRT model, `model.MIRT` for computing response probabilities.

Examples

```
# Basic 2D 2PL simulation
sim <- sim.data.MIRT(N = 20, I = 6, D = 2, model = "m2pl")

# With correlated factors and rotation
Corr <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
sim_rot <- sim.data.MIRT(N = 20, I = 6, D = 2, model = "m2pl",
                         Q.matrix = matrix(1, 6, 2),
                         Corr = Corr, rotate = "oblimin")

# Fit the simulated data
fit <- fit.MIRT(
  sim$response, model = "m2pl", D = 2, method = "iStEM",
  control.method = list(
    vis = FALSE, seed = 123,
```

```
    M = 2, B = 2, burnin.maxitr = 2,
    maxitr = 3, eps1 = 10, eps2 = 10,
    estimate.se = FALSE)
)
cor(fit$theta$est, sim$theta)  # trait recovery
```

`sim.data.TIRT` *Simulate Data from the Thurstonian IRT (TIRT) Model*

Description

Generates forced-choice pairwise comparison data under the Thurstonian IRT framework. Each statement loads on exactly one latent dimension; pairwise latent differences with probit link determine binary outcomes. Supports RANK, MOLE, and PICK response types. For each pair (i, k), the simulated comparison uses $-\gamma_{ik} + \lambda_i q_i' \theta_j - \lambda_k q_k' \theta_j + \epsilon_i - \epsilon_k$, with $\epsilon_i \sim N(0, \psi_i^2)$.

Usage

```
sim.data.TIRT(
  N.person = 1000,
  N.block = 10,
  I.block = 2,
  D = 3,
  fc.type = "RANK",
  control = NULL
)
```

Arguments

`N.person`	Integer; number of persons (default: 1000).
`N.block`	Integer; number of blocks (default: 10).
`I.block`	Integer; items per block (default: 2).
`D`	Integer; number of latent dimensions (default: 3, >= 2).
`fc.type`	Character; `"RANK"` (default), `"MOLE"`, or `"PICK"`.
`control`	Optional list with entries `Q.matrix`, `block.items`, `lambda`, `psi2`, `gamma.matrix`, `gamma.item`, `Corr`.

Value

An object of class `"data.TIRT"` containing `data`, `response`, `theta`, `par`, `Q.matrix`, `block.items`, `Corr`, `gamma.matrix`, `lambda`, `psi2`, `pairs.value`, and data-generating arguments.

See Also

`fit.TIRT`, `model.TIRT`

Examples

```
set.seed(123)
sim <- sim.data.TIRT(N.person = 20, N.block = 3, I.block = 2,
                     D = 2, fc.type = "RANK")
str(sim$data)
head(sim$response)
dim(sim$Q.matrix)
```

---

summary *S3 Methods: summary*

Description

Generates structured, comprehensive summaries of objects produced by the **ForceChoice** package. This generic function dispatches to class-specific methods that extract and organize key information including model configurations, fit statistics, parameter estimates, and convergence diagnostics. Designed for programmatic access and downstream reporting.

Usage

```
## S3 method for class 'good.of.fit'
summary(object, digits = 4, ...)

## S3 method for class 'MIRT'
summary(object, digits = 4, ...)

## S3 method for class 'MGPCM'
summary(object, digits = 4, ...)

## S3 method for class 'MGGUM'
summary(object, digits = 4, ...)

## S3 method for class 'FCMIRT'
summary(object, digits = 4, ...)

## S3 method for class 'FCDCM'
summary(object, digits = 4, ...)

## S3 method for class 'FCGDINA'
summary(object, digits = 4, ...)

## S3 method for class 'FCGGUM'
summary(object, digits = 4, ...)

## S3 method for class 'TIRT'
summary(object, digits = 4, ...)
```

Arguments

object	An object of one of the following classes: • Fitted model objects: "MIRT", "MGPCM", "MGGUM", "FCMIRT", "FCDCM", "FCGGUM", "FCGDINA", "TIRT" • Goodness-of-fit objects: "good.of.fit"
digits	Number of decimal places for numeric output (default: 4). Applied uniformly across all methods.
...	Additional arguments passed to or from other methods (currently ignored).

Details

Each method returns a named list with class-specific components optimized for structured access and formatted printing:

Fitted Model Objects Returns a summary.<Class> object with components:

call Original function call.

model.info List: family, description, D, method, and model-specific fields (model, fc.type, dcm.type, I.states, N.block).

data.info List: N, I or B, response.type.

fit.stats List: LogLik, npar, AIC, BIC.

par.summary Matrix: parameter summary (Min, Q1, Median, Mean, Q3, Max) across items/statements/blocks for each parameter type.

theta.summary Matrix: person parameter summary per dimension.

Corr Matrix: factor inter-trait correlation matrix (where applicable).

convergence List: algorithm, batch counts, convergence flags, latent grid length L (iStEM) or diagnostic note (Stan).

digits Numeric: precision used for formatting.

Additional model-specific components:

delta.summary FCDCM only: higher-order delta parameters.

classification FCDCM only: observed attribute patterns, mean maximum posterior.

lambda.summary TIRT only: factor loading summary.

Goodness-of-Fit Objects (good.of.fit) Returns a summary.good.of.fit object with components:

Sample_Size, Items, Response_Type, Parameters, Moments, LogLik, Deviance Overview statistics.

IC Named vector: AIC, AICc, BIC, CAIC, SABIC, HQIC.

M2 List: M2 statistic, df, p-value, RMSEA with CI.

Residual_Fit Named vector: SRMSR, RMSR, max standardized residual.

Comparative_Fit Named vector: CFI, TLI, IFI.

Pseudo_R2 Named vector: McFadden, Cox–Snell, Nagelkerke, Aldrich–Nelson, Veall–Zimmermann.

Local_Dependence Named vector: Q3 mean, max, adjusted max, P95.

Classification Named vector: posterior entropy, max posterior.

Null_Model Named vector: null model LogLik, deviance, M2, df.

tables List of data frames for formatted printing.

Value

Invisibly returns a structured list containing summary components. The exact structure depends on the class of `object`. All returned objects carry an appropriate S3 class (e.g., `"summary.MIRT"`, `"summary.good.of.fit"`) for use with corresponding `print` methods.

Methods (by class)

- `summary(good.of.fit)`: Summary method for `good.of.fit` objects. Extracts and structures all fit indices: information criteria, limited-information M2, RMSEA, SRMSR, CFI/TLI/IFI, pseudo-R^2, local dependence (Yen's Q3), posterior classification, and null-model diagnostics.
- `summary(MIRT)`: Summary method for `MIRT` objects. Multidimensional IRT (1PL–4PL): extracts binary-response model configuration, fit statistics (LogLik, AIC, BIC), factor correlation matrix, person and item parameter summaries, and convergence diagnostics.
- `summary(MGPCM)`: Summary method for `MGPCM` objects. Multidimensional Generalized Partial Credit Model: polytomous responses with category-specific step parameters.
- `summary(MGGUM)`: Summary method for `MGGUM` objects. Multidimensional Generalized Graded Unfolding Model: ideal-point polytomous responses with discrimination (a), location (delta), and threshold (tau) parameters.
- `summary(FCMIRT)`: Summary method for `FCMIRT` objects. Forced-Choice Multidimensional IRT: dominance model with sequential ranking over item endorsement logits at the block level. Extracts block configuration, FC type (RANK/MOLE/PICK), and statement-level parameter summaries.
- `summary(FCDCM)`: Summary method for `FCDCM` objects. Forced-Choice Diagnostic Classification Model: higher-order trait with exact marginalization over 2^D attribute profiles. Extracts DCM type (DINA/DINO), higher-order delta parameters, block eta parameters, and posterior classification diagnostics.
- `summary(FCGDINA)`: Summary method for `FCGDINA` objects. Forced-Choice GDINA model: CDM item model (DINA/DINO/ACDM/GDINA) with sequential forced-choice ranking over item endorsement logits.
- `summary(FCGGUM)`: Summary method for `FCGGUM` objects. Forced-Choice Generalized Graded Unfolding Model: ideal-point model with sequential ranking over binary GGUM endorsement logits. Extracts forced-choice block structure, unfolding parameters (a, delta, tau), and convergence info.
- `summary(TIRT)`: Summary method for `TIRT` objects. Thurstonian IRT for Forced-Choice: pairwise probit comparison of latent utility differences. Extracts statement loadings (lambda), uniquenesses (psi2), pairwise gamma matrix, and factor correlations.

`triplets` *Triplets of Pairwise Comparisons*

Description

A synthetic forced-choice data set containing pairwise comparison responses for the first 200 of 2000 participants on four triplets. The data were originally generated as part of Brown and Maydeu-Olivares (2012) and are distributed in the published R package **thurstonianIRT**.

In each triplet, participants ranked three alternative statements according to their preference. The rankings were then converted into dichotomous pairwise responses for all three within-triplet comparisons.

Format

A data frame with 200 rows and 12 integer columns. Values are 0/1 pairwise preference indicators. Overall, the 12 statements measure three traits. Items 1, 4, 7, and 10 load on trait 1; items 2, 5, 8, and 11 load on trait 2; and items 3, 6, 9, and 12 load on trait 3. Items 4, 9, and 11 are reverse-keyed.

i1i2 Response preference between item 1 and item 2.

i1i3 Response preference between item 1 and item 3.

i2i3 Response preference between item 2 and item 3.

i4i5 Response preference between item 4 and item 5.

i4i6 Response preference between item 4 and item 6.

i5i6 Response preference between item 5 and item 6.

i7i8 Response preference between item 7 and item 8.

i7i9 Response preference between item 7 and item 9.

i8i9 Response preference between item 8 and item 9.

i10i11 Response preference between item 10 and item 11.

i10i12 Response preference between item 10 and item 12.

i11i12 Response preference between item 11 and item 12.

Source

Brown, A. & Maydeu-Olivares, A. (2012). Fitting a Thurstonian IRT model to forced-choice data using Mplus. Behavior Research Methods, 44, 1135-1147. doi:10.3758/s13428-012-0217-x

The data file was taken from the published R package **thurstonianIRT**.

Examples

```
data("triplets")
dim(triplets)
head(triplets)
```

update *S3 Methods: update*

Description

The update function provides a unified and convenient interface to re-fit **ForceChoice** models with modified parameter settings while preserving all other original configurations. It allows users to change the estimation method, adjust control parameters, increase dimensionality, or switch model specifications without re-specifying the entire call.

Usage

```
## S3 method for class 'MIRT'
update(object, ...)

## S3 method for class 'MGPCM'
update(object, ...)

## S3 method for class 'MGGUM'
update(object, ...)

## S3 method for class 'FCMIRT'
update(object, ...)

## S3 method for class 'FCDCM'
update(object, ...)

## S3 method for class 'FCGDINA'
update(object, ...)

## S3 method for class 'FCGGUM'
update(object, ...)

## S3 method for class 'TIRT'
update(object, ...)
```

Arguments

object	An object of one of the following classes: • "MIRT" — Multidimensional IRT model. • "MGPCM" — Multidimensional Generalized Partial Credit Model. • "MGGUM" — Multidimensional Generalized Graded Unfolding Model. • "FCMIRT" — Forced-Choice Multidimensional IRT model. • "FCDCM" — Forced-Choice Diagnostic Classification Model. • "FCGDINA" — Forced-Choice GDINA model. • "FCGGUM" — Forced-Choice Generalized Graded Unfolding Model.

- "TIRT" — Thurstonian IRT for Forced-Choice.

... Additional named arguments passed to override or extend the original call. Valid arguments depend on the class of object and correspond to the formal parameters of the matching fit.*() function. Common overrides include method, control.model, control.method, and legacy control.

Details

Internally, each method extracts the stored arguments list from the input object, merges it with user-provided ... using modifyList, explicitly merges control lists from the previous fit and the update call, filters to only the formal parameters of the target fit.*() function, then re-invokes the fitting function via do.call.

This ensures that:

- Only explicitly overridden parameters are changed.
- Default values from the original call remain intact.
- Complex nested structures (e.g., control.model, control.method, and control lists) can be partially updated.
- Internal arguments (e.g., cores, vis, seed) are passed through to the method controls.

Value

An object of the same class as object, re-fitted using the original arguments updated with any provided in .... All unchanged parameters are preserved from the original call.

Methods (by class)

- update(MIRT): Update method for MIRT objects. Re-fits the MIRT model with modified arguments. Supports changing model (e.g., m2pl -> m3pl), method (iStEM -> Stan), D, Q.matrix, control.model, and control.method.
- update(MGPCM): Update method for MGPCM objects. Re-fits the MGPCM model with modified arguments.
- update(MGGUM): Update method for MGGUM objects. Re-fits the MGGUM model with modified arguments.
- update(FCMIRT): Update method for FCMIRT objects. Re-fits the FCMIRT model with modified arguments. Supports changing model, method, fc.type, block.items, and control settings.
- update(FCDCM): Update method for FCDCM objects. Re-fits the FCDCM model with modified arguments. Supports changing dcm.type, method, block.items, and control settings.
- update(FCGDINA): Update method for FCGDINA objects. Re-fits the FCGDINA model with modified arguments.
- update(FCGGUM): Update method for FCGGUM objects. Re-fits the FCGGUM model with modified arguments.
- update(TIRT): Update method for TIRT objects. Re-fits the TIRT model with modified arguments. Supports changing fc.type, method, pairs.value, and control settings.

Examples

```
sim <- sim.data.MIRT(N = 20, I = 6, D = 2, model = "m2pl")
fit <- fit.MIRT(sim$response, model = "m2pl", D = 2, method = "iStEM",
                control.method = list(
                  vis = FALSE, seed = 123,
                  M = 2, B = 2, burnin.maxitr = 2,
                  maxitr = 3, eps1 = 10, eps2 = 10,
                  estimate.se = FALSE))


# stan code, long time
# Switch to Stan estimation with more chains
fit_stan <- update(fit, method = "stan",
                   control.method = list(chains = 1, iter = 200))


# Change model to 3PL
fit_3pl <- update(fit, model = "m3pl")

# Adjust prior hyperparameters
fit_prior <- update(fit, control.model = list(a.mu = 0.5, a.sigma = 0.5))
```

---

vcov *S3 Methods: vcov*

Description

Returns the variance–covariance matrix of the estimated model parameters from a fitted **Force-Choice** model object. The package stores marginal standard errors rather than a joint covariance estimate, so the returned matrix is $\mathrm{diag}(\mathrm{SE}^2)$ for every backend.

Usage

```
## S3 method for class 'MIRT'
vcov(object, ...)

## S3 method for class 'MGPCM'
vcov(object, ...)

## S3 method for class 'MGGUM'
vcov(object, ...)

## S3 method for class 'FCMIRT'
vcov(object, ...)

## S3 method for class 'FCDCM'
vcov(object, ...)
```

```
## S3 method for class 'FCGDINA'
vcov(object, ...)

## S3 method for class 'FCGGUM'
vcov(object, ...)

## S3 method for class 'TIRT'
vcov(object, ...)
```

Arguments

object	A fitted model object of class "MIRT", "MGPCM", "MGGUM", "FCMIRT", "FCDCM", "FCGDINA", "FCGGUM", or "TIRT".
...	Additional arguments (currently ignored).

Details

Standard errors reflect the active estimation backend. Off-diagonal covariances are not retained in fitted objects and are therefore reported as zero. Parameters whose standard errors were not estimated have NA diagonal entries.

Value

A square numeric matrix. Rows and columns correspond to the free model parameters in the package's canonical parameter map.

Methods (by class)

- vcov(MIRT): Variance–covariance matrix of parameter estimates for MIRT objects.
- vcov(MGPCM): Variance–covariance matrix for MGPCM objects.
- vcov(MGGUM): Variance–covariance matrix for MGGUM objects.
- vcov(FCMIRT): Variance–covariance matrix for FCMIRT objects.
- vcov(FCDCM): Variance–covariance matrix for FCDCM objects.
- vcov(FCGDINA): Variance–covariance matrix for FCGDINA objects.
- vcov(FCGGUM): Variance–covariance matrix for FCGGUM objects.
- vcov(TIRT): Variance–covariance matrix for TIRT objects.

Index