Package 'irtQ'

June 7, 2026

Type Package

Title Unidimensional Item Response Theory Modeling

Version 1.1.0

Description Fit unidimensional item response theory (IRT) models to test data, which includes both dichotomous and polytomous items, calibrate pretest item parameters, estimate examinees' abilities, and examine the IRT model-data fit on item-level in different ways as well as provide useful functions related to IRT analyses such as IRT model-data fit evaluation and differential item functioning analysis.
The bring.flexmirt() and write.flexmirt() functions were written by modifying the read.flexmirt() function (Pritikin & Falk (2022) <doi:10.1177/0146621620929431>).
The bring.bilog() and bring.parscale() functions were written by modifying the read.bilog() and read.parscale() functions, respectively (Weeks (2010) <doi:10.18637/jss.v035.i12>).
The bisection() function was written by modifying the bisection() function (Howard (2017, ISBN:9780367657918)). The code of the inverse test characteristic curve scoring in the est_score() function was written by modifying the irt.eq.tse() function (González (2014) <doi:10.18637/jss.v059.i07>). In est_score() function, the code of weighted likelihood estimation method was written by referring to the Pi(), Ji(), and Ii() functions of the catR package (Magis & Barrada (2017) <doi:10.18637/jss.v076.c01>).

Depends R (>= 4.4)

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports stats, statmod, utils, tibble, dplyr, purrr, tidyr, rlang, reshape2, janitor, ggplot2, gridExtra, parallel, Matrix, Rfast, mirt

Suggests knitr, rmarkdown, testthat (>= 3.0.0), pkgdown, RhpcBLASctl

Config/testthat/edition 3

RoxygenNote 7.3.2

URL https://hwangQ.github.io/irtQ/, https://github.com/hwangQ/irtQ

BugReports https://github.com/hwangQ/irtQ/issues

NeedsCompilation no

Author Hwanggyu Lim [aut, cre],
Craig S. Wells [ctb],
James Howard [ctb],
Joshua Pritikin [ctb],
Jonathan P Weeks [ctb],
Jorge González [ctb],
David Magis [ctb]

Maintainer Hwanggyu Lim <hglim83@gmail.com>

Repository CRAN

Date/Publication 2026-06-07 14:30:08 UTC

Contents

`bind.fill` *Bind Fill*

Description

This function creates a matrix using either row-wise (`rbind`) or column-wise (`cbind`) binding of a list of numeric vectors with varying lengths. Shorter vectors are padded with a specified fill value.

Usage

```
bind.fill(List, type = c("rbind", "cbind"), fill = NA)
```

Arguments

`List`	A list containing numeric vectors of possibly different lengths.
`type`	A character string indicating the type of binding to perform. Options are `"rbind"` or `"cbind"`.
`fill`	A value used to fill missing elements when aligning the vectors. For `type = "cbind"`, this fills missing rows in shorter columns; for `type = "rbind"`, this fills missing columns in shorter rows. Accepts any R object (e.g., numeric, character, logical). Default is `NA`.

Value

A matrix formed by binding the elements of the list either row-wise or column-wise, with shorter vectors padded by the specified `fill` value.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

Examples

```
# Sample list
score_list <- list(item1 = 0:3, item2 = 0:2, item3 = 0:5, item4 = 0:4)

# 1) Create a row-bound matrix (rbind)
bind.fill(score_list, type = "rbind")

# 2) Create a column-bound matrix (cbind)
bind.fill(score_list, type = "cbind")

# 3) Create a column-bound matrix and fill missing values with 0
bind.fill(score_list, type = "cbind", fill = 0L)
```

---

bisection *The Bisection Method to Find a Root*

Description

This function is a modified version of the `bisection` function in the **cmna** R package (Howard, 2017), designed to find a root of the function `.fun` with respect to its first argument. Unlike the original `bisection()` in **cmna**, this version allows additional arguments to be passed to `.fun`.

Usage

```
bisection(.fun, ..., lb, ub, tol = 1e-04, max.it = 100)
```

Arguments

`.fun`	A function for which the root is to be found.
`...`	Additional arguments to be passed to `.fun`.
`lb`	A numeric value specifying the lower bound of the search interval.
`ub`	A numeric value specifying the upper bound of the search interval.
`tol`	A numeric value specifying the tolerance for convergence. Default is 1e-4.
`max.it`	An integer specifying the maximum number of iterations. Default is 100.

Details

The bisection method is a well-known root-finding numerical algorithm that applies to any continuous function, provided that the function values at the lower (`lb`) and upper (`ub`) bounds have opposite signs. The method repeatedly bisects the interval until the absolute difference between successive estimates is smaller than the error tolerance (`tol`) or the maximum number of iterations (`max.it`) is reached.

Value

A list with the following components:

- `root`: The estimated root of the function.
- `iter`: The number of iterations performed.
- `accuracy`: The final absolute difference between the last two interval points with opposite signs.

References

Howard, J. P. (2017). *Computational methods for numerical analysis with R*. New York: Chapman and Hall/CRC.

See Also

`est_score()`

Examples

```
## Example: Find the theta value corresponding to a given probability
## of a correct response using the item response function of a 2PLM
## (a = 1, b = 0.2)

# Define a function of theta
find.th <- function(theta, p) {
  p - drm(theta = theta, a = 1, b = 0.2, D = 1)
}

# Find the theta value corresponding to p = 0.2
bisection(.fun = find.th, p = 0.2, lb = -10, ub = 10)$root

# Find the theta value corresponding to p = 0.8
bisection(.fun = find.th, p = 0.8, lb = -10, ub = 10)$root
```

`bring.flexmirt` *Import Item and Ability Parameters from IRT Software*

Description

These functions import item and/or ability parameters from BILOG-MG 3, PARSCALE 4, flexMIRT, and the **mirt** R package.

Usage

```
bring.flexmirt(
  file,
  type = c("par", "sco"),
  rePar = TRUE,
  rePar.gpc = TRUE,
  n.factor = 1
)

bring.bilog(file, type = c("par", "sco"))

bring.parscale(file, type = c("par", "sco"))

bring.mirt(x)
```

Arguments

file	A file name (including the full path) containing the item or ability parameter estimates.
type	A character string indicating the type of output file. Available options are "par" for item parameter files and "sco" for ability parameter files.
rePar	Logical. If TRUE, and when a dichotomous IRT model (e.g., 3PLM) or the graded response model (GRM) is fit, the item intercept and the logit of the guessing parameter are reparameterized into the item difficulty and guessing parameters, respectively. Default is TRUE.
rePar.gpc	Logical. If TRUE, and when the generalized partial credit model (GPCM) is fit, the nominal model parameters in the flexMIRT output are reparameterized into GPCM slope and difficulty parameters. Default is TRUE.
n.factor	A numeric value indicating the number of latent traits (factors) estimated. This argument must be specified when type = "sco". Default is 1.
x	An object returned by the function mirt::mirt().

Details

The bring.flexmirt() function was developed by modifying the read.flexmirt() function (Pritikin & Falk, 2020). Similarly, bring.bilog() and bring.parscale() were based on modified versions of the read.bilog() and read.parscale() functions (Weeks, 2010), respectively.

The supported file extensions for item and ability parameter files are: ".par" and ".sco" for BILOG-MG and PARSCALE, and "-prm.txt" and "-sco.txt" for flexMIRT. For **mirt**, the user provides the object name directly.

Although bring.flexmirt() can extract multidimensional item and ability parameter estimates, the **irtQ** package is designed exclusively for unidimensional IRT applications.

For polytomous items, both bring.flexmirt() and bring.mirt() can import item parameters for the graded response model (GRM) and the generalized partial credit model (GPCM).

Value

These functions return a list containing several components. For flexMIRT output files, results from multiple-group analyses can be handled; in such cases, each list element corresponds to the estimation results for a separate group.

Sample Output Files of IRT software

To illustrate how to import item parameter estimate files from PARSCALE 4 and flexMIRT using `bring.parscale()` and `bring.flexmirt()`, two example output files are included in this package.

One file is from PARSCALE 4 with a ".PAR" extension (i.e., "parscale_sample.PAR"), and the other is from flexMIRT with a "-prm.txt" extension (i.e., "flexmirt_sample-prm.txt").

Both files contain item parameter estimates from a mixed-format test with 55 items: fifty dichotomous items following the 3PL model and five polytomous items with five response categories modeled using the graded response model (GRM). The examples below demonstrate how to import these output files.

Note

For item parameter files from any IRT software, only the internal object `"full_df"` in the returned list is required for various functions in the **irtQ** package. This object is a data frame containing item metadata (e.g., item parameters, number of categories, IRT model types). See `info()` or `simdat()` for more details on item metadata.

In addition, when item parameters are estimated using the partial credit model (PCM) or the generalized partial credit model (GPCM), item step parameters are included in the `"full_df"` object. These step parameters are calculated by subtracting the category threshold parameters from the overall item difficulty (or location) parameter. See the **IRT Models** section in irtQ-package for further details on the parameterization of the GPCM.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Cai, L. (2017). flexMIRT 3.5 Flexible multilevel multidimensional item analysis and test scoring (Computer software). Chapel Hill, NC: Vector Psychometric Group.

Chalmers, R. P. (2012). mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software, 48*(6), 1-29.

Weeks, J. P. (2010). plink: An R Package for Linking Mixed-Format Tests Using IRT-Based Methods. *Journal of Statistical Software, 35*(12), 1-33. URL http://www.jstatsoft.org/v35/i12/.

Pritikin, J. (2018). *rpf: Response Probability Functions*. R package version 0.59. https://CRAN.R-project.org/package=rpf.

Pritikin, J. N., & Falk, C. F. (2020). OpenMx: A modular research environment for item response theory method development. *Applied Psychological Measurement, 44*(7-8), 561-562.

Muraki, E. & Bock, R. D. (2003). PARSCALE 4: IRT item analysis and test scoring for rating scale data (Computer Software). Chicago, IL: Scientific Software International. URL http://www.ssicentral.com

Zimowski, M. F., Muraki, E., Mislevy, R. J., & Bock, R. D. (2003). BILOG-MG 3: Multiple-group IRT analysis and test maintenance for binary items (Computer Software). Chicago, IL: Scientific Software International. URL http://www.ssicentral.com

See Also

irtQ-package

Examples

```
## Example 1
# Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read item parameters and convert them to item metadata
bring.flexmirt(file = flex_sam, "par")$Group1$full_df

## Example 2
# Import the ".PAR" output file from PARSCALE
pscale_sam <- system.file("extdata", "parscale_sample.PAR", package = "irtQ")

# Read item parameters and convert them to item metadata
bring.parscale(file = pscale_sam, "par")$full_df
```

---

cac_lee *Classification Accuracy and Consistency Using Lee's (2010) Approach*

Description

This function computes classification accuracy and consistency indices for complex assessments based on the method proposed by Lee (2010). This function supports both dichotomous and polytomous item response theory (IRT) models.

Usage

```
cac_lee(x, cutscore, theta = NULL, weights = NULL, D = 1, cut.obs = TRUE)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
cutscore	A numeric vector specifying the cut scores for classification. Cut scores are the points that separate different performance categories (e.g., pass vs. fail, or different grades).

theta	A numeric vector of ability estimates. Ability estimates (theta values) are the individual proficiency estimates obtained from the IRT model. The theta parameter is optional and can be NULL.
weights	An optional two-column data frame or matrix where the first column is the quadrature points (nodes) and the second column is the corresponding weights. This is typically used in quadrature-based IRT analysis.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
cut.obs	Logical. If TRUE, it indicates the cutscores on the observed-summed score metric. If FALSE, it indicates they are on the IRT theta score metric. Default is TRUE.

Details

This function first validates the input arguments. If both theta and weights are NULL, the function will stop and return an error message. Either theta (a vector of ability estimates) or weights (a quadrature-based weight matrix) must be specified.

If cut.obs = FALSE, the provided cut scores are assumed to be on the theta (ability) scale, and they are internally converted to the observed summed score scale using the test characteristic curve (TCC). This transformation allows classification to be carried out on the summed score metric, even if theta-based cut points are provided.

When weights are provided (D method), the function uses the Lord-Wingersky recursive algorithm to compute the conditional distribution of observed total scores at each node. These conditional distributions are used to compute:

- the probability of being classified into each performance level,
- conditional classification accuracy (probability of correct classification), and
- conditional classification consistency (probability of being assigned to the same level upon repeated testing).

When theta values are provided instead (P method), the same logic applies, but using an empirical distribution of examinees instead of quadrature-based integration. In this case, uniform weights are assigned to all examinees.

Finally, marginal classification accuracy and consistency are computed as weighted averages of the conditional statistics across the ability distribution.

Value

A list containing the following elements:

- confusion: A confusion matrix showing the cross table between true and expected levels.
- marginal: A data frame showing the marginal classification accuracy and consistency indices.
- conditional: A data frame showing the conditional classification accuracy and consistency indices.
- prob.level: A data frame showing the probability of being assigned to each level category.
- cutscore: A numeric vector showing the cut scores used in the analysis.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Lee, W. C. (2010). Classification consistency and accuracy for complex assessments using item response theory. *Journal of Educational Measurement, 47*(1), 1-17.

See Also

gen.weight(), est_score(), cac_rud()

Examples

```
## ------------------------------------------------------------------------
## 1. When the empirical ability distribution of the population is available
##    (D method)
## ------------------------------------------------------------------------

# Import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read item parameter data and convert it to item metadata
x <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# Set the cut scores on the observed summed score scale
cutscore <- c(10, 20, 30, 50)

# Create a data frame containing the quadrature points and corresponding weights
node <- seq(-4, 4, 0.25)
weights <- gen.weight(dist = "norm", mu = 0, sigma = 1, theta = node)

# Calculate classification accuracy and consistency
cac_1 <- cac_lee(x = x, cutscore = cutscore, weights = weights, D = 1)
print(cac_1)

## ------------------------------------------------------------
## 2. When individual ability estimates are available (P method)
## ------------------------------------------------------------

# Randomly draw true ability values from N(0, 1)
set.seed(12)
theta <- rnorm(n = 1000, mean = 0, sd = 1)

# Simulate item response data
data <- simdat(x = x, theta = theta, D = 1)

# Estimate ability parameters using maximum likelihood (ML)
est_th <- est_score(
  x = x, data = data, D = 1, method = "ML",
  range = c(-4, 4), se = FALSE
)$est.theta
```

```
# Calculate classification accuracy and consistency
cac_2 <- cac_lee(x = x, cutscore = cutscore, theta = est_th, D = 1)
print(cac_2)

## ----------------------------------------------------------
## 3. When individual ability estimates are available,
##    but cut scores are specified on the IRT theta scale
## ----------------------------------------------------------
# Set the cut scores on the theta scale
cutscore <- c(-2, -0.4, 0.2, 1.0)

# Calculate classification accuracy and consistency
cac_3 <- cac_lee(
  x = x, cutscore = cutscore, theta = est_th, D = 1,
  cut.obs = FALSE
)
print(cac_3)
```

---

cac_rud *Classification Accuracy and Consistency Based on Rudner's (2001, 2005) Approach*

Description

This function computes classification accuracy and consistency indices using the method proposed by Rudner in 2001 and 2005. This function supports both scenarios: when the empirical ability distribution of the population is available, and when individual ability estimates are used.

Usage

```
cac_rud(x = NULL, cutscore, theta = NULL, se = NULL, weights = NULL, D = 1)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See est_irt() or simdat() for more details about the item metadata. This data frame can be easily created using the shape_df() function. If x = NULL, the se argument must be explicitly provided. Defaults to NULL.
cutscore	A numeric vector specifying the cut scores for classification. Cut scores are the points that separate different performance categories (e.g., pass vs. fail, or different grades).
theta	A numeric vector of ability estimates. Ability estimates (theta values) are the individual proficiency estimates obtained from the IRT model. The theta parameter is optional and can be NULL.

se	A numeric vector of the same length as theta representing the standard errors associated with each ability estimate. If NULL and x is supplied, standard errors are computed using the test information function. See the **Details** section for more information
weights	An optional two-column data frame or matrix where the first column is the quadrature points (nodes) and the second column is the corresponding weights. This is typically used in quadrature-based IRT analysis.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.

Details

This function first validates the input arguments. If both theta and weights are NULL, the function will stop and return an error message. Either theta or weights must be specified.

Either x or se must be specified. If se is not provided (i.e., se = NULL), it will be computed using the test information derived from the item metadata x. The length of se must match the length of theta, or the number of quadrature points in weights.

It then computes the probability that an examinee with a given ability is classified into each performance level using the normal distribution function centered at each theta (or quadrature point) with standard deviation se. These probabilities are used to calculate conditional classification accuracy (the probability of being correctly classified) and conditional classification consistency (the probability of being consistently classified upon repeated testing) for each ability value.

Finally, the function computes marginal classification accuracy and consistency across all examinees by aggregating the conditional indices with the associated weights.

Value

A list containing the following elements:

- confusion: A confusion matrix showing the cross table between true and expected levels.
- marginal: A data frame showing the marginal classification accuracy and consistency indices.
- conditional: A data frame showing the conditional classification accuracy and consistency indices.
- prob.level: A data frame showing the probability of being assigned to each level category.
- cutscore: A numeric vector showing the cut scores used in the analysis.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Rudner, L. M. (2001). Computing the expected proportions of misclassified examinees. *Practical Assessment, Research, and Evaluation, 7*(1), 14.

Rudner, L. M. (2005). Expected classification accuracy. *Practical Assessment, Research, and Evaluation, 10*(1), 13.

See Also

gen.weight(), est_score(), cac_lee()

Examples

```
## -----------------------------------------
# 1. Using the empirical ability distribution
## -----------------------------------------

# Import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read item parameter estimates and convert them into item metadata
x <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# Define cut scores on the theta scale
cutscore <- c(-2, -0.5, 0.8)

# Create quadrature points and corresponding weights
node <- seq(-4, 4, 0.25)
weights <- gen.weight(dist = "norm", mu = 0, sigma = 1, theta = node)

# Compute classification accuracy and consistency
cac_1 <- cac_rud(
  x = x,
  cutscore = cutscore,
  weights = weights,
  se = NULL,
  D = 1)
print(cac_1)

## -----------------------------------------
# 2. Using individual ability estimates
## -----------------------------------------

# Generate true abilities from N(0, 1)
set.seed(12)
theta <- rnorm(n = 1000, mean = 0, sd = 1)

# Simulate item response data
data <- simdat(x = x, theta = theta, D = 1)

# Estimate ability and standard errors using ML estimation
est_theta <- est_score(
  x = x, data = data, D = 1, method = "ML",
  range = c(-4, 4), se = TRUE
)
theta_hat <- est_theta$est.theta
se <- est_theta$se.theta

# Compute classification accuracy and consistency using provided SEs
cac_2 <- cac_rud(
```

```
  cutscore = cutscore,
  theta = theta_hat,
  se = se)
print(cac_2)

# Or compute classification accuracy and consistency using the item metadata
# instead of providing the SEs directly
cac_2 <- cac_rud(
  x = x,
  cutscore = cutscore,
  theta = theta_hat)
print(cac_2)
```

`catsib` *CATSIB DIF Detection Procedure*

Description

This function performs DIF analysis on items using the CATSIB procedure (Nandakumar & Roussos, 2004), a modified version of SIBTEST (Shealy & Stout, 1993). The CATSIB procedure is suitable for computerized adaptive testing (CAT) environments. In CATSIB, examinees are matched on IRT-based ability estimates that have been adjusted using a regression correction method (Shealy & Stout, 1993) to reduce statistical bias in the CATSIB statistic caused by impact.

Usage

```
catsib(
  x = NULL,
  data,
  score = NULL,
  se = NULL,
  group,
  focal.name,
  item.skip = NULL,
  D = 1,
  n.bin = c(80, 10),
  min.binsize = 3,
  max.del = 0.075,
  weight.group = c("comb", "foc", "ref"),
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  max.iter = 10,
```

```
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
data	A matrix of examinees' item responses corresponding to the items specified in the `x` argument. Rows represent examinees and columns represent items.
score	A numeric vector containing examinees' ability estimates (theta values). If not provided, `catsib()` will estimate ability parameters internally before computing the CATSIB statistics. See `est_score()` for more information on scoring methods. Default is `NULL`.
se	A vector of standard errors corresponding to the ability estimates. The order of the standard errors must match the order of the ability estimates provided in the `score` argument. Default is `NULL`.
group	A numeric or character vector indicating examinees' group membership. The length of the vector must match the number of rows in the response data matrix.
focal.name	A single numeric or character value specifying the focal group. For instance, given `group = c(0, 1, 0, 1, 1)` and '1' indicating the focal group, set `focal.name = 1`.
item.skip	A numeric vector of item indices to exclude from DIF analysis. If `NULL`, all items are included. Useful for omitting specific items based on prior insights.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
n.bin	A numeric vector of two positive integers specifying the maximum and minimum numbers of bins (or intervals) on the ability scale. The first and second values represent the maximum and minimum numbers of bins, respectively. Default is `c(80, 10)`. See the **Details** section below for more information.
min.binsize	A positive integer specifying the minimum number of examinees required in each bin. To ensure stable statistical estimation, each bin must contain at least the specified number of examinees from both the reference and focal groups in order to be included in the calculation of $\hat{\beta}$. Bins that do not meet this minimum are excluded from the computation. Default is 3. See the **Details** section for further explanation.

`max.del` A numeric value specifying the maximum allowable proportion of examinees that may be excluded from either the reference or focal group during the binning process. This threshold is used when determining the number of bins on the ability scale automatically. Default is 0.075. See the **Details** section for more information.

`weight.group` A character string specifying the target ability distribution used to compute the expected DIF measure $\hat{\beta}$ and its corresponding standard error. Available options are: `"comb"` for the combined distribution of both the reference and focal groups, `"foc"` for the focal group's distribution, and `"ref"` for the reference group's distribution. Default is `"comb"`. See the **Details** section below for more information.

`alpha` A numeric value specifying the significance level (α) for the hypothesis test associated with the CATSIB (*beta*) statistic. Default is 0.05.

`missing` A value indicating missing responses in the data set. Default is `NA`.

`purify` Logical. Indicates whether to apply a purification procedure. Default is `FALSE`.

`max.iter` A positive integer specifying the maximum number of iterations allowed for the purification process. Default is `10`.

`min.resp` A positive integer specifying the minimum number of valid item responses required from an examinee in order to compute an ability estimate. Default is `NULL`. See **Details** for more information.

`method` A character string indicating the scoring method to use. Available options are:

- `"ML"`: Maximum likelihood estimation
- `"WL"`: Weighted likelihood estimation (Warm, 1989)
- `"MAP"`: Maximum a posteriori estimation (Hambleton et al., 1991)
- `"EAP"`: Expected a posteriori estimation (Bock & Mislevy, 1982)

Default is `"ML"`.

`range` A numeric vector of length two specifying the lower and upper bounds of the ability scale. This is used for the following scoring methods: `"ML"`, `"WL"`, and `"MAP"`. Default is `c(-5, 5)`.

`norm.prior` A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. These values are used to generate the Gaussian quadrature points and weights. Ignored if `method` is `"ML"` or `"WL"`. Default is `c(0, 1)`.

`nquad` An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. Used only when `method` is `"EAP"`. Ignored for `"ML"`, `"WL"`, and `"MAP"`. Default is 41.

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and their corresponding weights (in the second column) for the latent variable prior distribution. The weights and points can be conveniently generated using the function `gen.weight()`.

If `NULL` and `method = "EAP"`, default quadrature values are generated based on the `norm.prior` and `nquad` arguments. Ignored if `method` is `"ML"`, `"WL"`, or `"MAP"`.

ncore	An integer specifying the number of logical CPU cores to use for parallel processing. Default is 1. See est_score() for details.
verbose	Logical. If TRUE, progress messages from the purification procedure will be displayed; if FALSE, the messages will be suppressed. Default is TRUE.
...	Additional arguments passed to the est_score() function.

Details

In the CATSIB procedure (Nandakumar & Roussos, 2004), $\hat{\theta}^*$— the expected value of θ regressed on $\hat{\theta}$—is a continuous variable. The range of $\hat{\theta}^*$ is divided into K equal-width intervals, and examinees are classified into one of these K intervals based on their $\hat{\theta}^*$ values. Any interval containing fewer than three examinees from either the reference or focal group is excluded from the computation of $\hat{\beta}$, the DIF effect size, to ensure statistical stability. According to Nandakumar and Roussos (2004), the default minimum bin size is 3, which can be controlled via the min.binsize argument.

To determine an appropriate number of intervals (K), catsib() automatically decreases K from a large starting value (e.g., 80) based on the rule proposed by Nandakumar and Roussos (2004). Specifically, if more than 7.5% of examinees in either the reference or focal group would be excluded due to small bin sizes, the number of bins is reduced by one and the process is repeated. This continues until the retained examinees in each group comprise at least 92.5\ few bins, they recommended a minimum of K = 10. Therefore, the default maximum and minimum number of bins are set to 80 and 10, respectively, via n.bin. Likewise, the maximum allowable proportion of excluded examinees is set to 0.075 by default through the max.del argument.

When it comes to the target ability distribution used to compute $\hat{\beta}$, Li and Stout (1996) and Nandakumar and Roussos (2004) employed the combined-group target ability distribution, which is the default option in weight.group. See Nandakumar and Roussos (2004) for further details about the CATSIB method.

Although Nandakumar and Roussos (2004) did not propose a purification procedure for DIF analysis using CATSIB, catsib() can implement an iterative purification process in a manner similar to that of Lim et al. (2022). Specifically, at each iteration, examinees' latent abilities are recalculated using the purified set of items and the scoring method specified in the method argument. The iterative purification process terminates either when no additional DIF items are detected or when the number of iterations reaches the limit set by max.iter. See Lim et al. (2022) for more details on the purification procedure.

Scoring based on a limited number of items may result in large standard errors, which can negatively affect the effectiveness of DIF detection using the CATSIB procedure. The min.resp argument can be used to prevent the use of scores with large standard errors, particularly during the purification process. For example, if min.resp is not NULL (e.g., min.resp = 5), item responses from examinees whose total number of valid responses is below the specified threshold are treated as missing (i.e., NA). As a result, their ability estimates are also treated as missing and are excluded from the CATSIB statistic computation. If min.resp = NULL, a score will be computed for any examinee with at least one valid item response.

Note that the regression correction (Eq. 7 in Nandakumar & Roussos, 2004) assumes $\hat{\rho}^2$ (the estimated reliability of ability estimates) lies in $[0, 1]$. In practice, however, $\hat{\rho}^2$ can become negative when the mean squared standard error of ability estimates exceeds the observed variance of ability estimates — a situation that can arise when (a) the number of items is very small, (b) a purification procedure removes many items, or (c) items exhibiting nonuniform DIF inflate the standard errors of

focal group examinees. A negative $\hat{\rho}^2$ causes the regression correction to amplify rather than attenuate group differences, leading to inflated Type I error rates. Even a small positive $\hat{\rho}^2$ (e.g., 0.03) can collapse the corrected ability scores so tightly around each group's mean that, when ability impact exists between groups, the two groups' corrected score distributions no longer overlap. This leaves no bins containing examinees from both groups, resulting in $\hat{\beta} = 0$ and $\text{SE}(\hat{\beta}) = 0$ for every item, which causes the purification loop to terminate early with invalid statistics. To prevent this correction collapse, `catsib()` enforces a floor of 0.05 on $\hat{\rho}^2$ — i.e., $\hat{\rho}^2 = \max(0.05, \min(1, 1 - \hat{\sigma}_e^2/\hat{\sigma}_{\hat{\theta}}^2))$ — so that a minimum degree of score spread is always preserved. When the unclamped $\hat{\rho}^2$ falls below 0.05 for either group, a warning is issued and DIF results from that iteration should be interpreted with caution. This situation typically arises during purification when too few items remain to yield reliable ability estimates. Users should also be aware that CATSIB, like its predecessor SIBTEST (Shealy & Stout, 1993), was originally designed and validated for detecting uniform DIF. Its statistical behavior under nonuniform or mixed DIF conditions has not been formally evaluated, and caution is warranted when interpreting results for items suspected of nonuniform DIF.

Value

This function returns a list consisting of four elements:

`no_purify` A list containing the results of the DIF analysis without applying a purification procedure. This list includes:

- **dif_stat** A data frame containing the results of the CATSIB statistics for all evaluated items. The columns include the item ID, CATSIB (*beta*) statistic, standard error of *beta*, standardized *beta*, p-value for *beta*, sample size of the reference group, sample size of the focal group, and total sample size.
- **dif_item** A numeric vector identifying items flagged as potential DIF items based on the CATSIB statistic.
- **contingency** A list of contingency tables used for computing the CATSIB statistics for each item.

`purify` A logical value indicating whether a purification procedure was applied.

`with_purify` A list containing the results of the DIF analysis with a purification procedure. This list includes:

- **dif_stat** A data frame containing the results of the CATSIB statistics for all evaluated items. The columns include the item ID, CATSIB (*beta*) statistic, standard error of *beta*, standardized *beta*, p-value for *beta*, sample size of the reference group, sample size of the focal group, total sample size, and the iteration number (*n*) in which the CATSIB statistics were computed.
- **dif_item** A numeric vector identifying items flagged as potential DIF items based on the CATSIB statistic.
- **n.iter** An integer indicating the total number of iterations performed during the purification process.
- **complete** A logical value indicating whether the purification process was completed. If FALSE, the process reached the maximum number of iterations without full convergence.
- **contingency** A list of contingency tables used for computing the CATSIB statistics for each item during the purification process.

`alpha` The significance level α used to compute the p-values of the CATSIB statistics.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Li, H. H., & Stout, W. (1996). A new procedure for detection of crossing DIF. *Psychometrika, 61*(4), 647-677.

Lim, H., Choe, E. M., & Han, K. T. (2022). A residual-based differential item functioning detection framework in item response theory. *Journal of Educational Measurement*.

Nandakumar, R., & Roussos, L. (2004). Evaluation of the CATSIB DIF procedure in a pretest setting. *Journal of Educational and Behavioral Statistics, 29*(2), 177-199.

Shealy, R. T., & Stout, W. F. (1993). A model-based standardization approach that separates true bias/DIF from group ability differences and detects test bias/DIF as well as item bias/DIF. *Psychometrika, 58*, 159–194.

See Also

rdif(), est_irt, est_item(), simdat(), shape_df(), est_score()

Examples

```
# Load required package
library("dplyr")

## Uniform DIF Detection
################################################
# (1) Simulate data with true uniform DIF
################################################

# Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Select 36 3PLM items that are non-DIF
par_nstd <-
  bring.flexmirt(file = flex_sam, "par")$Group1$full_df %>%
  dplyr::filter(.data$model == "3PLM") %>%
  dplyr::filter(dplyr::row_number() %in% 1:36) %>%
  dplyr::select(1:6)
par_nstd$id <- paste0("nondif", 1:36)

# Generate four new items to contain uniform DIF
difpar_ref <-
  shape_df(
    par.drm = list(a = c(0.8, 1.5, 0.8, 1.5), b = c(0.0, 0.0, -0.5, -0.5), g = 0.15),
    item.id = paste0("dif", 1:4), cats = 2, model = "3PLM"
  )

# Introduce uniform DIF in the focal group by shifting b-parameters
difpar_foc <-
  difpar_ref %>%
```

```
  dplyr::mutate_at(.vars = "par.2", .funs = function(x) x + rep(0.7, 4))

# Combine the 4 DIF and 36 non-DIF items for both reference and focal groups
# Threfore, the first four items now exhibit uniform DIF
par_ref <- rbind(difpar_ref, par_nstd)
par_foc <- rbind(difpar_foc, par_nstd)

# Generate true theta values
set.seed(123)
theta_ref <- rnorm(500, 0.0, 1.0)
theta_foc <- rnorm(500, 0.0, 1.0)

# Simulate response data
resp_ref <- simdat(par_ref, theta = theta_ref, D = 1)
resp_foc <- simdat(par_foc, theta = theta_foc, D = 1)
data <- rbind(resp_ref, resp_foc)

################################################
# (2) Estimate item and ability parameters
#     using the aggregated data
################################################

# Estimate item parameters
est_mod <- est_irt(data = data, D = 1, model = "3PLM")
est_par <- est_mod$par.est

# Estimate ability parameters using ML
theta_est <- est_score(x = est_par, data = data, method = "ML")
score <- theta_est$est.theta
se <- theta_est$se.theta

################################################
# (3) Conduct DIF analysis
################################################
# Create a vector of group membership indicators
# where '1' indicates the focal group
group <- c(rep(0, 500), rep(1, 500))

# (a)-1 Compute the CATSIB statistic using provided scores,
#       without purification
dif_1 <- catsib(
  x = NULL, data = data, D = 1, score = score, se = se, group = group, focal.name = 1,
  weight.group = "comb", alpha = 0.05, missing = NA, purify = FALSE
)
print(dif_1)

# (a)-2 Compute the CATSIB statistic using provided scores,
#       with purification
dif_2 <- catsib(
  x = est_par, data = data, D = 1, score = score, se = se, group = group, focal.name = 1,
  weight.group = "comb", alpha = 0.05, missing = NA, purify = TRUE
)
print(dif_2)
```

`covirt` *Asymptotic Variance-Covariance Matrices of Item Parameter Estimates*

Description

This function computes the analytical asymptotic variance-covariance matrices of item parameter estimates for dichotomous and polytomous IRT models, without requiring examinee response data. Given a set of item parameter estimates and the corresponding sample sizes, the function derives the matrices using analytical formulas (e.g., Li & Lissitz, 2004; Thissen & Wainer, 1982). The square roots of the diagonal elements (variances) provide the asymptotic standard errors of the maximum likelihood estimates.

Usage

```
covirt(
  x,
  D = 1,
  nstd = 1000,
  pcm.loc = NULL,
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL
)
```

Arguments

`x`	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
`D`	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
`nstd`	An integer or a vector of integers indicating the sample size(s). If a vector is provided, its length must match the number of items in the `x` argument. Default is 1,000. See **Details**.
`pcm.loc`	A vector of integers indicating the positions of items calibrated using the partial credit model (PCM). For PCM items, the variance-covariance matrices are computed only for the item category difficulty parameters. Default is `NULL`. See the **Details** for more information.
`norm.prior`	A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. These values are used to generate the Gaussian quadrature points and weights when `weights = NULL`. Default is `c(0, 1)`.

nquad	An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. The specified value is used when weights is not NULL. Default is 41.
weights	An optional two-column data frame or matrix where the first column is the quadrature points (nodes) and the second column is the corresponding weights. This is typically used in quadrature-based IRT analysis.

Details

The standard errors obtained from this analytical approach are generally considered lower bounds of the true standard errors (Thissen & Wainer, 1982). Thus, they may serve as useful approximations for evaluating the precision of item parameter estimates when empirical standard errors are not reported in the literature or research reports.

If the item parameters provided in the x argument were calibrated using different sample sizes, a corresponding vector of sample sizes must be specified via the nstd argument. For example, suppose you wish to compute the variance-covariance matrices of five 3PLM items that were calibrated using 500, 600, 1,000, 2,000, and 700 examinees, respectively. In this case, set `nstd = c(500, 600, 1000, 2000, 700)`.

Since the item metadata allows only "GPCM" to denote both the partial credit model (PCM) and the generalized partial credit model (GPCM), PCM items must be explicitly identified using the pcm.loc argument. This is necessary because the category difficulty parameters of PCM items require special handling when computing variance-covariance matrices. For instance, if you wish to compute the matrices for five polytomous items and the last two were calibrated using PCM, then specify `pcm.loc = c(4, 5)`.

Value

A named list with the following two components:

- cov: A named list of variance-covariance matrices for item parameter estimates. Each element corresponds to a single item and contains a square matrix whose dimensions match the number of estimated parameters for that item. For dichotomous items, this typically includes slopes and intercepts. For polytomous items, it includes category difficulty parameters (for PCM) or both slope and difficulty (or threshold) parameters (for GRM and GPCM).
- se: A named list of vectors containing the asymptotic standard errors (ASEs) of the item parameter estimates, computed as the square roots of the diagonal elements of each corresponding variance-covariance matrix in cov.

The names of the list elements in both cov and se correspond to the item identifiers (e.g., item names or labels) as given in the first column of the input x.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Li, Y. & Lissitz, R. (2004). Applications of the analytically derived asymptotic standard errors of item response theory item parameter estimates. *Journal of educational measurement, 41*(2), 85-117.

Thissen, D. & Wainer, H. (1982). Some standard errors in item response theory. *Psychometrika, 47*, 397-412.

See Also

est_irt(), simdat(), shape_df(), gen.weight()

Examples

```
# Example using a "-prm.txt" file exported from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Select the first two dichotomous items and the last polytomous item
x <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df[c(1:2, 55), ]

# Compute the variance-covariance matrices assuming a sample size of 2,000
covirt(x, D = 1, nstd = 2000, norm.prior = c(0, 1), nquad = 41)
```

---

crdif *Residual-Based DIF Detection Framework Using Categorical Residuals (RDIF-CR)*

Description

This function computes three statistics of the residual-based DIF detection framework using categorical residuals (RDIF-CR)—$RDIF_R - CR$, $RDIF_S - CR$, and $RDIF_{RS} - CR$—for detecting global differential item functioning (DIF), particularly in polytomously scored items. The RDIF-CR framework evaluates DIF by comparing categorical residual vectors, which are calculated as the difference between a one-hot encoded response vector (with 1 for the selected category and 0 for all others) and the IRT model–predicted probability vector across all score categories. This approach enables fine-grained detection of global DIF patterns at the category level.

Usage

```
crdif(x, ...)

## Default S3 method:
crdif(
  x,
  data,
  score = NULL,
  group,
  focal.name,
  item.skip = NULL,
  D = 1,
  alpha = 0.05,
  missing = NA,
```

```
  purify = FALSE,
  purify.by = c("crdifrs", "crdifr", "crdifs"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'est_irt'
crdif(
  x,
  score = NULL,
  group,
  focal.name,
  item.skip = NULL,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("crdifrs", "crdifr", "crdifs"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'est_item'
crdif(
  x,
  group,
  focal.name,
  item.skip = NULL,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("crdifrs", "crdifr", "crdifs"),
  max.iter = 10,
```

```
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class `est_irt` obtained from `est_irt()`, or `est_item` from `est_item()`. See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
...	Additional arguments passed to the `est_score()` function.
data	A matrix of examinees' item responses corresponding to the items specified in the `x` argument. Rows represent examinees and columns represent items.
score	A numeric vector containing examinees' ability estimates (theta values). If not provided, `crdif()` will estimate ability parameters internally before computing the RDIF statistics. See `est_score()` for more information on scoring methods. Default is `NULL`.
group	A numeric or character vector indicating examinees' group membership. The length of the vector must match the number of rows in the response data matrix.
focal.name	A single numeric or character value specifying the focal group. For instance, given `group = c(0, 1, 0, 1, 1)` and '1' indicating the focal group, set `focal.name = 1`.
item.skip	A numeric vector of item indices to exclude from DIF analysis. If `NULL`, all items are included. Useful for omitting specific items based on prior insights.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
alpha	A numeric value specifying the significance level (α) for hypothesis testing using the CRDIF statistics. Default is `0.05`.
missing	A value indicating missing responses in the data set. Default is `NA`.
purify	Logical. Indicates whether to apply a purification procedure. Default is `FALSE`.
purify.by	A character string specifying which RDIF statistic is used to perform the purification. Available options are `"crdifrs"` for $RDIF_{RS} - CR$, `"crdifr"` for $RDIF_R - CR$, and `"crdifs"` for $RDIF_S - CR$.
max.iter	A positive integer specifying the maximum number of iterations allowed for the purification process. Default is `10`.

`min.resp` A positive integer specifying the minimum number of valid item responses required from an examinee in order to compute an ability estimate. Default is `NULL`.

`method` A character string indicating the scoring method to use. Available options are:

- `"ML"`: Maximum likelihood estimation
- `"WL"`: Weighted likelihood estimation (Warm, 1989)
- `"MAP"`: Maximum a posteriori estimation (Hambleton et al., 1991)
- `"EAP"`: Expected a posteriori estimation (Bock & Mislevy, 1982)

Default is `"ML"`.

`range` A numeric vector of length two specifying the lower and upper bounds of the ability scale. This is used for the following scoring methods: `"ML"`, `"WL"`, and `"MAP"`. Default is `c(-5, 5)`.

`norm.prior` A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. These values are used to generate the Gaussian quadrature points and weights. Ignored if `method` is `"ML"` or `"WL"`. Default is `c(0, 1)`.

`nquad` An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. Used only when `method` is `"EAP"`. Ignored for `"ML"`, `"WL"`, and `"MAP"`. Default is 41.

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and their corresponding weights (in the second column) for the latent variable prior distribution. The weights and points can be conveniently generated using the function `gen.weight()`.

If `NULL` and `method = "EAP"`, default quadrature values are generated based on the `norm.prior` and `nquad` arguments. Ignored if `method` is `"ML"`, `"WL"`, or `"MAP"`.

`ncore` An integer specifying the number of logical CPU cores to use for parallel processing. Default is `1`. See `est_score()` for details.

`verbose` Logical. If `TRUE`, progress messages from the purification procedure will be displayed; if `FALSE`, the messages will be suppressed. Default is `TRUE`.

Details

According to Penfield (2010), differential item functioning (DIF) in polytomously scored items can be conceptualized in two forms: global DIF and net DIF. Global DIF refers to differences between groups in the conditional probabilities of responding in specific score categories, thus offering a fine-grained view of DIF at the category level. In contrast, net DIF summarizes these differences into a single value representing the overall impact of DIF on the item's expected score.

The RDIF framework using categorical residuals (RDIF-CR), implemented in `crdif()`, extends the original residual-based DIF framework proposed by Lim et al. (2022) to detect global DIF in polytomous items. This framework includes three statistics: $RDIF_R - CR$, $RDIF_S - CR$, and $RDIF_{RS} - CR$, each designed to capture different aspects of group-level differences in categorical response patterns.

To illustrate how the RDIF-CR framework operates, consider an item with five ordered score categories ($k \in \{0, 1, 2, 3, 4\}$). Suppose an examinee with latent ability θ responds with category

2. The one-hot encoded response vector for this response is $(0, 0, 1, 0, 0)^T$. Assume that the IRT model estimates the examinee's expected score as 2.5 and predicts the category probabilities as $(0.1, 0.2, 0.4, 0.25, 0.05)^T$. In the RDIF-CR framework, the categorical residual vector is calculated by subtracting the predicted probability vector from the one-hot response vector, resulting in $(-0.1, -0.2, 0.6, -0.25, -0.05)^T$.

In contrast to the RDIF-CR framework, net DIF is assessed using a unidimensional item score residual. In this example, the residual would be $2 - 2.5 = -0.5$. For detecting net DIF, the `rdif()` function should be used instead.

Note that for dichotomous items, `crdif()` and `rdif()` yield identical results. This is because the categorical probability vector for a binary item reduces to a scalar difference, making the global and net DIF evaluations mathematically equivalent.

Value

This function returns a list containing four main components:

`no_purify` A list of sub-objects containing the results of DIF analysis without applying a purification procedure. The sub-objects include:

- **dif_stat** A data frame summarizing the RDIF-CR analysis results for all items. Columns include item ID, $RDIF_R - CR$, degrees of freedom, $RDIF_S - CR$, degrees of freedom, $RDIF_{RS} - CR$, degrees of freedom, associated p-values, and sample sizes for the reference and focal groups.
- **moments** A list containing the first and second moments (means and covariance matrices) of the RDIF-CR statistics. The elements include: `mu.crdifr`, `mu.crdifs`, `mu.crdifrs` (means), and `cov.crdifr`, `cov.crdifs`, `cov.crdifrs` (covariances), each indexed by item ID.
- **dif_item** A list of three numeric vectors identifying items flagged as DIF based on each statistic: `crdifr`, `crdifs`, and `crdifrs`.
- **score** A numeric vector of ability estimates used to compute the RDIF-CR statistics. These may be user-supplied or internally estimated.

`purify` A logical value indicating whether a purification procedure was applied.

`with_purify` A list of sub-objects containing the results of DIF analysis after applying the purification procedure. The sub-objects include:

- **purify.by** A character string indicating the RDIF-CR statistic used for purification. Possible values are "crdifr", "crdifs", or "crdifrs".
- **dif_stat** A data frame summarizing the final RDIF-CR statistics after purification. Same structure as in `no_purify`, with an additional column indicating the iteration in which the result was obtained.
- **moments** A list of moments (means and covariance matrices) of the RDIF-CR statistics for all items, updated based on the final iteration.
- **dif_item** A list of three numeric vectors identifying items flagged as DIF at any iteration, by each statistic.
- **n.iter** An integer indicating the number of iterations performed during the purification procedure.
- **score** A numeric vector of updated ability estimates used in the final iteration.

complete A logical value indicating whether the purification process converged. If FALSE, the maximum number of iterations was reached before convergence.

alpha A numeric value indicating the significance level (α) used for hypothesis testing with RDIF-CR statistics.

Methods (by class)

- crdif(default): Default method for computing the three RDIF-CR statistics using a data frame x that contains item metadata
- crdif(est_irt): An object created by the function est_irt().
- crdif(est_item): An object created by the function est_item().

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Lim, H., Choe, E. M., & Han, K. T. (2022). A residual-based differential item functioning detection framework in item response theory. *Journal of Educational Measurement, 59*(1), 80-104. doi:10.1111/jedm.12313.

Penfield, R. D. (2010). Distinguishing between net and global DIF in polytomous items. *Journal of Educational Measurement, 47*(2), 129–149.

See Also

rdif(), est_irt(), est_item(), est_score()

Examples

```
##############################################################################
# This example demonstrates how to detect global DIF in polytomous items
# using the RDIF-CR framework implemented in `irtQ::crdif()`.
# Simulated response data are generated from 5 GRM items with 4 score
# categories. DIF is introduced in the 1st and 5th items.
##############################################################################

#############################################
# (1) Simulate response data with DIF
#############################################

set.seed(1)

# Generate ability parameters for 1000 examinees in each group
# Reference and focal groups follow N(0, 1.5^2)
theta_ref <- rnorm(1000, 0, 1.5)
theta_foc <- rnorm(1000, 0, 1.5)
```

```
# Combine abilities from both groups
theta_all <- c(theta_ref, theta_foc)

# Define item parameters using `irtQ::shape_df()`
# Items 1 and 5 are intentionally modified to exhibit DIF
par_ref <- irtQ::shape_df(
  par.prm = list(
    a = c(1, 1, 1, 2, 2),
    d = list(c(-2, 0, 1),
             c(-2, 0, 2),
             c(-2, 0, 1),
             c(-1, 0, 2),
             c(-2, 0, 0.5))
  ),
  cats = 4, model = "GRM"
)

par_foc <- irtQ::shape_df(
  par.prm = list(
    a = c(2, 1, 1, 2, 0.5),
    d = list(c(-0.5, 0, 0.5),
             c(-2, 0, 2),
             c(-2, 0, 1),
             c(-1, 0, 2),
             c(-1.5, -1, 0))
  ),
  cats = 4, model = "GRM"
)

# Generate response data
resp_ref <- irtQ::simdat(x = par_ref, theta = theta_ref, D = 1)
resp_foc <- irtQ::simdat(x = par_foc, theta = theta_foc, D = 1)

# Combine response data across groups
data <- rbind(resp_ref, resp_foc)

################################################
# (2) Estimate item and ability parameters
################################################

# Estimate GRM item parameters using `irtQ::est_irt()`
fit_mod <- irtQ::est_irt(data = data, D = 1, model = "GRM", cats = 4)

# Extract estimated item parameters
x <- fit_mod$par.est

# Estimate ability scores using ML method
score <- est_score(x = x, data = data, method = "ML")$est.theta

################################################
# (3) Perform RDIF-CR DIF analysis
################################################
```

```
# Define group membership: 1 = focal group
group <- c(rep(0, 1000), rep(1, 1000))

# (a) DIF detection without purification
dif_nopuri <- crdif(
  x = x, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05
)
print(dif_nopuri)

# (b) DIF detection with purification using RDIF_{R}-CR
dif_puri_1 <- crdif(
  x = x, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "crdifr"
)
print(dif_puri_1)

# (c) DIF detection with purification using RDIF_{S}-CR
dif_puri_2 <- crdif(
  x = x, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "crdifs"
)
print(dif_puri_2)

# (d) DIF detection with purification using RDIF_{RS}-CR
dif_puri_3 <- crdif(
  x = x, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "crdifrs"
)
print(dif_puri_3)
```

`drm` *Dichotomous Response Model (DRM) Probabilities*

Description

This function computes the probability of a correct response for multiple items given a set of theta values using the 1PL, 2PL, or 3PL item response models.

Usage

```
drm(theta, a, b, g = NULL, D = 1)
```

Arguments

theta	A numeric vector of ability values (latent traits).
a	A numeric vector of item discrimination (slope) parameters.
b	A numeric vector of item difficulty parameters.
g	A numeric vector of item guessing parameters. Not required for 1PL or 2PL models.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.

Details

If g is not specified, the function assumes a guessing parameter of 0 for all items, corresponding to the 1PL or 2PL model. The function automatically adjusts the model form based on the presence of g.

Value

A matrix of response probabilities, where rows represent ability values (theta) and columns represent items.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

prm()

Examples

```
## Example 1: theta and item parameters for 3PL model
drm(c(-0.1, 0.0, 1.5), a = c(1, 2), b = c(0, 1), g = c(0.2, 0.1), D = 1)

## Example 2: single theta value with 2PL item parameters
drm(0.0, a = c(1, 2), b = c(0, 1), D = 1)

## Example 3: multiple theta values with a single item (3PL model)
drm(c(-0.1, 0.0, 1.5), a = 1, b = 1, g = 0.2, D = 1)
```

`est_irt` *Item parameter estimation using MMLE-EM algorithm*

Description

This function fits unidimensional item response theory (IRT) models to mixed-format data comprising both dichotomous and polytomous items, using marginal maximum likelihood estimation via the expectation–maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981). It also supports fixed item parameter calibration (FIPC; Kim, 2006), a practical method for pretest (or newly developed) item calibration in computerized adaptive testing (CAT). FIPC enables the parameter estimates of pretest items to be placed on the same scale as those of operational items (Ban et al., 2001). For dichotomous items, the function supports the one-, two-, and three-parameter logistic models. For polytomous items, it supports the graded response model (GRM) and the (generalized) partial credit model (GPCM).

Usage

```
est_irt(
  x = NULL,
  data,
  D = 1,
  model = NULL,
  cats = NULL,
  item.id = NULL,
  fix.a.1pl = FALSE,
  fix.a.gpcm = FALSE,
  fix.g = FALSE,
  a.val.1pl = 1,
  a.val.gpcm = 1,
  g.val = 0.2,
  use.aprior = FALSE,
  use.bprior = FALSE,
  use.gprior = TRUE,
  aprior = list(dist = "lnorm", params = c(0, 0.5)),
  bprior = list(dist = "norm", params = c(0, 1)),
  gprior = list(dist = "beta", params = c(5, 16)),
  missing = NA,
  Quadrature = c(49, 6),
  weights = NULL,
  group.mean = 0,
  group.var = 1,
  EmpHist = FALSE,
  use.startval = FALSE,
  Etol = 1e-04,
  MaxE = 500,
  control = list(eval.max = 500, iter.max = 200, x.tol = 1e-04),
  fipc = FALSE,
```

```
  fipc.method = "MEM",
  fix.loc = NULL,
  fix.id = NULL,
  se = TRUE,
  verbose = TRUE
)
```

Arguments

x	A data frame containing item metadata. This metadata is required to retrieve essential information for each item (e.g., number of score categories, IRT model type, etc.) necessary for calibration. You can create an empty item metadata frame using the function shape_df(). When use.startval = TRUE, the item parameters specified in the metadata will be used as starting values for parameter estimation. If x = NULL, both model and cats arguments must be specified. Note that when fipc = TRUE to implement FIPC, item metadata for the test form must be supplied via the x argument. See **below** for more details. Default is NULL.
data	A matrix of examinees' item responses corresponding to the items specified in the x argument. Rows represent examinees and columns represent items.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
model	A character vector specifying the IRT model to fit each item. Available values are: • "1PLM", "2PLM", "3PLM", "DRM" for dichotomous items • "GRM", "GPCM" for polytomous items Here, "GRM" denotes the graded response model and "GPCM" the (generalized) partial credit model. Note that "DRM" serves as a general label covering all three dichotomous IRT models. If a single model name is provided, it is recycled for all items. This argument is only used when x = NULL and fipc = FALSE. Default is NULL.
cats	Numeric vector specifying the number of score categories per item. For dichotomous items, this should be 2. If a single value is supplied, it will be recycled across all items. When cats = NULL and all models specified in the model argument are dichotomous ("1PLM", "2PLM", "3PLM", or "DRM"), the function defaults to 2 categories per item. This argument is used only when x = NULL and fipc = FALSE. Default is NULL.
item.id	Character vector of item identifiers. If NULL, IDs are generated automatically. When fipc = TRUE, a provided item.id will override any IDs present in x. Default is NULL.
fix.a.1pl	Logical. If TRUE, the slope parameters of all 1PLM items are fixed to a.val.1pl; otherwise, they are constrained to be equal and estimated. Default is FALSE.
fix.a.gpcm	Logical. If TRUE, GPCM items are calibrated as PCM with slopes fixed to a.val.gpcm; otherwise, each item's slope is estimated. Default is FALSE.

`fix.g` Logical. If `TRUE`, all 3PLM guessing parameters are fixed to `g.val`; otherwise, each guessing parameter is estimated. Default is `FALSE`.

`a.val.1pl` Numeric. Value to which the slope parameters of 1PLM items are fixed when `fix.a.1pl = TRUE`. Default is 1.

`a.val.gpcm` Numeric. Value to which the slope parameters of GPCM items are fixed when `fix.a.gpcm = TRUE`. Default is 1.

`g.val` Numeric. Value to which the guessing parameters of 3PLM items are fixed when `fix.g = TRUE`. Default is 0.2.

`use.aprior` Logical. If `TRUE`, applies a prior distribution to all item discrimination (slope) parameters during calibration. Default is `FALSE`.

`use.bprior` Logical. If `TRUE`, applies a prior distribution to all item difficulty (or threshold) parameters during calibration. Default is `FALSE`.

`use.gprior` Logical. If `TRUE`, applies a prior distribution to all 3PLM guessing parameters during calibration. Default is `TRUE`.

`aprior, bprior, gprior`
A list specifying the prior distribution for all item discrimination (slope), difficulty (or threshold), guessing parameters. Three distributions are supported: Beta, Log-normal, and Normal. The list must have two elements:

- `dist`: A character string, one of `"beta"`, `"lnorm"`, or `"norm"`.
- `params`: A numeric vector of length two giving the distribution's parameters. For details on each parameterization, see `stats::dbeta()`, `stats::dlnorm()`, and `stats::dnorm()`.

Defaults are:

- `aprior = list(dist = "lnorm", params = c(0.0, 0.5))`
- `bprior = list(dist = "norm", params = c(0.0, 1.0))`
- `gprior = list(dist = "beta", params = c(5, 16))`

for discrimination, difficulty, and guessing parameters, respectively.

`missing` A value indicating missing responses in the data set. Default is `NA`.

`Quadrature` A numeric vector of length two:

- first element: number of quadrature points
- second element: symmetric bound (absolute value) for those points For example, `c(49, 6)` specifies 49 evenly spaced points from –6 to 6. These points are used in the E-step of the EM algorithm. Default is `c(49, 6)`.

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and their corresponding weights (in the second column) for the latent variable prior distribution. If not `NULL`, the scale of the latent ability distribution is fixed to match the scale of the provided quadrature points and weights. The weights and points can be conveniently generated using the function `gen.weight()`.

If `NULL`, a normal prior density is used instead, based on the information provided in the `Quadrature`, `group.mean`, and `group.var` arguments. Default is `NULL`.

`group.mean` A numeric value specifying the mean of the latent variable prior distribution when `weights = NULL`. Default is 0. This value is fixed to resolve the indeterminacy of the item parameter scale during calibration. However, the scale of the prior distribution is updated when FIPC is implemented.

`group.var` A positive numeric value specifying the variance of the latent variable prior distribution when `weights = NULL`. Default is 1. This value is fixed to resolve the indeterminacy of the item parameter scale during calibration. However, the scale of the prior distribution is updated when FIPC is implemented.

`EmpHist` Logical. If `TRUE`, the empirical histogram of the latent variable prior distribution is estimated simultaneously with the item parameters using the approach proposed by Woods (2007). Item calibration is conducted relative to the estimated empirical prior. See below for details.

`use.startval` Logical. If `TRUE`, the item parameters provided in the item metadata (i.e., the `x` argument) are used as starting values for item parameter estimation. Otherwise, internally generated starting values are used. Default is `FALSE`.

`Etol` A positive numeric value specifying the convergence criterion for the E-step of the EM algorithm. Default is 1e-4. Specifically, the EM algorithm terminates when the largest absolute difference in item parameter estimates between consecutive iterations is smaller than this value.

`MaxE` A positive integer specifying the maximum number of iterations for the E-step in the EM algorithm. Default is `500`.

`control` A named list of options passed directly to `stats::nlminb()` in each M-step optimization of the EM algorithm. By default: `control = list(eval.max = 500, iter.max = 200, x.tol = 1e-4)`, where

- `eval.max` = 500 limits the number of function evaluations
- `iter.max` = 200 caps the number of internal optimizer iterations
- `x.tol` = 1e-4 sets the absolute change threshold in parameter values below which `stats::nlminb()` considers the solution to have converged Users may additionally supply other `nlminb()` control options (such as `abs.tol`, `rel.tol`, `trace`, etc.) as needed.

`fipc` Logical. If `TRUE`, fixed item parameter calibration (FIPC) is applied during item parameter estimation. When `fipc = TRUE`, the information on which items are fixed must be provided via either `fix.loc` or `fix.id`. See below for details.

`fipc.method` A character string specifying the FIPC method. Available options are:

- `"OEM"`: No Prior Weights Updating and One EM Cycle (NWU-OEM; Wainer & Mislevy, 1990)
- `"MEM"`: Multiple Prior Weights Updating and Multiple EM Cycles (MWU-MEM; Kim, 2006) When `fipc.method = "OEM"`, the maximum number of E-steps is automatically set to 1, regardless of the value specified in `MaxE`.

`fix.loc` A vector of positive integers specifying the row positions of the items to be fixed in the item metadata (i.e., `x`) when FIPC is implemented (i.e., `fipc = TRUE`). For example, suppose that five items located in the 1st, 2nd, 4th, 7th, and 9th rows of `x` should be fixed. Then use `fix.loc = c(1, 2, 4, 7, 9)`. Note that if `fix.id` is not `NULL`, the information provided in `fix.loc` is ignored. See below for details.

fix.id	A character vector specifying the IDs of the items to be fixed when FIPC is implemented (i.e., fipc = TRUE). For example, suppose five items with IDs "CMC1", "CMC2", "CMC3", "CMC4", and "CMC5" are to be fixed, and that all item IDs are supplied via item.id column in the x argument. Then use fix.id = c("CMC1", "CMC2", "CMC3", "CMC4", "CMC5"). Note that if fix.id is not NULL, the information in fix.loc is ignored. See below for details.
se	Logical. If FALSE, standard errors of the item parameter estimates are not computed. Default is TRUE.
verbose	Logical. If FALSE, all progress messages, including information about the EM algorithm process, are suppressed. Default is TRUE.

Details

A specific format of data frame should be used for the argument x. The first column should contain item IDs, the second column should contain the number of unique score categories for each item, and the third column should specify the IRT model to be fitted to each item. Available IRT models are:

- "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous item data
- "GRM" and "GPCM" for polytomous item data

Note that "DRM" serves as a general label covering all dichotomous IRT models (i.e., "1PLM", "2PLM", and "3PLM"), while "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively.

The subsequent columns should contain the item parameters for the specified models. For dichotomous items, the fourth, fifth, and sixth columns represent item discrimination (slope), item difficulty, and item guessing parameters, respectively. When "1PLM" or "2PLM" is specified in the third column, NAs must be entered in the sixth column for the guessing parameters.

For polytomous items, the item discrimination (slope) parameter should appear in the fourth column, and the item difficulty (or threshold) parameters for category boundaries should occupy the fifth through the last columns. When the number of unique score categories differs across items, unused parameter cells should be filled with NAs.

In the **irtQ** package, the threshold parameters for GPCM items are expressed as the item location (or overall difficulty) minus the threshold values for each score category. Note that when a GPCM item has *K* unique score categories, *K - 1* threshold parameters are required, since the threshold for the first category boundary is always fixed at 0. For example, if a GPCM item has five score categories, four threshold parameters must be provided.

An example of a data frame for a single-format test is shown below:

ITEM1	2	1PLM	1.000	1.461	NA
ITEM2	2	2PLM	1.921	-1.049	NA
ITEM3	2	3PLM	1.736	1.501	0.203
ITEM4	2	3PLM	0.835	-1.049	0.182
ITEM5	2	DRM	0.926	0.394	0.099

An example of a data frame for a mixed-format test is shown below:

ITEM1	2	1PLM	1.000	1.461	NA	NA	NA
ITEM2	2	2PLM	1.921	-1.049	NA	NA	NA
ITEM3	2	3PLM	0.926	0.394	0.099	NA	NA
ITEM4	2	DRM	1.052	-0.407	0.201	NA	NA
ITEM5	4	GRM	1.913	-1.869	-1.238	-0.714	NA
ITEM6	5	GRM	1.278	-0.724	-0.068	0.568	1.072
ITEM7	4	GPCM	1.137	-0.374	0.215	0.848	NA
ITEM8	5	GPCM	1.233	-2.078	-1.347	-0.705	-0.116

See the *IRT Models* section in the irtQ-package documentation for more details about the IRT models used in the **irtQ** package. A convenient way to create a data frame for the argument `x` is by using the function `shape_df()`.

To fit IRT models to data, the item response data must be accompanied by information on the IRT model and the number of score categories for each item. There are two ways to provide this information:

1. Supply item metadata to the argument `x`. As explained above, such metadata can be easily created using `shape_df()`.
2. Specify the IRT models and score category information directly through the arguments `model` and `cats`.

If `x = NULL`, the function uses the information specified in `model` and `cats`.

To implement FIPC, the item metadata must be provided via the `x` argument. This is because the item parameters of the fixed items in the metadata are used to estimate the characteristics of the underlying latent variable prior distribution when calibrating the remaining (freely estimated) items. More specifically, the latent prior distribution is estimated based on the fixed items, and then used to calibrate the new (pretest) items so that their parameters are placed on the same scale as those of the fixed items (Kim, 2006).The full item metadata, including both fixed and non-fixed items, can be conveniently created using the `shape_df_fipc()` function.

In terms of approaches for FIPC, Kim (2006) described five different methods. Among them, two methods are available in the `est_irt()` function. The first method is `"NWU-OEM"`, which uses a single E-step in the EM algorithm (involving only the fixed items) followed by a single M-step (involving only the non-fixed items). This method was proposed by Wainer and Mislevy (1990) in the context of online calibration and can be implemented by setting `fipc.method = "OEM"`.

The second method is `"MWU-MEM"`, which iteratively updates the latent variable prior distribution and estimates the parameters of the non-fixed items. In this method, the same procedure as the NWU-OEM approach is applied during the first EM cycle. From the second cycle onward, both the parameters of the non-fixed items and the weights of the prior distribution are concurrently updated. This method can be implemented by setting `fipc.method = "MEM"`. See Kim (2006) for more details.

When `fipc = TRUE`, information about which items are to be fixed must be provided via either the `fix.loc` or `fix.id` argument. For example, suppose that five items with IDs "CMC1", "CMC2", "CMC3", "CMC4", and "CMC5" should be fixed, and all item IDs are provided via the `x` or `item.id` argument. Also, assume these five items are located in the 1st through 5th rows of the item metadata (i.e., `x`). In this case, the fixed items can be specified using either `fix.loc = c(1, 2, 3, 4, 5)` or `fix.id = c("CMC1", "CMC2", "CMC3", "CMC4", "CMC5")`. Note that if both `fix.loc` and `fix.id` are not `NULL`, the information in `fix.loc` is ignored.

When EmpHist = TRUE, the empirical histogram of the latent variable prior distribution (i.e., the densities at the quadrature points) is estimated simultaneously with the item parameters. If EmpHist = TRUE and fipc = TRUE, the scale parameters of the empirical prior distribution (e.g., mean and variance) are also estimated. If EmpHist = TRUE and fipc = FALSE, the scale parameters are fixed to the values specified in group.mean and group.var. When EmpHist = FALSE, a normal prior distribution is used instead. If fipc = TRUE, the scale parameters of this normal prior are estimated along with the item parameters. If fipc = FALSE, they are fixed to the values specified in group.mean and group.var.

Value

This function returns an object of class est_irt. The returned object contains the following components:

estimates	A data frame containing both the item parameter estimates and their corresponding standard errors.
par.est	A data frame of item parameter estimates, structured according to the item metadata format.
se.est	A data frame of standard errors for the item parameter estimates, computed using the cross-product approximation method (Meilijson, 1989).
pos.par	A data frame indicating the position index of each estimated item parameter. The position information is useful for interpreting the variance-covariance matrix of item parameter estimates
covariance	A variance-covariance matrix of the item parameter estimates.
loglikelihood	The marginal log-likelihood, calculated as the sum of the log-likelihoods across all items.
aic	Akaike Information Criterion (AIC) based on the log-likelihood.
bic	Bayesian Information Criterion (BIC) based on the log-likelihood.
group.par	A data frame containing the mean, variance, and standard deviation of the latent variable prior distribution.
weights	A two-column data frame of quadrature points (column 1) and corresponding weights (column 2) of the (updated) latent prior distribution.
posterior.dist	A matrix of normalized posterior densities for all response patterns at each quadrature point. Rows and columns represent response patterns and quadrature points, respectively.
data	A data frame of examinees' response data.
scale.D	The scaling factor used in the IRT model.
ncase	The total number of response patterns.
nitem	The total number of items in the response data.
Etol	The convergence criterion for the E-step of the EM algorithm.
MaxE	The maximum number of E-steps allowed in the EM algorithm.
aprior	A list describing the prior distribution used for discrimination parameters.
bprior	A list describing the prior distribution used for difficulty parameters.

gprior	A list describing the prior distribution used for guessing parameters.
npar.est	The total number of parameters estimated.
niter	The number of completed EM cycles.
maxpar.diff	The maximum absolute change in parameter estimates at convergence.
EMtime	Time (in seconds) spent on EM cycles.
SEtime	Time (in seconds) spent computing standard errors.
TotalTime	Total computation time (in seconds).
test.1	First-order test result indicating whether the gradient sufficiently vanished for solution stability.
test.2	Second-order test result indicating whether the information matrix is positive definite, a necessary condition for identifying a local maximum.
var.note	A note indicating whether the variance-covariance matrix was successfully obtained from the information matrix.
fipc	Logical. Indicates whether FIPC was used.
fipc.method	The method used for FIPC.
fix.loc	A vector of integers specifying the row locations of fixed items when FIPC was applied.

Note that you can easily extract components from the output using the `getirt()` function.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Ban, J. C., Hanson, B. A., Wang, T., Yi, Q., & Harris, D., J. (2001) A comparative study of on-line pretest item calibration/scaling methods in computerized adaptive testing. *Journal of Educational Measurement, 38*(3), 191-212.

Bock, R. D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika, 46*, 443-459.

Kim, S. (2006). A comparative study of IRT fixed parameter calibration methods. *Journal of Educational Measurement, 43*(4), 355-381.

Meilijson, I. (1989). A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society: Series B (Methodological), 51*, 127-138.

Stocking, M. L. (1988). *Scale drift in on-line calibration* (Research Rep. 88-28). Princeton, NJ: ETS.

Wainer, H., & Mislevy, R. J. (1990). Item response theory, item calibration, and proficiency estimation. In H. Wainer (Ed.), *Computer adaptive testing: A primer* (Chap. 4, pp.65-102). Hillsdale, NJ: Lawrence Erlbaum.

Woods, C. M. (2007). Empirical histograms in item response theory with ordinal data. *Educational and Psychological Measurement, 67*(1), 73-87.

See Also

shape_df(), shape_df_fipc(), getirt()

Examples

```
## ------------------------------------------------------------
## 1. Item parameter estimation for dichotomous item data (LSAT6)
## ------------------------------------------------------------
# Fit the 1PL model to LSAT6 data and estimate a common slope parameter
# (i.e., constrain slope parameters to be equal)
(mod.1pl.c <- est_irt(data = LSAT6, D = 1, model = "1PLM", cats = 2,
                      fix.a.1pl = FALSE))

# Display a summary of the estimation results
summary(mod.1pl.c)

# Extract the item parameter estimates
getirt(mod.1pl.c, what = "par.est")

# Extract the standard error estimates
getirt(mod.1pl.c, what = "se.est")

# Fit the 1PL model to LSAT6 data and fix slope parameters to 1.0
(mod.1pl.f <- est_irt(data = LSAT6, D = 1, model = "1PLM", cats = 2,
                      fix.a.1pl = TRUE, a.val.1pl = 1))

# Display a summary of the estimation results
summary(mod.1pl.f)

# Fit the 2PL model to LSAT6 data
(mod.2pl <- est_irt(data = LSAT6, D = 1, model = "2PLM", cats = 2))

# Display a summary of the estimation results
summary(mod.2pl)

# Assess the model fit for the 2PL model using the S-X2 fit statistic
(sx2fit.2pl <- sx2_fit(x = mod.2pl))

# Compute item and test information functions at a range of theta values
theta <- seq(-4, 4, 0.1)
(info.2pl <- info(x = mod.2pl, theta = theta))

# Plot the test characteristic curve (TCC)
(trace.2pl <- traceline(x = mod.2pl, theta = theta))
plot(trace.2pl)

# Plot the item characteristic curve (ICC) for the first item
plot(trace.2pl, item.loc = 1)

# Fit the 2PL model and simultaneously estimate an empirical histogram
# of the latent variable prior distribution
```

```
# Also apply a looser convergence threshold for the E-step
(mod.2pl.hist <- est_irt(data = LSAT6, D = 1, model = "2PLM", cats = 2,
                         EmpHist = TRUE, Etol = 0.001))
(emphist <- getirt(mod.2pl.hist, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# Fit the 3PL model and apply a Beta prior to the guessing parameters
(mod.3pl <- est_irt(
  data = LSAT6, D = 1, model = "3PLM", cats = 2, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 16))
))

# Display a summary of the estimation results
summary(mod.3pl)

# Fit the 3PL model and fix the guessing parameters at 0.2
(mod.3pl.f <- est_irt(data = LSAT6, D = 1, model = "3PLM", cats = 2,
                      fix.g = TRUE, g.val = 0.2))

# Display a summary of the estimation results
summary(mod.3pl.f)

# Fit different dichotomous models to each item in the LSAT6 data:
# Fit the constrained 1PL model to items 1-3, the 2PL model to item 4,
# and the 3PL model with a Beta prior on guessing to item 5
(mod.drm.mix <- est_irt(
  data = LSAT6, D = 1, model = c("1PLM", "1PLM", "1PLM", "2PLM", "3PLM"),
  cats = 2, fix.a.1pl = FALSE, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 16))
))

# Display a summary of the estimation results
summary(mod.drm.mix)

## ---------------------------------------------------------------------
## 2. Item parameter estimation for mixed-format data (simulated data)
## ---------------------------------------------------------------------
## Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Extract item metadata
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df

# Modify the item metadata so that the 39th and 40th items use the GPCM
x[39:40, 3] <- "GPCM"

# Generate 1,000 examinees' latent abilities from N(0, 1)
set.seed(37)
score1 <- rnorm(1000, mean = 0, sd = 1)

# Simulate item response data
sim.dat1 <- simdat(x = x, theta = score1, D = 1)
```

```
# Fit the 3PL model to all dichotomous items, the GPCM to items 39 and 40,
# and the GRM to items 53, 54, and 55.
# Use a Beta prior for guessing parameters, a log-normal prior for slope
# parameters, and a normal prior for difficulty (threshold) parameters.
# Also, specify the argument `x` to provide IRT model and score category information.
item.meta <- shape_df(item.id = x$id, cats = x$cats, model = x$model,
  default.par = TRUE)
(mod.mix1 <- est_irt(
  x = item.meta, data = sim.dat1, D = 1, use.aprior = TRUE, use.bprior = TRUE,
  use.gprior = TRUE,
  aprior = list(dist = "lnorm", params = c(0.0, 0.5)),
  bprior = list(dist = "norm", params = c(0.0, 2.0)),
  gprior = list(dist = "beta", params = c(5, 16))
))

# Display a summary of the estimation results
summary(mod.mix1)

# Estimate examinees' latent scores using MLE and the estimated item parameters
(score.mle <- est_score(x = mod.mix1, method = "ML", range = c(-4, 4), ncore = 2))

# Compute traditional model-fit statistics
(fit.mix1 <- irtfit(
  x = mod.mix1, score = score.mle$est.theta, group.method = "equal.width",
  n.width = 10, loc.theta = "middle"
))

# Residual plot for the first item (dichotomous)
plot(
  x = fit.mix1, item.loc = 1, type = "both", ci.method = "wald",
  show.table = TRUE, ylim.sr.adjust = TRUE
)

# Residual plot for the last item (polytomous)
plot(
  x = fit.mix1, item.loc = 55, type = "both", ci.method = "wald",
  show.table = FALSE, ylim.sr.adjust = TRUE
)

# Fit the 2PL model to all dichotomous items, the GPCM to items 39 and 40,
# and the GRM to items 53, 54, and 55.
# Provide IRT model and score category information via `model` and `cats`
# arguments.
(mod.mix2 <- est_irt(
  data = sim.dat1, D = 1,
  model = c(rep("2PLM", 38), rep("GPCM", 2), rep("2PLM", 12), rep("GRM", 3)),
  cats = c(rep(2, 38), rep(5, 2), rep(2, 12), rep(5, 3))
))

# Display a summary of the estimation results
summary(mod.mix2)

# Fit the 2PL model to all dichotomous items, the GPCM to items 39 and 40,
```

```
# and the GRM to items 53, 54, and 55.
# Also estimate the empirical histogram of the latent prior distribution.
# Provide IRT model and score category information via `model` and `cats` arguments.
(mod.mix3 <- est_irt(
  data = sim.dat1, D = 1,
  model = c(rep("2PLM", 38), rep("GPCM", 2), rep("2PLM", 12), rep("GRM", 3)),
  cats = c(rep(2, 38), rep(5, 2), rep(2, 12), rep(5, 3)), EmpHist = TRUE
))
(emphist <- getirt(mod.mix3, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# Fit the 2PL model to all dichotomous items, the PCM to items 39 and 40 by
# fixing slope parameters to 1, and the GRM to items 53, 54, and 55.
# Provide IRT model and score category information via `model` and `cats` arguments.
(mod.mix4 <- est_irt(
  data = sim.dat1, D = 1,
  model = c(rep("2PLM", 38), rep("GPCM", 2), rep("2PLM", 12), rep("GRM", 3)),
  cats = c(rep(2, 38), rep(5, 2), rep(2, 12), rep(5, 3)),
  fix.a.gpcm = TRUE, a.val.gpcm = 1
))

# Display a summary of the estimation results
summary(mod.mix4)

## -----------------------------------------------------------------
## 3. Fixed item parameter calibration (FIPC) for mixed-format data
##    (simulated)
## -----------------------------------------------------------------
## Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Select item metadata
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df

# Generate 1,000 examinees' latent abilities from N(0.4, 1.3)
set.seed(20)
score2 <- rnorm(1000, mean = 0.4, sd = 1.3)

# Simulate response data
sim.dat2 <- simdat(x = x, theta = score2, D = 1)

# Fit the 3PL model to all dichotomous items and the GRM to all polytomous items
# Fix five 3PL items (1st-5th) and three GRM items (53rd-55th)
# Also estimate the empirical histogram of the latent variable distribution
# Use the MEM method
fix.loc <- c(1:5, 53:55)
(mod.fix1 <- est_irt(
  x = x, data = sim.dat2, D = 1, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 16)), EmpHist = TRUE,
  Etol = 1e-3, fipc = TRUE, fipc.method = "MEM", fix.loc = fix.loc
))

# Extract group-level parameter estimates
```

```
(prior.par <- mod.fix1$group.par)

# Visualize the empirical prior distribution
(emphist <- getirt(mod.fix1, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# Display a summary of the estimation results
summary(mod.fix1)

# Alternatively, fix the same items by providing their item IDs
# using the `fix.id` argument. In this case, set `fix.loc = NULL`
fix.id <- c(x$id[1:5], x$id[53:55])
(mod.fix1 <- est_irt(
  x = x, data = sim.dat2, D = 1, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 16)), EmpHist = TRUE,
  Etol = 1e-3, fipc = TRUE, fipc.method = "MEM", fix.loc = NULL,
  fix.id = fix.id
))

# Display a summary of the estimation results
summary(mod.fix1)

# Fit the 3PL model to all dichotomous items and the GRM to all polytomous items
# Fix the same items as before (1st-5th and 53rd-55th)
# This time, do not estimate the empirical histogram of the latent prior
# Instead, estimate the scale of the normal prior distribution
# Use the MEM method
fix.loc <- c(1:5, 53:55)
(mod.fix2 <- est_irt(
  x = x, data = sim.dat2, D = 1, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 16)), EmpHist = FALSE,
  Etol = 1e-3, fipc = TRUE, fipc.method = "MEM", fix.loc = fix.loc
))

# Extract group-level parameter estimates
(prior.par <- mod.fix2$group.par)

# Visualize the prior distribution
(emphist <- getirt(mod.fix2, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# Fit the 3PL model to all dichotomous items and the GRM to all polytomous items
# Fix only the five 3PL items (1st-5th) and estimate the empirical histogram
# Use the OEM method (i.e., only one EM cycle is used)
fix.loc <- c(1:5)
(mod.fix3 <- est_irt(
  x = x, data = sim.dat2, D = 1, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 16)), EmpHist = TRUE,
  Etol = 1e-3, fipc = TRUE, fipc.method = "OEM", fix.loc = fix.loc
))

# Extract group-level parameter estimates
(prior.par <- mod.fix3$group.par)
```

```
# Visualize the prior distribution
(emphist <- getirt(mod.fix3, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# Display a summary of the estimation results
summary(mod.fix3)

# Fit the 3PL model to all dichotomous items and the GRM to all polytomous items
# Fix all 55 items and estimate only the latent ability distribution
# Use the MEM method
fix.loc <- c(1:55)
(mod.fix4 <- est_irt(
  x = x, data = sim.dat2, D = 1, EmpHist = TRUE,
  Etol = 1e-3, fipc = TRUE, fipc.method = "MEM", fix.loc = fix.loc
))

# Extract group-level parameter estimates
(prior.par <- mod.fix4$group.par)

# Visualize the prior distribution
(emphist <- getirt(mod.fix4, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# Display a summary of the estimation results
summary(mod.fix4)

# Alternatively, fix all 55 items by providing their item IDs
# using the `fix.id` argument. In this case, set `fix.loc = NULL`
fix.id <- x$id
(mod.fix4 <- est_irt(
  x = x, data = sim.dat2, D = 1, EmpHist = TRUE,
  Etol = 1e-3, fipc = TRUE, fipc.method = "MEM", fix.loc = NULL,
  fix.id = fix.id
))

# Display a summary of the estimation results
summary(mod.fix4)
```

`est_item` *Fixed ability parameter calibration*

Description

This function performs fixed ability parameter calibration (FAPC), often called Stocking's (1988) Method A, which is the maximum likelihood estimation of item parameters given ability estimates (Baker & Kim, 2004; Ban et al., 2001; Stocking, 1988). It can also be considered a special case of

joint maximum likelihood estimation in which only one cycle of item parameter estimation is conducted, conditioned on the given ability estimates (Birnbaum, 1968). FAPC is a potentially useful method for calibrating pretest (or newly developed) items in computerized adaptive testing (CAT), as it enables placing their parameter estimates on the same scale as operational items. In addition, it can be used to recalibrate operational items in the item bank to evaluate potential parameter drift (Chen & Wang, 2016; Stocking, 1988).

Usage

```
est_item(
  x = NULL,
  data,
  score,
  D = 1,
  model = NULL,
  cats = NULL,
  item.id = NULL,
  fix.a.1pl = FALSE,
  fix.a.gpcm = FALSE,
  fix.g = FALSE,
  a.val.1pl = 1,
  a.val.gpcm = 1,
  g.val = 0.2,
  use.aprior = FALSE,
  use.bprior = FALSE,
  use.gprior = TRUE,
  aprior = list(dist = "lnorm", params = c(0, 0.5)),
  bprior = list(dist = "norm", params = c(0, 1)),
  gprior = list(dist = "beta", params = c(5, 17)),
  missing = NA,
  use.startval = FALSE,
  control = list(eval.max = 500, iter.max = 200, x.tol = 1e-04),
  verbose = TRUE
)
```

Arguments

x	A data frame containing item metadata. This metadata is required to retrieve essential information for each item (e.g., number of score categories, IRT model type, etc.) necessary for calibration. You can create an empty item metadata frame using the function shape_df(). When use.startval = TRUE, the item parameters specified in the metadata will be used as starting values for parameter estimation. If x = NULL, both model and cats arguments must be specified. See est_irt() or simdat() for more details about the item metadata. Default is NULL.
data	A matrix of examinees' item responses corresponding to the items specified in the x argument. Rows represent examinees and columns represent items.
score	A numeric vector of examinees' ability estimates (theta values). The length of this vector must match the number of rows in the response data.

D — A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.

model — A character vector specifying the IRT model to fit each item. Available values are:

- "1PLM", "2PLM", "3PLM", "DRM" for dichotomous items
- "GRM", "GPCM" for polytomous items

Here, "GRM" denotes the graded response model and "GPCM" the (generalized) partial credit model. Note that "DRM" serves as a general label covering all three dichotomous IRT models. If a single model name is provided, it is recycled for all items. This argument is only used when x = NULL. Default is NULL.

cats — Numeric vector specifying the number of score categories per item. For dichotomous items, this should be 2. If a single value is supplied, it will be recycled across all items. When cats = NULL and all models specified in the model argument are dichotomous ("1PLM", "2PLM", "3PLM", or "DRM"), the function defaults to 2 categories per item. This argument is used only when x = NULL. Default is NULL.

item.id — Character vector of item identifiers. If NULL, IDs are generated automatically. When fipc = TRUE, a provided item.id will override any IDs present in x. Default is NULL.

fix.a.1pl — Logical. If TRUE, the slope parameters of all 1PLM items are fixed to a.val.1pl; otherwise, they are constrained to be equal and estimated. Default is FALSE.

fix.a.gpcm — Logical. If TRUE, GPCM items are calibrated as PCM with slopes fixed to a.val.gpcm; otherwise, each item's slope is estimated. Default is FALSE.

fix.g — Logical. If TRUE, all 3PLM guessing parameters are fixed to g.val; otherwise, each guessing parameter is estimated. Default is FALSE.

a.val.1pl — Numeric. Value to which the slope parameters of 1PLM items are fixed when fix.a.1pl = TRUE. Default is 1.

a.val.gpcm — Numeric. Value to which the slope parameters of GPCM items are fixed when fix.a.gpcm = TRUE. Default is 1.

g.val — Numeric. Value to which the guessing parameters of 3PLM items are fixed when fix.g = TRUE. Default is 0.2.

use.aprior — Logical. If TRUE, applies a prior distribution to all item discrimination (slope) parameters during calibration. Default is FALSE.

use.bprior — Logical. If TRUE, applies a prior distribution to all item difficulty (or threshold) parameters during calibration. Default is FALSE.

use.gprior — Logical. If TRUE, applies a prior distribution to all 3PLM guessing parameters during calibration. Default is TRUE.

aprior, bprior, gprior — A list specifying the prior distribution for all item discrimination (slope), difficulty (or threshold), guessing parameters. Three distributions are supported: Beta, Log-normal, and Normal. The list must have two elements:

- dist: A character string, one of "beta", "lnorm", or "norm".

- params: A numeric vector of length two giving the distribution's parameters. For details on each parameterization, see stats::dbeta(), stats::dlnorm(), and stats::dnorm().

Defaults are:

- aprior = list(dist = "lnorm", params = c(0.0, 0.5))
- bprior = list(dist = "norm", params = c(0.0, 1.0))
- gprior = list(dist = "beta", params = c(5, 16))

for discrimination, difficulty, and guessing parameters, respectively.

missing — A value indicating missing responses in the data set. Default is NA.

use.startval — Logical. If TRUE, the item parameters provided in the item metadata (i.e., the x argument) are used as starting values for item parameter estimation. Otherwise, internally generated starting values are used. Default is FALSE.

control — A named list of options passed directly to stats::nlminb(). These parameters define settings for the item parameter estimation process, such as the maximum number of iterations. By default: control = list(eval.max = 500, iter.max = 200, x.tol = 1e-4), where

- eval.max = 500 limits the number of function evaluations
- iter.max = 200 caps the number of internal optimizer iterations
- x.tol = 1e-4 sets the absolute change threshold in parameter values below which stats::nlminb() considers the solution to have converged Users may additionally supply other nlminb() control options (such as abs.tol, rel.tol, trace, etc.) as needed.

verbose — Logical. If FALSE, all progress messages are suppressed. Default is TRUE.

Details

In most cases, the function est_item() returns successfully converged item parameter estimates using its default internal starting values. However, if convergence issues arise during calibration, one possible solution is to use alternative starting values. If item parameter values are already specified in the item metadata (i.e., the x argument), they can be used as starting values for item parameter calibration by setting use.startval = TRUE.

Value

This function returns an object of class est_item. The returned object contains the following components:

estimates — A data frame containing both the item parameter estimates and their corresponding standard errors.

par.est — A data frame of item parameter estimates, structured according to the item metadata format.

se.est — A data frame of standard errors for the item parameter estimates, computed based on the observed information functions

pos.par — A data frame indicating the position of each item parameter within the estimation vector. Useful for interpreting the variance-covariance matrix.

covariance	A variance-covariance matrix of the item parameter estimates.
loglikelihood	The total log-likelihood value computed across all estimated items based on the complete response data.
data	A data frame of examinees' response data.
score	A vector of examinees' ability estimates used as fixed values during item parameter estimation.
scale.D	The scaling factor used in the IRT model.
convergence	A message indicating whether item parameter estimation successfully converged.
nitem	The total number of items in the response data.
deleted.item	Items with no response data. These items are excluded from the item parameter estimation.
npar.est	The total number of parameters estimated.
n.response	An integer vector indicating the number of valid responses for each item used in the item parameter estimation.
TotalTime	Total computation time in seconds.

Note that you can easily extract components from the output using the `getirt()` function.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Baker, F. B., & Kim, S. H. (2004). *Item response theory: Parameter estimation techniques.* CRC Press.

Ban, J. C., Hanson, B. A., Wang, T., Yi, Q., & Harris, D., J. (2001) A comparative study of on-line pretest item calibration/scaling methods in computerized adaptive testing. *Journal of Educational Measurement, 38*(3), 191-212.

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

Chen, P., & Wang, C. (2016). A new online calibration method for multidimensional computerized adaptive testing. *Psychometrika, 81*(3), 674-701.

Stocking, M. L. (1988). *Scale drift in on-line calibration* (Research Rep. 88-28). Princeton, NJ: ETS.

See Also

`est_irt()`, `shape_df()`, `getirt()`

Examples

```
## Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Extract the item metadata
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df

# Modify the item metadata so that some items follow 1PLM, 2PLM, and GPCM
x[c(1:3, 5), 3] <- "1PLM"
x[c(1:3, 5), 4] <- 1
x[c(1:3, 5), 6] <- 0
x[c(4, 8:12), 3] <- "2PLM"
x[c(4, 8:12), 6] <- 0
x[54:55, 3] <- "GPCM"

# Generate examinees' abilities from N(0, 1)
set.seed(23)
score <- rnorm(500, mean = 0, sd = 1)

# Simulate response data based on the item metadata and ability values
data <- simdat(x = x, theta = score, D = 1)


# 1) Estimate item parameters: constrain the slope parameters of 1PLM items
#    to be equal
(mod1 <- est_item(x, data, score,
  D = 1, fix.a.1pl = FALSE, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 17)), use.startval = FALSE
))
summary(mod1)

# Extract the item parameter estimates
getirt(mod1, what = "par.est")

# 2) Estimate item parameters: fix the slope parameters of 1PLM items to 1
(mod2 <- est_item(x, data, score,
  D = 1, fix.a.1pl = TRUE, a.val.1pl = 1, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 17)), use.startval = FALSE
))
summary(mod2)

# Extract the standard error estimates
getirt(mod2, what = "se.est")

# 3) Estimate item parameters: fix the guessing parameters of 3PLM items to 0.2
(mod3 <- est_item(x, data, score,
  D = 1, fix.a.1pl = TRUE, fix.g = TRUE, a.val.1pl = 1, g.val = .2,
  use.startval = FALSE
))
summary(mod3)

# Extract both item parameter and standard error estimates
```

```
getirt(mod2, what = "estimates")
```

---

est_mg *Multiple-group item calibration using MMLE-EM algorithm*

Description

This function performs multiple-group item calibration (Bock & Zimowski, 1997) using marginal maximum likelihood estimation via the expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981). It also supports multiple-group fixed item parameter calibration (MG-FIPC; e.g., Kim & Kolen, 2016), which extends the single-group FIPC method (Kim, 2006) to multiple-group settings. For dichotomous items, the function supports one-, two-, and three-parameter logistic IRT models. For polytomous items, the graded response model (GRM) and the (generalized) partial credit model (GPCM) are available.

Usage

```
est_mg(
  x = NULL,
  data,
  group.name = NULL,
  D = 1,
  model = NULL,
  cats = NULL,
  item.id = NULL,
  free.group = NULL,
  fix.a.1pl = FALSE,
  fix.a.gpcm = FALSE,
  fix.g = FALSE,
  a.val.1pl = 1,
  a.val.gpcm = 1,
  g.val = 0.2,
  use.aprior = FALSE,
  use.bprior = FALSE,
  use.gprior = TRUE,
  aprior = list(dist = "lnorm", params = c(0, 0.5)),
  bprior = list(dist = "norm", params = c(0, 1)),
  gprior = list(dist = "beta", params = c(5, 16)),
  missing = NA,
  Quadrature = c(49, 6),
  weights = NULL,
  group.mean = 0,
  group.var = 1,
  EmpHist = FALSE,
```

```
  use.startval = FALSE,
  Etol = 0.001,
  MaxE = 500,
  control = list(eval.max = 500, iter.max = 200, x.tol = 1e-04),
  fipc = FALSE,
  fipc.method = "MEM",
  fix.loc = NULL,
  fix.id = NULL,
  se = TRUE,
  verbose = TRUE
)
```

Arguments

x	A list containing item metadata for all groups to be analyzed. For example, if five groups are analyzed, the list should contain five elements, each representing the item metadata for one group. The order of the elements in the list must match the order of group names specified in the `group.name` argument. Each group's item metadata includes essential information for each item (e.g., number of score categories, IRT model type, etc.) required for calibration. See `est_irt()` or `simdat()` for more details about the item metadata. When `use.startval = TRUE`, the item parameters specified in the metadata will be used as starting values for parameter estimation. If `x = NULL`, both `model` and `cats` arguments must be specified. Note that when `fipc = TRUE` to implement MG-FIPC, the `x` argument must be specified and cannot be NULL. Default is `NULL`.
data	A list containing item response matrices for all groups to be analyzed. For example, if five groups are analyzed, the list should include five elements, each representing the response data matrix for one group. The elements in the list must be ordered to match the group names specified in the `group.name` argument. Each matrix contains examinees' item responses corresponding to the item metadata for that group. In each matrix, rows represent examinees and columns represent items.
group.name	A character vector indicating the names of the groups. For example, if five groups are analyzed, use `group.name = c("G1", "G2", "G3", "G4", "G5")`. Group names can be any valid character strings.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
model	A list containing character vectors specifying the IRT models used to calibrate items across all groups. For example, if five groups are analyzed, the list should contain five elements, each being a character vector of IRT model names for one group. The elements in the list must be ordered to match the group names specified in the `group.name` argument. Available IRT models include: • `"1PLM"`, `"2PLM"`, `"3PLM"`, `"DRM"` for dichotomous items • `"GRM"`, `"GPCM"` for polytomous items

Here, "GRM" denotes the graded response model and "GPCM" the (generalized) partial credit model. Note that "DRM" serves as a general label covering all three dichotomous IRT models.If a single model name is provided in any element of the list, it will be recycled across all items within that group.

This argument is used only when x = NULL and fipc = FALSE. Default is NULL.

cats — A list containing numeric vectors specifying the number of score categories for items in each group. For example, if five groups are analyzed, the list should contain five numeric vectors corresponding to the five groups. The elements in the list must be ordered consistently with the group names specified in the group.name argument.

If a single numeric value is specified in any element of the list, it will be recycled across all items in the corresponding group. If cats = NULL and all models specified in the model argument are dichotomous (i.e., "1PLM", "2PLM", "3PLM", or "DRM"), the function assumes that all items have two score categories across all groups.

This argument is used only when x = NULL and fipc = FALSE. Default is NULL.

item.id — A list containing character vectors of item IDs for each group to be analyzed. For example, if five groups are analyzed, the list should contain five character vectors corresponding to the five groups. The elements in the list must be ordered consistently with the group names specified in the group.name argument.

When fipc = TRUE and item IDs are provided via the item.id argument, the item IDs in the x argument will be overridden. Default is NULL.

free.group — A numeric or character vector indicating the groups for which the scales (i.e., mean and standard deviation) of the latent ability distributions are freely estimated. The scales of the remaining groups (those not specified in this argument) are fixed using the values provided in the group.mean and group.var arguments, or from the weights argument.

For example, suppose that five groups are analyzed with group names "G1", "G2", "G3", "G4", and "G5". To freely estimate the scales for groups 2 through 5, set free.group = c(2, 3, 4, 5) or free.group = c("G2", "G3", "G4", "G5"). In this case, the first group ("G1") will have a fixed scale (e.g., a mean of 0 and variance of 1 when group.mean = 0, group.var = 1, and weights = NULL).

fix.a.1pl — Logical. If TRUE, the slope parameters of all 1PLM items are fixed to a.val.1pl; otherwise, they are constrained to be equal and estimated. Default is FALSE.

fix.a.gpcm — Logical. If TRUE, GPCM items are calibrated as PCM with slopes fixed to a.val.gpcm; otherwise, each item's slope is estimated. Default is FALSE.

fix.g — Logical. If TRUE, all 3PLM guessing parameters are fixed to g.val; otherwise, each guessing parameter is estimated. Default is FALSE.

a.val.1pl — Numeric. Value to which the slope parameters of 1PLM items are fixed when fix.a.1pl = TRUE. Default is 1.

a.val.gpcm — Numeric. Value to which the slope parameters of GPCM items are fixed when fix.a.gpcm = TRUE. Default is 1.

g.val — Numeric. Value to which the guessing parameters of 3PLM items are fixed when fix.g = TRUE. Default is 0.2.

use.aprior Logical. If TRUE, applies a prior distribution to all item discrimination (slope) parameters during calibration. Default is FALSE.

use.bprior Logical. If TRUE, applies a prior distribution to all item difficulty (or threshold) parameters during calibration. Default is FALSE.

use.gprior Logical. If TRUE, applies a prior distribution to all 3PLM guessing parameters during calibration. Default is TRUE.

aprior, bprior, gprior
A list specifying the prior distribution for all item discrimination (slope), difficulty (or threshold), guessing parameters. Three distributions are supported: Beta, Log-normal, and Normal. The list must have two elements:

- dist: A character string, one of "beta", "lnorm", or "norm".
- params: A numeric vector of length two giving the distribution's parameters. For details on each parameterization, see stats::dbeta(), stats::dlnorm(), and stats::dnorm().

Defaults are:

- aprior = list(dist = "lnorm", params = c(0.0, 0.5))
- bprior = list(dist = "norm", params = c(0.0, 1.0))
- gprior = list(dist = "beta", params = c(5, 16))

for discrimination, difficulty, and guessing parameters, respectively.

missing A value indicating missing responses in the data set. Default is NA.

Quadrature A numeric vector of length two:

- first element: number of quadrature points
- second element: symmetric bound (absolute value) for those points For example, c(49, 6) specifies 49 evenly spaced points from –6 to 6. These points are used in the E-step of the EM algorithm. Default is c(49, 6).

weights A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) for the latent ability prior distribution. If not NULL, the latent ability distributions for the groups not specified in the free.group argument are fixed to match the scale defined by the provided quadrature points and weights. The weights and points can be conveniently generated using the function gen.weight().

If NULL, a normal prior density is used instead, based on the information provided in the Quadrature, group.mean, and group.var arguments. Default is NULL.

group.mean A numeric value specifying the mean of the latent variable prior distribution when weights = NULL. Default is 0. For groups not specified in the free.group argument, their distribution means are fixed to this value in order to resolve the indeterminacy of the item parameter scale.

group.var A positive numeric value specifying the variance of the latent variable prior distribution when weights = NULL. Default is 1. For groups not specified in the free.group argument, their distribution variances are fixed to this value in order to resolve the indeterminacy of the item parameter scale.

`EmpHist` Logical. If `TRUE`, the empirical histograms of the latent ability prior distributions across all groups are estimated simultaneously with the item parameters using the approach proposed by Woods (2007). Item calibration is then performed relative to the estimated empirical priors.

`use.startval` Logical. If `TRUE`, the item parameters provided in the item metadata (i.e., the `x` argument) are used as starting values for item parameter estimation. Otherwise, internally generated starting values are used. Default is `FALSE`.

`Etol` A positive numeric value specifying the convergence criterion for the E-step of the EM algorithm. Default is 1e-3. Specifically, the EM algorithm terminates when the largest absolute difference in item parameter estimates between consecutive iterations is smaller than this value.

`MaxE` A positive integer specifying the maximum number of iterations for the E-step in the EM algorithm. Default is `500`.

`control` A named list of options passed directly to `stats::nlminb()` in each M-step optimization of the EM algorithm. By default: `control = list(eval.max = 500, iter.max = 200, x.tol = 1e-4)`, where

- `eval.max` = 500 limits the number of function evaluations
- `iter.max` = 200 caps the number of internal optimizer iterations
- `x.tol` = 1e-4 sets the absolute change threshold in parameter values below which `stats::nlminb()` considers the solution to have converged Users may additionally supply other `nlminb()` control options (such as `abs.tol`, `rel.tol`, `trace`, etc.) as needed.

`fipc` Logical. If `TRUE`, multiple-group fixed item parameter calibration (MG-FIPC) is applied during item parameter estimation. When `fipc = TRUE`, the information on which items are fixed must be provided via either `fix.loc` or `fix.id`. See below for details.

`fipc.method` A character string specifying the FIPC method. Available options are:

- `"OEM"`: No Prior Weights Updating and One EM Cycle (NWU-OEM; Wainer & Mislevy, 1990)
- `"MEM"`: Multiple Prior Weights Updating and Multiple EM Cycles (MWU-MEM; Kim, 2006) When `fipc.method = "OEM"`, the maximum number of E-steps is automatically set to 1, regardless of the value specified in `MaxE`.

`fix.loc` A list of positive integer vectors. Each internal vector specifies the positions of the items to be fixed in the item metadata (i.e., `x`) for each group when MG-FIPC is implemented (i.e., `fipc = TRUE`). The internal objects in the list must follow the same order as the group names provided in the `group.name` argument.

For example, suppose three groups are analyzed. In the first group, the 1st, 3rd, and 5th items are fixed; in the second group, the 2nd, 3rd, 4th, and 7th items are fixed; and in the third group, the 1st, 2nd, and 6th items are fixed. Then `fix.loc = list(c(1, 3, 5), c(2, 3, 4, 7), c(1, 2, 6))`. Note that if the `fix.id` argument is not NULL, the information in `fix.loc` will be ignored. See below for details.

`fix.id` A vector of character strings specifying the IDs of items to be fixed when MG-FIPC is implemented (i.e., `fipc = TRUE`).

For example, suppose that three groups are analyzed. In the first group, three items with IDs G1I1, C1I1, and C1I2 are fixed. In the second group, four items with IDs C1I1, C1I2, C2I1, and C2I2 are fixed. In the third group, three items with IDs C2I1, C2I2, and G3I1 are fixed.

In this case, there are six unique items fixed across the groups—namely, G1I1, C1I1, C1I2, C2I1, C2I2, and G3I1, because C1I1 and C1I2 appear in both the first and second groups, while C2I1 and C2I2 appear in both the second and third groups. Thus, you should specify `fix.id = c("G1I1", "C1I1", "C1I2", "C2I1", "C2I2", "G3I1")`. Note that if the `fix.id` argument is not NULL, the information provided in `fix.loc` is ignored. See below for details.

`se` Logical. If `FALSE`, standard errors of the item parameter estimates are not computed. Default is `TRUE`.

`verbose` Logical. If `FALSE`, all progress messages, including information about the EM algorithm process, are suppressed. Default is `TRUE`.

Details

Multiple-group (MG) item calibration (Bock & Zimowski, 1996) provides a unified framework for handling testing scenarios involving multiple groups, such as nonequivalent groups equating, vertical scaling, and the identification of differential item functioning (DIF). In such applications, examinees from different groups typically respond to either the same test form or to different forms that share common (anchor) items.

The goal of MG item calibration is to estimate both item parameters and latent ability distributions for all groups simultaneously (Bock & Zimowski, 1996). The **irtQ** package implements MG calibration via the `est_mg()` function, which uses marginal maximum likelihood estimation through the expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981). In addition, the function supports multiple-group fixed item parameter calibration (MG-FIPC; e.g., Kim & Kolen, 2016), which allows the parameters of specific items to be fixed across groups.

In MG IRT analyses, it is common for multiple groups' test forms to share some common (anchor) items. By default, the `est_mg()` function automatically constrains items with identical item IDs across groups to share the same parameter estimates.

Most of the features of the `est_mg()` function are similar to those of the `est_irt()` function. The main difference is that several arguments in `est_mg()` accept list objects containing elements for each group to be analyzed. These arguments include `x`, `data`, `model`, `cats`, `item.id`, `fix.loc` and `fix.id`.

Additionally, `est_mg()` introduces two new arguments: `group.name` and `free.group`. The `group.name` argument is required to assign a unique identifier to each group. The order of the list elements provided in `x`, `data`, `model`, `cats`, `item.id`, `fix.loc` and `fix.id` must match the order of group names specified in the `group.name` argument.

The `free.group` argument is required to indicate which groups have their latent ability distribution scales (i.e., mean and standard deviation) freely estimated. When no item parameters are fixed (i.e., `fipc = FALSE`), at least one group must have a fixed latent ability scale (e.g., mean = 0 and variance = 1) among the multiple groups sharing common items, in order to resolve the scale indeterminacy inherent in IRT estimation. By specifying the groups in the `free.group` argument, the scales for those groups will be freely estimated, while the scales for all other groups not included in `free.group` will be fixed using the values provided in the `group.mean` and `group.var` arguments or from the `weights` argument.

Situations requiring the implementation of MG-FIPC typically arise when new latent ability scales from multiple-group (MG) test data need to be linked to an established scale (e.g., that of an existing item bank). In a single run of the MG-FIPC procedure, the parameters of non-fixed (freed) items across multiple test forms, as well as the latent ability distributions for multiple groups, can be estimated on the same scale as the fixed items (Kim & Kolen, 2016).

For example, suppose that three different test forms—Form 1, Form 2, and Form 3—are administered to three nonequivalent groups: Group1, Group2, and Group3. Form 1 and Form 2 share 12 common items (C1I1 to C1I12), while Form 2 and Form 3 share 10 common items (C2I1 to C2I10). There are no common items between Form 1 and Form 3. Also, assume that all unique items in Form 1 are from an existing item bank and have already been calibrated on the item bank's scale.

In this case, the goal of MG-FIPC is to estimate the parameters of all items across the three test forms—except the unique items in Form 1— and the latent ability distributions of the three groups, all on the same scale as the item bank. To achieve this, the unique items in Form 1 must be fixed during MG-FIPC to link the current MG test data to the item bank scale.

The `est_mg()` function can implement MG-FIPC by setting `fipc = TRUE`. In this case, the information on which items to fix must be provided through either the `fix.loc` or `fix.id` argument. When using `fix.loc`, you must supply a list of item positions (locations) to be fixed in each group's test form. For example, suppose that the test data from the three groups above are analyzed. In the first group, the 1st, 3rd, and 5th items are fixed; in the second group, the 2nd, 3rd, 4th, and 7th items are fixed; and in the third group, the 1st, 2nd, and 6th items are fixed. In this case, you would specify: `fix.loc = list(c(1, 3, 5), c(2, 3, 4, 7), c(1, 2, 6))`.

Alternatively, you can use `fix.id` to specify a character vector of item IDs to be fixed across groups. In the first group, the items with IDs G1I1, C1I1, and C1I2 are fixed; in the second group, the items with IDs C1I1, C1I2, C2I1, and C2I2 are fixed; and in the third group, the items with IDs C2I1, C2I2, and G3I1 are fixed. In this case, there are six unique items to be fixed across all groups: G1I1, C1I1, C1I2, C2I1, C2I2, and G3I1. You would then specify: `fix.id = c("G1I1", "C1I1", "C1I2", "C2I1", "C2I2", "G3I1")`.

Note that when both `fix.loc` and `fix.id` are provided, the information in `fix.id` takes precedence and overrides `fix.loc`.

Value

This function returns an object of class `est_irt`. The returned object contains the following components:

`estimates` A list containing two internal elements: `overall` and `group`. The `overall` element is a data frame with item parameter estimates and their standard errors, computed from the combined data across all groups. This data frame includes only the unique items across all groups. The `group` element is a list of group-specific data frames, each containing item parameter estimates and standard errors for that particular group.

`par.est` A list with the same structure as `estimates`, containing only the item parameter estimates (excluding standard errors), formatted according to the item metadata structure.

`se.est` A list with the same structure as `estimates`, but containing only the standard errors of the item parameter estimates. Note that the standard errors are calculated using the cross-product approximation method (Meilijson, 1989).

pos.par	A data frame indicating the position index of each estimated item parameter. This index is based on the combined data set across all groups (i.e., the first internal object of estimates). The position information is useful for interpreting the variance-covariance matrix of item parameter estimates.
covariance	A variance-covariance matrix of the item parameter estimates, based on the combined data set across all groups (i.e., the first internal object of estimates).
loglikelihood	A list containing two internal objects (i.e., overall and group) of marginal log-likelihood values based on the observed data. The structure of the list matches that of estimates. Specifically, the overall component contains the total log-likelihood summed across all unique items from all groups, while the group component provides group-specific log-likelihood values.
aic	A model fit statistic based on the Akaike Information Criterion (AIC), calculated from the log-likelihood of all unique items.
bic	A model fit statistic based on the Bayesian Information Criterion (BIC), calculated from the log-likelihood of all unique items.
group.par	A list containing summary statistics (i.e., mean, variance, and standard deviation) of the latent variable prior distributions across all groups.
weights	A list of two-column data frames, where the first column contains quadrature points and the second column contains the corresponding weights of the (updated) latent variable prior distributions for each group.
posterior.dist	A matrix of normalized posterior densities for all response patterns at each quadrature point. Rows and columns represent response patterns and quadrature points, respectively.
data	A list containing two internal objects (i.e., overall and group) representing the examinees' response data sets. The structure of this list matches that of the estimates component.
scale.D	The scaling factor used in the IRT model.
ncase	A list containing two internal objects (i.e., overall and group) representing the total number of response patterns. The structure of this list matches that of the estimates component.
nitem	A list containing two internal objects (i.e., overall and group) representing the total number of items included in the response data. The structure of this list matches that of the estimates component.
Etol	The convergence criterion for the E-step of the EM algorithm.
MaxE	The maximum number of E-steps allowed in the EM algorithm.
aprior	A list describing the prior distribution used for discrimination parameters.
bprior	A list describing the prior distribution used for difficulty parameters.
gprior	A list describing the prior distribution used for guessing parameters.
npar.est	The total number of parameters estimated across all unique items.
niter	The number of completed EM cycles.
maxpar.diff	The maximum absolute change in parameter estimates at convergence.
EMtime	Time (in seconds) spent on EM cycles.

SEtime	Time (in seconds) spent computing standard errors.
TotalTime	Total computation time (in seconds).
test.1	First-order test result indicating whether the gradient sufficiently vanished for solution stability.
test.2	Second-order test result indicating whether the information matrix is positive definite, a necessary condition for identifying a local maximum.
var.note	A note indicating whether the variance-covariance matrix was successfully obtained from the information matrix.
fipc	Logical. Indicates whether FIPC was used.
fipc.method	The method used for FIPC.
fix.loc	A list containing two internal objects (i.e., overall and group) indicating the locations of fixed items when FIPC is applied. The structure of the list matches that of the 'estimates' component.

Note that you can easily extract components from the output using the `getirt()` function.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Bock, R. D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika, 46*, 443-459.

Bock, R. D., & Zimowski, M. F. (1997). Multiple group IRT. In W. J. van der Linden & R. K. Hambleton (Eds.), *Handbook of modern item response theory* (pp. 433-448). New York: Springer.

Kim, S. (2006). A comparative study of IRT fixed parameter calibration methods. *Journal of Educational Measurement, 43*(4), 355-381.

Kim, S., & Kolen, M. J. (2016). Multiple group IRT fixed-parameter estimation for maintaining an established ability scale. *Center for Advanced Studies in Measurement and Assessment Report, 49.*

Meilijson, I. (1989). A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society: Series B (Methodological), 51*, 127-138.

Woods, C. M. (2007). Empirical histograms in item response theory with ordinal data. *Educational and Psychological Measurement, 67*(1), 73-87.

See Also

`est_irt()`, `shape_df()`, `shape_df_fipc()`, `getirt()`

Examples

```
## ------------------------------------------------------------------------------
# 1. MG calibration using the simMG data
#  - Details:
#    (a) Constrain common items between groups to have
#        identical item parameters (i.e., items C1I1-C1I12 between
```

```
#        Groups 1 and 2, and items C2I1–C2I10 between Groups 2 and 3).
#    (b) Freely estimate the means and variances of the ability
#        distributions for all groups except the reference group,
#        where the mean and variance are fixed to 0 and 1, respectively.
## ------------------------------------------------------------------------------
# 1-(1). Freely estimate the means and variances of Groups 2 and 3
# Import the true item metadata for the three groups
x <- simMG$item.prm

# Extract model, score category, and item ID information
# from the item metadata for the three groups
model <- list(x$Group1$model, x$Group2$model, x$Group3$model)
cats <- list(x$Group1$cats, x$Group2$cats, x$Group3$cats)
item.id <- list(x$Group1$id, x$Group2$id, x$Group3$id)

# Import the simulated response data sets for the three groups
data <- simMG$res.dat

# Import the group names for the three groups
group.name <- simMG$group.name

# Specify Groups 2 and 3 as the free groups where the scale
# of the ability distributions will be freely estimated.
# Group 1 will serve as the reference group, where the scale
# of the ability distribution is fixed to the values specified
# via the 'group.mean' and 'group.var' arguments
free.group <- c(2, 3) # or use 'free.group <- group.name[2:3]'

# Estimate IRT parameters:
# As long as common items across groups share the same item IDs,
# their item parameters will be constrained to be equal across groups
# unless FIPC is implemented.
fit.1 <-
  est_mg(
    data = data, group.name = group.name, model = model,
    cats = cats, item.id = item.id, D = 1, free.group = free.group,
    use.gprior = TRUE, gprior = list(dist = "beta", params = c(5, 16)),
    group.mean = 0, group.var = 1, EmpHist = TRUE, Etol = 0.001, MaxE = 500
  )

# Summary of the estimation
summary(fit.1)

# Extract the item parameter estimates
getirt(fit.1, what = "par.est")

# Extract the standard error estimates
getirt(fit.1, what = "se.est")

# Extract the group-level parameter estimates (i.e., scale parameters)
getirt(fit.1, what = "group.par")

# Extract the posterior latent ability distributions for each group
```

```
getirt(fit.1, what = "weights")

# 1-(2). Alternatively, the same parameter estimation can be performed by
# inserting a list of item metadata for the groups into the 'x' argument.
# If the item metadata contains item parameters to be used as starting values,
# set 'use.startval = TRUE'.
# Also, specify the groups in which the ability distribution scales
# will be freely estimated using their group names.
free.group <- group.name[2:3]
fit.2 <-
  est_mg(
    x = x, data = data, group.name = group.name, D = 1,
    free.group = free.group, use.gprior = TRUE,
    gprior = list(dist = "beta", params = c(5, 16)),
    group.mean = 0, group.var = 1, EmpHist = TRUE, use.startval = TRUE,
    Etol = 0.001, MaxE = 500
  )

# Summary of the estimation
summary(fit.2)

## ------------------------------------------------------------------------------
# 2. MG calibration with FIPC using simMG data
#  - Details:
#    (a) Fix the parameters of the common items between the groups
#        (i.e., items C1I1-C1I12 between Groups 1 and 2, and
#        items C2I1-C2I10 between Groups 2 and 3)
#    (b) Freely estimate the means and variances of the ability
#        distributions for all three groups
## ------------------------------------------------------------------------------
# 2-(1). Freely estimate the means and variances for all three groups
# Set all three groups as free groups in which the scales
# of the ability distributions are to be freely estimated
free.group <- 1:3 # or use 'free.group <- group.name'

# Specify the locations of items to be fixed in each group's metadata
# For Group 1: C1I1-C1I12 are located in rows 1-10 and 49-50
# For Group 2: C1I1-C1I12 are in rows 1-12, and
#              C2I1-C2I10 are in rows 41-50
# For Group 3: C2I1-C2I10 are in rows 1-10
fix.loc <- list(
  c(1:10, 49:50),
  c(1:12, 41:50),
  c(1:10)
)

# Estimate IRT parameters using MG-FIPC:
# When FIPC is implemented, item metadata for all groups
# must be provided via the 'x' argument.
# For fixed items, their item parameters must be specified
# in the metadata. For non-fixed items, any placeholder values
# can be used in the metadata.
# Also set fipc = TRUE and fipc.method = "MEM"
```

```
fit.3 <-
  est_mg(
    x = x, data = data, group.name = group.name, D = 1,
    free.group = free.group, use.gprior = TRUE,
    gprior = list(dist = "beta", params = c(5, 16)),
    EmpHist = TRUE, Etol = 0.001, MaxE = 500, fipc = TRUE,
    fipc.method = "MEM", fix.loc = fix.loc
  )

# Summary of the estimation
summary(fit.3)

# Extract the item parameter estimates
getirt(fit.3, what = "par.est")

# Extract the standard error estimates
getirt(fit.3, what = "se.est")

# Extract the group parameter estimates (i.e., scale parameters)
getirt(fit.3, what = "group.par")

# Extract the posterior latent ability distributions of the groups
getirt(fit.3, what = "weights")

# 2-(2). Alternatively, MG-FIPC can be implemented by specifying the
# IDs of the items to be fixed using the 'fix.id' argument.
# Provide a character vector of fixed item IDs to 'fix.id'
fix.id <- c(paste0("C1I", 1:12), paste0("C2I", 1:10))
fit.4 <-
  est_mg(
    x = x, data = data, group.name = group.name, D = 1,
    free.group = free.group, use.gprior = TRUE,
    gprior = list(dist = "beta", params = c(5, 16)),
    EmpHist = TRUE, Etol = 0.001, MaxE = 500, fipc = TRUE,
    fipc.method = "MEM", fix.id = fix.id
  )

# Summary of the estimation
summary(fit.4)

## ----------------------------------------------------------------------------
# 3. MG calibration with FIPC using simMG data
#    (Estimate group parameters only)
#  - Details:
#    (a) Fix all item parameters across all three groups
#    (b) Freely estimate the means and variances of the ability
#        distributions for all three groups
## ----------------------------------------------------------------------------
# 3-(1). Freely estimate the means and variances for all three groups
# Set all three groups as free groups in which the scales
# of the ability distributions will be freely estimated
free.group <- 1:3 # or use 'free.group <- group.name'
```

```
# Specify the locations of all fixed items in each group's metadata
fix.loc <- list(1:50, 1:50, 1:38)

# Estimate group parameters only using MG-FIPC
fit.5 <-
  est_mg(
    x = x, data = data, group.name = group.name, D = 1,
    free.group = free.group, use.gprior = TRUE,
    gprior = list(dist = "beta", params = c(5, 16)),
    EmpHist = TRUE, Etol = 0.001, MaxE = 500, fipc = TRUE,
    fipc.method = "MEM", fix.loc = fix.loc
  )

# Summary of the estimation
summary(fit.5)

# Extract the group parameter estimates (i.e., scale parameters)
getirt(fit.5, what = "group.par")

## -----------------------------------------------------------------------------
# 4. MG calibration with FIPC using simMG data
#    (Fix only the unique items of Group 1)
#  - Details:
#    (a) Fix item parameters of the unique items in Group 1 only
#    (b) Constrain the common items across groups to have
#        the same item parameters (i.e., C1I1-C1I12 between
#        Groups 1 and 2, and C2I1-C2I10 between Groups 2 and 3)
#    (c) Freely estimate the means and variances of the ability
#        distributions for all three groups
## -----------------------------------------------------------------------------
# 4-(1). Freely estimate the means and variances for all three groups
# Set all three groups as free groups in which the scales
# of the ability distributions will be freely estimated
free.group <- group.name # or use 'free.group <- 1:3'

# Specify the item IDs of the unique items in Group 1 to be fixed using
# the `fix.id` argument.
fix.id <- paste0("G1I", 1:38)

# Alternatively, use the 'fix.loc' argument as
# 'fix.loc = list(11:48, NULL, NULL)'

# Estimate IRT parameters using MG-FIPC
fit.6 <-
  est_mg(
    x = x, data = data, group.name = group.name, D = 1,
    free.group = free.group, use.gprior = TRUE,
    gprior = list(dist = "beta", params = c(5, 16)),
    EmpHist = TRUE, Etol = 0.001, MaxE = 500, fipc = TRUE,
    fipc.method = "MEM", fix.loc = NULL, fix.id = fix.id
  )

# Summary of the estimation
```

```
summary(fit.6)

# Extract the group parameter estimates (i.e., scale parameters)
getirt(fit.6, what = "group.par")
```

`est_score` *Estimate examinees' ability (proficiency) parameters*

Description

This function estimates examinees' latent ability parameters. Available scoring methods include maximum likelihood estimation (ML), maximum likelihood estimation with fences (MLF; Han, 2016), weighted likelihood estimation (WL; Warm, 1989), maximum a posteriori estimation (MAP; Hambleton et al., 1991), expected a posteriori estimation (EAP; Bock & Mislevy, 1982), EAP summed scoring (Thissen et al., 1995; Thissen & Orlando, 2001), and inverse test characteristic curve (TCC) scoring (e.g., Kolen & Brennan, 2004; Kolen & Tong, 2010; Stocking, 1996).

Usage

```
est_score(x, ...)

## Default S3 method:
est_score(
  x,
  data,
  D = 1,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  fence.a = 3,
  fence.b = NULL,
  tol = 1e-04,
  max.iter = 100,
  se = TRUE,
  stval.opt = 1,
  intpol = TRUE,
  range.tcc = c(-7, 7),
  missing = NA,
  ncore = 1,
  ...
)
```

```
## S3 method for class 'est_irt'
est_score(
  x,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  fence.a = 3,
  fence.b = NULL,
  tol = 1e-04,
  max.iter = 100,
  se = TRUE,
  stval.opt = 1,
  intpol = TRUE,
  range.tcc = c(-7, 7),
  missing = NA,
  ncore = 1,
  ...
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class `est_irt` obtained from `est_irt()`, or `est_item` from `est_item()`. See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
...	Additional arguments passed to `parallel::makeCluster()`.
data	A matrix of examinees' item responses corresponding to the items specified in the `x` argument. Rows represent examinees and columns represent items.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
method	A character string indicating the scoring method to use. Available options are: • `"ML"`: Maximum likelihood estimation • `"MLF"`: Maximum likelihood estimation with fences (Han, 2016) • `"WL"`: Weighted likelihood estimation (Warm, 1989) • `"MAP"`: Maximum a posteriori estimation (Hambleton et al., 1991) • `"EAP"`: Expected a posteriori estimation (Bock & Mislevy, 1982) • `"EAP.SUM"`: Expected a posteriori summed scoring (Thissen et al., 1995; Thissen & Orlando, 2001) • `"INV.TCC"`: Inverse test characteristic curve scoring (e.g., Kolen & Brennan, 2004; Kolen & Tong, 2010; Stocking, 1996) Default is `"ML"`.

`range` A numeric vector of length two specifying the lower and upper bounds of the ability scale. This is used for the following scoring methods: `"ML"`, `"MLF"`, `"WL"`, and `"MAP"`. Default is `c(-5, 5)`.

`norm.prior` A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. These values are used to generate the Gaussian quadrature points and weights. Ignored if `method` is `"ML"`, `"MLF"`, `"WL"`, or `"INV.TCC"`. Default is `c(0, 1)`.

`nquad` An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. Used only when `method` is `"EAP"` or `"EAP.SUM"`. Ignored for `"ML"`, `"MLF"`, `"WL"`, `"MAP"`, and `"INV.TCC"`. Default is `41`.

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and their corresponding weights (in the second column) for the latent variable prior distribution. The weights and points can be conveniently generated using the function `gen.weight()`.

If `NULL` and `method` is either `"EAP"` or `"EAP.SUM"`, default quadrature values are generated based on the `norm.prior` and `nquad` arguments. Ignored if `method` is `"ML"`, `"MLF"`, `"WL"`, `"MAP"`, or `"INV.TCC"`.

`fence.a` A numeric value specifying the item slope parameter (i.e., *a*-parameter) for the two imaginary items used in MLF. See **Details** below. Default is 3.0.

`fence.b` A numeric vector of length two specifying the lower and upper bounds of the item difficulty parameters (i.e., *b*-parameters) for the two imaginary items in MLF. If `fence.b = NULL`, the values specified in the `range` argument are used instead. Default is NULL.

`tol` A numeric value specifying the convergence tolerance for the ML, MLF, WL, MAP, and inverse TCC scoring methods. Newton-Raphson optimization is used for ML, MLF, WL, and MAP, while the bisection method is used for inverse TCC. Default is 1e-4.

`max.iter` A positive integer specifying the maximum number of iterations allowed for the Newton-Raphson optimization. Default is 100.

`se` Logical. If `TRUE`, standard errors of ability estimates are computed. If `method` is "EAP.SUM" or "INV.TCC", standard errors are always returned regardless of this setting. Default is `TRUE`.

`stval.opt` A positive integer specifying the starting value option for the ML, MLF, WL, and MAP scoring methods. Available options are:

- 1: Brute-force search (default)
- 2: Based on observed sum scores
- 3: Fixed at 0

See **Details** below for more information.

`intpol` Logical. If `TRUE` and `method = "INV.TCC"`, linear interpolation is applied to approximate ability estimates for sum scores that cannot be directly mapped using the TCC (e.g., when the observed sum score is less than the total of item guessing parameters). Default is `TRUE`. See **Details** below.

`range.tcc` A numeric vector of length two specifying the lower and upper bounds of ability estimates when `method = "INV.TCC"`. Default is `c(-7, 7)`.

`missing` A value indicating missing responses in the data set. Default is `NA`. See **Details** below.

`ncore` An integer specifying the number of logical CPU cores to use for parallel processing. Default is 1. See **Details** below.

Details

For the MAP scoring method, only a normal prior distribution is supported for the population distribution.

When there are missing responses in the data set, the missing value must be explicitly specified using the `missing` argument. Missing data are properly handled when using the ML, MLF, WL, MAP, or EAP methods. However, when using the "EAP.SUM" or "INV.TCC" methods, any missing responses are automatically treated as incorrect (i.e., recoded as 0s).

In the maximum likelihood estimation with fences (MLF; Han, 2016), two imaginary items based on the 2PL model are introduced. The first imaginary item functions as the lower fence, and its difficulty parameter (b) should be smaller than any of the difficulty parameters in the test form. Similarly, the second imaginary item serves as the upper fence, and its b parameter should be greater than any difficulty value in the test form. Both imaginary items should also have very steep slopes (i.e., high a-parameter values). See Han (2016) for more details. If `fence.b = NULL`, the function will automatically assign the lower and upper fences based on the values provided in the `range` argument.

When the "INV.TCC" method is used with the 3PL model, ability estimates cannot be obtained for observed sum scores that are less than the sum of the items' guessing parameters. In such cases, linear interpolation can be applied by setting `intpol = TRUE`.

Let θ_{min} and θ_{max} denote the minimum and maximum ability estimates, respectively, and let θ_X be the ability estimate corresponding to the smallest observed sum score, X, that is greater than or equal to the sum of the guessing parameters.When linear interpolation is applied, the first value in the `range.tcc` argument is treated as θ_{min}. A line is then constructed between the points $(x = \theta_{min}, y = 0)$ and $(x = \theta_X, y = X)$. The second value in `range.tcc` is interpreted as θ_{max}, which corresponds to the ability estimate for the maximum observed sum score.

For the "INV.TCC" method, standard errors of ability estimates are computed using the approach proposed by Lim et al. (2021). The implementation of inverse TCC scoring in this function is based on a modified version of the `SNSequate::irt.eq.tse()` function from the **SNSequate** package (González, 2014).

For the ML, MLF, WL, and MAP scoring methods, different strategies can be used to determine the starting value for ability estimation based on the `stval.opt` argument:

- When `stval.opt = 1` (default), a brute-force search is performed by evaluating the log-likelihood at discrete theta values within the range specified by `range`, using 0.1 increments. The theta value yielding the highest log-likelihood is chosen as the starting value.
- When `stval.opt = 2`, the starting value is derived from the observed sum score using a logistic transformation. For example, if the maximum possible score (`max.score`) is 30 and the examinee's observed sum score (`obs.score`) is 20, the starting value is `log(obs.score / (max.score - obs.score))`.
  - If all responses are incorrect (i.e., `obs.score = 0`), the starting value is `log(1 / max.score)`.

  - If all responses are correct (`obs.score` = `max.score`), the starting value is `log(max.score / 1)`.
- When `stval.opt` = 3, the starting value is fixed at 0.

To accelerate ability estimation using the ML, MLF, WL, MAP, and EAP methods, this function supports parallel processing across multiple logical CPU cores. The number of cores can be specified via the `ncore` argument (default is 1).

Note that the standard errors of ability estimates are computed based on the Fisher expected information for the ML, MLF, WL, and MAP methods.

For the implementation of the WL method, the function references the `catR::Pi()`, `catR::Ji()`, and `catR::Ii()` functions from the **catR** package (Magis & Barrada, 2017).

Value

When `method` is one of `"ML"`, `"MLF"`, `"WL"`, `"MAP"`, or `"EAP"`, a two-column data frame is returned:

- Column 1: Ability estimates
- Column 2: Standard errors of the ability estimates

When `method` is either `"EAP.SUM"` or `"INV.TCC"`, a list with two components is returned:

- Object 1: A three-column data frame including:
  - Column 1: Observed sum scores
  - Column 2: Ability estimates
  - Column 3: Standard errors of the ability estimates
- Object 2: A score table showing possible raw sum scores and the corresponding ability and standard error estimates

Methods (by class)

- `est_score(default)`: Default method to estimate examinees' latent ability parameters using a data frame `x` containing the item metadata.
- `est_score(est_irt)`: An object created by the function `est_irt()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Bock, R. D., & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Psychometrika, 35*, 179-198.

González, J. (2014). SNSequate: Standard and nonstandard statistical models and methods for test equating. *Journal of Statistical Software, 59*, 1-30.

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991).*Fundamentals of item response theory*. Newbury Park, CA: Sage.

Han, K. T. (2016). Maximum likelihood score estimation method with fences for short-length tests and computerized adaptive tests. *Applied psychological measurement, 40*(4), 289-301.

Howard, J. P. (2017). *Computational methods for numerical analysis with R*. New York: Chapman and Hall/CRC.

Kolen, M. J. & Brennan, R. L. (2004). *Test Equating, Scaling, and Linking* (2nd ed.). New York: Springer

Kolen, M. J. & Tong, Y. (2010). Psychometric properties of IRT proficiency estimates. *Educational Measurement: Issues and Practice, 29*(3), 8-14.

Lim, H., Davey, T., & Wells, C. S. (2021). A recursion-based analytical approach to evaluate the performance of MST. *Journal of Educational Measurement, 58*(2), 154-178.

Magis, D., & Barrada, J. R. (2017). Computerized adaptive testing with R: Recent updates of the package catR. *Journal of Statistical Software, 76*, 1-19.

Stocking, M. L. (1996). An alternative method for scoring adaptive tests. *Journal of Educational and Behavioral Statistics, 21*(4), 365-389.

Thissen, D. & Orlando, M. (2001). Item response theory for items scored in two categories. In D. Thissen & H. Wainer (Eds.), *Test scoring* (pp.73-140). Mahwah, NJ: Lawrence Erlbaum.

Thissen, D., Pommerich, M., Billeaud, K., & Williams, V. S. (1995). Item Response Theory for Scores on Tests Including Polytomous Items with Ordered Responses. *Applied Psychological Measurement, 19*(1), 39-49.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427-450.

See Also

`est_irt()`, `simdat()`, `shape_df()`, `gen.weight()`

Examples

```
## Import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read item parameters and convert them into item metadata
x <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# Generate examinee ability values
set.seed(12)
theta <- rnorm(10)

# Simulate item response data based on the item metadata and abilities
data <- simdat(x, theta, D = 1)


# Estimate abilities using maximum likelihood (ML)
est_score(x, data, D = 1, method = "ML", range = c(-4, 4), se = TRUE)

# Estimate abilities using weighted likelihood (WL)
est_score(x, data, D = 1, method = "WL", range = c(-4, 4), se = TRUE)

# Estimate abilities using MLF with default fences
# based on the `range` argument
```

```
est_score(x, data,
  D = 1, method = "MLF",
  fence.a = 3.0, fence.b = NULL, se = TRUE
)

# Estimate abilities using MLF with user-specified fences
est_score(x, data,
  D = 1, method = "MLF", fence.a = 3.0,
  fence.b = c(-7, 7), se = TRUE
)

# Estimate abilities using maximum a posteriori (MAP)
est_score(x, data,
  D = 1, method = "MAP", norm.prior = c(0, 1),
  nquad = 30, se = TRUE
)

# Estimate abilities using expected a posteriori (EAP)
est_score(x, data,
  D = 1, method = "EAP", norm.prior = c(0, 1),
  nquad = 30, se = TRUE
)

# Estimate abilities using EAP summed scoring
est_score(x, data,
  D = 1, method = "EAP.SUM", norm.prior = c(0, 1),
  nquad = 30
)

# Estimate abilities using inverse TCC scoring
est_score(x, data,
  D = 1, method = "INV.TCC", intpol = TRUE,
  range.tcc = c(-7, 7)
)
```

`gen.weight` *Generate Weights*

Description

This function generates a set of normalized weights based on theta (ability) values to be used in functions such as `est_score()`, `sx2_fit()`, and `covirt()`.

Usage

```
gen.weight(n = 41, dist = "norm", mu = 0, sigma = 1, l = -4, u = 4, theta)
```

Arguments

n	An integer specifying the number of theta (node) values for which weights are to be generated. Default is 41.
dist	A character string indicating the distribution type used to generate weights. Available options are "norm" for a normal distribution, "unif" for a uniform distribution, and "emp" for an empirical distribution. • If dist = "norm", either n or theta must be provided. • If dist = "unif", only n is applicable. • If dist = "emp", only theta must be specified.
mu, sigma	Mean and standard deviation of the normal distribution (used when dist = "norm").
l, u	Lower and upper bounds of the uniform distribution (used when dist = "unif").
theta	A numeric vector of empirical theta (node) values for which weights are generated.

Details

If theta is not specified, *n* equally spaced quadrature points and corresponding weights are generated from either the normal or uniform distribution:

- When dist = "norm", Gaussian quadrature points and weights are computed using gauss.quad.prob() from the **statmod** package.
- When dist = "unif", equally spaced points are drawn from the specified interval [l, u], and weights are proportional to the uniform density.

If theta is specified:

- When dist = "norm", the weights are proportional to the normal density evaluated at each theta value and normalized to sum to 1.
- When dist = "emp", equal weights are assigned to each provided theta value.

Value

A data frame with two columns:

- theta: The theta (node) values.
- weight: The corresponding normalized weights.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

est_score(), sx2_fit(), covirt()

Examples

```
## Example 1:
## Generate 41 Gaussian quadrature points and weights from the normal distribution
gen.weight(n = 41, dist = "norm", mu = 0, sigma = 1)

## Example 2:
## Generate 41 theta values and weights from the uniform distribution,
## given a minimum value of -4 and a maximum value of 4
gen.weight(n = 41, dist = "unif", l = -4, u = 4)

## Example 3:
## Generate normalized weights from the standard normal distribution,
## given a user-defined set of theta values
theta <- seq(-4, 4, by = 0.1)
gen.weight(dist = "norm", mu = 0, sigma = 1, theta = theta)

## Example 4:
## Generate equal normalized weights for theta values
## randomly sampled from the standard normal distribution
theta <- rnorm(100)
gen.weight(dist = "emp", theta = theta)
```

---

getirt *Extract Components from 'est_irt', 'est_mg', or 'est_item' Objects*

Description

Extracts internal components from an object of class `est_irt` (from `est_irt()`), `est_mg` (from `est_mg()`), or `est_item` (from `est_item()`).

Usage

```
getirt(x, ...)

## S3 method for class 'est_irt'
getirt(x, what, ...)

## S3 method for class 'est_mg'
getirt(x, what, ...)

## S3 method for class 'est_item'
getirt(x, what, ...)
```

Arguments

`x`	An object of class `est_irt`, `est_mg`, or `est_item` as returned by `est_irt()`, `est_mg()`, or `est_item()`, respectively.

...	Additional arguments passed to or from other methods.
what	A character string specifying the name of the internal component to extract.

Details

The following components can be extracted from an object of class `est_irt` created by `est_irt()`:

estimates A data frame containing both the item parameter estimates and their corresponding standard errors.

par.est A data frame containing only the item parameter estimates.

se.est A data frame containing the standard errors of the item parameter estimates, calculated using the cross-product approximation method (Meilijson, 1989).

pos.par A data frame indicating the position index of each estimated item parameter. This is useful when interpreting the variance-covariance matrix.

covariance A variance-covariance matrix of the item parameter estimates.

loglikelihood The total marginal log-likelihood value summed across all items.

aic Akaike Information Criterion (AIC) based on the marginal log-likelihood.

bic Bayesian Information Criterion (BIC) based on the marginal log-likelihood.

group.par A data frame containing the mean, variance, and standard deviation of the latent variable's prior distribution.

weights A two-column data frame containing quadrature points (first column) and corresponding weights (second column) of the (updated) latent trait prior.

posterior.dist A matrix of normalized posterior densities for all response patterns at each quadrature point. Rows represent examinees, and columns represent quadrature points.

data A data frame of the examinee response dataset used in estimation.

scale.D The scaling constant (usually 1 or 1.7) used in the IRT model.

ncase The number of unique response patterns.

nitem The number of items included in the dataset.

Etol The convergence criterion used for the E-step in the EM algorithm.

MaxE The maximum number of E-steps allowed during EM estimation.

aprior A list describing the prior distribution for item slope parameters.

bprior A list describing the prior distribution for item difficulty (or threshold) parameters.

gprior A list describing the prior distribution for item guessing parameters.

npar.est The total number of parameters estimated.

niter The number of EM cycles completed.

maxpar.diff The maximum change in parameter estimates at convergence.

EMtime Computation time (in seconds) for the EM algorithm.

SEtime Computation time (in seconds) for estimating standard errors.

TotalTime Total computation time (in seconds) for model estimation.

test.1 Result of the first-order test indicating whether the gradients were sufficiently close to zero.

test.2 Result of the second-order test indicating whether the information matrix was positive definite (a condition for maximum likelihood).

var.note A note indicating whether the variance-covariance matrix was successfully derived from the information matrix.

fipc Logical value indicating whether Fixed Item Parameter Calibration (FIPC) was applied.

fipc.method The specific method used for FIPC.

fix.loc An integer vector indicating the positions of fixed items used during FIPC.

Components that can be extracted from an object of class `est_mg` created by `est_mg()` include:

estimates A list with two components: `overall` and `group`.

- `overall`: A data frame containing item parameter estimates and their standard errors, based on the combined data set across all groups.
- `group`: A list of group-specific data frames containing item parameter estimates and standard errors for each group.

par.est Same structure as `estimates`, but containing only the item parameter estimates (without standard errors).

se.est Same structure as `estimates`, but containing only the standard errors of the item parameter estimates. The standard errors are computed using the cross-product approximation method (Meilijson, 1989).

pos.par A data frame indicating the position index of each estimated parameter. This index is based on the combined item set across all groups and is useful when interpreting the variance-covariance matrix.

covariance A variance-covariance matrix for the item parameter estimates based on the combined data from all groups.

loglikelihood A list with `overall` and `group` components:

- `overall`: The marginal log-likelihood summed over all unique items across all groups.
- `group`: Group-specific marginal log-likelihood values.

aic Akaike Information Criterion (AIC) computed from the overall log-likelihood.

bic Bayesian Information Criterion (BIC) computed from the overall log-likelihood.

group.par A list of group-specific summary statistics (mean, variance, and standard deviation) of the latent trait prior distribution.

weights A list of two-column data frames (one per group) containing the quadrature points (first column) and the corresponding weights (second column) for the updated prior distributions.

posterior.dist A matrix of normalized posterior densities for all response patterns at each quadrature point. Rows correspond to individuals, and columns to quadrature points.

data A list with `overall` and `group` components, each containing examinee response data.

scale.D The scaling constant used in the IRT model (typically 1 or 1.7).

ncase A list with `overall` and `group` components indicating the number of response patterns in each.

nitem A list with `overall` and `group` components indicating the number of items in the respective response sets.

Etol Convergence criterion used for the E-step in the EM algorithm.

MaxE Maximum number of E-steps allowed in the EM algorithm.

aprior A list describing the prior distribution for item slope parameters.

gprior A list describing the prior distribution for item guessing parameters.

npar.est Total number of parameters estimated across all unique items.

niter Number of EM cycles completed.

maxpar.diff Maximum change in item parameter estimates at convergence.

EMtime Computation time (in seconds) for EM estimation.

SEtime Computation time (in seconds) for estimating standard errors.

TotalTime Total computation time (in seconds) for model estimation.

test.1 First-order condition test result indicating whether gradients converged sufficiently.

test.2 Second-order condition test result indicating whether the information matrix is positive definite.

var.note A note indicating whether the variance-covariance matrix was successfully derived from the information matrix.

fipc Logical value indicating whether Fixed Item Parameter Calibration (FIPC) was used.

fipc.method The method used for FIPC.

fix.loc A list with `overall` and `group` components specifying the locations of fixed items when FIPC was applied.

Components that can be extracted from an object of class `est_item` created by `est_item()` include:

estimates A data frame containing both the item parameter estimates and their corresponding standard errors.

par.est A data frame containing only the item parameter estimates.

se.est A data frame containing the standard errors of the item parameter estimates, computed using observed information functions.

pos.par A data frame indicating the position index of each estimated item parameter. This is useful when interpreting the variance-covariance matrix.

covariance A variance-covariance matrix of the item parameter estimates.

loglikelihood The sum of log-likelihood values across all items in the complete data set.

data A data frame of examinee response data.

score A numeric vector of examinees' ability values used as fixed effects during estimation.

scale.D The scaling constant (typically 1 or 1.7) used in the IRT model.

convergence A character string indicating the convergence status of the item parameter estimation.

nitem The total number of items included in the response data.

deleted.item Items that contained no response data and were excluded from estimation.

npar.est The total number of estimated item parameters.

n.response An integer vector indicating the number of responses used to estimate parameters for each item.

TotalTime Total computation time (in seconds) for the estimation process.

See `est_irt()`, `est_mg()`, and `est_item()` for more details.

Value

The internal component extracted from an object of class `est_irt`, `est_mg`, or `est_item`, depending on the input to the `x` argument.

Methods (by class)

- `getirt(est_irt)`: An object created by the function `est_irt()`.
- `getirt(est_mg)`: An object created by the function `est_mg()`.
- `getirt(est_item)`: An object created by the function `est_item()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

`est_irt()`, `est_mg()`, `est_item()`

Examples

```
# Fit a 2PL model to the LSAT6 data
mod.2pl <- est_irt(data = LSAT6, D = 1, model = "2PLM", cats = 2)

# Extract item parameter estimates
(est.par <- getirt(mod.2pl, what = "par.est"))

# Extract standard error estimates
(est.se <- getirt(mod.2pl, what = "se.est"))

# Extract the variance-covariance matrix of item parameter estimates
(cov.mat <- getirt(mod.2pl, what = "covariance"))
```

---

grdif *Generalized IRT residual-based DIF detection framework for multiple groups (GRDIF)*

Description

This function computes three GRDIF statistics, $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$, for analyzing differential item functioning (DIF) among multiple groups (Lim et al., 2024). They are specialized to capture uniform DIF, nonuniform DIF, and mixed DIF, respectively.

Usage

```
grdif(x, ...)

## Default S3 method:
grdif(
  x,
  data,
  score = NULL,
  group,
  focal.name,
  D = 1,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("grdifrs", "grdifr", "grdifs"),
  max.iter = 10,
  min.resp = NULL,
  post.hoc = TRUE,
  method = "ML",
  range = c(-4, 4),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'est_irt'
grdif(
  x,
  score = NULL,
  group,
  focal.name,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("grdifrs", "grdifr", "grdifs"),
  max.iter = 10,
  min.resp = NULL,
  post.hoc = TRUE,
  method = "ML",
  range = c(-4, 4),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
```

```
  ...
)

## S3 method for class 'est_item'
grdif(
  x,
  group,
  focal.name,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("grdifrs", "grdifr", "grdifs"),
  max.iter = 10,
  min.resp = NULL,
  post.hoc = TRUE,
  method = "ML",
  range = c(-4, 4),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class `est_irt` obtained from `est_irt()`, or `est_item` from `est_item()`. See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
...	Additional arguments passed to the `est_score()` function.
data	A matrix of examinees' item responses corresponding to the items specified in the `x` argument. Rows represent examinees and columns represent items.
score	A numeric vector containing examinees' ability estimates (theta values). If not provided, `grdif()` will estimate ability parameters internally before computing the GRDIF statistics. See `est_score()` for more information on scoring methods. Default is `NULL`.
group	A numeric or character vector indicating examinees' group membership. The length of the vector must match the number of rows in the response data matrix.
focal.name	A numeric or character vector specifying the levels associated with the focal groups. For example, consider `group = c(0, 0, 1, 2, 2, 3, 3)`, where '1', '2', and '3' indicate three distinct focal groups and '0' represents the reference group. In this case, set `focal.name = c(1, 2, 3)`.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.

`alpha` A numeric value specifying the significance level (α) for hypothesis testing using the GRDIF statistics. Default is `0.05`.

`missing` A value indicating missing responses in the data set. Default is `NA`.

`purify` Logical. Indicates whether to apply a purification procedure. Default is `FALSE`.

`purify.by` A character string specifying which GRDIF statistic is used to perform the purification. Available options are "grdifrs" for $GRDIF_{RS}$, "grdifr" for $GRDIF_R$, and "grdifs" for $GRDIF_S$.

`max.iter` A positive integer specifying the maximum number of iterations allowed for the purification process. Default is `10`.

`min.resp` A positive integer specifying the minimum number of valid item responses required from an examinee in order to compute an ability estimate. Default is `NULL`. See **Details** for more information.

`post.hoc` A logical value indicating whether to perform post-hoc RDIF analyses for all possible pairwise group comparisons on items flagged as statistically significant. Default is TRUE. See details below.

`method` A character string indicating the scoring method to use. Available options are:

- `"ML"`: Maximum likelihood estimation
- `"WL"`: Weighted likelihood estimation (Warm, 1989)
- `"MAP"`: Maximum a posteriori estimation (Hambleton et al., 1991)
- `"EAP"`: Expected a posteriori estimation (Bock & Mislevy, 1982)

Default is `"ML"`.

`range` A numeric vector of length two specifying the lower and upper bounds of the ability scale. This is used for the following scoring methods: `"ML"`, `"WL"`, and `"MAP"`. Default is `c(-5, 5)`.

`norm.prior` A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. These values are used to generate the Gaussian quadrature points and weights. Ignored if `method` is `"ML"` or `"WL"`. Default is `c(0, 1)`.

`nquad` An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. Used only when `method` is `"EAP"`. Ignored for `"ML"`, `"WL"`, and `"MAP"`. Default is 41.

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and their corresponding weights (in the second column) for the latent variable prior distribution. The weights and points can be conveniently generated using the function `gen.weight()`.

If `NULL` and `method = "EAP"`, default quadrature values are generated based on the `norm.prior` and `nquad` arguments. Ignored if `method` is `"ML"`, `"WL"`, or `"MAP"`.

`ncore` An integer specifying the number of logical CPU cores to use for parallel processing. Default is `1`. See `est_score()` for details.

`verbose` Logical. If `TRUE`, progress messages from the purification procedure will be displayed; if `FALSE`, the messages will be suppressed. Default is `TRUE`.

Details

The GRDIF framework (Lim et al., 2024) is a generalized version of the RDIF detection framework, designed to assess DIF across multiple groups. The framework includes three statistics: $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$, which are tailored to detect uniform, nonuniform, and mixed DIF, respectively. Under the null hypothesis that the test contains no DIF items, the statistics $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$ asymptotically follow χ^2 distributions with G-1, G-1, and 2(G-1) degrees of freedom, respectively, where *G* represents the number of groups being compared. For more information on the GRDIF framework, see Lim et al. (2024).

The `grdif()` function computes all three GRDIF statistics: $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$. It supports both dichotomous and polytomous item response data. To compute these statistics, `grdif()` requires: (1) item parameter estimates obtained from aggregate data (regardless of group membership); (2) examinees' ability estimates (e.g., ML); and (3) their item response data. Note that ability estimates must be computed using the item parameters estimated from the aggregate data. The item parameters should be provided via the `x` argument, the ability estimates via the `score` argument, and the response data via the `data` argument. If ability estimates are not supplied (`score = NULL`), `grdif()` automatically computes them using the scoring method specified in the `method` argument (e.g., `method = "ML"`).

The `group` argument accepts a vector of numeric or character values, indicating the group membership of examinees. The vector may contain multiple distinct values, where one represents the reference group and the others represent focal groups. Its length must match the number of rows in the response data, with each value corresponding to an examinee's group membership. Once `group` is specified, a numeric or character vector must be supplied via the `focal.name` argument to define which group(s) in `group` represent the focal groups. The reference group is defined as the group not included in `focal.name`.

Similar to the original RDIF framework for two-group comparisons, the GRDIF framework supports an iterative purification process. When `purify = TRUE`, purification is conducted based on the GRDIF statistic specified in the `purify.by` argument (e.g., `purify.by = "grdifrs"`). During each iteration, examinees' latent abilities are re-estimated using only the purified items, with the scoring method determined by the `method` argument. The process continues until no additional DIF items are flagged or until the number of iterations reaches the specified `max.iter` limit. For details on the purification procedure, see Lim et al. (2022).

Scoring based on a limited number of items can lead to large standard errors, which may compromise the effectiveness of DIF detection within the GRDIF framework. The `min.resp` argument can be used to exclude ability estimates with substantial standard errors, especially during the purification process. For example, if `min.resp` is not NULL (e.g., `min.resp = 5`), examinees whose total number of responses falls below the specified threshold will have their responses treated as missing values (i.e., NA). Consequently, their ability estimates will also be missing and will not be used when computing the GRDIF statistics. If `min.resp = NULL`, an examinee's score will be computed as long as at least one response is available.

The `post.hoc` argument enables post-hoc RDIF analyses across all possible pairwise group comparisons for items flagged as statistically significant. For instance, consider four groups of examinees: A, B, C, and D. If `post.hoc = TRUE`, the `grdif()` function will perform pairwise RDIF analyses for each flagged item across all group pairs (A-B, A-C, A-D, B-C, B-D, and C-D). This provides a more detailed understanding of which specific group pairs exhibit DIF. Note that when purification is enabled (i.e., `purify = TRUE`), post-hoc RDIF analyses are conducted for each flagged item at each iteration of the purification process.

Value

This function returns a list containing four main components:

`no_purify` A list of sub-objects presenting the results of DIF analysis without a purification procedure. These include:

- **dif_stat** A data frame summarizing the results of the three GRDIF statistics for all evaluated items. Columns include item ID, $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$ statistics; their corresponding p-values; the sample size of the reference group; the sample sizes of the focal groups; and the total sample size.
- **moments** A list of three data frames reporting the moments of mean raw residuals (MRRs) and mean squared residuals (MSRs) across all compared groups. The first contains means, the second variances, and the third covariances of MRRs and MSRs.
- **dif_item** A list of three numeric vectors indicating the items flagged as potential DIF items by each of the GRDIF statistics ($GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$).
- **score** A numeric vector of ability estimates used to compute the GRDIF statistics.
- **post.hoc** A list of three data frames containing post-hoc RDIF analysis results for all possible pairwise group comparisons. Each data frame corresponds to the results for items flagged by $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$, respectively.

`purify` A logical value indicating whether the purification process was applied.

`with_purify` A list of sub-objects presenting the results of DIF analysis with a purification procedure. These include:

- **purify.by** A character string indicating which GRDIF statistic was used for purification: "grdifr", "grdifs", or "grdifrs", corresponding to $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$, respectively.
- **dif_stat** A data frame summarizing the GRDIF results across iterations. Columns include item ID, the three GRDIF statistics and their p-values, sample size of the reference group, sample sizes of the focal groups, total sample size, and the iteration number at which each statistic was computed.
- **moments** A list of three data frames showing the MRR and MSR moments across iterations. The final column in each data frame indicates the iteration in which the statistics were computed.
- **n.iter** The total number of iterations executed during the purification process.
- **score** A numeric vector of the final purified ability estimates used for computing GRDIF statistics.
- **post.hoc** A data frame containing the post-hoc RDIF analysis results for flagged items across all possible pairwise group comparisons, updated at each iteration.
- **complete** A logical value indicating whether the purification process was completed. If `FALSE`, the process reached the maximum number of iterations without convergence.

`alpha` The significance level (α) used for hypothesis testing of the GRDIF statistics.

Methods (by class)

- `grdif(default)`: Default method to compute the three GRDIF statistics for multiple-group data using a data frame `x` that contains item metadata.
- `grdif(est_irt)`: An object created by the function `est_irt()`.
- `grdif(est_item)`: An object created by the function `est_item()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Lim, H., & Choe, E. M. (2023). Detecting differential item functioning in CAT using IRT residual DIF approach. *Journal of Educational Measurement, 60*(4), 626-650. doi:10.1111/jedm.12366.

Lim, H., Choe, E. M., & Han, K. T. (2022). A residual-based differential item functioning detection framework in item response theory. *Journal of Educational Measurement, 59*(1), 80-104. doi:10.1111/jedm.12313.

Lim, H., Zhu, D., Choe, E. M., & Han, K. T. (2024). Detecting differential item functioning among multiple groups using IRT residual DIF framework. *Journal of Educational Measurement, 61*(4), 656-681.

See Also

`rdif()` `est_irt()`, `est_item()`, `simdat()`, `shape_df()`, `est_score()`

Examples

```
# Load required library
library("dplyr")

## Uniform DIF detection for four groups (1 reference, 3 focal)
#########################################################
# (1) Manipulate uniform DIF for all three focal groups
#########################################################

# Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Select 36 non-DIF items modeled under 3PLM
par_nstd <-
  bring.flexmirt(file = flex_sam, "par")$Group1$full_df %>%
  dplyr::filter(.data$model == "3PLM") %>%
  dplyr::filter(dplyr::row_number() %in% 1:36) %>%
  dplyr::select(1:6)
par_nstd$id <- paste0("nondif", 1:36)

# Generate four new items on which uniform DIF will be imposed
difpar_ref <-
  shape_df(
```

```
    par.drm = list(a = c(0.8, 1.5, 0.8, 1.5), b = c(0.0, 0.0, -0.5, -0.5), g = .15),
    item.id = paste0("dif", 1:4), cats = 2, model = "3PLM"
  )

# Introduce DIF by shifting b-parameters differently for each focal group
difpar_foc1 <-
  difpar_ref %>%
  dplyr::mutate_at(.vars = "par.2", .funs = function(x) x + c(0.7, 0.7, 0, 0))
difpar_foc2 <-
  difpar_ref %>%
  dplyr::mutate_at(.vars = "par.2", .funs = function(x) x + c(0, 0, 0.7, 0.7))
difpar_foc3 <-
  difpar_ref %>%
  dplyr::mutate_at(.vars = "par.2", .funs = function(x) x + c(-0.4, -0.4, -0.5, -0.5))

# Combine the 4 DIF and 36 non-DIF items for all four groups
# Therefore, the first four items contain uniform DIF across all focal groups
par_ref <- rbind(difpar_ref, par_nstd)
par_foc1 <- rbind(difpar_foc1, par_nstd)
par_foc2 <- rbind(difpar_foc2, par_nstd)
par_foc3 <- rbind(difpar_foc3, par_nstd)

# Generate true abilities from different distributions
set.seed(128)
theta_ref <- rnorm(500, 0.0, 1.0)
theta_foc1 <- rnorm(500, -1.0, 1.0)
theta_foc2 <- rnorm(500, 1.0, 1.0)
theta_foc3 <- rnorm(500, 0.5, 1.0)

# Simulate response data for each group
resp_ref <- irtQ::simdat(par_ref, theta = theta_ref, D = 1)
resp_foc1 <- irtQ::simdat(par_foc1, theta = theta_foc1, D = 1)
resp_foc2 <- irtQ::simdat(par_foc2, theta = theta_foc2, D = 1)
resp_foc3 <- irtQ::simdat(par_foc3, theta = theta_foc3, D = 1)
data <- rbind(resp_ref, resp_foc1, resp_foc2, resp_foc3)

#########################################################
# (2) Estimate item and ability parameters
#     using aggregated data
#########################################################

# Estimate item parameters
est_mod <- irtQ::est_irt(data = data, D = 1, model = "3PLM")
est_par <- est_mod$par.est

# Estimate ability parameters using MLE
score <- irtQ::est_score(x = est_par, data = data, method = "ML")$est.theta

#########################################################
# (3) Conduct DIF analysis
#########################################################

# Create a group membership vector:
```

```
# 0 = reference group; 1, 2, 3 = focal groups
group <- c(rep(0, 500), rep(1, 500), rep(2, 500), rep(3, 500))

# (a) Compute GRDIF statistics without purification,
#     and perform post-hoc pairwise comparisons for flagged items
dif_nopuri <- grdif(
  x = est_par, data = data, score = score, group = group,
  focal.name = c(1, 2, 3), D = 1, alpha = 0.05,
  purify = FALSE, post.hoc = TRUE
)
print(dif_nopuri)

# Display post-hoc pairwise comparison results
print(dif_nopuri$no_purify$post.hoc)

# (b) Compute GRDIF statistics with purification
#     based on GRDIF_R, including post-hoc comparisons
dif_puri_r <- grdif(
  x = est_par, data = data, score = score, group = group,
  focal.name = c(1, 2, 3), D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "grdifr", post.hoc = TRUE
)
print(dif_puri_r)

# Display post-hoc results before purification
print(dif_puri_r$no_purify$post.hoc)

# Display post-hoc results after purification
print(dif_puri_r$with_purify$post.hoc)
```

`info` *Item and Test Information Function*

Description

This function computes item and test information functions (Hambleton et al., 1991) for a given set of theta values.

Usage

```
info(x, ...)

## Default S3 method:
info(x, theta, D = 1, tif = TRUE, ...)

## S3 method for class 'est_item'
info(x, theta, tif = TRUE, ...)
```

```
## S3 method for class 'est_irt'
info(x, theta, tif = TRUE, ...)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class est_irt obtained from est_irt(), or est_item from est_item(). See est_irt() or simdat() for more details about the item metadata. This data frame can be easily created using the shape_df() function.
...	Further arguments passed to or from other methods.
theta	A numeric vector of theta values at which item and test information are computed.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
tif	Logical. If TRUE, the test information function is computed. Default is TRUE.

Details

This function calculates the amount of statistical information provided by each item (item information function, IIF) and the total test (test information function, TIF) across a range of ability (theta) values. Higher information values at a particular theta level indicate greater measurement precision at that ability level.

The input x must follow a specific data frame format if not already an est_irt or est_item object. The structure of this data frame is explained in the documentation of est_irt() and simdat(). Items of different models (e.g., 3PLM, GPCM) can be combined in a single test.

The information is computed for each item appropriately and aggregated for the TIF if tif = TRUE. The TIF is often used to assess where the test provides the most precision, and is critical when designing adaptive tests or evaluating test coverage across the ability continuum.

The returned object is a list of class "info", which contains the item information matrix and the test information vector. The plot() method for info objects can be used to visualize the IIFs and TIF (see plot.info()).

Value

This function returns an object of class info, which is a list containing the following components:

iif	A matrix of item information values. Each row corresponds to an item, and each column represents the information value computed at a given theta point. The row names are the item IDs, and the column names indicate the theta points (e.g., "theta.1", "theta.2", ...).
tif	A numeric vector containing the test information values at each theta value, computed as the sum of item information values across all items. This component is included only when tif = TRUE.

`theta` A numeric vector of theta values at which the item and test information functions are evaluated. This matches the user-supplied `theta` argument.

The returned object is of class `info` and can be visualized using the function `plot.info()`. This output structure is consistent across input types (`data.frame`, `est_item`, `est_irt`), and facilitates downstream plotting, comparison, or export of information function values.

Methods (by class)

- `info(default)`: Default method to compute item and test information functions for a data frame `x` containing the item metadata.
- `info(est_item)`: An object created by the function `est_item()`.
- `info(est_irt)`: An object created by the function `est_irt()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Hambleton, R. K., & Swaminathan, H. (1985) *Item response theory: Principles and applications*. Boston, MA: Kluwer.

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991) *Fundamentals of item response theory*. Newbury Park, CA: Sage.

See Also

`plot.info()`, `shape_df()`, `est_irt()`, `est_item()`

Examples

```
## Example 1.
## Using the function `shape_df()` to create a data frame of test metadata

# Create a list of dichotomous item parameters
par.drm <- list(
  a = c(1.1, 1.2, 0.9, 1.8, 1.4),
  b = c(0.1, -1.6, -0.2, 1.0, 1.2),
  g = rep(0.2, 5)
)

# Create a list of polytomous item parameters
par.prm <- list(
  a = c(1.4, 0.6),
  d = list(c(-1.9, 0.0, 1.2), c(0.4, -1.1, 1.5, 0.2))
)

# Create a numeric vector for the number of score categories for each item
cats <- c(2, 4, 2, 2, 5, 2, 2)

# Create a character vector of IRT model types for each item
```

```
model <- c("DRM", "GRM", "DRM", "DRM", "GPCM", "DRM", "DRM")

# Create item metadata using `shape_df()`
test <- shape_df(
  par.drm = par.drm, par.prm = par.prm,
  cats = cats, model = model
) # create a data frame

# Define a sequence of theta values
theta <- seq(-2, 2, 0.1)

# Compute item and test information values based on theta
info(x = test, theta = theta, D = 1, tif = TRUE)


## Example 2.
## Using a "-prm.txt" file exported from flexMIRT

# Import a sample "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt",
  package = "irtQ"
)

# Read item parameters and convert them into item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# Define a sequence of theta values
theta <- seq(-2, 2, 0.1)

# Compute item and test information values based on theta
info(x = test_flex, theta = theta, D = 1, tif = TRUE)
```

`irtfit` *Traditional IRT Item Fit Statistics*

Description

This function computes traditional IRT item fit statistics, including the χ^2 fit statistic (e.g., Bock, 1960; Yen, 1981), the log-likelihood ratio χ^2 fit statistic (G^2; McKinley & Mills, 1985), and the infit and outfit statistics (Ames et al., 2015). It also returns contingency tables used to compute the χ^2 and G^2 statistics.

Usage

```
irtfit(x, ...)

## Default S3 method:
irtfit(
  x,
```

```
  score,
  data,
  group.method = c("equal.width", "equal.freq"),
  n.width = 10,
  loc.theta = "average",
  range.score = NULL,
  D = 1,
  alpha = 0.05,
  missing = NA,
  overSR = 2,
  min.collapse = 1,
  pcm.loc = NULL,
  ...
)

## S3 method for class 'est_item'
irtfit(
  x,
  group.method = c("equal.width", "equal.freq"),
  n.width = 10,
  loc.theta = "average",
  range.score = NULL,
  alpha = 0.05,
  missing = NA,
  overSR = 2,
  min.collapse = 1,
  pcm.loc = NULL,
  ...
)

## S3 method for class 'est_irt'
irtfit(
  x,
  score,
  group.method = c("equal.width", "equal.freq"),
  n.width = 10,
  loc.theta = "average",
  range.score = NULL,
  alpha = 0.05,
  missing = NA,
  overSR = 2,
  min.collapse = 1,
  pcm.loc = NULL,
  ...
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class est_irt obtained from est_irt(), or est_item from est_item(). See est_irt() or simdat() for more details about the item metadata. This data frame can be easily created using the shape_df() function.
...	Further arguments passed to or from other methods.
score	A numeric vector containing examinees' ability estimates (theta values).
data	A matrix of examinees' item responses corresponding to the items specified in the x argument. Rows represent examinees and columns represent items.
group.method	A character string specifying the method used to group examinees along the ability scale when computing the χ^2 and G^2 fit statistics. Available options are: • "equal.width": Divides the ability scale into intervals of equal width. • "equal.freq": Divides the examinees into groups with (approximately) equal numbers of examinees. Note that "equal.freq" does not guarantee exactly equal group sizes due to ties in ability estimates. Default is "equal.width". The number of groups and the range of the ability scale are controlled by the n.width and range.score arguments, respectively.
n.width	An integer specifying the number of intervals (groups) into which the ability scale is divided for computing the fit statistics. Default is 10.
loc.theta	A character string indicating the point on the ability scale at which the expected category probabilities are calculated for each group. Available options are: • "average": Uses the average ability estimate of examinees within each group. • "middle": Uses the midpoint of each group's ability interval. Default is "average".
range.score	A numeric vector of length two specifying the lower and upper bounds of the ability scale. Ability estimates below the lower bound or above the upper bound are truncated to the respective bound. If NULL, the observed minimum and maximum of the score vector are used. Note that this range restriction is independent of the grouping method specified in group.method. Default is NULL.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
alpha	A numeric value specifying the significance level (α) for the hypothesis tests of the χ^2 and G^2 item fit statistics. Default is 0.05.
missing	A value indicating missing responses in the data set. Default is NA. See **Details** below.
overSR	A numeric threshold used to identify ability groups (intervals) whose standardized residuals exceed the specified value. This is used to compute the proportion of misfitting groups per item. Default is 2.

`min.collapse` An integer specifying the minimum expected frequency required for a cell in the contingency table. Neighboring groups will be merged if any expected cell frequency falls below this threshold when computing the χ^2 and G^2 statistics. Default is 1.

`pcm.loc` A vector of integers indicating the locations (indices) of partial credit model (PCM) items for which slope parameters are fixed.

Details

To compute the χ^2 and G^2 item fit statistics, the `group.method` argument determines how the ability scale is divided into groups:

- `"equal.width"`: Examinees are grouped based on intervals of equal width a long the ability scale.
- `"equal.freq"`: Examinees are grouped such that each group contains (approximately) the same number of individuals.

Note that `"equal.freq"` does not guarantee *exactly* equal frequencies across all groups, since grouping is based on quantiles.

When dividing the ability scale into intervals to compute the χ^2 and G^2 fit statistics, the intervals should be:

- **Wide enough** to ensure that each group contains a sufficient number of examinees (to avoid unstable estimates),
- **Narrow enough** to ensure that examinees within each group are relatively homogeneous in ability (Hambleton et al., 1991).

If you want to divide the ability scale into a number of groups other than the default of 10, specify the desired number using the `n.width` argument. For reference:

- Yen (1981) used 10 fixed-width groups,
- Bock (1960) allowed for flexibility in the number of groups.

Regarding degrees of freedom (*df*):

- The χ^2 statistic is approximately chi-square distributed with degrees of freedom equal to the number of ability groups minus the number of item parameters (Ames et al., 2015).
- The G^2 statistic is approximately chi-square distributed with degrees of freedom equal to the number of ability groups (Ames et al., 2015; Muraki & Bock, 2003).

  Note that if `"DRM"` is specified for an item in the item metadata set, the item is treated as a `"3PLM"` when computing the degrees of freedom for the χ^2 fit statistic.

Note that infit and outfit statistics should be interpreted with caution when applied to non-Rasch models. The returned object—particularly the contingency tables—can be passed to `plot.irtfit()` to generate raw and standardized residual plots (Hambleton et al., 1991).

Value

This function returns an object of class `irtfit`, which includes the following components:

`fit_stat` A data frame containing the results of three IRT item fit statistics— χ^2, G^2, infit, and outfit—for all evaluated items. Each row corresponds to one item, and the columns include: the item ID; χ^2 statistic; G^2 statistic; degrees of freedom for χ^2 and G^2; critical values and p-values for both statistics; outfit and infit values; the number of examinees used to compute these statistics; and the proportion of ability groups (prior to cell collapsing) that have standardized residuals greater than the threshold specified in the `overSR` argument.

`contingency.fitstat` A list of contingency tables used to compute the χ^2 and G^2 fit statistics for all items. Note that the cell-collapsing strategy is applied to these tables to ensure sufficient expected frequencies.

`contingency.plot` A list of contingency tables used to generate raw and standardized residual plots (Hambleton et al., 1991) via the `plot.irtfit()`. Note that these tables are based on the original, uncollapsed groupings.

`individual.info` A list of data frames containing individual residuals and corresponding variance values. This information is used to compute infit and outfit statistics.

`item_df` A data frame containing the item metadata provided in the argument `x`.

`ancillary` A list of ancillary information used during the item fit analysis.

Methods (by class)

- `irtfit(default)`: Default method for computing traditional IRT item fit statistics using a data frame `x` that contains item metadata.
- `irtfit(est_item)`: An object created by the function `est_item()`.
- `irtfit(est_irt)`: An object created by the function `est_irt()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Ames, A. J., & Penfield, R. D. (2015). An NCME Instructional Module on Item-Fit Statistics for Item Response Theory Models. *Educational Measurement: Issues and Practice, 34*(3), 39-48.

Bock, R.D. (1960), *Methods and applications of optimal scaling*. Chapel Hill, NC: L.L. Thurstone Psychometric Laboratory.

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991).*Fundamentals of item response theory*. Newbury Park, CA: Sage.

McKinley, R., & Mills, C. (1985). A comparison of several goodness-of-fit statistics. *Applied Psychological Measurement, 9*, 49-57.

Muraki, E. & Bock, R. D. (2003). PARSCALE 4: IRT item analysis and test scoring for rating scale data (Computer Software). Chicago, IL: Scientific Software International. URL http://www.ssicentral.com

Wells, C. S., & Bolt, D. M. (2008). Investigation of a nonparametric procedure for assessing goodness-of-fit in item response theory. *Applied Measurement in Education, 21*(1), 22-40.

Yen, W. M. (1981). Using simulation results to choose a latent trait model. *Applied Psychological Measurement, 5*, 245-262.

See Also

plot.irtfit(), shape_df(), est_irt(), est_item()

Examples

```
## Example 1
## Use the simulated CAT data
# Identify items with more than 10,000 responses
over10000 <- which(colSums(simCAT_MX$res.dat, na.rm = TRUE) > 10000)

# Select items with more than 10,000 responses
x <- simCAT_MX$item.prm[over10000, ]

# Extract response data for the selected items
data <- simCAT_MX$res.dat[, over10000]

# Extract examinees' ability estimates
score <- simCAT_MX$score

# Compute item fit statistics
fit1 <- irtfit(
  x = x, score = score, data = data, group.method = "equal.width",
  n.width = 10, loc.theta = "average", range.score = NULL, D = 1, alpha = 0.05,
  missing = NA, overSR = 2
)

# View the fit statistics
fit1$fit_stat

# View the contingency tables used to compute fit statistics
fit1$contingency.fitstat


## Example 2
## Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Select the first two dichotomous items and the last polytomous item
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df[c(1:2, 55), ]

# Generate ability values from a standard normal distribution
set.seed(10)
score <- rnorm(1000, mean = 0, sd = 1)
```

```
# Simulate response data
data <- simdat(x = x, theta = score, D = 1)

# Compute item fit statistics
fit2 <- irtfit(
  x = x, score = score, data = data, group.method = "equal.freq",
  n.width = 11, loc.theta = "average", range.score = c(-4, 4), D = 1, alpha = 0.05
)

# View the fit statistics
fit2$fit_stat

# View the contingency tables used to compute fit statistics
fit2$contingency.fitstat

# Plot raw and standardized residuals for the first item (dichotomous)
plot(x = fit2, item.loc = 1, type = "both", ci.method = "wald",
     show.table = TRUE, ylim.sr.adjust = TRUE)

# Plot raw and standardized residuals for the third item (polytomous)
plot(x = fit2, item.loc = 3, type = "both", ci.method = "wald",
     show.table = FALSE, ylim.sr.adjust = TRUE)
```

`llike_score` *Log-Likelihood of Ability Parameters*

Description

This function computes the log-likelihood values for a set of ability parameters, given item parameters and response data

Usage

```
llike_score(
  x,
  data,
  theta,
  D = 1,
  method = "ML",
  norm.prior = c(0, 1),
  fence.a = 3,
  fence.b = NULL,
  missing = NA
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
data	A matrix of examinees' item responses corresponding to the items specified in the x argument. Rows represent examinees and columns represent items.
theta	A numeric vector of ability values at which to evaluate the log-likelihood function.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
method	A character string specifying the estimation method. Available options include: • "ML": Maximum Likelihood Estimation • "MLF": Maximum Likelihood Estimation with Fences • "MAP": Maximum A Posteriori Estimation Default is "ML".
norm.prior	A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution (used only when method = "MAP"). Default is c(0, 1). Ignored for "ML" and "MLF".
fence.a	A numeric value specifying the item slope parameter (i.e., *a*-parameter) for the two imaginary items used in MLF. See **Details** below. Default is 3.0.
fence.b	A numeric vector of length two specifying the lower and upper bounds of the item difficulty parameters (i.e., *b*-parameters) for the two imaginary items in MLF. If fence.b = NULL, the values specified in the range argument are used instead. Default is NULL.
missing	A value indicating missing responses in the data set. Default is NA.

Details

This function evaluates the log-likelihood of a given ability (theta) for one or more examinees, based on item parameters (x) and item response data (data).

If method = "MLF" is selected, the function appends two virtual "fence" items to the item pool with fixed parameters. These artificial items help avoid unstable likelihood functions near the boundaries of the ability scale.

For example, to compute the log-likelihood curves of two examinees' responses to the same test items, supply a 2-row matrix to data and a vector of ability values to theta.

Value

A data frame of log-likelihood values.

- Each **row** corresponds to an ability value (theta).
- Each **column** corresponds to an examinee's response pattern.

Examples

```
## Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read item parameters and convert them to item metadata
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df

# Generate ability values from N(0, 1)
set.seed(10)
score <- rnorm(5, mean = 0, sd = 1)

# Simulate response data
data <- simdat(x = x, theta = score, D = 1)

# Specify ability values for log-likelihood evaluation
theta <- seq(-3, 3, 0.5)

# Compute log-likelihood values (using MLE)
llike_score(x = x, data = data, theta = theta, D = 1, method = "ML")
```

LSAT6 *LSAT6 Data*

Description

A well-known dichotomous response dataset from the Law School Admission Test (LSAT), Section 6, as used in Thissen (1982).

Usage

```
LSAT6
```

Format

A data frame with 1,000 rows and 5 columns, where each row represents a unique examinee's response pattern to five dichotomously scored items (0 = incorrect, 1 = correct).

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Thissen, D. (1982). Marginal maximum likelihood estimation for the one-parameter logistic model. *Psychometrika, 47*, 175–186.

`lwrc` *Lord-Wingersky Recursion Formula*

Description

This function computes the conditional distributions of number-correct (or observed) scores given either the probabilities of category responses for each item or a set of theta values, using the Lord and Wingersky recursion formula (1984).

Usage

```
lwrc(x = NULL, theta, prob = NULL, cats, D = 1)
```

Arguments

`x`	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function. If `prob` is `NULL`, the item metadata in `x` is used in the recursion formula. See **Details** below.
`theta`	A vector of theta values at which the conditional distributions of observed scores are computed. This argument is required only when item metadata is provided via the `x` argument.
`prob`	A matrix containing the category response probabilities for each item. Each row corresponds to an item, and each column represents a score category. If items have different numbers of categories, empty cells should be filled with zeros or `NA` values. If `x` is `NULL`, this matrix is used in the recursion formula.
`cats`	A numeric vector specifying the number of score categories for each item. For example, a dichotomous item has two categories. This argument is required only when a probability matrix is provided via the `prob` argument.
`D`	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.

Details

The Lord and Wingersky recursive algorithm provides an efficient method for calculating the compound probabilities of all possible number-correct (i.e., observed) scores on a test, based on IRT models. This algorithm is particularly useful for obtaining the IRT model-based distribution of observed scores.

The conditional distributions of observed scores can be computed using either the item metadata specified in `x` or the category probability matrix specified in `prob`.

Value

When the prob argument is provided, the function returns a vector of probabilities for all possible observed scores on a test.

When the x argument is specified, it returns a matrix of conditional probabilities for each observed score across all specified theta values.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Kolen, M. J. & Brennan, R. L. (2004) *Test Equating, Scaling, and Linking* (2nd ed.). New York: Springer.

Lord, F. & Wingersky, M. (1984). Comparison of IRT true score and equipercentile observed score equatings. *Applied Psychological Measurement, 8*(4), 453-461.

Examples

```
## Example 1: Using a matrix of category probabilities
## This example is from Kolen and Brennan (2004, p. 183)
# Create a matrix of probabilities for correct and incorrect responses to three items
probs <- matrix(c(.74, .73, .82, .26, .27, .18), nrow = 3, ncol = 2, byrow = FALSE)

# Create a vector specifying the number of score categories for each item
cats <- c(2, 2, 2)

# Compute the conditional distribution of observed scores
lwrc(prob = probs, cats = cats)

## Example 2: Using a matrix of category probabilities for a mixed-format test
# Category probabilities for a dichotomous item
p1 <- c(0.2, 0.8, 0, 0, 0)

# Category probabilities for another dichotomous item
p2 <- c(0.4, 0.6, NA, NA, NA)

# Category probabilities for a polytomous item with five categories
p3 <- c(0.1, 0.2, 0.2, 0.4, 0.1)

# Category probabilities for a polytomous item with three categories
p4 <- c(0.5, 0.3, 0.2, NA, NA)

# Combine the probability vectors into a matrix
p <- rbind(p1, p2, p3, p4)

# Create a vector specifying the number of score categories for each item
cats <- c(2, 2, 5, 3)

# Compute the conditional distribution of observed scores
lwrc(prob = p, cats = cats)
```

```
## Example 3: Using a data frame of item metadata for a mixed-format test
# Import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read item parameters and convert them to item metadata
x <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# Compute the conditional distribution of observed scores for a range of theta values
lwrc(x = x, theta = seq(-4, 4, 0.2), D = 1)
```

---

pcd2 *Pseudo-count D2 method*

Description

This function calculates the Pseudo-count D^2 statistic to evaluate item parameter drift, as described by Cappaert et al. (2018) and Stone (2000). The Pseudo-count D^2 statistic is designed to detect item parameter drift efficiently without requiring item recalibration, making it especially valuable in computerized adaptive testing (CAT) environments. This method compares observed and expected response frequencies across quadrature points, which represent latent ability levels. The expected frequencies are computed using the posterior distribution of each examinee's ability (Stone, 2000), providing a robust and sensitive measure of item parameter drift, ensuring the stability and accuracy of the test over time.

Usage

```
pcd2(
  x,
  data,
  D = 1,
  item.skip = NULL,
  missing = NA,
  Quadrature = c(49, 6),
  weights = NULL,
  group.mean = 0,
  group.var = 1,
  crit.val = NULL,
  min.resp = NULL,
  purify = FALSE,
  max.iter = 10,
  verbose = TRUE
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See est_irt() or simdat() for more details about the item metadata. This data frame can be easily created using the shape_df() function.
data	A matrix of examinees' item responses corresponding to the items specified in the x argument. Rows represent examinees and columns represent items.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
item.skip	A numeric vector of item indices to exclude from IPD analysis. If NULL, all items are included. Useful for omitting specific items based on prior insights.
missing	A value indicating missing responses in the data set. Default is NA.
Quadrature	A numeric vector of length two: • first element: number of quadrature points • second element: symmetric bound (absolute value) for those points For example, c(49, 6) specifies 49 evenly spaced points from –6 to 6. These points are used in the E-step of the EM algorithm. Default is c(49, 6).
weights	A two-column matrix or data frame containing the quadrature points (in the first column) and their corresponding weights (in the second column) for the latent variable prior distribution. If not NULL, the scale of the latent ability distribution is fixed to match the scale of the provided quadrature points and weights. The weights and points can be conveniently generated using the function gen.weight(). If NULL, a normal prior density is used instead, based on the information provided in the Quadrature, group.mean, and group.var arguments. Default is NULL.
group.mean	A numeric value specifying the mean of the latent variable prior distribution when weights = NULL. Default is 0. This value is fixed to resolve the indeterminacy of the item parameter scale during calibration.
group.var	A positive numeric value specifying the variance of the latent variable prior distribution when weights = NULL. Default is 1. This value is fixed to resolve the indeterminacy of the item parameter scale during calibration.
crit.val	A critical value applied in hypothesis testing using the Pseudo-count D^2 statistic. Default is NULL.
min.resp	A positive integer specifying the minimum required number of responses for each evaluated item. Defaults to NULL.
purify	Logical. Indicates whether to apply a purification procedure. Default is FALSE.
max.iter	A positive integer specifying the maximum number of iterations allowed for the purification process. Default is 10.
verbose	Logical. If TRUE, progress messages from the purification procedure will be displayed; if FALSE, the messages will be suppressed. Default is TRUE.

Details

The Pseudo-count D^2 statistic quantifies item parameter drift (IPD) by computing the weighted squared differences between the observed and expected response frequencies for each score category across ability levels. The expected frequencies are determined using the posterior distribution of each examinee's ability (Stone, 2000).

The Pseudo-count D^2 statistic is calculated as:

$$Pseudo - count D^2 = \sum_{k=1}^{Q} \left(\frac{r_{0k} + r_{1k}}{N} \right) \left(\frac{r_{1k}}{r_{0k} + r_{1k}} - E_{1k} \right)^2$$

where r_{0k} and r_{1k} are the pseudo-counts for the incorrect and correct responses at each ability level k, E_{1k} is the expected proportion of correct responses at each ability level k, calculated using item parameters from the item bank, and N is the total count of examinees who received each item.

Critical Value (`crit.val`): The `crit.val` argument specifies the threshold used to flag an item as exhibiting potential parameter drift. If an item's Pseudo-count D^2 value exceeds this threshold, it is identified as a drifted item. If `crit.val = NULL`, the function reports the raw statistic without flagging.

Minimum Response Count (`min.resp`): The `min.resp` argument sets a lower bound on the number of responses required for an item to be included in the analysis. Items with fewer responses than `min.resp` are automatically excluded by replacing all their responses with `NA`. This avoids unreliable estimates based on small sample sizes.

Purification Procedure: Although Cappaert et al. (2018) did not incorporate purification into their method, `pcd2()` implements an optional iterative purification process similar to Lim et al. (2022). When `purify = TRUE` and a `crit.val` is provided:

- The procedure begins by identifying items flagged for drift using the initial Pseudo-count D^2 statistics.
- In each subsequent iteration, the item with the highest flagged Pseudo-count D^2 value is removed from the item set, and the statistics are recalculated using only the remaining items.
- The process continues until no additional items are flagged or the number of iterations reaches `max.iter`.
- All flagged items and statistics are saved, and convergence status is reported.

This process ensures that drift detection is not distorted by already-flagged items, improving the robustness of the results.

Value

This function returns a list containing four main components:

`no_purify` A list containing the results of Pseudo-count D^2 analysis without applying the purification procedure. It includes:

ipd_stat A data frame summarizing the Pseudo-count D^2 statistics for all items. The columns include: `id` (item ID), `pcd2` (the computed D^2 value), and `N` (the number of valid examinee responses per item).

ipd_item A numeric vector of item indices that were flagged as exhibiting item parameter drift (IPD), based on the specified critical value `crit.val`. If no items are flagged or `crit.val = NULL`, this is `NULL`.

`purify` A logical value indicating whether the iterative purification procedure was applied (`TRUE`) or not (`FALSE`).

`with_purify` A list containing the results of Pseudo-count D^2 analysis after applying the purification procedure. This list is populated only when both `purify = TRUE` and `crit.val` is not `NULL`. It includes:

ipd_stat A data frame reporting the final Pseudo-count D^2 statistics after purification. Columns include: `id` (item ID), `pcd2` (the computed D^2 value), `N` (the number of valid responses), and `n.iter` (the iteration number in which each item was evaluated).

ipd_item A numeric vector of item indices flagged as IPD items during purification. Items are ordered by the iteration in which they were flagged.

n.iter An integer indicating the number of purification iterations completed.

complete A logical value indicating whether the purification procedure converged before reaching the maximum number of iterations (`max.iter`). If `FALSE`, the iteration limit was reached before convergence.

`crit.val` A numeric value indicating the critical threshold used to flag items for parameter drift. If not specified by the user, this will be `NULL`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Cappaert, K. J., Wen, Y., & Chang, Y. F. (2018). Evaluating CAT-adjusted approaches for suspected item parameter drift detection. *Measurement: Interdisciplinary Research and Perspectives, 16*(4), 226-238.

Stone, C. A. (2000). Monte Carlo based null distribution for an alternative goodness-of-fit test statistic in IRT models. *Journal of educational measurement, 37*(1), 58-75.

Examples

```
## Example 1: No critical value specified
## Compute the Pseudo-count D² statistics for dichotomous items
## Import the "-prm.txt" output file generated by flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Extract metadata for the first 30 3PLM items
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df[1:30, 1:6]

# Generate abilities for 500 examinees from N(0, 1)
set.seed(25)
score <- rnorm(500, mean = 0, sd = 1)

# Simulate response data using the item metadata and ability values
```

```
data <- simdat(x = x, theta = score, D = 1)

# Compute the Pseudo-count D² statistics (no purification applied)
ps_d2 <- pcd2(x = x, data = data)
print(ps_d2)

## Example 2: Applying a critical value with purification
# Compute the Pseudo-count D² statistics with purification enabled
ps_d2_puri <- pcd2(x = x, data = data, crit.val = 0.002, purify = TRUE)
print(ps_d2_puri)
```

`plot.info` *Plot Item and Test Information Functions*

Description

This method plots item or test information functions for a specified set of theta values. It can also display the conditional standard error of estimation (CSEE) at the test level.

Usage

```
## S3 method for class 'info'
plot(
  x,
  item.loc = NULL,
  overlap = FALSE,
  csee = FALSE,
  xlab.text,
  ylab.text,
  main.text,
  lab.size = 15,
  main.size = 15,
  axis.size = 15,
  line.color,
  line.size = 1,
  layout.col = 4,
  strip.size = 12,
  ...
)
```

Arguments

`x`	x An object of class `info` obtained from `info()`.
`item.loc`	A numeric vector indicating which item information functions to plot, specified by item position (e.g., 1 for the first item). If `NULL` (default), the test information function for the entire test form is plotted.

`overlap` Logical. If `TRUE`, multiple item information functions are plotted in a single panel. If `FALSE` (default), each item information function is displayed in a separate panel.

`csee` Logical. If `TRUE`, plots the conditional standard error of estimation (CSEE) at the test level. Note that the CSEE plot is only available at the test level, not for individual items. If `FALSE` (default), item or test information functions are plotted.

`xlab.text, ylab.text` Character strings specifying the labels for the x and y axes, respectively.

`main.text` Character string specifying the overall title of the plot.

`lab.size` Numeric value specifying the font size of axis titles. Default is 15.

`main.size` Numeric value specifying the font size of the plot title. Default is 15.

`axis.size` Numeric value specifying the font size of axis tick labels. Default is 15.

`line.color` A character string specifying the color of the plot lines. See `http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/` for available color names.

`line.size` Numeric value specifying the thickness of plot lines. Default is 1.

`layout.col` Integer. Number of columns to use when faceting multiple item information functions. Used only when `overlap = FALSE`. Default is 4.

`strip.size` Numeric value specifying the font size of facet labels when multiple items are displayed.

`...` Additional arguments passed to `ggplot2::geom_line()` from the **ggplot2** package.

Details

All of the plots are drawn using the **ggplot2** package. The object of class `info` can be obtained from the function `info()`.

Value

This method function displays the item or test information function plot. When `csee = TRUE`, the CSEE is returned at the test level.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

`info()`

Examples

```
## Not run:
## Example using a "-prm.txt" file exported from flexMIRT

# Import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read the item parameters and convert them to item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# Define a sequence of theta values
theta <- seq(-4, 4, 0.1)

# Compute item and test information values for the given theta values
x <- info(x = test_flex, theta = theta, D = 1, tif = TRUE)

# Plot the test information function
plot(x)

# Plot the item information function for the second item
plot(x, item.loc = 2)

# Plot multiple item information functions, each in a separate panel
plot(x, item.loc = 1:8, overlap = FALSE)

# Plot multiple item information functions in a single panel
plot(x, item.loc = 1:8, overlap = TRUE)

# Plot the conditional standard error of estimation (CSEE) at the test level
plot(x, csee = TRUE)


## End(Not run)
```

`plot.irtfit` *Draw Raw and Standardized Residual Plots*

Description

This method provides graphical displays of the residuals between observed data and model-based predictions (Hambleton et al., 1991). For each score category of an item, it generates two types of residual plots: (a) the raw residual plot and (b) the standardized residual plot. Note that for dichotomous items, residual plots are drawn only for score category 1.

Usage

```
## S3 method for class 'irtfit'
plot(
```

```
  x,
  item.loc = NULL,
  type = "both",
  ci.method = c("wald", "wilson", "wilson.cr"),
  show.table = TRUE,
  layout.col = 2,
  xlab.text,
  ylab.text,
  main.text,
  lab.size = 15,
  main.size = 15,
  axis.size = 15,
  line.size = 1,
  point.size = 2.5,
  strip.size = 12,
  ylim.icc = c(0, 1),
  ylim.sr.adjust = FALSE,
  ylim.sr = c(-4, 4),
  ...
)
```

Arguments

x	x An object of class `irtfit` obtained from `irtfit()`.
item.loc	An integer specifying the position of the item to be plotted (i.e., the *n*th item in the item set). See **Details** below.
type	A character string indicating the type of residual plot to be displayed. Available options are: • `"icc"` for the raw residual plot • `"sr"` for the standardized residual plot • `"both"` for displaying both plots Default is `"both"`.
ci.method	A character string specifying the method used to compute confidence intervals for the raw residual plot. Available options are: • `"wald"` for the Wald method • `"wilson"` for the Wilson score interval • `"wilson.cr"` for the Wilson interval with continuity correction Default is `"wald"`. See **Details** below.
show.table	A logical value indicating whether to return the contingency table used for drawing the residual plots of the specified item. If `TRUE`, the function returns the same contingency table stored in the internal `contingency.plot` object of the `irtfit` object. Default is `TRUE`.
layout.col	An integer specifying the number of columns in the panel layout when plotting residuals for a polytomous item. Default is 2.
xlab.text	A character string specifying the title for the x-axis. If omitted, a default label is used.

ylab.text	A character string specifying the title for the y-axis. If type = "both", a character vector of length two can be provided, corresponding to the y-axis titles for the raw residual and standardized residual plots, respectively. If omitted, default labels are used.
main.text	A character string specifying the main title for the plot. If type = "both", a character vector of length two can be provided, corresponding to the main titles for the raw residual and standardized residual plots, respectively. If omitted, default titles are used.
lab.size	Numeric value specifying the font size of axis titles. Default is 15.
main.size	Numeric value specifying the font size of the plot title. Default is 15.
axis.size	Numeric value specifying the font size of axis tick labels. Default is 15.
line.size	Numeric value specifying the thickness of plot lines. Default is 1.
point.size	A numeric value specifying the size of points. Default is 2.5.
strip.size	A numeric value specifying the size of facet label text. Default is 12.
ylim.icc	A numeric vector of length two specifying the y-axis limits for the raw residual plot. Default is c(0, 1).
ylim.sr.adjust	Logical. If TRUE, the y-axis range for the standardized residual plot is automatically adjusted based on the maximum residual value for each item. If FALSE, the range is fixed according to the values specified in the ylim.sr argument. Default is FALSE.
ylim.sr	A numeric vector of length two specifying the y-axis limits for the standardized residual plot. Default is c(-4, 4).
...	Additional arguments passed to ggplot2::ggplot() from the **ggplot2** package.

Details

All plots are generated using the **ggplot2** package.

Once the IRT model fit analysis is completed using irtfit(), the resulting object of class irtfit can be used to draw raw and standardized residual plots.These plots are primarily based on the information stored in the internal object contingency.plot.

Because residual plots are generated for one item at a time, you must specify which item to evaluate by providing an integer value for the item.loc argument, indicating the item's position in the test form. For example, to draw residual plots for the third item, set item.loc = 3.

For the raw residual plot, the ci.method argument determines the method used to estimate confidence intervals. The available methods are:

- "wald": Wald interval based on the normal approximation (Laplace, 1812)
- "wilson": Wilson score interval (Wilson, 1927)
- "wilson.cr": Wilson score interval with continuity correction (Newcombe, 1998)

For more information, see https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval. Note that the width of the confidence interval is governed by the α-level specified in the alpha argument of the irtfit() function.

For the standardized residual plot, residuals exceeding the threshold specified in the overSR argument of the irtfit() function are displayed as circles. Residuals that do not exceed the threshold are displayed as crosses.

Value

This method displays the IRT raw residual plot, standardized residual plot, or both for the specified item. When `show.table = TRUE`, a contingency table used to generate the residual plots is also returned. See `irtfit()` for more details about the contingency table.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991).*Fundamentals of item response theory*. Newbury Park, CA: Sage.

Laplace, P. S. (1820).*Theorie analytique des probabilites* (in French). Courcier.

Newcombe, R. G. (1998). Two-sided confidence intervals for the single proportion: comparison of seven methods. *Statistics in medicine, 17*(8), 857-872.

Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association, 22*(158), 209-212.

See Also

`irtfit()`

Examples

```
## Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Select the first two dichotomous items and the last polytomous item
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df[c(1:2, 55), ]

# Generate examinees' abilities from N(0, 1)
set.seed(23)
score <- rnorm(1000, mean = 0, sd = 1)

# Simulate response data
data <- simdat(x = x, theta = score, D = 1)


# Compute fit statistics
fit <- irtfit(
  x = x, score = score, data = data, group.method = "equal.freq",
  n.width = 11, loc.theta = "average", range.score = c(-4, 4), D = 1,
  alpha = 0.05, overSR = 1.5
)

# Residual plots for the first item (dichotomous item)
plot(x = fit, item.loc = 1, type = "both", ci.method = "wald",
     show.table = TRUE, ylim.sr.adjust = TRUE)
```

```
# Residual plots for the third item (polytomous item)
plot(x = fit, item.loc = 3, type = "both", ci.method = "wald",
     show.table = FALSE, ylim.sr.adjust = TRUE)

# Raw residual plot for the third item (polytomous item)
plot(x = fit, item.loc = 3, type = "icc", ci.method = "wald",
     show.table = TRUE, ylim.sr.adjust = TRUE)

# Standardized residual plot for the third item (polytomous item)
plot(x = fit, item.loc = 3, type = "sr", ci.method = "wald",
     show.table = TRUE, ylim.sr.adjust = TRUE)
```

`plot.traceline` *Plot Item and Test Characteristic Curves*

Description

This method visualizes item characteristic curves (ICCs), item score curves, or the test characteristic curve (TCC) using the **ggplot2** package. ICCs or item score curves can be plotted for one or more selected items, while the TCC is plotted for the entire test form.

Usage

```
## S3 method for class 'traceline'
plot(
  x,
  item.loc = NULL,
  score.curve = FALSE,
  overlap = FALSE,
  layout.col = 2,
  xlab.text,
  ylab.text,
  main.text,
  lab.size = 15,
  main.size = 15,
  axis.size = 15,
  line.color,
  line.size = 1,
  strip.size = 12,
  ...
)
```

Arguments

x	x An object of class `traceline` obtained from `traceline()`.

item.loc	A numeric vector specifying the position(s) of the item(s) to plot. If NULL (default), the test characteristic curve (TCC) for the entire test form is plotted.
score.curve	Logical. If TRUE, plots the item score curve, defined as the weighted sum of category probabilities across score categories, in a panel. If FALSE, plots item characteristic curves (ICCs) for all score categories, either in separate panels or in a single panel depending on the overlap setting. For dichotomous items, the item score curve is equivalent to the ICC for score category 1. Ignored when item.loc = NULL. Default is FALSE.
overlap	Logical. Determines how multiple curves are displayed when plotting ICCs or item score curves. If TRUE, curves are overlaid in a single panel using different colors. If FALSE, each curve is drawn in a separate panel—either one panel per item or per score category, depending on the setting of score.curve.
layout.col	An integer value indicating the number of columns in the plot when displaying multiple panels. Used only when overlap = FALSE. Default is 2.
xlab.text, ylab.text	Character strings specifying the labels for the x and y axes, respectively.
main.text	Character string specifying the overall title of the plot.
lab.size	Numeric value specifying the font size of axis titles. Default is 15.
main.size	Numeric value specifying the font size of the plot title. Default is 15.
axis.size	Numeric value specifying the font size of axis tick labels. Default is 15.
line.color	A character string specifying the color of the plot lines. See http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/ for available color names.
line.size	Numeric value specifying the thickness of plot lines. Default is 1.
strip.size	Numeric. Font size of facet labels when ICCs are plotted.
...	Additional arguments passed to ggplot2::geom_line() from the **ggplot2** package.

Details

All plots are generated using the **ggplot2** package. If item.loc = NULL, the test characteristic curve (TCC) for the entire test form is plotted. If item.loc is specified, it should be a vector of positive integers indicating the position(s) of the items to be plotted. For example, if the test form includes ten items and you wish to plot the score curves of the 1st, 2nd, and 3rd items, set item.loc = 1:3.

Value

This method displays item characteristic curves (ICCs), item score curves, or the test characteristic curve (TCC), depending on the specified arguments.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

traceline()

Examples

```
## Example using a "-prm.txt" file exported from flexMIRT

# Import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read the item parameters and convert them to item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# Define a sequence of theta values
theta <- seq(-3, 3, 0.1)

# Compute item category probabilities and item/test characteristic functions
x <- traceline(x = test_flex, theta, D = 1)

# Plot the test characteristic curve (TCC) for the full test form
plot(x, item.loc = NULL)


# Plot ICCs for the first item (dichotomous),
# with a separate panel for each score category
plot(x, item.loc = 1, score.curve = FALSE, layout.col = 2)

# Plot ICCs for the first item in a single panel
# (all score categories overlaid)
plot(x, item.loc = 1, score.curve = FALSE, overlap = TRUE)

# Plot ICCs for multiple items (both dichotomous and polytomous),
# with each item's ICCs shown in a single panel
plot(x, item.loc = c(1:3, 53:55), score.curve = FALSE, overlap = TRUE)

# Plot the item score curve for the first item (dichotomous)
plot(x, item.loc = 1, score.curve = TRUE)

# Plot item score curves for the first six dichotomous items
# using multiple panels
plot(x, item.loc = 1:6, score.curve = TRUE, overlap = FALSE)

# Plot item score curves for the first six dichotomous items
# overlaid in a single panel
plot(x, item.loc = 1:6, score.curve = TRUE, overlap = TRUE)

# Plot ICCs for the last item (polytomous),
# with each score category in a separate panel
plot(x, item.loc = 55, score.curve = FALSE, layout.col = 2)

# Plot the item score curve for the last item (polytomous)
plot(x, item.loc = 55, score.curve = TRUE)
```

```
# Plot item score curves for the last three polytomous items
# using multiple panels
plot(x, item.loc = 53:55, score.curve = TRUE, overlap = FALSE)

# Plot item score curves for the last three polytomous items
# overlaid in a single panel
plot(x, item.loc = 53:55, score.curve = TRUE, overlap = TRUE)
```

---

prm *Polytomous Response Model (PRM) Probabilities (GRM and GPCM)*

Description

This function computes the probability of selecting each response category for an item, given a set of theta values, using the graded response model (GRM) or the (generalized) partial credit model (GPCM).

Usage

```
prm(theta, a, d, D = 1, pr.model = c("GRM", "GPCM"))
```

Arguments

theta	A numeric vector of ability values (latent traits).
a	A numeric vector of item discrimination (slope) parameters.
d	A numeric vector of item difficulty (or threshold) parameters.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
pr.model	A character string specifying the polytomous IRT model. Available options are "GRM" for the graded response model and "GPCM" for the (generalized) partial credit model.

Details

When computing category probabilities using the partial credit model (PCM), set a = 1.

For pr.model = "GPCM", the vector d should contain threshold parameters that define the boundaries between adjacent score categories. In the **irtQ** package, these thresholds are expressed as the item location (overall difficulty) minus the step parameters for each category. If an item has *K* score categories, *K - 1* threshold parameters must be provided; the first is assumed to be 0. For example, for a GPCM item with five categories, provide four threshold parameters.

For more details on the parameterization of the (generalized) partial credit model, refer to the *IRT Models* section in the irtQ-package documentation.

Value

A matrix of category response probabilities, where each row corresponds to a theta value and each column represents a score category of the item.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

drm()

Examples

```
## Category probabilities for an item with four categories
## using the generalized partial credit model (GPCM)
prm(theta = c(-0.2, 0, 0.5), a = 1.4, d = c(-0.2, 0, 0.5), D = 1, pr.model = "GPCM")

## Category probabilities for an item with five categories
## using the graded response model (GRM)
prm(theta = c(-0.2, 0, 0.5), a = 1.2, d = c(-0.4, -0.2, 0.4, 1.5), D = 1, pr.model = "GRM")
```

---

rdif *IRT Residual-Based Differential Item Functioning (RDIF) Detection Framework*

Description

This function computes three RDIF statistics for each item: $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$ (Lim & Choe, 2023; Lim, et al., 2022). $RDIF_R$ primarily captures differences in raw residuals between two groups, which are typically associated with uniform DIF. $RDIF_S$ primarily captures differences in squared residuals, which are typically associated with nonuniform DIF. $RDIF_{RS}$ jointly considers both types of differences and is capable of detecting both uniform and nonuniform DIF.

Usage

```
rdif(x, ...)

## Default S3 method:
rdif(
  x,
  data,
  score = NULL,
  group,
  focal.name,
  item.skip = NULL,
```

```
  D = 1,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("rdifrs", "rdifr", "rdifs"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'est_irt'
rdif(
  x,
  score = NULL,
  group,
  focal.name,
  item.skip = NULL,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("rdifrs", "rdifr", "rdifs"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'est_item'
rdif(
  x,
  group,
  focal.name,
  item.skip = NULL,
  alpha = 0.05,
  missing = NA,
```

```
  purify = FALSE,
  purify.by = c("rdifrs", "rdifr", "rdifs"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class `est_irt` obtained from `est_irt()`, or `est_item` from `est_item()`. See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
...	Additional arguments passed to the `est_score()` function.
data	A matrix of examinees' item responses corresponding to the items specified in the `x` argument. Rows represent examinees and columns represent items.
score	A numeric vector containing examinees' ability estimates (theta values). If not provided, `rdif()` will estimate ability parameters internally before computing the RDIF statistics. See `est_score()` for more information on scoring methods. Default is `NULL`.
group	A numeric or character vector indicating examinees' group membership. The length of the vector must match the number of rows in the response data matrix.
focal.name	A single numeric or character value specifying the focal group. For instance, given `group = c(0, 1, 0, 1, 1)` and '1' indicating the focal group, set `focal.name = 1`.
item.skip	A numeric vector of item indices to exclude from DIF analysis. If `NULL`, all items are included. Useful for omitting specific items based on prior insights.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
alpha	A numeric value specifying the significance level (α) for hypothesis testing using the RDIF statistics. Default is `0.05`.
missing	A value indicating missing responses in the data set. Default is `NA`.
purify	Logical. Indicates whether to apply a purification procedure. Default is `FALSE`.
purify.by	A character string specifying which RDIF statistic is used to perform the purification. Available options are "rdifrs" for $RDIF_{RS}$, "rdifr" for $RDIF_{R}$, and "rdifs" for $RDIF_{S}$.

`max.iter` A positive integer specifying the maximum number of iterations allowed for the purification process. Default is `10`.

`min.resp` A positive integer specifying the minimum number of valid item responses required from an examinee in order to compute an ability estimate. Default is `NULL`. See **Details** for more information.

`method` A character string indicating the scoring method to use. Available options are:

- `"ML"`: Maximum likelihood estimation
- `"WL"`: Weighted likelihood estimation (Warm, 1989)
- `"MAP"`: Maximum a posteriori estimation (Hambleton et al., 1991)
- `"EAP"`: Expected a posteriori estimation (Bock & Mislevy, 1982)

Default is `"ML"`.

`range` A numeric vector of length two specifying the lower and upper bounds of the ability scale. This is used for the following scoring methods: `"ML"`, `"WL"`, and `"MAP"`. Default is `c(-5, 5)`.

`norm.prior` A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. These values are used to generate the Gaussian quadrature points and weights. Ignored if `method` is `"ML"` or `"WL"`. Default is `c(0, 1)`.

`nquad` An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. Used only when `method` is `"EAP"`. Ignored for `"ML"`, `"WL"`, and `"MAP"`. Default is 41.

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and their corresponding weights (in the second column) for the latent variable prior distribution. The weights and points can be conveniently generated using the function `gen.weight()`.

If `NULL` and `method = "EAP"`, default quadrature values are generated based on the `norm.prior` and `nquad` arguments. Ignored if `method` is `"ML"`, `"WL"`, or `"MAP"`.

`ncore` An integer specifying the number of logical CPU cores to use for parallel processing. Default is `1`. See `est_score()` for details.

`verbose` Logical. If `TRUE`, progress messages from the purification procedure will be displayed; if `FALSE`, the messages will be suppressed. Default is `TRUE`.

Details

The RDIF framework (Lim & Choe, 2023; Lim et al., 2022) consists of three IRT residual-based statistics: $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$. Under the null hypothesis that a test contains no DIF items, $RDIF_R$ and $RDIF_S$ asymptotically follow standard normal distributions. $RDIF_{RS}$ is based on a bivariate normal distribution of the $RDIF_R$ and $RDIF_S$ statistics, and under the null hypothesis, it asymptotically follows a χ^2 distribution with 2 degrees of freedom. See Lim et al. (2022) for more details about the RDIF framework.

The `rdif()` function computes all three RDIF statistics: $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$. The current version of `rdif()` supports both dichotomous and polytomous item response data. Note that for polytomous items, net DIF are assessed. To evaluate global DIF for polytomous items, use `crdif()` function.

To compute the RDIF statistics, the `rdif()` function requires: (1) item parameter estimates obtained from aggregate data (regardless of group membership), (2) examinees' ability estimates (e.g., ML), and (3) examinees' item response data. Note that the ability estimates must be based on the aggregate-data item parameters. The item parameter estimates should be provided in the `x` argument, the ability estimates in the `score` argument, and the response data in the `data` argument. If ability estimates are not provided (i.e., `score = NULL`), `rdif()` will estimate them automatically using the scoring method specified via the `method` argument (e.g., `method = "ML"`).

The `group` argument should be a vector containing exactly two distinct values (either numeric or character), representing the reference and focal groups. Its length must match the number of rows in the response data, where each element corresponds to an examinee. Once `group` is specified, a single numeric or character value must be provided in the `focal.name` argument to indicate which level in `group` represents the focal group.

Similar to other DIF detection approaches, the RDIF framework supports an iterative purification process. When `purify = TRUE`, purification is conducted using one of the RDIF statistics specified in the `purify.by` argument (e.g., `purify.by = "rdifrs"`). At each iteration, examinees' ability estimates are recalculated based on the set of purified items using the scoring method specified in the `method` argument. The purification process continues until no additional DIF items are identified or the maximum number of iterations specified in `max.iter` is reached. See Lim et al. (2022) for more details on the purification procedure.

Scoring based on a small number of item responses can lead to large standard errors, potentially reducing the accuracy of DIF detection in the RDIF framework. The `min.resp` argument can be used to exclude examinees with insufficient response data from scoring, especially during the purification process. For example, if `min.resp` is not NULL (e.g., `min.resp = 5`), examinees who responded to fewer than five items will have all their responses treated as missing (i.e., NA). As a result, their ability estimates will also be missing and will not be used in the computation of RDIF statistics. If `min.resp = NULL`, a score will be computed for any examinee with at least one valid item response.

Value

This function returns a list containing four main components:

`no_purify` A list of sub-objects containing the results of DIF analysis without applying a purification procedure. The sub-objects include:

- **dif_stat** A data frame summarizing the RDIF analysis results for all items. The columns include: item ID, $RDIF_R$ statistic, standardized $RDIF_R$, $RDIF_S$ statistic, standardized $RDIF_S$, $RDIF_{RS}$ statistic, p-values for $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$, sample sizes for the reference and focal groups, and total sample size. Note that $RDIF_{RS}$ does not have a standardized value because it is a χ^2-based statistic.
- **moments** A data frame reporting the first and second moments of the RDIF statistics. The columns include: item ID, mean and standard deviation of $RDIF_R$, mean and standard deviation of $RDIF_S$, and the covariance between $RDIF_R$ and $RDIF_S$.
- **dif_item** A list of three numeric vectors identifying items flagged as DIF by each RDIF statistic: $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$.
- **score** A numeric vector of ability estimates used to compute the RDIF statistics.

`purify` A logical value indicating whether the purification procedure was applied.

`with_purify` A list of sub-objects containing the results of DIF analysis with a purification procedure. The sub-objects include:

purify.by A character string indicating the RDIF statistic used for purification. Possible values are "rdifr", "rdifs", and "rdifrs", corresponding to $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$, respectively.

dif_stat A data frame reporting the RDIF analysis results for all items across the final iteration. Same structure as in `no_purify`, with one additional column indicating the iteration number in which each result was obtained.

moments A data frame reporting the moments of RDIF statistics across the final iteration. Includes the same columns as in `no_purify`, with an additional column for the iteration number.

dif_item A list of three numeric vectors identifying DIF items flagged by each RDIF statistic.

n.iter An integer indicating the total number of iterations performed during the purification process.

score A numeric vector of purified ability estimates used to compute the final RDIF statistics.

complete A logical value indicating whether the purification process converged. If FALSE, the maximum number of iterations was reached without convergence.

`alpha` A numeric value indicating the significance level (α) used in hypothesis testing for RDIF statistics.

Methods (by class)

- `rdif(default)`: Default method for computing the three RDIF statistics using a data frame `x` that contains item metadata
- `rdif(est_irt)`: An object created by the function `est_irt()`.
- `rdif(est_item)`: An object created by the function `est_item()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Lim, H., & Choe, E. M. (2023). Detecting differential item functioning in CAT using IRT residual DIF approach. *Journal of Educational Measurement, 60*(4), 626-650. doi:10.1111/jedm.12366.

Lim, H., Choe, E. M., & Han, K. T. (2022). A residual-based differential item functioning detection framework in item response theory. *Journal of Educational Measurement, 59*(1), 80-104. doi:10.1111/jedm.12313.

See Also

`est_irt()`, `est_item()`, `simdat()`, `shape_df()`, `est_score()`

Examples

```
# Load required package
library("dplyr")

## Uniform DIF detection
################################################
# (1) Generate data with known uniform DIF
################################################

# Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Select 36 non-DIF items using the 3PLM model
par_nstd <-
  bring.flexmirt(file = flex_sam, "par")$Group1$full_df %>%
  dplyr::filter(.data$model == "3PLM") %>%
  dplyr::filter(dplyr::row_number() %in% 1:36) %>%
  dplyr::select(1:6)
par_nstd$id <- paste0("nondif", 1:36)

# Generate 4 new DIF items for the reference group
difpar_ref <-
  shape_df(
    par.drm = list(a = c(0.8, 1.5, 0.8, 1.5), b = c(0.0, 0.0, -0.5, -0.5), g = 0.15),
    item.id = paste0("dif", 1:4), cats = 2, model = "3PLM"
  )

# Add uniform DIF by shifting the b-parameters for the focal group
difpar_foc <-
  difpar_ref %>%
  dplyr::mutate_at(.vars = "par.2", .funs = function(x) x + rep(0.7, 4))

# Combine the DIF and non-DIF items for both reference and focal groups
# Therefor, the first 4 items exhibit uniform DIF
par_ref <- rbind(difpar_ref, par_nstd)
par_foc <- rbind(difpar_foc, par_nstd)

# Generate true ability values
set.seed(123)
theta_ref <- rnorm(500, 0.0, 1.0)
theta_foc <- rnorm(500, 0.0, 1.0)

# Simulate response data
resp_ref <- simdat(par_ref, theta = theta_ref, D = 1)
resp_foc <- simdat(par_foc, theta = theta_foc, D = 1)
data <- rbind(resp_ref, resp_foc)

################################################
# (2) Estimate item and ability parameters
#     from the combined response data
################################################
```

```
# Estimate item parameters
est_mod <- est_irt(data = data, D = 1, model = "3PLM")
est_par <- est_mod$par.est

# Estimate ability parameters using ML
score <- est_score(x = est_par, data = data, method = "ML")$est.theta

################################################
# (3) Perform DIF analysis
################################################

# Define group membership: 1 = focal group
group <- c(rep(0, 500), rep(1, 500))

# (a)-1 Compute RDIF statistics with provided ability scores
#       (no purification)
dif_nopuri_1 <- rdif(
  x = est_par, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05
)
print(dif_nopuri_1)

# (a)-2 Compute RDIF statistics without providing ability scores
#       (no purification)
dif_nopuri_2 <- rdif(
  x = est_par, data = data, score = NULL,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  method = "ML"
)
print(dif_nopuri_2)

# (b)-1 Compute RDIF statistics with purification based on RDIF(R)
dif_puri_r <- rdif(
  x = est_par, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "rdifr"
)
print(dif_puri_r)

# (b)-2 Compute RDIF statistics with purification based on RDIF(S)
dif_puri_s <- rdif(
  x = est_par, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "rdifs"
)
print(dif_puri_s)

# (b)-3 Compute RDIF statistics with purification based on RDIF(RS)
dif_puri_rs <- rdif(
  x = est_par, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "rdifrs"
)
```

```
print(dif_puri_rs)
```

`reval_mst` *Recursion-based MST evaluation method*

Description

This function evaluates the measurement precision and bias in Multistage-adaptive Test (MST) panels using a recursion-based evaluation method introduced by Lim et al. (2021). This function computes conditional biases and standard errors of measurement (CSEMs) across a range of IRT ability levels, facilitating efficient and accurate MST panel assessments without extensive simulations.

Usage

```
reval_mst(
  x,
  D = 1,
  route_map,
  module,
  cut_score,
  theta = seq(-5, 5, 1),
  intpol = TRUE,
  range.tcc = c(-7, 7),
  tol = 1e-04
)
```

Arguments

`x`	A data frame containing the metadata for the item bank, which includes important information for each item such as the number of score categories and the IRT model applied. This metadata is essential for evaluating the MST panel, with items selected based on the specifications in the `module` argument. To construct this item metadata efficiently, the `shape_df()` function is recommended. Further details on utilizing item bank metadata along with `module` for MST panel evaluation are provided below.
`D`	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
`route_map`	A binary square matrix that defines the MST structure, illustrating transitions between modules and stages. This concept and structure are inspired by the `transMatrix` argument in the `randomMST()` function from the **mstR** package (Magis et al., 2017), which provides a framework for representing MST pathways. For comprehensive understanding and examples of constructing `route_map`, refer to the **mstR** package (Magis et al., 2017) documentation. Also see below for details.

`module` A binary matrix that maps items from the item bank specified in `x` to modules within the MST framework. This parameter's structure is analogous to the `modules` argument in the `randomMST()` function of the **mstR** package, enabling precise item-to-module assignments for MST configurations. For detailed instructions on creating and utilizing the `module` matrix effectively, consult the documentation of the **mstR** package (Magis et al., 2017). Also see below for details.

`cut_score` A list defining cut scores for routing test takers through MST stages. Each list element is a vector of cut scores for advancing participants to subsequent stage modules. In a 1-3-3 MST configuration, for example, `cut_score` might be defined as `cut_score = list(c(-0.5, 0.5), c(-0.6, 0.6))`, where `c(-0.5, 0.5)` are thresholds for routing from the first to the second stage, and `c(-0.6, 0.6)` for routing from the second to the third stage. This setup facilitates tailored test progression based on performance. Further examples and explanations are available below.

`theta` A vector of ability levels (theta) at which the MST panel's performance is assessed. This allows for the evaluation of measurement precision and bias across a continuum of ability levels. The default range is `theta = seq(-5, 5, 0.1)`.

`intpol` A logical value to enable linear interpolation in the inverse test characteristic curve (TCC) scoring, facilitating ability estimate approximation for observed sum scores not directly obtainable from the TCC, such as those below the sum of item guessing parameters. Default is TRUE, applying interpolation to bridge gaps in the TCC. Refer to `est_score()` for more details and consult Lim et al. (2021) for insights into the interpolation technique within inverse TCC scoring.

`range.tcc` A vector to define the range of ability estimates for inverse TCC scoring, expressed as the two numeric values for lower and upper bounds. Default is to c(-7, 7).

`tol` A numeric value of the convergent tolerance for the inverse TCC scoring. For the inverse TCC scoring, the bisection method is used for optimization. Default is 1e-4.

Details

The `reval_mst()` function evaluates an MST panel by implementing a recursion-based method to assess measurement precision across IRT ability levels. This approach, detailed in Lim et al. (2021), enables the computation of conditional biases and CSEMs efficiently, bypassing the need for extensive simulations traditionally required for MST evaluation.

The `module` argument, used in conjunction with the item bank metadata `x`, systematically organizes items into modules for MST panel evaluation. Each row of `x` corresponds to an item, detailing its characteristics like score categories and IRT model. The `module` matrix, structured with the same number of rows as `x` and columns representing modules, indicates item assignments with 1s. This precise mapping enables the `reval_mst()` function to evaluate the MST panel's performance by analyzing how items within each module contribute to measurement precision and bias, reflecting the tailored progression logic inherent in MST designs.

The `route_map` argument is essential for defining the MST's structure by indicating possible transitions between modules. Similar to the `transMatrix()` in the **mstR** package (Magis et al., 2017), `route_map` is a binary matrix that outlines which module transitions are possible within an MST

design. Each "1" in the matrix represents a feasible transition from one module (row) to another (column), effectively mapping the flow of test takers through the MST based on their performance. For instance, a "1" at the intersection of row *i* and column *j* indicates the possibility for test takers to progress from the module corresponding to row *i* directly to the module denoted by column *j*. This structure allows `reval_mst()` to simulate and evaluate the dynamic routing of test takers through various stages and modules of the MST panel.

To further detail the `cut_score` argument with an illustration: In a 1-3-3 MST configuration, the list `cut_score = list(c(-0.5, 0.5), c(-0.6, 0.6))` operates as a decision guide at each stage. Initially, all test takers start in the first module. Upon completion, their scores determine their next stage module: scores below -0.5 route to the first module of the next stage, between -0.5 and 0.5 to the second, and above 0.5 to the third. This pattern allows for dynamic adaptation, tailoring the test path to individual performance levels.

Value

This function returns a list of seven internal objects. The four objects are:

`panel.info` A list of several sub-objects containing detailed information about the MST panel configuration, including:

- **config** A nested list indicating the arrangement of modules across stages, showing which modules are included in each stage. For example, the first stage includes module 1, the second stage includes modules 2 to 4, and so forth.
- **pathway** A matrix detailing all possible pathways through the MST panel. Each row represents a unique path a test taker might take, based on their performance and the cut scores defined.
- **n.module** A named vector indicating the number of modules available at each stage.
- **n.stage** A single numeric value representing the total number of stages in the MST panel.

`item.by.mod` A list where each entry represents a module in the MST panel, detailing the item metadata within that module. Each module's metadata includes item IDs, the number of categories, the IRT model used (model), and the item parameters (e.g., par.1, par.2, par.3).

`item.by.path` A list containing item metadata arranged according to the paths through the MST structure. This detailed breakdown allows for an analysis of item characteristics along specific MST paths. Each list entry corresponds to a testing stage and path, providing item metadata. This structure facilitates the examination of how items function within the context of each unique path through the MST.

`eq.theta` Estimated ability levels (θ) corresponding to the observed scores, derived from the inverse TCC scoring method. This provides the estimated θ values for each potential pathway through the MST stages. For each stage, θ values are calculated for each path, indicating the range of ability levels across the test takers. For instance, in a three-stage MST, the `eq.theta` list may contain θ estimates for multiple paths within each stage, reflecting the progression of ability estimates as participants move through the test. The example below illustrates the structure of `eq.theta` output for a 1-3-3 MST panel with varying paths:

stage.1 `path.1` shows θ estimates ranging from -7 to +7, demonstrating the initial spread of abilities.

stage.2 Multiple paths (`path.1`, `path.2`, ...) each with their own θ estimates, indicating divergence in ability levels based on test performance.

stage.3 Further refinement of θ estimates across paths, illustrating the final estimation of abilities after the last stage.

`cdist.by.mod` A list where each entry contains the conditional distributions of the observed scores for each module given the ability levels.

`jdist.by.path` Joint distributions of observed scores for different paths at each stage in a MST panel. The example below outlines the organization of `jdist.by.path` data in a hypothetical 1-3-3 MST panel:

stage.1 Represents the distribution at the initial stage, indicating the broad spread of test-taker abilities at the outset.

stage.2 Represents the conditional joint distributions of the observed scores as test-takers move through different paths at the stage 2, based on their performance in earlier stages.

stage.3 Represents a further refinement of joint distribution of observed scores as test-takers move through different paths at the final stage 3, based on their performance in earlier stages.

`eval.tb` A table summarizing the measurement precision of the MST panel. It contains the true ability levels (`theta`) with the mean ability estimates (`mu`), variance (`sigma2`), bias, and conditional standard error of measurement (CSEM) given the true ability levels. This table highlights the MST panel's accuracy and precision across different ability levels, providing insights into its effectiveness in estimating test-taker abilities.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Magis, D., Yan, D., & Von Davier, A. A. (2017). *Computerized adaptive and multistage testing with R: Using packages catR and mstR*. Springer.

Lim, H., Davey, T., & Wells, C. S. (2021). A recursion-based analytical approach to evaluate the performance of MST. *Journal of Educational Measurement, 58*(2), 154-178.

See Also

`shape_df()`, `est_score()`

Examples

```
## ------------------------------------------------------------------------------
# Evaluation of a 1-3-3 MST panel using simMST data.
# This simulation dataset was utilized in Lim et al.'s (2021) simulation study.
# Details:
#    (a) Panel configuration: 1-3-3 MST panel
```

```
#    (b) Test length: 24 items (each module contains 8 items across all stages)
#    (c) IRT model: 3-parameter logistic model (3PLM)
## ----------------------------------------------------------------------------
# Load the necessary library
library(dplyr)
library(tidyr)
library(ggplot2)

# Import item bank metadata
x <- simMST$item_bank

# Import module information
module <- simMST$module

# Import routing map
route_map <- simMST$route_map

# Import cut scores for routing to subsequent modules
cut_score <- simMST$cut_score

# Import ability levels (theta) for evaluating measurement precision
theta <- simMST$theta

# Evaluate MST panel using the reval_mst() function
eval <-
  reval_mst(x,
    D = 1.702, route_map = route_map, module = module,
    cut_score = cut_score, theta = theta, range.tcc = c(-5, 5)
  )

# Review evaluation results
# The evaluation result table below includes conditional biases and
# standard errors of measurement (CSEMs) across ability levels
print(eval$eval.tb)

# Generate plots for biases and CSEMs
p_eval <-
  eval$eval.tb %>%
  dplyr::select(theta, bias, csem) %>%
  tidyr::pivot_longer(
    cols = c(bias, csem),
    names_to = "criterion", values_to = "value"
  ) %>%
  ggplot2::ggplot(mapping = ggplot2::aes(x = theta, y = value)) +
  ggplot2::geom_point(mapping = ggplot2::aes(shape = criterion), size = 3) +
  ggplot2::geom_line(
    mapping = ggplot2::aes(
      color = criterion,
      linetype = criterion
    ),
    linewidth = 1.5
  ) +
  ggplot2::labs(x = expression(theta), y = NULL) +
```

```
  ggplot2::theme_classic() +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.key.width = unit(1.5, "cm"))
print(p_eval)
```

---

ripd *Residual-based Item Parameter Drift (RIPD) Detection Framework*

Description

This function computes three RIPD statistics—$RIPD_R$, $RIPD_S$, and $RIPD_{RS}$—for each item. $RIPD_R$ captures differences in mean raw residuals between groups, which is typically indicative of uniform item parameter drift (IPD). $RIPD_S$ captures differences in mean squared residuals between groups, reflecting nonuniform IPD. $RIPD_{RS}$, a combined chi-square-based index, is sensitive to both uniform and nonuniform IPD.

Usage

```
ripd(x, ...)

## Default S3 method:
ripd(
  x,
  data,
  score = NULL,
  group,
  focal.name,
  item.skip = NULL,
  D = 1,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("ripdrs", "ripdr", "ripds"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)
```

```
## S3 method for class 'est_irt'
ripd(
  x,
  score = NULL,
  group,
  focal.name,
  item.skip = NULL,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("ripdrs", "ripdr", "ripds"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'est_item'
ripd(
  x,
  group,
  focal.name,
  item.skip = NULL,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("ripdrs", "ripdr", "ripds"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class `est_irt` obtained from

est_irt(), or est_item from est_item().

See est_irt() or simdat() for more details about the item metadata. This data frame can be easily created using the shape_df() function.

`...` Additional arguments passed to the est_score() function.

`data` A matrix of examinees' item responses corresponding to the items specified in the x argument. Rows represent examinees and columns represent items.

`score` A numeric vector containing examinees' ability estimates (theta values). If not provided, ripd() will estimate ability parameters internally before computing the RIPD statistics. See est_score() for more information on scoring methods. Default is NULL.

`group` A numeric or character vector indicating group membership of examinees. The length of vector should the same with the number of rows in the response data matrix.

`focal.name` A single numeric or character value specifying the focal group. For instance, given group = c(0, 1, 0, 1, 1) and '1' indicating the focal group, set focal.name = 1.

`item.skip` A numeric vector of item indices to exclude from IPD analysis. If NULL, all items are included. Useful for omitting specific items based on prior insights.

`D` A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.

`alpha` A numeric value specifying the significance level (α) for hypothesis testing using the RIPD statistics. Default is 0.05.

`missing` A value indicating missing values in the response data set. Default is NA.

`purify` Logical. Indicates whether to apply a purification procedure. Default is FALSE.

`purify.by` A character string specifying which RIPD statistic is used to perform the purification. Available options are "ripdrs" for $RIPD_{RS}$, "ripdr" for $RIPD_R$, and "ripds" for $RIPD_S$.

`max.iter` A positive integer specifying the maximum number of iterations allowed for the purification process. Default is 10.

`min.resp` A positive integer specifying the minimum number of valid item responses required from an examinee in order to compute an ability estimate. Default is NULL.

`method` A character string indicating the scoring method to use. Available options are:

- "ML": Maximum likelihood estimation
- "WL": Weighted likelihood estimation (Warm, 1989)
- "MAP": Maximum a posteriori estimation (Hambleton et al., 1991)
- "EAP": Expected a posteriori estimation (Bock & Mislevy, 1982)

Default is "ML".

`range` A numeric vector of length two specifying the lower and upper bounds of the ability scale. This is used for the following scoring methods: "ML", "WL", and "MAP". Default is c(-5, 5).

`norm.prior` A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. These values are used to generate the Gaussian quadrature points and weights. Ignored if `method` is `"ML"` or `"WL"`. Default is `c(0, 1)`.

`nquad` An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. Used only when `method` is `"EAP"`. Ignored for `"ML"`, `"WL"`, and `"MAP"`. Default is 41.

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and their corresponding weights (in the second column) for the latent variable prior distribution. The weights and points can be conveniently generated using the function `gen.weight()`.

If `NULL` and `method = "EAP"`, default quadrature values are generated based on the `norm.prior` and `nquad` arguments. Ignored if `method` is `"ML"`, `"WL"`, or `"MAP"`.

`ncore` An integer specifying the number of logical CPU cores to use for parallel processing. Default is `1`. See `est_score()` for details.

`verbose` Logical. If `TRUE`, progress messages from the purification procedure will be displayed; if `FALSE`, the messages will be suppressed. Default is `TRUE`.

Value

This function returns a list containing four main components:

`no_purify` A list of sub-objects containing the results of IPD analysis without applying a purification procedure. The sub-objects include:

- **ipd_stat** A data frame summarizing the RIPD analysis results for all items. The columns include: item ID, $RIPD_R$ statistic, standardized $RIPD_R$, $RIPD_S$ statistic, standardized $RIPD_S$, $RIPD_{RS}$ statistic, p-values for $RIPD_R$, $RIPD_S$, and $RIPD_{RS}$, sample sizes for the reference and focal groups, and total sample size. Note that $RIPD_{RS}$ does not have a standardized value because it is a χ^2-based statistic.
- **moments** A data frame reporting the first and second moments of the RIPD statistics. The columns include: item ID, mean and standard deviation of $RIPD_R$, mean and standard deviation of $RIPD_S$, and the covariance between $RIPD_R$ and $RIPD_S$.
- **ipd_item** A list of three numeric vectors identifying items flagged as drifting by each RIPD statistic: $RIPD_R$, $RIPD_S$, and $RIPD_{RS}$.
- **score** A numeric vector of ability estimates used to compute the RIPD statistics.

`purify` A logical value indicating whether the purification procedure was applied.

`with_purify` A list of sub-objects containing the results of IPD analysis with a purification procedure. The sub-objects include:

- **purify.by** A character string indicating the RIPD statistic used for purification. Possible values are "ripdr", "ripds", and "ripdrs", corresponding to $RIPD_R$, $RIPD_S$, and $RIPD_{RS}$, respectively.
- **ipd_stat** A data frame reporting the RIPD analysis results for all items from the final iteration. Same structure as in `no_purify`, with one additional column indicating the iteration number in which each result was obtained.

moments A data frame reporting the moments of RIPD statistics from the final iteration. Includes the same columns as in `no_purify`, with an additional column for the iteration number.

ipd_item A list of numeric item indices identified as IPD items across iterations.

n.iter An integer indicating the total number of iterations performed during the purification process.

score A numeric vector of purified ability estimates used to compute the final RIPD statistics.

complete A logical value indicating whether the purification process converged. If `FALSE`, the maximum number of iterations was reached without convergence.

`alpha` A numeric value indicating the significance level (α) used in hypothesis testing for RIPD statistics.

Methods (by class)

- `ripd(default)`: Default method for computing the three RIPD statistics using a data frame `x` that contains item metadata
- `ripd(est_irt)`: An object created by the function `est_irt()`.
- `ripd(est_item)`: An object created by the function `est_item()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Lim, H., Choe, E. M., & Han, K. T. (2022). A residual-based differential item functioning detection framework in item response theory. *Journal of Educational Measurement, 59*(1), 80-104. doi:10.1111/jedm.12313.

Lim, H., & H, K. T. (2025, April). *IRT residual-based approach to detecting item parameter drift in CAT*. Paper presented at the annual conference of the National Council on Measurement in Education (NCME), Denver, CO

See Also

`rdif()`, `est_irt()`, `est_item()`, `simdat()`, `shape_df()`, `est_score()`

run_flexmirt *Run flexMIRT from Within R*

Description

This function runs flexMIRT (Cai, 2017) from within R by executing a model specified in a flexMIRT syntax file (i.e., *.flexmirt). To use this function, the flexMIRT software must be installed on your system. This interface is especially useful for conducting simulation studies or automating batch analyses involving flexMIRT.

Usage

```
run_flexmirt(file.syntax, dir.flex = NULL, show.output.on.console = FALSE, ...)
```

Arguments

file.syntax	A single string or character vector specifying the path(s) to one or more flexMIRT syntax files (with extension *.flexmirt) to be executed. For example: "C:/Users/Data/irtmodel.flexmi
dir.flex	A character string specifying the directory where flexMIRT is installed. The folder name typically includes "flexMIRT" (e.g., "flexMIRT3", "flexMIRT 3.6"). If set to NULL, the function searches for flexMIRT in "C:/Program Files" and uses a default path if found (e.g., "C:/Program Files/flexMIRT3").
show.output.on.console	Logical. If TRUE, the output of the system command is printed to the R console. Default is FALSE. See base::system().
...	Additional arguments passed to base::system().

Details

When using a version of flexMIRT earlier than 3.6, the directory specified in dir.flex must contain the following six files:

- WinFlexMIRT.exe
- FlexMIRT_x64.exe
- FlexMIRT_x86.exe
- vpg.dll
- vpg.licensing.client.dll
- vpg.licensing.dll

For flexMIRT version 3.6 or later, the directory must include the following five files:

- WinFlexMIRT.exe
- vpg.dll
- vpg.licensing.client.dll
- vpg.licensing.dll

- VPGLicenseClientNet.dll

along with a subdirectory named Resources that contains the following two files:

- flexMIRT_x64_AVX.exe
- flexMIRT_x86_AVX.exe

Value

Output files generated by flexMIRT.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Cai, L. (2017). flexMIRT 3.5 Flexible multilevel multidimensional item analysis and test scoring (Computer Software). Chapel Hill, NC: Vector Psychometric Group.

Examples

```
# Examples below will run if the flexMIRT software is installed
# in the default directory "C:/Program Files/flexMIRT3".
# Otherwise, specify the directory where flexMIRT is installed
# using the 'dir.flex' argument.

## Not run:
# (1) Run a single syntax file
# Load an example flexMIRT syntax file for estimating item parameters using the 2PL model
file.syntax <- system.file("extdata", "2PLM_example.flexmirt", package = "irtQ")

# Run flexMIRT to estimate item parameters for the 2PL model
run_flexmirt(file.syntax = file.syntax, dir.flex = NULL, show.output = TRUE)

# Check the output file
out.file <- system.file("extdata", "2PLM_example-prm.txt", package = "irtQ")
bring.flexmirt(out.file, type = "par")

# (2) Run multiple syntax files
# Load two example flexMIRT syntax files
file.syntax1 <- system.file("extdata", "2PLM_example.flexmirt", package = "irtQ")
file.syntax2 <- system.file("extdata", "3PLM_example.flexmirt", package = "irtQ")

# Run flexMIRT to estimate item parameters for both models
run_flexmirt(file.syntax = c(file.syntax1, file.syntax2), dir.flex = NULL, show.output = FALSE)

# Check the output files
out.file1 <- system.file("extdata", "2PLM_example-prm.txt", package = "irtQ")
out.file2 <- system.file("extdata", "3PLM_example-prm.txt", package = "irtQ")
bring.flexmirt(out.file1, type = "par")
bring.flexmirt(out.file2, type = "par")
```

```
## End(Not run)
```

---

shape_df *Create a Data Frame of Item Metadata*

Description

This function creates a data frame of item metadata—including item parameters, the number of score categories, and IRT model specifications—to be used in various IRT-related analyses within the **irtQ** package.

Usage

```
shape_df(
  par.drm = list(a = NULL, b = NULL, g = NULL),
  par.prm = list(a = NULL, d = NULL),
  item.id = NULL,
  cats,
  model,
  default.par = FALSE
)
```

Arguments

par.drm	A list containing three numeric vectors for dichotomous item parameters: item discrimination (a), item difficulty (b), and guessing parameters (g).
par.prm	A list containing polytomous item parameters. The list must include a numeric vector a for item discrimination (slope) parameters, and a list d of numeric vectors specifying difficulty (or threshold) parameters for each item. See the **Details** section for more information.
item.id	A character vector of item IDs. If NULL, default IDs (e.g., "V1", "V2", ...) are assigned automatically.
cats	A numeric vector indicating the number of score categories for each item.
model	A character vector specifying the IRT model for each item. Available options are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous items, and "GRM" and "GPCM" for polytomous items. The label "DRM" serves as a general category that encompasses all dichotomous models ("1PLM", "2PLM", and "3PLM"), while "GRM" and "GPCM" refer to the graded response model and (generalized) partial credit model, respectively.
default.par	Logical. If TRUE, default item parameters are generated based on the specified cats and model. In this case, the slope parameter is set to 1, all difficulty (or threshold) parameters are set to 0, and the guessing parameter is set to 0.2 for "3PLM" or "DRM" items. The default is FALSE.

Details

For any item where "1PLM" or "2PLM" is specified in model, the guessing parameter will be set to NA. If model is a vector of length 1, the specified model will be replicated across all items.

As in the simdat() function, when constructing a mixed-format test form, it is important to specify the cats argument to reflect the correct number of score categories for each item, in the exact order that the items appear. See simdat() for further guidance on how to specify cats.

When specifying item parameters using par.drm and/or par.prm, the internal structure and ordering of elements must be followed.

- par.drm should be a list with three components:
  - a: a numeric vector of slope parameters
  - b: a numeric vector of difficulty parameters
  - g: a numeric vector of guessing parameters
- par.prm should be a list with two components:
  - a: a numeric vector of slope parameters for polytomous items
  - d: a list of numeric vectors specifying threshold (or step) parameters for each polytomous item

For items following the (generalized) partial credit model ("GPCM"), the threshold (or step) parameters are computed as the overall item difficulty (location) minus the category-specific thresholds. Therefore, for an item with m score categories, m - 1 step parameters must be provided, since the first category threshold is fixed and does not contribute to category probabilities.

Value

A data frame containing item metadata, including item IDs, number of score categories, IRT model types, and associated item parameters. This data frame can be used as input for other functions in the **irtQ** package, such as est_irt() or simdat().

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

est_irt(), simdat(), shape_df_fipc()

Examples

```
## A mixed-format test form
## containing five dichotomous items and two polytomous items
# Create a list of dichotomous item parameters
par.drm <- list(
  a = c(1.1, 1.2, 0.9, 1.8, 1.4),
  b = c(0.1, -1.6, -0.2, 1.0, 1.2),
  g = rep(0.2, 5)
)
```

```
# Create a list of polytomous item parameters
par.prm <- list(
  a = c(1.4, 0.6),
  d = list(
    c(0.0, -1.9, 1.2),
    c(0.4, -1.1, 1.5, 0.2)
  )
)

# Create a numeric vector indicating the number of score categories for each item
cats <- c(2, 4, 2, 2, 5, 2, 2)

# Create a character vector specifying the IRT model for each item
model <- c("DRM", "GRM", "DRM", "DRM", "GPCM", "DRM", "DRM")

# Generate an item metadata set using the specified parameters
shape_df(par.drm = par.drm, par.prm = par.prm, cats = cats, model = model)

## An empty item metadata frame with five dichotomous items and two polytomous items
# Create a numeric vector indicating the number of score categories for each item
cats <- c(2, 4, 3, 2, 5, 2, 2)

# Create a character vector specifying the IRT model for each item
model <- c("1PLM", "GRM", "GRM", "2PLM", "GPCM", "DRM", "3PLM")

# Generate an item metadata frame with default parameters
shape_df(cats = cats, model = model, default.par = TRUE)

## A single-format test form consisting of five dichotomous items
# Generate the item metadata
shape_df(par.drm = par.drm, cats = rep(2, 5), model = "DRM")
```

---

shape_df_fipc *Combine fixed and new item metadata for fixed-item parameter calibration (FIPC)*

Description

This function merges existing fixed-item metadata with automatically generated metadata for new items, producing a single data frame ordered by specified test positions, to facilitate fixed item parameter calibration using `est_irt()`.

Usage

```
shape_df_fipc(x, fix.loc = NULL, item.id = NULL, cats, model)
```

Arguments

x	A data.frame of metadata for items whose parameters remain fixed (e.g., output from shape_df()).
fix.loc	An integer vector specifying the row positions in the final output where fixed items should be placed.
item.id	A character vector of IDs for new items whose parameters will be estimated.If NULL, default IDs (e.g., "V1", "V2", ...) are assigned automatically.
cats	An integer vector indicating the number of response categories for each new item; order must match item.id.
model	A character vector of IRT model names for each new item. Valid options for dichotomous items: "1PLM", "2PLM", "3PLM", "DRM"; for polytomous items: "GRM", "GPCM".

Details

To use this function, first prepare a metadata frame x containing only fixed items—either created by shape_df() or imported from external software (e.g., via bring.flexmirt()), which must include columns id, cats, model, and all relevant parameter columns (par.1, par.2, etc.). The fix.loc argument should then specify the exact row positions in the final test form where these fixed items should remain. The length of fix.loc must match the number of rows in x, and the order of positions in fix.loc determines where each fixed-item row is placed.

Next, provide information for the new items whose parameters will be estimated. Supply vectors for item.id, cats, and model matching the number of new items (equal to total form length minus length of fix.loc). If item.id is NULL, unique IDs are generated automatically.

Value

A data.frame containing combined metadata for all items (fixed and new), ordered by test position.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

shape_df()

Examples

```
## Import the flexMIRT parameter output file
prm_file <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")
x_fixed <- bring.flexmirt(file = prm_file, "par")$Group1$full_df

## Define positions of fixed items in the test form
fixed_pos <- c(1:40, 43:57)

## Specify IDs, models, and category counts for new items
new_ids <- paste0("NI", 1:6)
```

```
new_models <- c("3PLM", "1PLM", "2PLM", "GRM", "GRM", "GPCM")
new_cats <- c(2, 2, 2, 4, 5, 6)

## Generate combined metadata for FIPC
shape_df_fipc(x = x_fixed, fix.loc = fixed_pos, item.id = new_ids,
  cats = new_cats, model = new_models)
```

`simCAT_DC` *Simulated Single-Item Format CAT Data*

Description

A simulated dataset containing an item pool, sparse response data, and examinee ability estimates, designed for single-item computerized adaptive testing (CAT).

Usage

```
simCAT_DC
```

Format

A list of length three:

item_pool A data frame in item metadata format containing 100 dichotomous items.

- Items 1–90: Generated and calibrated under the IRT 2PL model.
- Items 91–100: Generated under the IRT 3PL model but calibrated using the 2PL model.

response_data A sparse matrix of item responses from 10,000 examinees.

theta_estimates A numeric vector of ability estimates for the 10,000 examinees.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

`simCAT_MX` *Simulated Mixed-Item Format CAT Data*

Description

A simulated dataset for computerized adaptive testing (CAT), containing an item pool, sparse response data, and examinee ability estimates. The item pool includes both dichotomous and polytomous items.

Usage

```
simCAT_MX
```

Format

A list of length three:

item_pool A data frame in item metadata format consisting of 200 dichotomous items and 30 polytomous items.

- Dichotomous items: Calibrated using the IRT 3PL model.
- Polytomous items: Calibrated using the Generalized Partial Credit Model (GPCM), with three score categories (0, 1, 2).

response_data A sparse matrix of item responses from 30,000 examinees.

theta_estimates A numeric vector of ability estimates for the 30,000 examinees.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

---

simdat *Simulated Response Data*

Description

This function generates simulated response data for single-format or mixed-format test forms. For dichotomous item response data, the IRT 1PL, 2PL, and 3PL models are supported. For polytomous item response data, the graded response model (GRM), the partial credit model (PCM), and the generalized partial credit model (GPCM) are supported.

Usage

```
simdat(
  x = NULL,
  theta,
  a.drm,
  b.drm,
  g.drm = NULL,
  a.prm,
  d.prm,
  cats,
  pr.model,
  D = 1
)
```

Arguments

x	A data frame containing item metadata. This metadata is required to retrieve essential information for each item (e.g., number of score categories, IRT model type, etc.) necessary for calibration. You can create an empty item metadata frame using the function shape_df(). See **below** for more details. Default is NULL.

theta	A numeric vector of ability (theta) values.
a.drm	A numeric vector of item discrimination (slope) parameters for dichotomous IRT models.
b.drm	A numeric vector of item difficulty parameters for dichotomous IRT models.
g.drm	A numeric vector of guessing parameters for dichotomous IRT models.
a.prm	A numeric vector of item discrimination (slope) parameters for polytomous IRT models.
d.prm	A list of numeric vectors, where each vector contains difficulty (threshold) parameters for a polytomous item.
cats	A numeric vector indicating the number of score categories for each item.
pr.model	A character vector specifying the polytomous IRT model used to simulate responses for each polytomous item. Each element should be either "GRM" (graded response model) or "GPCM" (generalized partial credit model).
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.

Details

There are two ways to generate simulated response data. The first is by providing a data frame of item metadata using the argument `x`. This data frame must follow a specific structure: the first column should contain item IDs, the second column should contain the number of unique score categories for each item, and the third column should specify the IRT model to be fitted to each item. Available IRT models are:

- `"1PLM"`, `"2PLM"`, `"3PLM"`, and `"DRM"` for dichotomous item data
- `"GRM"` and `"GPCM"` for polytomous item data

Note that `"DRM"` serves as a general label covering all dichotomous IRT models (i.e., `"1PLM"`, `"2PLM"`, and `"3PLM"`), while `"GRM"` and `"GPCM"` represent the graded response model and (generalized) partial credit model, respectively.

The subsequent columns should contain the item parameters for the specified models. For dichotomous items, the fourth, fifth, and sixth columns represent item discrimination (slope), item difficulty, and item guessing parameters, respectively. When `"1PLM"` or `"2PLM"` is specified in the third column, `NA`s must be entered in the sixth column for the guessing parameters.

For polytomous items, the item discrimination (slope) parameter should appear in the fourth column, and the item difficulty (or threshold) parameters for category boundaries should occupy the fifth through the last columns. When the number of unique score categories differs across items, unused parameter cells should be filled with `NA`s.

In the **irtQ** package, the threshold parameters for GPCM items are expressed as the item location (or overall difficulty) minus the threshold values for each score category. Note that when a GPCM item has K unique score categories, K - 1 threshold parameters are required, since the threshold for the first category boundary is always fixed at 0. For example, if a GPCM item has five score categories, four threshold parameters must be provided.

An example of a data frame for a single-format test is shown below:

ITEM1	2	1PLM	1.000	1.461	NA
ITEM2	2	2PLM	1.921	-1.049	NA
ITEM3	2	3PLM	1.736	1.501	0.203
ITEM4	2	3PLM	0.835	-1.049	0.182
ITEM5	2	DRM	0.926	0.394	0.099

An example of a data frame for a mixed-format test is shown below:

ITEM1	2	1PLM	1.000	1.461	NA	NA	NA
ITEM2	2	2PLM	1.921	-1.049	NA	NA	NA
ITEM3	2	3PLM	0.926	0.394	0.099	NA	NA
ITEM4	2	DRM	1.052	-0.407	0.201	NA	NA
ITEM5	4	GRM	1.913	-1.869	-1.238	-0.714	NA
ITEM6	5	GRM	1.278	-0.724	-0.068	0.568	1.072
ITEM7	4	GPCM	1.137	-0.374	0.215	0.848	NA
ITEM8	5	GPCM	1.233	-2.078	-1.347	-0.705	-0.116

See the *IRT Models* section in the irtQ-package documentation for more details about the IRT models used in the **irtQ** package. A convenient way to create a data frame for the argument `x` is by using the function `shape_df()`.

The second approach is to simulate response data by directly specifying item parameters, instead of providing a metadata data frame via the `x` argument (see examples below). In this case, the following arguments must also be specified: `theta`, `cats`, `pr.model`, and `D`.

The `g.drm` argument is only required when simulating dichotomous item responses under the 3PL model. It can be omitted entirely if all dichotomous items follow the 1PL or 2PL model. However, if the test includes a mixture of 1PL, 2PL, and 3PL items, the `g.drm` vector must be specified for all items, using `NA` for non-3PL items. For example, if a test consists of four dichotomous items where the first two follow the 3PL model and the third and fourth follow the 1PL and 2PL models respectively, then `g.drm = c(0.2, 0.1, NA, NA)` should be used.

For dichotomous items, each element in `cats` should be set to 2. For polytomous items, the number of unique score categories should be specified in `cats`. When simulating data for a mixed-format test, it is important to specify `cats` in the correct item order. For example, suppose responses are simulated for 10 examinees across 5 items, including 3 dichotomous items and 2 polytomous items (each with 3 categories), where the second and fourth items are polytomous. In this case, `cats = c(2, 3, 2, 3, 2)` should be used.

Furthermore, if the two polytomous items are modeled using the graded response model and the generalized partial credit model, respectively, then `pr.model = c("GRM", "GPCM")`.

Value

A matrix or vector of simulated item responses. If a matrix is returned, rows correspond to examinees (theta values) and columns to items.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

drm(), prm()

Examples

```
## Example 1:
## Simulate response data for a mixed-format test.
## The first two polytomous items use the generalized partial credit model (GPCM),
## and the last polytomous item uses the graded response model (GRM).
# Generate theta values for 100 examinees
theta <- rnorm(100)

# Set item parameters for three dichotomous items under the 3PL model
a.drm <- c(1, 1.2, 1.3)
b.drm <- c(-1, 0, 1)
g.drm <- rep(0.2, 3)

# Set item parameters for three polytomous items
# These items have 4, 4, and 5 response categories, respectively
a.prm <- c(1.3, 1.2, 1.7)
d.prm <- list(c(-1.2, -0.3, 0.4), c(-0.2, 0.5, 1.6), c(-1.7, 0.2, 1.1, 2.0))

# Specify the number of score categories for all items
# This vector also determines the location of polytomous items
cats <- c(2, 2, 4, 4, 5, 2)

# Specify the IRT models for the polytomous items
pr.model <- c("GPCM", "GPCM", "GRM")

# Simulate the response data
simdat(
  theta = theta, a.drm = a.drm, b.drm = b.drm, g.drm = NULL,
  a.prm = a.prm, d.prm = d.prm, cats = cats, pr.model = pr.model, D = 1
)

## Example 2:
## Simulate response data for a single-format test using the 2PL model
# Specify score categories (2 for each dichotomous item)
cats <- rep(2, 3)

# Simulate the response data
simdat(theta = theta, a.drm = a.drm, b.drm = b.drm, cats = cats, D = 1)

## Example 3:
## Simulate response data using a "-prm.txt" file exported from flexMIRT
# Load the flexMIRT parameter file
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Convert the flexMIRT parameters to item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# Simulate the response data using the item metadata
```

```
simdat(x = test_flex, theta = theta, D = 1)
```

`simMG` *Simulated multiple-group data*

Description

This data set has a list consisting of item metadata, item response data, and group names of three simulated groups.

Usage

```
simMG
```

Format

This data set includes a list of three internal objects: (1) a list of item metadata (item.prm) for three groups, (2) a list of item response data (res.dat) for the three groups, and (3) a vector of group names (group.name) for the three groups.

The first internal object (item.prm) contains a list of item metadata of three test forms for the three groups. In terms of test forms, the test forms for the first and second groups have fifty items consisting of forty seven 3PLM items and three GRM items. The test form for the third group has thirty eight items consisting of thirty seven 3PLM items and one GRM item. Among the three forms, the first and second test forms share twelve common items (C1I1 through C1I12) and the second and third test forms share ten common items (C2I1 through c2I10). There is no common item between the first and third forms. The item parameters in the item metadata were used to simulate the item response data sets for the three groups (see the second object of the list).

Regrading the second internal object, all three response data sets were simulated with 2,000 latent abilities randomly sampled from $N(0, 1)$ (Group 1), $N(0.5, 0.8^2)$ (Group 2), and $N(-0.3, 1.3^2)$ (Group 3), respectively, using the true item parameters provided in the item metadata.

The third internal object is a vector of three group names which are "Group1", "Group2", and "Group3".

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

`simMST` *Simulated 1-3-3 MST Panel Data*

Description

A simulated multistage testing (MST) dataset based on a 1-3-3 panel structure, used in the simulation study by Lim et al. (2021).

Usage

```
simMST
```

Format

A list containing five internal objects:

item_bank A data frame of item metadata including item parameters and related information.

module A binary matrix that maps items in the item bank to MST modules. This structure specifies the item-to-module assignment, similar to the `modules` argument in the `randomMST()` function from the **mstR** package (Magis et al., 2017).

route_map A binary square matrix that defines the MST transition structure, showing module pathways across stages. This corresponds to the `transMatrix` argument in the `randomMST()` function in the **mstR** package.

cut_score A list of numeric vectors specifying the routing cut scores between MST stages. Each vector represents the cut scores used to determine module transitions for a particular stage.

theta A numeric vector of ability (theta) values used to evaluate the panel's measurement precision across the latent trait continuum.

This 1-3-3 MST panel includes 7 modules across 3 stages. Each module contains 8 dichotomously scored items calibrated under the IRT 3-parameter logistic (3PL) model.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Magis, D., Yan, D., & von Davier, A. A. (2017). *Computerized adaptive and multistage testing with R: Using packages catR and mstR*. Springer.

Lim, H., Davey, T., & Wells, C. S. (2021). A recursion-based analytical approach to evaluate the performance of MST. *Journal of Educational Measurement, 58*(2), 154–178.

summary *Summary of Item Calibration Results*

Description

This S3 method summarizes the IRT calibration results from an object of class `est_irt`, `est_mg`, or `est_item`, which are returned by the functions `est_irt()`, `est_mg()`, and `est_item()`, respectively.

Usage

```
summary(object, ...)

## S3 method for class 'est_irt'
summary(object, ...)

## S3 method for class 'est_mg'
summary(object, ...)

## S3 method for class 'est_item'
summary(object, ...)
```

Arguments

`object`	An object of class `est_irt`, `est_mg`, or `est_item`.
`...`	Additional arguments passed to or from other methods (currently not used).

Value

A list of internal components extracted from the given object. In addition, the summary method prints an overview of the IRT calibration results to the console.

Methods (by class)

- `summary(est_irt)`: An object created by the function `est_irt()`.
- `summary(est_mg)`: An object created by the function `est_mg()`.
- `summary(est_item)`: An object created by the function `est_item()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

`est_irt()`, `est_mg()`, `est_item()`

Examples

```
# Fit the 1PL model to LSAT6 data and constrain the slope parameters to be equal
fit.1pl <- est_irt(data = LSAT6, D = 1, model = "1PLM", cats = 2, fix.a.1pl = FALSE)

# Display the calibration summary
summary(fit.1pl)
```

`sx2_fit` *S-X2 Fit Statistic*

Description

Computes the S-X^2 item fit statistic proposed by Orlando and Thissen (2000, 2003). This statistic evaluates the fit of IRT models by comparing observed and expected item response frequencies across summed score groups.

Usage

```
sx2_fit(x, ...)

## Default S3 method:
sx2_fit(
  x,
  data,
  D = 1,
  alpha = 0.05,
  min.collapse = 1,
  norm.prior = c(0, 1),
  nquad = 30,
  weights,
  pcm.loc = NULL,
  ...
)

## S3 method for class 'est_item'
sx2_fit(
  x,
  alpha = 0.05,
  min.collapse = 1,
  norm.prior = c(0, 1),
  nquad = 30,
  weights,
  pcm.loc = NULL,
  ...
)
```

```
## S3 method for class 'est_irt'
sx2_fit(
  x,
  alpha = 0.05,
  min.collapse = 1,
  norm.prior = c(0, 1),
  nquad = 30,
  weights,
  pcm.loc = NULL,
  ...
)
```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class est_irt obtained from est_irt(), or est_item from est_item(). See est_irt() or simdat() for more details about the item metadata. This data frame can be easily created using the shape_df() function.
...	Additional arguments passed to or from other methods.
data	A matrix of examinees' item responses corresponding to the items specified in the x argument. Rows represent examinees and columns represent items.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.
alpha	A numeric value specifying the significance level (α) for the hypothesis test associated with the S-X^2 statistic. Default is 0.05.
min.collapse	An integer specifying the minimum expected frequency required per cell before adjacent cells are collapsed. Default is 1. See **Details**.
norm.prior	A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. These values are used to generate the Gaussian quadrature points and weights. Ignored if method is "ML", "MLF", "WL", or "INV.TCC". Default is c(0, 1).
nquad	An integer specifying the number of Gaussian quadrature points used to approximate the normal prior distribution. Default is 30.
weights	A two-column matrix or data frame containing the quadrature points (first column) and their corresponding weights (second column) for the latent ability distribution. If omitted, default values are generated using gen.weight() according to the norm.prior and nquad arguments.
pcm.loc	An optional integer vector indicating the row indices of items that follow the partial credit model (PCM), where slope parameters are fixed. Default is NULL.

Details

The accuracy of the χ^2 approximation in item fit statistics can be compromised when expected cell frequencies in contingency tables are too small (Orlando & Thissen, 2000). To address this

issue, Orlando and Thissen (2000) proposed collapsing adjacent summed score groups to ensure a minimum expected frequency of at least 1.

However, applying this collapsing approach directly to polytomous item data can result in excessive information loss (Kang & Chen, 2008). To mitigate this, Kang and Chen (2008) instead collapsed adjacent response categories *within* each summed score group, maintaining a minimum expected frequency of 1 per category. The same collapsing strategies are implemented in `sx2_fit()`. If a different minimum expected frequency is desired, it can be specified via the `min.collapse` argument.

When an item is labeled as "DRM" in the item metadata, it is treated as a 3PLM item when computing the degrees of freedom for the S-X^2 statistic.

Additionally, any missing responses in the `data` are automatically replaced with incorrect responses (i.e., 0s).

Value

A list containing the following components:

`fit_stat`	A data frame summarizing the S-X^2 fit statistics for all items, including the chi-square value, degrees of freedom, critical value, and p-value.
`item_df`	A data frame containing the item metadata as specified in the input argument `x`.
`exp_freq`	A list of collapsed expected frequency tables for all items.
`obs_freq`	A list of collapsed observed frequency tables for all items.
`exp_prob`	A list of collapsed expected probability tables for all items.
`obs_prop`	A list of collapsed observed proportion tables for all items.

Methods (by class)

- `sx2_fit(default)`: Default method for computing S-X^2 fit statistics from a data frame `x` containing item metadata.
- `sx2_fit(est_item)`: An object created by the function `est_item()`.
- `sx2_fit(est_irt)`: An object created by the function `est_irt()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Kang, T., & Chen, T. T. (2008). Performance of the generalized S-X2 item fit index for polytomous IRT models. *Journal of Educational Measurement, 45*(4), 391-406.

Orlando, M., & Thissen, D. (2000). Likelihood-based item-fit indices for dichotomous item response theory models. *Applied Psychological Measurement, 24*(1), 50-64.

Orlando, M., & Thissen, D. (2003). Further investigation of the performance of S-X2: An item fit index for use with dichotomous item response theory models. *Applied Psychological Measurement, 27*(4), 289-298.

See Also

irtfit(), simdat(), shape_df(), est_irt(), est_item()

Examples

```
## Example 1: All five polytomous IRT items follow the GRM
## Import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Select the item metadata
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df

# Generate examinees' abilities from N(0, 1)
set.seed(23)
score <- rnorm(500, mean = 0, sd = 1)

# Simulate response data
data <- simdat(x = x, theta = score, D = 1)


# Compute fit statistics
fit1 <- sx2_fit(x = x, data = data, nquad = 30)

# Display fit statistics
fit1$fit_stat


## Example 2: Items 39 and 40 follow the GRM, and items 53, 54, and 55
##            follow the PCM (with slope parameters fixed to 1)
# Replace the model names with "GPCM" and
# set the slope parameters of items 53-55 to 1
x[53:55, 3] <- "GPCM"
x[53:55, 4] <- 1

# Generate examinees' abilities from N(0, 1)
set.seed(25)
score <- rnorm(1000, mean = 0, sd = 1)

# Simulate response data
data <- simdat(x = x, theta = score, D = 1)


# Compute fit statistics
fit2 <- sx2_fit(x = x, data = data, nquad = 30, pcm.loc = 53:55)

# Display fit statistics
fit2$fit_stat
```

traceline *Compute Item/Test Characteristic Functions*

Description

This function computes item category probabilities, item characteristic curves (ICCs), and the test characteristic curve (TCC) for a given set of theta values. The returned object can be used to visualize these functions using `plot.traceline()`.

Usage

```
traceline(x, ...)

## Default S3 method:
traceline(x, theta, D = 1, ...)

## S3 method for class 'est_item'
traceline(x, theta, ...)

## S3 method for class 'est_irt'
traceline(x, theta, ...)
```

Arguments

`x`	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.); or an object of class `est_irt` obtained from `est_irt()`, or `est_item` from `est_item()`. See `est_irt()` or `simdat()` for more details about the item metadata. This data frame can be easily created using the `shape_df()` function.
`...`	Further arguments passed to or from other methods.
`theta`	A numeric vector of theta values at which item and test characteristic curves are computed.
`D`	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.7 is commonly used for this purpose. Default is 1.

Details

This function computes the item and test characteristic functions commonly used in IRT. For each item, the function computes the category response probabilities across a specified set of theta values. These probabilities are used to derive:

- The item characteristic curve (ICC), which represents the expected score of each item as a function of theta.
- The test characteristic curve (TCC), which is the sum of expected item scores at each theta value.

The output object can be visualized using the plot.traceline to inspect the relationship between ability levels (theta) and expected item/test scores.

If the input `x` is an object of class `est_item` or `est_irt`, the function automatically extracts item parameter estimates and the scaling constant `D` from the object. Otherwise, a properly formatted item metadata data frame must be provided.

Value

This function returns an object of class `traceline`, which is a list containing the following components:

`prob.cats`	A list of data frames containing the category response probabilities for each item across the specified theta values. Each data frame corresponds to an item, with rows representing theta values and columns representing response categories (e.g., `"resp.0"`, `"resp.1"`, ...).
`icc`	A numeric matrix representing ICCs. Each column corresponds to an item, and each row represents the expected item score at a given theta value. The column names are the item IDs.
`tcc`	A numeric vector representing the TCC, computed as the sum of expected item scores across all items at each theta value.
`theta`	A numeric vector of theta values at which the item and test information functions are evaluated. This matches the user-supplied `theta` argument.

Methods (by class)

- `traceline(default)`: Default method to compute the item category probabilities, item characteristic function, and test characteristic function for a data frame `x` containing the item metadata.
- `traceline(est_item)`: An object created by the function `est_item()`.
- `traceline(est_irt)`: An object created by the function `est_irt()`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

`plot.traceline()`, `est_irt()`, `est_item()`

Examples

```
## Example using a "-prm.txt" file exported from flexMIRT

# Import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Read the item parameters and convert them into item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df
```

```
# Define a sequence of theta values
theta <- seq(-3, 3, 0.5)

# Compute item category probabilities, ICCs,
# and the TCC for the given theta values
traceline(x = test_flex, theta, D = 1)
```

`write.flexmirt` *Write a "-prm.txt" File for flexMIRT*

Description

This function writes a flexMIRT-compatible "-prm.txt" file (Cai, 2017). It currently supports only unidimensional IRT models. This function was developed by modifying `read.flexmirt()` from Pritikin & Falk (2020).

Usage

```
write.flexmirt(
  x,
  file = NULL,
  norm.pop = c(0, 1),
  rePar = TRUE,
  mgroup = FALSE,
  group.name = NULL
)
```

Arguments

`x`	A data frame of item metadata (e.g., item parameters, number of categories, model types) for a single group, or a list of such data frames for multiple groups. See `est_irt()` or `simdat()` for item metadata format. You can also create metadata using `shape_df()`.
`file`	A character string specifying the destination file path (with a ".txt" extension).
`norm.pop`	A numeric vector of length two specifying the mean and standard deviation of the normal population ability distribution for a single group, or a list of such vectors for multiple groups. When a list is provided, each internal vector must contain the mean and standard deviation for a group's ability distribution (e.g., `norm.pop = list(c(0, 1), c(0, 0.8), c(0.5, 1.2))` for three groups). If `mgroup = TRUE` and a single vector is provided (e.g., `norm.pop = c(0, 1)`), it will be recycled across all groups. The default is `c(0, 1)`.
`rePar`	A logical value indicating whether the item parameters are reparameterized. If `TRUE`, item intercepts and logits of guessing parameters are assumed. If `FALSE`, item difficulty and guessing parameters are assumed.
`mgroup`	A logical value indicating whether the file includes multiple groups. Default is `FALSE`.

group.name A character vector of group names. If NULL, group names are automatically generated (e.g., "Group1", "Group2", ...).

Value

This function creates a flexMIRT-style "-prm.txt" file at the specified path.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Cai, L. (2017). flexMIRT 3.5 Flexible multilevel multidimensional item analysis and test scoring (Computer Software). Chapel Hill, NC: Vector Psychometric Group.

Pritikin, J. N., & Falk, C. F. (2020). OpenMx: A modular research environment for item response theory method development. *Applied Psychological Measurement, 44*(7-8), 561-562.

Examples

```
## 1. Create a "-prm.txt" file for a single group
##    using the simulated CAT data
# 1-(1) Extract the item metadata
x <- simCAT_MX$item.prm

# 1-(2) Set the name of the "-prm.txt" file
temp_prm <- file.path(tempdir(), "single_group_temp-prm.txt")

# 1-(3) Write the "-prm.txt" file
write.flexmirt(x, file = temp_prm, norm.pop = c(0, 1), rePar = FALSE)

## 2. Create a "-prm.txt" file for multiple groups
##    using simulated multi-group data
# 2-(1) Extract the item metadata
x <- simMG$item.prm

# Set the name of the "-prm.txt" file
temp_prm <- file.path(tempdir(), "mg_group_temp-prm1.txt")

# Write the "-prm.txt" file
write.flexmirt(x,
  file = temp_prm, norm.pop = list(c(0, 1), c(0.5, 0.8), c(-0.3, 1.3)),
  rePar = FALSE, mgroup = TRUE, group.name = c("GR1", "GR2", "GR3")
)

# Or write the "-prm.txt" file so that
# all groups share the same ability distribution
# and group names are generated automatically
temp_prm <- file.path(tempdir(), "mg_group_temp-prm2.txt")
write.flexmirt(x,
  file = temp_prm, norm.pop = c(0, 1),
  rePar = FALSE, mgroup = TRUE, group.name = NULL
```

)

Index