Package 'irtQ'

October 8, 2026

Type Package

Title Unidimensional Item Response Theory Modeling

Version 1.3.1

Date 2026-10-08

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 differential item functioning analysis.
In addition, the package provides a set of classical test theory functions for computing item- and test-level statistics (e.g., item difficulty, item-total correlation, and coefficient alpha) and for scoring and analyzing selected-response item data.
The bring.flexmirt() and write.flexmirt() functions were written by modifying the read.flexmirt() function (Pritikin & Falk (2020) <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 (Gonzalez (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 (>= 4.5)

License GPL (>= 2)

Encoding UTF-8

LazyData true

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

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

VignetteBuilder knitr

Config/testthat/edition 3

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

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

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Maintainer Hwanggyu Lim <hglim83@gmail.com>

Repository CRAN

Date/Publication 2026-10-08 14:30:55 UTC

Contents

irtQ-package . . . 3
bind.fill . . . 16
bisection . . . 17
bring.flexmirt . . . 18
cac_lee . . . 21
cac_rud . . . 24
catsib . . . 27
covirt . . . 34
crdif . . . 36
ctt . . . 44
ctt_distr . . . 48
drm . . . 52
est_irt . . . 53
est_item . . . 67
est_mg . . . 72
est_score . . . 85
find_cut . . . 92
freq_score . . . 95
gen.weight . . . 97
getirt . . . 98
grdif . . . 103
info . . . 111
irtfit . . . 114
llike_score . . . 120
LSAT6 . . . 122
lwrc . . . 122
panel_info . . . 124

pcd2 . . . 126
plot.find_cut . . . 131
plot.info . . . 133
plot.irtfit . . . 135
plot.traceline . . . 139
print.ctt . . . 142
print.summary.ctt . . . 143
prm . . . 144
rdif . . . 145
reval_mst . . . 153
ripd . . . 158
run_flexmirt . . . 167
run_mst . . . 169
score_resp . . . 178
shape_df . . . 182
shape_df_fipc . . . 184
simCAT_DC . . . 186
simCAT_MX . . . 187
simdat . . . 188
simIPD . . . 191
simMG . . . 194
simMST . . . 195
summary . . . 196
summary.ctt . . . 197
sx2_fit . . . 198
traceline . . . 202
write.flexmirt . . . 204

Index **207**

`irtQ-package` *irtQ: Unidimensional Item Response Theory Modeling*

Description

The **irtQ** package provides tools for fitting unidimensional item response theory (IRT) models to test data that include both dichotomous and polytomous items. It enables the calibration of pretest item parameters, estimation of examinees' abilities, and offers a comprehensive suite of utilities for unidimensional IRT analysis, including model-data fit evaluation, differential item functioning (DIF) detection, and more.

For item parameter estimation, the package employs marginal maximum likelihood estimation via the expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981). To calibrate pretest (newly developed) items, it supports both fixed item parameter calibration (FIPC; Kim, 2006) and fixed ability parameter calibration (FAPC; Ban et al., 2001; Stocking, 1988), also known as Stocking's Method A. In addition, the package offers multiple-group item calibration via the MMLE-EM algorithm.

For ability estimation, several widely used scoring methods are available, including:

- 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)
- Inverse test characteristic curve (TCC) scoring (e.g., Kolen & Brennan, 2004; Kolen & Tong, 2010; Stocking, 1996)

In addition, the package offers a variety of utilities for IRT analysis, including:

- Evaluating model-data fit
- Detecting DIF
- Computing classification accuracy and consistency indices
- Importing item or ability parameters from popular IRT software
- Running flexMIRT (Cai, 2017) directly from R
- Simulating response data
- Computing the conditional distribution of observed scores using the Lord-Wingersky recursion
- Calculating item and test information and characteristic functions
- Visualizing item and test characteristic and information curves
- Supporting additional tools for flexible and practical IRT analyses

Beyond these IRT-based analyses, the package also provides a small set of classical test theory (CTT) functions (`ctt()`, `freq_score()`, `ctt_distr()`, and `score_resp()`) for computing traditional item- and test-level statistics and for scoring selected-response item data.

Package:	irtQ
Version:	1.3.1
Date:	2026-10-08
Depends:	R (>= 4.5)
License:	GPL (>= 2)

Details

The following five sections provide an overview of:

1. The IRT models implemented in the **irtQ** package
2. How to estimate item parameters for a linear test form
3. How to perform pretest item calibration using the fixed item parameter calibration (FIPC) method
4. How to perform pretest item calibration using the fixed ability parameter calibration (FAPC) method
5. Illustrative examples of item calibration for a linear test form and for pretest items

IRT Models

The **irtQ** package supports both dichotomous and polytomous item response theory (IRT) models. For dichotomous items, the one-, two-, and three-parameter logistic models (1PLM, 2PLM, and 3PLM) are available. For polytomous items, the graded response model (GRM) and the (generalized) partial credit model (GPCM) are implemented. When fitting the partial credit model (PCM), note that item discrimination (or slope) parameters should be fixed to 1.

In the following, let Y be the response of an examinee with latent ability θ on an item, and suppose that each polytomous item has K unique score categories.

IRT 1-3PL models For the IRT 1PLM, 2PLM, and 3PLM models, the probability that an examinee with ability θ responds correctly to an item is given by:

$$P(Y = 1 \mid \theta) = g + \frac{(1-g)}{1 + \exp(-Da(\theta - b))},$$

where a is the item discrimination (slope) parameter, b is the item difficulty parameter, and g is the guessing parameter. The constant D is a scaling factor used to make the logistic function approximate the normal ogive function, typically set to $D = 1.702$. When the 1PLM is used, the discrimination parameter a is either fixed to a constant value (e.g., $a = 1$) or constrained to be equal across all items. For 1PLM and 2PLM models, the guessing parameter is fixed at $g = 0$.

GRM For the graded response model (GRM), the probability that an examinee with latent ability θ responds in score category k ($k = 0, 1, \ldots, K-1$) is given by:

$$P(Y = k \mid \theta) = P^*(Y \geq k \mid \theta) - P^*(Y \geq k+1 \mid \theta),$$

$$P^*(Y \geq k \mid \theta) = \frac{1}{1 + \exp(-Da(\theta - b_k))},$$

$$P^*(Y \geq k+1 \mid \theta) = \frac{1}{1 + \exp(-Da(\theta - b_{k+1}))},$$

where $P^*(Y \geq k \mid \theta)$ is the cumulative probability (or category boundary function) for score category k, with a functional form similar to the 2PL model. The parameter b_k represents the difficulty (threshold) parameter associated with the boundary between categories $k-1$ and k. Note that the probability of responding in the lowest or highest score category can be written as:

$$P(Y = 0 \mid \theta) = 1 - P^*(Y \geq 1 \mid \theta),$$

$$P(Y = K-1 \mid \theta) = P^*(Y \geq K-1 \mid \theta).$$

GPCM For the generalized partial credit model (GPCM), the probability that an examinee with latent ability θ responds in score category k ($k = 0, 1, \ldots, K-1$) is given by:

$$P(Y = k \mid \theta) = \frac{\exp\left(\sum_{v=0}^{k} Da(\theta - b_v)\right)}{\sum_{h=0}^{K-1} \exp\left(\sum_{v=0}^{h} Da(\theta - b_v)\right)},$$

where b_v is the threshold (or step) parameter associated with category boundary v of the item. In alternative parameterizations, b_v can be expressed as $b_v = \beta - \tau_v$, where β is the overall location (difficulty) parameter of the item, and τ_v is the threshold for score category v. In the

irtQ package, an item with K unique score categories requires $K - 1$ threshold parameters. The $v = 0$ term of the sums is defined as 0 (b_0 is not a free parameter), so the numerator for the lowest category is $\exp(0) = 1$. When fitting the partial credit model, the item discrimination parameter a is fixed to 1.

Item Calibration for a Linear Test Form

Item parameter estimation for a linear test form can be performed using the `est_irt()` function, which implements marginal maximum likelihood estimation via the expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981). The function returns item parameter estimates along with their standard errors, computed using the cross-product approximation method (Meilijson, 1989).

The **irtQ** package supports calibration for mixed-format tests containing both dichotomous and polytomous items. It also provides a flexible set of options to address various practical calibration needs. For example, users can:

- Specify prior distributions for item parameters
- Fix specific parameters (e.g., the guessing parameter in the 3PL model)
- Estimate the latent ability distribution using a nonparametric histogram method (Woods, 2007)

In the **irtQ** package, item calibration for a linear test form typically involves two main steps:

1. Prepare the examinees' response data set for the linear test form
   To estimate item parameters using the `est_irt()` function, a response data set for the linear test form must first be prepared. The data should be provided in either a matrix or data frame format, where rows represent examinees and columns represent items. If there are missing responses, they should be properly coded (e.g., `NA`).
2. Estimate item parameters using the `est_irt()` function
   To estimate item parameters, several key input arguments must be specified in the `est_irt()` function:
   - `data`: A matrix or data frame containing examinees' item responses.
   - `model`: A character vector specifying the IRT model for each item (e.g., `"1PLM"`, `"2PLM"`, `"3PLM"`, `"GRM"`, `"GPCM"`).
   - `cats`: A numeric vector indicating the number of score categories for each item. For dichotomous items, use 2.
   - `D`: A scaling constant (typically 1.702) to align the logistic function with the normal ogive model.

   Optionally, you may incorporate prior distributions for item parameters:
   - `use.aprior`, `use.bprior`, `use.gprior`: Logical indicators specifying whether to apply prior distributions to the discrimination (`a`), difficulty (`b`), and guessing (`g`) parameters, respectively.
   - `aprior`, `bprior`, `gprior`: Lists specifying the distributional form and corresponding parameters for each prior. Supported distributions include Beta, Log-normal, and Normal.

   If the response data contain missing values, you must specify the missing value code via the `missing` argument.

   By default, the latent ability distribution is assumed to follow a standard normal distribution (i.e., N(0, 1)). However, users can estimate the empirical histogram of the latent distribution by setting `EmpHist = TRUE`, based on the nonparametric method proposed by Woods (2007).

Pretest Item Calibration with the Fixed Item Parameter Calibration (FIPC) Method (e.g., Kim, 2006)

The fixed item parameter calibration (FIPC) method is a widely used approach for calibrating pretest items in computerized adaptive testing (CAT). It enables the placement of parameter estimates for newly developed items onto the same scale as the operational item parameters (i.e., the scale of the item bank), without the need for post hoc linking or rescaling procedures (Ban et al., 2001; Chen & Wang, 2016).

In FIPC, the parameters of the operational items are fixed, and the prior distribution of the latent ability variable is estimated during the calibration process. This estimated prior is used to place the pretest item parameters on the same scale as the fixed operational items (Kim, 2006).

In the **irtQ** package, FIPC is implemented through the following three steps:

1. Prepare the item metadata, including both the operational items (to be fixed) and the pretest items.

   To perform FIPC using the `est_irt()` function, the item metadata must first be prepared. The item metadata is a structured data frame that includes essential information for each item, such as the number of score categories and the IRT model type. For more details, refer to the **Details** section of the `est_irt()` documentation.

   In the FIPC procedure, the metadata must contain both:

   - Operational items (whose parameters will be fixed), and
   - Pretest items (whose parameters will be freely estimated).

   For the pretest items, the `cats` (number of score categories) and `model` (IRT model type) must be accurately specified. However, the item parameter values (e.g., `par.1`, `par.2`, `par.3`) in the metadata serve only as placeholders and can be arbitrary, since the actual parameter estimates will be obtained during calibration.

   To facilitate creation of the metadata for FIPC, the helper function `shape_df_fipc()` can be used.

2. Prepare the response data set from examinees who answered both the operational and pretest items.

   To implement FIPC using the `est_irt()` function, examinees' response data for the test form must be provided, including both operational and pretest items. The response data should be in a matrix or data frame format, where rows represent examinees and columns represent items. Note that the column order of the response data must exactly match the row order of the item metadata.

3. Perform FIPC using the `est_irt()` function to calibrate the pretest items.

   When FIPC is performed using the `est_irt()` function, the parameters of pretest items are estimated while the parameters of operational items are fixed.

   To implement FIPC, you must provide the following arguments to `est_irt()`:

   - `x`: The item metadata, including both operational and pretest items.
   - `data`: The examinee response data corresponding to the item metadata.
   - `fipc = TRUE`: Enables fixed item parameter calibration.
   - `fipc.method`: Specifies the FIPC method to be used (e.g., `"MEM"`).
   - `fix.loc`: A vector indicating the positions of the operational items to be fixed.

Optionally, you may estimate the empirical histogram and scale of the latent ability distribution by setting `EmpHist = TRUE`. If `EmpHist = FALSE`, a normal prior is assumed and its scale is updated iteratively during the EM algorithm.

For additional details on implementing FIPC, refer to the documentation for `est_irt()`.

Pretest Item Calibration with the Fixed Ability Parameter Calibration (FAPC) Method (e.g., Stocking, 1988)

In computerized adaptive testing (CAT), the fixed ability parameter calibration (FAPC) method (also known as Stocking's Method A; Stocking, 1988) is one of the simplest and most straightforward approaches for calibrating pretest items. It involves estimating item parameters using maximum likelihood estimation, conditional on known or estimated proficiency values.

FAPC is primarily used to place the parameter estimates of pretest items onto the same scale as the operational item parameters. It can also be used to recalibrate operational items when evaluating potential item parameter drift (Chen & Wang, 2016; Stocking, 1988). This method is known to produce accurate and unbiased item parameter estimates when items are randomly administered to examinees, rather than adaptively, which is often the case for pretest items (Ban et al., 2001; Chen & Wang, 2016).

In the **irtQ** package, FAPC can be conducted in two main steps:

1. Prepare a data set containing both the item response data and the corresponding ability (proficiency) estimates.

   To use the `est_item()` function, two input data sets are required:

   - Ability estimates: A numeric vector containing examinees' ability (or proficiency) estimates.
   - Item response data: A matrix or data frame containing item responses, where rows represent examinees and columns represent items. The order of examinees in the response data must exactly match the order of the ability estimates.

2. Estimate the item parameters using the `est_item()` function.

   The `est_item()` function estimates pretest item parameters based on provided ability estimates. To use this function, you must specify the following arguments:

   - `data`: A matrix or data frame containing examinees' item responses.
   - `score`: A numeric vector of examinees' ability (proficiency) estimates.
   - `model`: A character vector specifying the IRT model for each item (e.g., `"1PLM"`, `"2PLM"`, `"3PLM"`, `"GRM"`, `"GPCM"`).
   - `cats`: A numeric vector indicating the number of score categories for each item. For dichotomous items, use 2.
   - `D`: A scaling constant (typically 1.702) to align the logistic function with the normal ogive model.

   For additional details on implementing FAPC, refer to the documentation for `est_item()`.

Three examples of R scripts

The examples below demonstrate how to estimate item parameters for a linear test form, and how to calibrate pretest items using the FIPC and FAPC methods. All examples use simulated data sets, where examinees' response data are generated based on true item parameters and true ability values using the `simdat()` function.

The examples utilize item parameter estimates imported from a flexMIRT output file sample, which includes a mixed-format test of 55 items: 50 dichotomous items modeled with the 3PLM and 5 polytomous items modeled with the GRM. These item parameters are treated as the "true" values for data generation. Specifically:

- Items 1-38: 3PLM items
- Items 39-40: GRM items
- Items 41-52: 3PLM items
- Items 53-55: GRM items (all with 5 score categories)

Example 1 - Estimating item parameters for a linear test form This example uses the first 40 items (38 3PLM and 2 GRM) to form a linear test. The test is assumed to be administered to a reference group of 2,000 examinees whose latent ability follows a standard normal distribution, N(0, 1).

Examples 2 and 3 - Calibrating pretest items using FIPC and FAPC These examples assume a new test form consisting of all 55 items described above. The first 40 items are treated as the fixed operational items (i.e., the same as in Example 1), while the remaining 15 items are newly developed pretest items to be calibrated.

The new test form is administered to a separate group of 2,000 examinees, whose ability distribution is assumed to follow $N(0.5, 1.3^2)$. The calibration of the pretest items is then conducted on the scale of the old form by using FIPC and FAPC procedures, respectively.

```
# Attach the packages
library(irtQ)

##-----------------------------------------------------------------------------
## 1. Item parameter estimation for a linear test form
##-----------------------------------------------------------------------------

## Step 1: Prepare response data for the reference group
## Import the "-prm.txt" output file from flexMIRT
meta_true <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# Extract item metadata using `irtQ::bring.flexmirt()`
# This will serve as the base test form for later pretest item examples
x_new <- irtQ::bring.flexmirt(file = meta_true, "par")$Group1$full_df

# Extract items 1 to 40 to define the linear test form used in this illustration
x_ref <- x_new[1:40, ]

# Generate true ability values (N = 2,000) from N(0, 1) for the reference group
set.seed(20)
theta_ref <- rnorm(2000, mean = 0, sd = 1)

# Simulate response data for the linear test form
# Scaling factor D = 1 assumes a logistic IRT model
data_ref <- irtQ::simdat(x = x_ref, theta = theta_ref, D = 1)
```

```
## Step 2: Estimate item parameters for the linear test form
mod_ref <- irtQ::est_irt(
  data       = data_ref,                                # Response data
  D          = 1,                                       # Scaling factor
  model      = c(rep("3PLM", 38), rep("GRM", 2)),       # Item models
  cats       = c(rep(2, 38), rep(5, 2)),              # Score categories per item
  item.id    = paste0("Ref_I", 1:40),                   # Item IDs
  use.gprior = TRUE,                              # Use prior for guessing parameter
  gprior     = list(dist = "beta", params = c(5, 16)),# Prior: Beta(5,16) for g
  Quadrature = c(49, 6),                          # 49 quadrature points from -6 to 6
  group.mean = 0,
  group.var  = 1,                                    # Fixed latent ability: N(0,1)
  EmpHist   = TRUE,                          # Estimate empirical ability distribution
  Etol      = 1e-3,                                  # E-step convergence tolerance
  MaxE       = 500)                                  # Max EM iterations

# Summarize estimation results
irtQ::summary(mod_ref)

# Extract item parameter estimates
est_ref <- mod_ref$par.est
print(est_ref)

##-----------------------------------------------------------------------------
## 2. Pretest item calibration using Fixed Item Parameter Calibration (FIPC)
##-----------------------------------------------------------------------------

## Step 1: Prepare item metadata for both fixed operational items and pretest items
# Define anchor item positions (items to be fixed)
fixed_pos <- c(1:40)

# Specify IDs, models, and categories for 15 pretest items
# Includes 12 3PLM and 3 GRM items (each GRM has 5 categories)
new_ids <- paste0("New_I", 1:15)
new_models <- c(rep("3PLM", 12), rep("GRM", 3))
new_cats <- c(rep(2, 12), rep(5, 3))

# Construct item metadata using `shape_df_fipc()`. See Details of `shape_df_fipc()`
# for more information
# First 40 items are anchor items (fixed); last 15 are pretest (freely estimated)
meta_fipc <- irtQ::shape_df_fipc(x = est_ref, fix.loc = fixed_pos, item.id = new_ids,
                                 cats = new_cats, model = new_models)

## Step 2: Prepare response data for the new test form
# Generate latent abilities for 2,000 new examinees from N(0.5, 1.3^2)
set.seed(21)
theta_new <- rnorm(2000, mean = 0.5, sd = 1.3)
```

```
# Simulate response data using true item parameters and true abilities
data_new <- irtQ::simdat(x = x_new, theta = theta_new, D = 1)

## Step 3: Calibrate pretest items using FIPC
# Fit 3PLM to dichotomous and GRM to polytomous items
# Fix first 40 items and freely estimate the remaining 15 pretest items
mod_fipc <- irtQ::est_irt(
  x           = meta_fipc,                    # Combined item metadata
  data        = data_new,                     # Response data
  D           = 1,                            # Scaling constant
  use.gprior  = TRUE,                        # Use prior for guessing parameter
  gprior     = list(dist = "beta", params = c(5, 16)),  # Prior: Beta(5,16) for g
  Quadrature  = c(49, 6),                    # 49 quadrature points from -6 to 6
  EmpHist    = TRUE,                   # Estimate empirical ability distribution
  Etol        = 1e-3,                         # E-step convergence tolerance
  MaxE        = 500,                          # Max EM iterations
  fipc        = TRUE,                         # Enable FIPC
  fipc.method = "MEM",                        # Use Multiple EM cycles
  fix.loc     = c(1:40))                      # Anchor item positions to fix

# Summarize estimation results
irtQ::summary(mod_fipc)

# Extract item parameter estimates
est_new_fipc <- mod_fipc$par.est
print(est_new_fipc)

# Plot estimated empirical distribution of ability
emphist <- irtQ::getirt(mod_fipc, what="weights")
plot(emphist$weight ~ emphist$theta, xlab="Theta", ylab="Density", type = "h")


##-----------------------------------------------------------------------------
## 3. Pretest item calibration using Fixed Ability Parameter Calibration (FAPC)
##-----------------------------------------------------------------------------

## Step 1: Prepare response data and ability estimates
# In FAPC, ability estimates are assumed known and fixed.
# Estimate abilities for new examinees using the first 40 fixed operational (anchor) items only.
# Pretest items are not used for scoring, as their parameters are not yet calibrated.

# Estimate abilities using ML method via `irtQ::est_score()`
# Based on fixed anchor item parameters and corresponding responses
score_ml <- irtQ::est_score(
  x      = est_ref,            # Metadata with operational item parameters
  data   = data_new[, 1:40],   # Responses to anchor items
  D      = 1,                  # Scaling constant
  method = "ML",               # Scoring method: Maximum Likelihood
```

```
  range  = c(-5, 5))           # Scoring bounds

# Extract estimated abilities
theta_est <- score_ml$est.theta

## Step 2: Calibrate pretest items using FAPC
# Only the 15 pretest items are included in the calibration
mod_fapc <- irtQ::est_item(
  data      = data_new[, 41:55],                    # Responses to pretest items
  score      = theta_est,                              # Fixed ability estimates
  D          = 1,                                        # Scaling constant
  model      = c(rep("3PLM", 12), rep("GRM", 3)),       # Item models
  cats       = c(rep(2, 12), rep(5, 3)),                # Score categories
  item.id    = paste0("New_I", 1:15),                   # Item IDs
  use.gprior = TRUE,                              # Use prior for guessing parameter
  gprior     = list(dist = "beta", params = c(5, 16))   # Prior: Beta(5,16) for g
)

# Summarize estimation results
irtQ::summary(mod_fapc)

# Extract item parameter estimates
est_new_fapc <- mod_fapc$par.est
print(est_new_fapc)
```

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.

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. doi:10.1111/j.17453984.2001.tb01123.x.

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.

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

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., & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement, 6*(4), 431-444. doi:10.1177/014662168200600405.

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). Springer. doi:10.1007/9781475726916_25.

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

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.

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

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

Crocker, L., & Algina, J. (1986). *Introduction to classical and modern test theory*. Holt, Rinehart and Winston.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297-334. doi:10.1007/BF02310555.

Gonzalez, J. (2014). SNSequate: Standard and nonstandard statistical models and methods for test equating. *Journal of Statistical Software, 59*(7), 1-30.

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.

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. doi:10.1177/0146621616631317.

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

Jung, H., & Lim, H. (2026, April). Detecting global and net DIF in polytomous items using RDIF. Paper presented at the annual meeting of the National Council on Measurement in Education, Los Angeles, CA.

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.

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.*

Kolen, M. J., & Brennan, R. L. (2004). *Test equating, scaling, and linking* (2nd ed.). Springer.

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

Laplace, P. S. (1820). *Theorie analytique des probabilites* (in French). Courcier.

Lee, W.-C. (2010). Classification consistency and accuracy for complex assessments using item response theory. *Journal of Educational Measurement, 47*(1), 1-17. doi:10.1111/j.17453984.2009.00096.x.

Li, H.-H., & Stout, W. (1996). A new procedure for detection of crossing DIF. *Psychometrika, 61*(4), 647-677. doi:10.1007/BF02294041.

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.

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., 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. doi:10.1111/jedm.12276.

Lim, H., & Han, K. T. (2026). IRT residual-based approach to detecting item parameter drift in CAT. *Journal of Educational and Behavioral Statistics*. doi:10.3102/10769986261460852.

Lim, H., Malatesta, J., & Lee, Y. (2024, July). Advancing polytomous DIF detection with the residual DIF framework. Paper presented at the annual International Meeting of the Psychometric Society, Prague, Czech Republic.

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.

Lord, F., & Wingersky, M. (1984). Comparison of IRT true score and equipercentile observed score equatings. *Applied Psychological Measurement, 8*(4), 453-461.

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

Magis, D., Yan, D., & von Davier, A. A. (2017). *Computerized adaptive and multistage testing with R: Using packages catR and mstR*. Springer. doi:10.1007/9783319692180.

McKinley, R., & Mills, C. (1985). A comparison of several goodness-of-fit statistics. *Applied Psychological Measurement, 9*, 49-57.

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.

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

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. doi:10.3102/10769986029002177.

Newcombe, R. G. (1998). Two-sided confidence intervals for the single proportion: comparison of seven methods. *Statistics in Medicine, 17*(8), 857-872.

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.

Osburn, H. G. (2000). Coefficient alpha and related internal consistency reliability coefficients. *Psychological Methods, 5*(3), 343-355. doi:10.1037/1082989X.5.3.343.

Penfield, R. D. (2010). Distinguishing between net and global DIF in polytomous items. *Journal of Educational Measurement, 47*(2), 129-149. doi:10.1111/j.17453984.2010.00105.x.

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. doi:10.1177/0146621620929431.

Rudner, L. M. (2001). Computing the expected proportions of misclassified examinees. *Practical Assessment, Research & Evaluation, 7*(14). doi:10.7275/an9m2035.

Rudner, L. M. (2005). Expected classification accuracy. *Practical Assessment, Research & Evaluation, 10*(13). doi:10.7275/56a56b14.

Shealy, R., & Stout, W. (1993). A model-based standardization approach that separates true bias/DIF from group ability differences and detects test bias/DTF as well as item bias/DIF. *Psychometrika, 58*(2), 159-194. doi:10.1007/BF02294572.

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

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

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.

Thissen, D. (1982). Marginal maximum likelihood estimation for the one-parameter logistic model. *Psychometrika, 47*, 175-186.

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. L. (1995). Item response theory for scores on tests including polytomous items with ordered responses. *Applied Psychological Measurement, 19*(1), 39-49. doi:10.1177/014662169501900105.

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

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.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427-450. doi:10.1007/BF02294627.

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. doi:10.18637/jss.v035.i12.

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.

Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association, 22*(158), 209-212.

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

Yen, W. M. (1981). Using simulation results to choose a latent trait model. *Applied Psychological Measurement, 5*, 245-262.

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

---

`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 mirt::mirt(). The **mirt** package, listed in Suggests, must be installed to use bring.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.

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

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. doi:10.1177/0146621620929431.

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. doi:10.18637/jss.v035.i12.

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

## Example 3
# Import the item parameters of a model fitted with the mirt package

if (requireNamespace("mirt", quietly = TRUE)) {
  fit <- mirt::mirt(as.data.frame(LSAT6), 1, itemtype = "2PL", verbose = FALSE)
  bring.mirt(fit)$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. doi:10.1111/j.17453984.2009.00096.x.

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. Standard errors from `est_score()` that are set to 99.9999 (ability estimates at a limit of `range`) should be handled before they are supplied.

`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 & Evaluation, 7*(14). doi:10.7275/an9m2035.

Rudner, L. M. (2005). Expected classification accuracy. *Practical Assessment, Research & Evaluation, 10*(13). doi:10.7275/56a56b14.

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. It must be supplied when `score` is supplied. 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`. If the limit is reached while flagged items remain, a warning is issued and the `complete` element of the purification results is `FALSE`; for tests with many items or many DIF items, consider increasing `max.iter`.

`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. doi:10.1007/BF02294041.

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.

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. doi:10.3102/10769986029002177.

Shealy, R., & Stout, W. (1993). A model-based standardization approach that separates true bias/DIF from group ability differences and detects test bias/DTF as well as item bias/DIF. *Psychometrika, 58*(2), 159-194. doi:10.1007/BF02294572.

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
```

```
# Therefore, 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`. If the limit is reached while flagged items remain, a warning is issued and the `complete` element of the purification results is `FALSE`; for tests with many items or many DIF items, consider increasing `max.iter`.

`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 (Jung & Lim, 2026; Lim, Malatesta, & Lee, 2024). 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 numeric vector of the positions (rows of `x`) of the items flagged as DIF by the `purify.by` statistic across all purification iterations, sorted in ascending order.

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

Jung, H., & Lim, H. (2026, April). Detecting global and net DIF in polytomous items using RDIF. Paper presented at the annual meeting of the National Council on Measurement in Education, Los Angeles, CA.

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., Malatesta, J., & Lee, Y. (2024, July). Advancing polytomous DIF detection with the residual DIF framework. Paper presented at the annual International Meeting of the Psychometric Society, Prague, Czech Republic.

Penfield, R. D. (2010). Distinguishing between net and global DIF in polytomous items. *Journal of Educational Measurement, 47*(2), 129-149. doi:10.1111/j.17453984.2010.00105.x.

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)
```

---

ctt *Classical Test Theory (CTT) Item and Test Analysis*

Description

Computes classical test theory (CTT) statistics from a scored item response data set (dichotomous or polytomous). For each item, computes its difficulty, an item-total correlation reflecting how well the item discriminates between high- and low-scoring examinees, and Cronbach's alpha recomputed with that item removed, with optional flagging of items whose difficulty or discrimination falls outside conventional quality thresholds. At the test level, computes Cronbach's alpha as a reliability estimate (in both raw and standardized forms), the associated standard error of measurement (SEM), and the average item difficulty and discrimination across the test. The distribution of total (raw) scores across examinees is also tabulated. The result is returned as an object of class `"ctt"`, with `print.ctt()` and `summary.ctt()` methods that display a condensed and a full report, respectively, analogous to how `est_irt()` pairs with `print.est_irt()`/`summary.est_irt()`.

Usage

```
ctt(
  data,
  item.id = NULL,
  cats = NULL,
  correct = FALSE,
  missing = NA,
  flag = TRUE,
  crit.p = c(0.1, 0.95),
  crit.dis = 0.2
)
```

Arguments

`data`	A data frame or matrix of already-scored item responses, with examinees in rows and items in columns. Item scores must range from 0 to `cats[j] - 1` for each item j (0/1 for a dichotomous item; 0, 1, 2, ... for a polytomous/partial-credit item). An error is raised when a score is not a whole number between 0 and `cats[j] - 1`.
`item.id`	A character vector of item identifiers, in the same order as the columns of `data`. If `NULL` (default), item IDs are generated automatically as `paste0("V", 1:ncol(data))`, following the convention used elsewhere in irtQ (e.g., `shape_df()`, `est_irt()`). Note that this generated ID does not fall back to `colnames(data)`; pass `item.id` explicitly to label items using the column names of `data` or any other identifier scheme.

`cats` A numeric vector giving the number of score categories for each item (e.g., 2 for a dichotomous item), following the `cats` convention used elsewhere in irtQ (see, e.g., `shape_df()`). If `NULL` (default), the number of categories for each item is inferred from the observed maximum score in `data` plus one, with a minimum of two (i.e., `max(max(data[, j]) + 1, 2)`); supply `cats` explicitly whenever the maximum possible score may not have been observed in the sample. Each value must be a whole number of at least 2.

`correct` Logical. Both the raw (uncorrected) item-total correlation - where an item is correlated with the total score that includes its own contribution - and the corrected item-total correlation - excluding its own contribution - are always computed and reported as separate columns. This argument only controls which of the two is used for flagging: if `TRUE`, flagging is based on the corrected value; if `FALSE` (default), on the raw value.

`missing` A value indicating missing responses in `data`, analogous to the `missing` argument in `est_irt()` and `score_resp()`. Any cell equal to `missing` is recoded to `NA` before analysis. Default is `NA`. Examinees with any remaining missing item response are excluded listwise from all statistics computed by this function.

`flag` Logical. If `TRUE` (default), items are flagged when their difficulty or discrimination falls outside the thresholds given in `crit.p` and `crit.dis`.

`crit.p` A numeric vector of length two giving the lower and upper difficulty bounds used for flagging: difficulty below the first value is flagged as too difficult, and difficulty above the second value is flagged as too easy. Default is `c(0.10, 0.95)`.

`crit.dis` A single numeric value giving the minimum acceptable discrimination (item-total correlation); items strictly below this value are flagged as poorly discriminating. Default is `0.20`.

Details

Difficulty for item j is the mean observed item score divided by the item's maximum possible score,

$$p_j = \frac{\bar{X}_j}{m_j},$$

where m_j is `cats[j] - 1`. For a dichotomous item (`cats[j] = 2`) this reduces to the familiar proportion-correct difficulty index; for a polytomous item it expresses the average score as a proportion of the maximum attainable score, keeping difficulty on a common 0-1 scale regardless of the number of score categories.

Discrimination for item j is the Pearson correlation between the item score and the total score, always computed and reported in two forms: the raw (uncorrected) item-total correlation, which correlates the item with the total score that includes the item's own contribution, and the corrected (item-excluded) item-total correlation, which excludes it. For a dichotomous item, the raw item-total correlation is mathematically equivalent to the point-biserial correlation. `correct` selects which of the two feeds the discrimination flagging criterion (`crit.dis`); both are always reported as separate columns regardless of `correct` (this mirrors how, e.g., `psych::alpha()` reports the raw item-total correlation as `raw.r` and the corrected version as `r.drop`).

Alpha-with-item-removed for item j is Cronbach's alpha (see below) recomputed using only the remaining items, so that a value above the overall alpha identifies an item whose removal would increase the reliability of the test.

At the test level, two forms of Cronbach's alpha are always computed and reported. Raw alpha uses the standard variance-based formula

$$\alpha = \frac{k}{k-1}\left(1 - \frac{\sum_j \sigma_j^2}{\sigma_X^2}\right),$$

where k is the number of items, σ_j^2 is the variance of item j, and σ_X^2 is the variance of the total score X. Raw alpha reflects the reliability of the actual (unweighted) total score obtained by simply summing the item scores - the score most tests actually use for reporting and decisions. Standardized alpha instead first standardizes every item to unit variance before combining them,

$$\alpha_{std} = \frac{k\bar{r}}{1 + (k-1)\bar{r}},$$

where $\bar{r}$ is the average pairwise correlation among all items. Standardized alpha differs meaningfully from raw alpha mainly when items vary substantially in scale or format (e.g., a mix of dichotomous and polytomous items with very different score ranges); when all items share the same scale and format, the two are typically close, and raw alpha remains the more directly interpretable of the two since it matches the reliability of the score actually used in practice. See Cronbach (1951) for the original derivation of coefficient alpha, and Osburn (2000) for a discussion contrasting the raw (covariance-based) and standardized (correlation-based) forms.

The standard error of measurement (SEM) is $SEM = SD(X)\sqrt{1-\alpha}$ (using raw alpha), the standard CTT relationship between test reliability and measurement precision.

The total-score frequency distribution is computed internally by `freq_score()`, using the row sums of `data` after applying the same `missing` recoding and listwise deletion of incomplete rows used throughout the rest of this function, so that the frequency distribution reflects exactly the same set of examinees and the same total-score definition used elsewhere in the analysis. `freq_score()` is also exported separately, for callers who only need a frequency distribution for an arbitrary vector of integer total scores.

Value

An object of class `"ctt"`, a list with elements:

`item`	A data frame with one row per item, containing the item label, number of score categories, difficulty, the raw (uncorrected) item-total correlation (`discrimination_raw`), the corrected (item-excluded) item-total correlation (`discrimination_corrected`), alpha-with-item-removed, and, if `flag = TRUE`, a `flag` column.
`crit`	A list echoing the `crit.p` and `crit.dis` thresholds used for flagging.
`alpha`	A one-row test-level summary data frame containing `n_examinee`, `n_item`, `alpha`, `alpha_std`, `sem`, `mean_difficulty`, `mean_discrimination_raw`, and `mean_discrimination_correct` Both the raw (`alpha`) and standardized (`alpha_std`) forms of Cronbach's alpha are always included; standardized alpha standardizes every item to unit variance before combining them, which differs meaningfully from raw alpha mainly when items vary widely in scale or format.

`freq` The total-score frequency distribution table returned by `freq_score()` (`score`, `freq`, `pct`, `cum_pct`).

`call` The matched call, as used by the `print()`/`summary()` methods for `"ctt"` objects.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297-334. doi:10.1007/BF02310555.

Osburn, H. G. (2000). Coefficient alpha and related internal consistency reliability coefficients. *Psychological Methods, 5*(3), 343-355. doi:10.1037/1082989X.5.3.343.

See Also

`freq_score()`, `print.ctt()`, `summary.ctt()`

Examples

```
# A dichotomous example: simulate the responses of 300 examinees to 15
# 3PLM items, then run ctt() on the simulated data
set.seed(1)
x_bin <- shape_df(
  par.drm = list(a = rep(1.5, 15), b = seq(-1.5, 1.5, length.out = 15),
                 g = rep(0.2, 15)),
  cats = 2, model = "3PLM"
)
dat <- simdat(x = x_bin, theta = rnorm(300), D = 1)
out <- ctt(data = dat)
out
summary(out)

# A more realistic, mixed-format example: simulate response data for a
# 55-item test (50 dichotomous 3PLM items + 5 polytomous GRM items) from
# item parameters imported from a flexMIRT output file bundled with irtQ,
# then run ctt() on the simulated data
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df
set.seed(2)
theta <- rnorm(1000)
dat_mixed <- simdat(x = x, theta = theta, D = 1)
out_mixed <- ctt(data = dat_mixed, item.id = x$id, cats = x$cats)
summary(out_mixed)
```

ctt_distr *Option/Category Response Distribution and Distractor Analysis*

Description

Computes, for each item, the percentage of examinees choosing each response option (or, for already-scored polytomous data, each score category), together with each option's/category's point-biserial correlation with the total score. When an answer key is supplied, the function operates in "selected-response" mode - treating `data` as raw, unscored option responses (as consumed by `score_resp()`) - and additionally reports omission/double-marking rates and flags distractors (incorrect options) whose correlation with the total score is positive, a common indicator of a miskeyed or poorly written item. When no key is supplied, the function operates in "scored-category" mode - treating `data` as an already-scored response matrix (0, 1, 2, ..., cats - 1) - and simply reports each score category's response proportion and category-total correlation, with no notion of a "correct" option to flag distractors against.

Usage

```
ctt_distr(
  data,
  item.id = NULL,
  key = NULL,
  opt = NULL,
  cats = NULL,
  total = NULL,
  correct = FALSE,
  missing = NA,
  crit.distractor = 0
)
```

Arguments

`data`	In selected-response mode (`key` supplied), a data frame or matrix of raw item responses in the same format as the `data` argument of `score_resp()` (a selected option number, a missing-response indicator, or a comma-separated string such as `"1,5"` for a double-marked response). In scored-category mode (`key = NULL`), a data frame or matrix of already-scored item responses with scores 0 to `cats[j] - 1` for each item j; an error is raised when a score is not a whole number between 0 and `cats[j] - 1`. Either way, examinees are rows and items are columns.
`item.id`	A character vector of item identifiers, in the same order as the columns of `data`. If `NULL` (default), item IDs are generated automatically as `paste0("V", 1:ncol(data))`, following the convention used elsewhere in irtQ (e.g., `shape_df()`, `est_irt()`). Note that this generated ID does not fall back to `colnames(data)`; pass `item.id` explicitly to label items using the column names of `data` or any other identifier scheme.

key — NULL (default) for scored-category mode, or - to use selected-response mode - either a vector of correct options in item order, or a data frame with columns item and key, exactly as accepted by score_resp(). As in score_resp(), each item's option-coding scheme (numeric, Latin-letter, or general label such as a Korean syllable label) is inferred independently from its own key value; see **Details**.

opt — A vector giving the full set of possible response option values in selected-response mode (e.g., 1:5 for a five-option item, or c("A", "B", "C", "D", "E") for a letter-coded item), applied to every item. Only used when key is supplied; it is silently ignored in scored-category mode (key = NULL), which instead derives its category set from cats. If NULL (default), the option set is inferred as the sorted set of all single (non-blank, non-double-marked) response values observed anywhere in data, using each item's own option-coding scheme to decide what counts as a single response (see **Details**). Because opt is shared across every item, ctt_distr() assumes a test form uses one consistent option-coding scheme throughout; see **Details** for what happens if items actually differ.

cats — A numeric vector giving the number of score categories per item, following the same convention as ctt(). Only used in scored-category mode (key = NULL); silently ignored when key is supplied. If NULL (default), inferred per item as the observed maximum score plus one, with a minimum of two. Each value must be a whole number of at least 2.

total — An optional numeric vector of total scores, one per examinee, aligned with the rows of data. If NULL (default), the total score is computed internally: via score_resp() in selected-response mode, or as rowSums(data) in scored-category mode (see **Details** for how total interacts with missing/listwise deletion in scored-category mode).

correct — Logical. Both the raw (uncorrected) item-total correlation - where an item (or, in ctt_distr(), a response option/category) is correlated with the total score that includes its own contribution - and the corrected item-total correlation - excluding its own contribution - are always computed and reported as separate columns (see **Details/Value**). This argument only controls which of the two is used for flagging: if TRUE, flagging (in ctt_item()'s flag column, ctt_alpha()'s summary discrimination figure, and, in ctt_distr(), the crit.distractor check) is based on the corrected value; if FALSE (default), on the raw value.

missing — A value indicating missing responses, analogous to the missing argument in est_irt() and score_resp(). Its effect differs by mode (see **Details**): in selected-response mode, a cell equal to missing is recoded to NA and then treated as a blank (omitted) response, contributing to that item's omission rate, without removing the examinee from the analysis. In scored-category mode, a cell equal to missing is recoded to NA and the examinee's row is then excluded listwise, as in ctt(). Default is NA.

crit.distractor — A single numeric value; in selected-response mode, a distractor (non-key option) is flagged when its point-biserial correlation with the total score exceeds this value. Default is 0.

Details

In selected-response mode, each item's option-coding scheme is inferred from its own `key` value, using exactly the same three-way rule as `score_resp()`: a key value that parses as a number is compared numerically; a key value consisting of one or more Latin letters is compared as letters, case-insensitively, and a response must itself consist only of Latin letters to count as a single valid option; any other key value (e.g., a Korean syllable label, a Roman numeral written in a non-Latin numeral script, or a circled-number symbol) is treated as a general option label, under which any non-blank, non-double-marked response is accepted as a single valid option and compared to the key (and to `opt`) with a case-insensitive exact string match. This ensures `ctt_distr()` interprets a given `data`/`key` pair identically to `score_resp()`. Because `opt` (the option universe used for every item's frequency table) is a single vector shared across all items, `ctt_distr()` is intended for a test form that uses one consistent option-coding scheme throughout (e.g., every item numbered 1-5, or every item lettered A-E); mixing schemes across items within a single call is not well-supported, since the shared `opt` cannot represent two different option universes at once.

The two modes handle missing responses differently because they represent different kinds of data. In selected-response mode, an omitted response is itself a meaningful, common outcome for a selected-response item (the examinee saw the item and chose not to answer it), so omissions are tabulated as their own category rather than causing the examinee to be dropped, exactly as in `score_resp()`. In scored-category mode, `data` is assumed to already be a scored response matrix (as used by `ctt()`), where a missing cell instead means the item was not administered to, or not observed for, that examinee, so the examinee's row is excluded listwise for that analysis. When `total` is supplied together with scored-category mode's listwise deletion, `total` is assumed to align with the *original* rows of `data`; it is subset using the same completeness mask before use, so its length need only match the original `nrow(data)`.

Point-biserial correlations are computed as the Pearson correlation between a 0/1 indicator (whether the examinee chose that particular option, or scored in that particular category) and the total score. As with the item-level discrimination statistics computed by `ctt()`, both the raw (uncorrected, item/option-included) and corrected (item-excluded) versions are always computed and returned as separate columns (`pb_raw`/`pb_corrected`); `correct` only selects which of the two feeds the `crit.distractor` flagging criterion in selected-response mode. A point-biserial correlation is undefined, and reported as `NA`, whenever an option/category was chosen by zero or all examinees (a constant indicator).

Value

A list with these elements:

`distr` A data frame in long format with one row per item x option (selected-response mode) or item x category (scored-category mode), containing `item`, `option` (the option number, option label, or score category; numeric for a numeric-coded or scored-category test form, character for a letter- or general-label-coded test form), `is_key` (logical; whether this option is the item's correct answer, or `NA` in scored-category mode), `freq` (the raw number of examinees choosing this option/category), `pct` (the same count expressed as a percentage of examinees), `pb_raw` (raw, uncorrected point-biserial correlation with the total score), `pb_corrected` (corrected, item-excluded point-biserial correlation with the total score), and `flag` (a description of `crit.distractor` violations, based

on whichever of pb_raw/pb_corrected is selected by correct, in selected-response mode, or "" otherwise).

omit — In selected-response mode, a data frame with one row per item giving the blank/omission and double-marked response percentages (pct_blank, pct_double). NULL in scored-category mode.

Note

Selected-response mode (key supplied) is restricted to **dichotomous, single-key items only**, exactly like score_resp(), which it calls internally to obtain the 0/1 scored responses and total score: a selected-response item has exactly one correct option and therefore cannot yield a partial-credit score by construction, so a mixed dichotomous/polytomous test form cannot be analyzed in this mode. Scored-category mode (key = NULL) has no such restriction: because cats is always a per-item vector, a single call fully supports a genuinely mixed dichotomous/polytomous test form (see **Examples**). If raw selected-response data and already-scored polytomous data need to be analyzed together, score the selected-response items first (via score_resp() or this function's own selected-response mode), then combine the result with the polytomous scores and call ctt_distr() again in scored-category mode on the combined matrix.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

score_resp(), ctt()

Examples

```
# Selected-response mode: raw option responses + an answer key
raw <- data.frame(
  V1 = c("1", "2", "1", "1", "3", "2", "1", "4"),
  V2 = c("4", "4", "2", "2,4", "4", "1", "4", "4")
)
key <- c(1, 4)
out <- ctt_distr(data = raw, key = key)
out$distr
out$omit

# Scored-category mode: an already-scored polytomous matrix, no key
set.seed(1)
scored <- data.frame(matrix(sample(0:2, 8 * 3, replace = TRUE), nrow = 8))
out2 <- ctt_distr(data = scored)
out2$distr

# Scored-category mode also handles a genuinely mixed dichotomous/
# polytomous test form in a single call, since `cats` is always a
# per-item vector (see Note): item 3 below contributes 3 category rows
# while items 1-2 contribute 2 each
mixed <- data.frame(
  I1 = c(1, 0, 1, 1, 0, 1, 0, 1),   # dichotomous item (cats = 2)
```

```
  I2 = c(0, 1, 0, 1, 1, 0, 0, 1),   # dichotomous item (cats = 2)
  I3 = c(2, 0, 1, 2, 1, 0, 2, 1)    # polytomous item (cats = 3)
)
out2b <- ctt_distr(data = mixed, cats = c(2, 2, 3))
out2b$distr

# Selected-response mode with letter-coded (A-E) options: each item's
# coding scheme is inferred from its own key value, exactly as in
# irtQ::score_resp()
raw3 <- data.frame(
  V1 = c("A", "B", "A", "A", "C", "B", "A", "D"),
  V2 = c("D", "D", "B", "B,D", "D", "A", "D", "D")
)
key3 <- c("A", "D")
out3 <- ctt_distr(data = raw3, key = key3)
out3$distr
```

---

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 less than or equal to this value. When FIPC is used and all items are fixed, so that only the latent ability distribution is estimated, the criterion is applied to the largest absolute change in the mean and variance of the prior distribution.

`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, `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()`.

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 largest absolute change in the estimates in the last EM cycle.
EMtime	Time (in seconds) spent on EM cycles.
SEtime	Time (in seconds) spent computing standard errors.
TotalTime	Total computation time (in seconds).
test.1	A message indicating whether the convergence criteria were met: the M-step optimization converged for every item and the largest absolute change in the parameter estimates between two consecutive EM cycles was less than or equal to Etol. When fipc.method = "OEM" with new items to calibrate, the single EM cycle is not judged by Etol and only the M-step check applies. For FIPC with all items fixed, see Etol.
test.2	Second-order test result indicating whether the information matrix is positive definite, a necessary condition for identifying a local maximum. The message reports a possible local maximum only when the first-order test is also satisfied.
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. doi:10.1111/j.17453984.2001.tb01123.x.

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, 16)),
  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. doi:10.1111/j.17453984.2001.tb01123.x.

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, 16)), 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, 16)), 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 less than or equal to this value. When MG-FIPC is used and all items are fixed, so that only the latent ability distributions are estimated, the criterion is applied to the largest absolute change in the means and variances of the prior distributions.

`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, 1997) 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, 1997). 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_mg`. 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 largest absolute change in the estimates in the last EM cycle.
EMtime	Time (in seconds) spent on EM cycles.
SEtime	Time (in seconds) spent computing standard errors.
TotalTime	Total computation time (in seconds).
test.1	A message indicating whether the convergence criteria were met: the M-step optimization converged for every item and the largest absolute change in the parameter estimates between two consecutive EM cycles was less than or equal to Etol. When fipc.method = "OEM" with new items to calibrate, the single EM cycle is not judged by Etol and only the M-step check applies. For FIPC with all items fixed, see Etol.
test.2	Second-order test result indicating whether the information matrix is positive definite, a necessary condition for identifying a local maximum. The message reports a possible local maximum only when the first-order test is also satisfied.
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). Springer. doi:10.1007/9781475726916_25.

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 (Gonzalez, 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 one of "ML", "MLF", "WL", or "MAP", the standard error is set to 99.9999 when the ability estimate equals a limit of range.

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. *Applied Psychological Measurement, 6*(4), 431-444. doi:10.1177/014662168200600405.

Gonzalez, J. (2014). SNSequate: Standard and nonstandard statistical models and methods for test equating. *Journal of Statistical Software, 59*(7), 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. doi:10.1177/0146621616631317.

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.). 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. doi:10.1111/jedm.12276.

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. L. (1995). Item response theory for scores on tests including polytomous items with ordered responses. *Applied Psychological Measurement, 19*(1), 39-49. doi:10.1177/014662169501900105.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427-450. doi:10.1007/BF02294627.

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)
)
```

`find_cut` *Find TIF-Crossing Cut Scores for MST Routing*

Description

Computes routing cut scores for a multistage test (MST) by identifying the theta values at which adjacent modules' Test Information Functions (TIFs) cross in the correct direction. The resulting cut scores can be passed directly to `run_mst` as its `cut_score` argument (with `route_method = NULL`), providing a principled and psychometrically grounded alternative to heuristic cut-score selection.

Usage

```
find_cut(
  x,
  module,
  route_map,
  D = 1,
  theta_range = c(-6, 6),
  n_grid = 2001L,
  ref_theta = 0
)
```

Arguments

`x`	A data frame of item metadata in the standard **irtQ** format (columns: `id`, `cats`, `model`, `par.1`, `par.2`, ...). See `shape_df` for details.
`module`	A binary integer matrix of dimensions $J \times M$, where J is the number of items and M is the number of modules. A value of 1 at row j, column m indicates that item j belongs to module m. This is the same `module` argument used in `run_mst`.
`route_map`	A binary square matrix of dimensions $M \times M$ defining the MST transition structure. A value of 1 at row i, column j indicates that examinees can be routed from module i to module j. This is the same `route_map` argument used in `run_mst`.
`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.

`theta_range`	A numeric vector of length 2 specifying the theta range over which to search for TIF crossings. Default is `c(-6, 6)`.
`n_grid`	An integer specifying the number of equally spaced theta points used for the initial sign-change scan. A finer grid reduces the chance of missing a crossing but increases computation time. Default is `2001`.
`ref_theta`	A single numeric value used to break ties when multiple proper crossings are found for an adjacent module pair. The crossing closest to `ref_theta` is selected. Default is `0`.

Details

Motivation: the MFI routing problem: When `run_mst` uses Maximum Fisher Information (MFI) routing (`route_method = "mfi"`), it selects the next-stage module that maximizes TIF at the examinee's current ability estimate. In principle, a harder module should always have a higher TIF for high-ability examinees, and a lower TIF for low-ability examinees, making MFI routing work as intended. However, in practice, a harder module's TIF curve can peak at a theta value that is unexpectedly low, so that the harder module *also* has the highest TIF near the low end of the ability scale. In this case, a low-ability examinee is incorrectly routed into the harder module – the opposite of what the MST is designed to do, which is called an *anomalous routing* or *path reversal*.

Solution: TIF-crossing cut scores: `find_cut()` resolves this problem by computing cut scores from the TIF curves themselves, rather than relying on pure MFI at the time of routing. Specifically, for each pair of adjacent modules (ordered by mean item difficulty), the function finds the theta value at which the harder module's TIF first overtakes the easier module's TIF *from below* as theta increases. This crossing point is the natural boundary between the two modules: below it, the easier module is more informative; above it, the harder module is more informative. Using this point as a fixed cut score pre-empts the path reversals that MFI routing produces at the extremes of the scale.

Proper vs. anomalous crossings: Two TIF curves may cross more than once. `find_cut()` distinguishes two types of crossing:

- **Proper crossing**: the difference $\text{TIF}_{\text{harder}}(\theta) - \text{TIF}_{\text{easier}}(\theta)$ changes sign from negative to positive (positive slope at the crossing). This is the correct, expected crossing in the middle of the theta scale.
- **Anomalous crossing**: the difference changes sign from positive to negative (negative slope at the crossing). This corresponds to the pathological situation in which the harder module temporarily dominates at the low end of the theta scale – exactly the pattern that causes path reversals under MFI routing.

Only proper crossings are used as cut scores. Anomalous crossings are reported in `$details` for diagnostic purposes but excluded from the returned `cut_score`. If no proper crossing is found for a module pair, an informative error is thrown. If multiple proper crossings are found (unusual for well-designed modules), a warning is issued and the one closest to `ref_theta` is selected.

Algorithm: The function proceeds stage by stage (from stage 2 onward). At each stage:

1. For each adjacent pair of modules in the order supplied by the user (ascending module index), the difference $\text{TIF}_{\text{right}}(\theta) - \text{TIF}_{\text{left}}(\theta)$ is evaluated on a fine theta grid of `n_grid` points, where "left" and "right" refer to the lower- and higher-index module in the pair. Sign changes on the grid are detected and then refined with `uniroot`.

2. The slope at each refined root is estimated by a finite difference (step size 0.01) to classify it as proper or anomalous.
3. The selected proper crossing is stored as the cut score for that pair.

Module index vs. difficulty order: `run_mst` internally orders candidate modules by ascending module index and maps routing rank 1 to the lowest-index module. For cut-score routing to assign low-theta examinees to the easiest module, the modules at each stage must be numbered in ascending difficulty order (easiest module = lowest index). `find_cut()` checks this assumption and issues a warning if the difficulty order does not match the index order. Regardless of this warning, cut scores are always computed between consecutive module pairs in the supplied index order (e.g., modules 2 & 3 and 3 & 4, not 2 & 4 and 4 & 3). The returned cut scores within each stage are sorted in ascending order, as required by `cut`.

Value

An object of class `"find_cut"`, which is a named list with three elements:

`cut_score` A list of length S - 1 (where S is the number of stages), compatible with the `cut_score` argument of `run_mst`. Each element is a numeric vector of cut scores for the transition into that stage, sorted in ascending order. Stages with only one module have an empty numeric vector (`numeric(0)`).

`details` A named list of per-stage diagnostic information, including module indices, mean item locations, difficulty-sorted module order, and per-pair crossing details (proper crossings, anomalous crossings, and the selected cut score).

`tif_data` A `tibble` with columns `theta`, `module`, `stage`, and `tif`, providing TIF curves for all modules at all stages (including stage 1). Used by `plot.find_cut` to visualise TIF curves and crossing points.

See Also

`run_mst`, `panel_info`, `reval_mst`, `info`

Examples

```
## -- Setup: use the built-in simMST 1-3-3 panel --------------------------
## simMST is a 7-module, 3-stage MST panel (8 dichotomous 3PLM items each).
## Modules 1 (routing), 2-4 (stage 2: easy/medium/hard),
##         5-7 (stage 3: easy/medium/hard).
x         <- simMST$item_bank
module    <- simMST$module
route_map <- simMST$route_map

## -- Find TIF-crossing cut scores ----------------------------------------
## For each adjacent module pair at stages 2 and 3, find_cut() identifies
## the theta at which the harder module's TIF first exceeds the easier
## module's TIF (proper crossing), and returns it as a cut score.
cut_result <- find_cut(x = x, module = module, route_map = route_map, D = 1.702)

## Print a summary: crossing points found, anomalous crossings excluded,
## and the final selected cut scores per stage transition.
```

```
print(cut_result)

## -- Visualise TIF curves and cut scores ---------------------------------
## plot() shows TIF curves for every module faceted by stage.
## Selected cut scores appear as solid black vertical lines with theta
## labels at the crossing point. Anomalous crossings (if any) appear
## as dashed red lines.
plot(cut_result)                        # stages stacked vertically (default)
plot(cut_result, layout = "horizontal")  # stages side by side
plot(cut_result, show_anomalous = FALSE) # hide anomalous crossing markers

## Inspect the cut_score element -- a list directly compatible with run_mst()
## cut_score[[1]]: two cut scores for the stage-1 -> stage-2 transition
## cut_score[[2]]: cut scores for the stage-2 -> stage-3 transition
cut_result$cut_score

## Compare with the cut scores stored in simMST, which were obtained the
## same way; the two sets of values are identical
simMST$cut_score

## -- Use the cut scores in run_mst() -------------------------------------
## Pass cut_result$cut_score directly to run_mst() with route_method = NULL.
## This replaces pure MFI routing with TIF-crossing-based fixed cut scores,
## preventing anomalous path reversals at the extremes of the theta scale.
## Not run:
set.seed(1)
theta_true <- rnorm(500)
result <- run_mst(
  x            = x,
  route_map    = route_map,
  module       = module,
  theta        = theta_true,
  route_method = NULL,
  cut_score    = cut_result$cut_score
)

## End(Not run)
```

`freq_score` *Frequency Distribution Table for Total Scores*

Description

Computes a frequency distribution table for a vector of total (raw) scores - frequency, percentage, and cumulative percentage for each score value - commonly reported alongside classical test theory (CTT) item analysis results. This is the same function `ctt()` calls internally to build the total-score frequency distribution included in its output, but it is also exported and fully usable on its own for any vector of integer total scores.

Usage

```
freq_score(score, missing = NA)
```

Arguments

score	A numeric vector of total (raw) scores, one value per examinee.
missing	A value indicating missing scores in score, analogous to the missing argument in est_irt() and score_resp(). Any element equal to missing is recoded to NA before tabulation. Default is NA. Examinees with a (remaining) missing score are excluded from the table, with a warning reporting how many were dropped.

Details

The table includes every integer score value spanning the observed minimum to maximum score (not just the values that were actually observed), so that a score with zero examinees still appears in the table with a frequency of 0, matching how a raw-score frequency table is conventionally reported. Percentages are computed relative to the number of non-missing scores and rounded to two decimal places; cumulative percentages are the running sum of the unrounded percentages, rounded to two decimal places only in the final output, so that rounding error does not accumulate across rows and the final cumulative percentage totals almost exactly 100 (subject only to ordinary rounding).

This function assumes scores already lie on an integer (or otherwise evenly spaced discrete) scale, as is standard for a raw total score; it does not bin or group continuous values, and raises an error if any non-integer score is supplied rather than silently dropping it.

The output table always spans min(score) to max(score), so its size scales with the observed score *range*, not the sample size; a single unusually large or small outlier score will produce a correspondingly large table.

Value

A data frame with one row per score value from min(score) to max(score), containing:

score	the score value.
freq	the number of examinees with that score.
pct	the percentage of examinees with that score.
cum_pct	the cumulative percentage up to and including that score.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

ctt()

Examples

```
set.seed(1)
score <- rbinom(500, size = 20, prob = 0.6)
freq_score(score)
```

---

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 for the E-step of the EM algorithm: the largest absolute change in the item parameter estimates between consecutive cycles. For FIPC with all items fixed, it applies to the change in the mean and variance of the prior distribution.

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 largest absolute change in the estimates in the last EM cycle.

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 A message indicating whether the convergence criteria were met (M-step convergence and the EM criterion).

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 The convergence criterion for the E-step of the EM algorithm: the largest absolute change in the item parameter estimates between consecutive cycles. For FIPC with all items fixed, it applies to the change in the mean and variance of the prior distribution.

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

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

bprior A list describing the prior distribution for item difficulty 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 The largest absolute change in the estimates in the last EM cycle.

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 A message indicating whether the convergence criteria were met (M-step convergence and the EM criterion).

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`. If the limit is reached while flagged items remain, a warning is issued and the `complete` element of the purification results is `FALSE`; for tests with many items or many DIF items, consider increasing `max.iter`.
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.

dif_item A numeric vector of the positions (rows of `x`) of the items flagged as DIF by the `purify.by` statistic across all purification iterations, sorted in ascending order.

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 & Penfield, 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 & Penfield, 2015).
- The G^2 statistic is approximately chi-square distributed with degrees of freedom equal to the number of ability groups (Ames & Penfield, 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.

Examples

```
# structure of the data
head(LSAT6)

# fit the 2PL model to the LSAT6 data
est_irt(data = LSAT6, D = 1, model = "2PLM", cats = 2, verbose = FALSE)
```

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.). 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 three items
# (column 1: incorrect responses, column 2: correct responses)
probs <- matrix(c(.26, .27, .18, .74, .73, .82), 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)
```

---

panel_info *Extract Panel Structure from an MST Route Map*

Description

Given a binary transition matrix that defines the Multistage-Adaptive Test (MST) structure, this function extracts key panel information: the module configuration by stage, all valid test pathways, the number of modules per stage, and the total number of stages.

Usage

```
panel_info(route_map)
```

Arguments

`route_map`	A binary square matrix defining the MST transition structure, where a value of 1 at row *i*, column *j* indicates that a test taker can be routed from module *i* to module *j*. This is equivalent to the `transMatrix` argument in the `randomMST()` function of the **mstR** package (Magis et al., 2017). See `reval_mst` for details on how to construct this matrix.

Details

The function identifies stage-1 modules as those with no incoming transitions (column sum of `route_map` equals zero). It then traverses the route map stage by stage, collecting modules reachable from the previous stage. All logically possible pathways are enumerated via `expand.grid()`, and pathways that violate the transition constraints in `route_map` are removed. The remaining pathways are sorted and returned as a matrix.

Value

A named list with four elements:

`config` A named list of length equal to the number of stages. Each element is an integer vector of module indices belonging to that stage. Names are `"stage.1"`, `"stage.2"`, etc.

`pathway` An integer matrix of all valid test pathways, with one row per pathway and one column per stage. Row names are `"path.1"`, `"path.2"`, etc. Columns correspond to the module index administered at each stage.

`n.module` A named integer vector giving the number of modules in each stage.

`n.stage` An integer scalar giving the total number of stages.

References

Magis, D., Yan, D., & von Davier, A. A. (2017). *Computerized adaptive and multistage testing with R: Using packages catR and mstR*. Springer. doi:10.1007/9783319692180.

See Also

`reval_mst`, `run_mst`

Examples

```
# Use the 1-3-3 simMST panel route map
route_map <- simMST$route_map
pinfo <- panel_info(route_map)

# Stage configuration: which modules belong to which stage
pinfo$config

# All valid pathways (rows = pathways, columns = stages)
pinfo$pathway

# Number of modules per stage
pinfo$n.module

# Total number of stages
pinfo$n.stage
```

---

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 observed pseudo-frequencies are computed using the posterior distribution of each examinee's ability, whereas the expected frequencies are the response probabilities implied by the item parameters in the item bank (Stone, 2000). This provides a sensitive measure of item parameter drift, helping to ensure 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. The user must supply it; no default threshold is built in. If NULL (default), the statistic is reported but no item is flagged and purification is not performed.
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`. If the limit is reached while flagged items remain, a warning is issued and the `complete` element of the purification results is `FALSE`; for tests with many items or many DIF items, consider increasing `max.iter`.

`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 observed pseudo-frequencies are obtained from the posterior distribution of each examinee's ability, and the expected frequencies are the category probabilities computed from the item bank parameters (Stone, 2000).

The Pseudo-count D^2 statistic is calculated as:

$$Pseudo-countD^2 = \sum_{k=1}^{Q} \left(\frac{r_{0k} + r_{1k}}{N} \right) \left(\frac{r_{1k}}{r_{0k} + r_{1k}} - E_{1k} \right)^2$$

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)


## -- Example 3: CAT-based IPD detection using simIPD --------------------------
##
## The Pseudo-count D2 statistic has no closed-form null distribution.
## Following Lim & Han (2026), the critical value is estimated empirically
## via bootstrap:
##   (1) Select drift-free (anchor) items to form the null D2 distribution.
##   (2) Repeatedly resample from those values and take the 95th percentile.
##   (3) Average across bootstrap iterations to obtain the critical value.
##
## The simIPD dataset provides:
##   item_par  : original 360-item pool (3PLM, irtQ format)
##   foc_resp  : focal group CAT responses (N = 3,000; TL = 30; ~92% NA)
##   key_item  : indices of 90 highly-exposed key items (IPD analysis targets)
##   item.skip : indices of 270 non-key items (excluded from analysis)
##   ipd_item  : 18 truly drifted items (ground truth; a & b each -0.5)
## ------------------------------------------------------------------------------

data(simIPD)

## -- Step 1. Select anchor items for bootstrap ----------------------------
## From the full item pool, select items that (a) have sufficient observed
## responses (>= boot_size) and (b) are not known IPD items. These items
## serve as the empirical null distribution of D2.
## boot_size = 300: chosen to match the minimum response count used in the
##   bootstrap procedure of the simulation study (Lim & Han, 2026).
## In practice, exclude items you know or suspect have drifted; here the
## ground truth (simIPD$ipd_item) is used for illustration.
boot_size    <- 300
n_resp       <- colSums(!is.na(simIPD$foc_resp))   # observed responses per item
anchor_items <- setdiff(which(n_resp >= boot_size), simIPD$ipd_item)

## -- Step 2. Compute null D2 values for the anchor items ------------------
```

```
pcd2_null <- pcd2(
  x        = simIPD$item_par[anchor_items, ],
  data     = simIPD$foc_resp[, anchor_items],
  D        = 1.7,
  crit.val = NULL,
  purify   = FALSE
)$no_purify$ipd_stat$pcd2

## -- Step 3. Bootstrap critical value (Lim & Han, 2026) --------------
## For each bootstrap iteration: resample boot_size D2 values from the null
## distribution and take the 95th percentile. The critical value is the mean
## of these percentiles across all iterations.
## n_boots = 500: reduced for demo speed; use 10,000 in practice for a
##   stable critical value estimate.
set.seed(2024)
n_boots <- 500

crit_val <- purrr::map_dbl(
  .x = 1:n_boots,
  .f = ~ sample(x = pcd2_null, size = boot_size, replace = TRUE) |>
           stats::quantile(probs = 0.95)
) |> mean()

cat("Bootstrap critical value (alpha = 0.05):", round(crit_val, 6), "\n")

## -- Step 4. Run PCD2 with the bootstrap critical value + purification --
pcd2_result <- pcd2(
  x         = simIPD$item_par,
  data      = simIPD$foc_resp,
  D         = 1.7,
  item.skip = simIPD$item.skip,
  crit.val  = crit_val,
  purify    = TRUE,
  max.iter  = 30
)

## -- Step 5. Compare detected items to ground truth --------------------
detected <- pcd2_result$with_purify$ipd_item
cat("Truly drifted items (ground truth):", simIPD$ipd_item, "\n")
cat("PCD2-detected items:               ", detected, "\n")
cat("True positives:", sum(detected %in% simIPD$ipd_item), "of",
    length(simIPD$ipd_item), "\n")
```

`plot.find_cut` *Plot TIF Curves and Cut Scores from a* `find_cut` *Result*

Description

Produces a **ggplot2** visualisation of the Test Information Function (TIF) curves for each module, faceted by stage, with vertical lines marking the selected cut scores (proper TIF crossings) and, optionally, any anomalous crossings detected by `find_cut`.

Each stage occupies one panel. Stage 1 (routing module) appears first. Stages with routing decisions show TIF curves for all candidate modules and mark the selected cut score(s) with a solid black vertical line and a theta value label at the crossing point.

Usage

```
## S3 method for class 'find_cut'
plot(
  x,
  theta_range = NULL,
  show_anomalous = TRUE,
  label_cuts = TRUE,
  layout = c("vertical", "horizontal"),
  ...
)
```

Arguments

`x`	An object of class `"find_cut"` returned by `find_cut`.
`theta_range`	A numeric vector of length 2 specifying the theta range shown on the x-axis. Defaults to the full range stored in `x$tif_data`.
`show_anomalous`	Logical. If `TRUE` (default), anomalous TIF crossings (negative-slope crossings excluded from the cut scores) are shown as dashed red vertical lines.
`label_cuts`	Logical. If `TRUE` (default), each selected cut score is annotated with its theta value above the crossing point.
`layout`	Character string controlling the panel arrangement. `"vertical"` (default) stacks each stage in its own row (one column of panels, stage 1 at the top). `"horizontal"` places each stage in its own column (one row of panels, stage 1 at the left).
`...`	Currently unused. Reserved for future arguments.

Value

A `ggplot` object. The object is printed automatically when called interactively. It can be further customised with standard **ggplot2** functions such as `ggplot2::theme()`, `ggplot2::scale_color_manual()`, etc.

See Also

`find_cut`, `run_mst`

Examples

```
## Use the built-in simMST 1-3-3 panel
cut_result <- find_cut(
  x         = simMST$item_bank,
  module    = simMST$module,
  route_map = simMST$route_map,
  D         = 1.702
)

## Default: stages stacked vertically (stage 1 at the top)
plot(cut_result)

## Horizontal layout: stages side by side
plot(cut_result, layout = "horizontal")

## Hide anomalous crossing markers
plot(cut_result, show_anomalous = FALSE)
```

`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)
```

`print.ctt` *Print a Combined CTT Analysis*

Description

Prints a condensed, headline-style summary of an object of class `"ctt"` returned by `ctt()`, following the same brief/full print-summary split used by `est_irt()` (compare `print.est_irt()` vs. `summary.est_irt()`/`print.summary.est_irt()`).

Usage

```
## S3 method for class 'ctt'
print(x, digits = 3, ...)
```

Arguments

`x`	An object of class `"ctt"`, as returned by `ctt()`.
`digits`	Number of decimal places used when rounding numeric values for display. Default is `3`. Note that the item- and test-level statistics bundled into `x` are already rounded to 3 decimal places before `ctt()` returns them, so a `digits` value greater than 3 here cannot recover precision that was already discarded upstream.
`...`	Additional arguments passed to or from other methods (currently not used).

Value

`x`, invisibly.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

`ctt()`, `summary.ctt()`

Examples

```
# simulate the responses of 300 examinees to 15 dichotomous 3PLM items
set.seed(1)
x <- shape_df(
  par.drm = list(a = rep(1.5, 15), b = seq(-1.5, 1.5, length.out = 15),
                 g = rep(0.2, 15)),
  cats = 2, model = "3PLM"
)
dat <- simdat(x = x, theta = rnorm(300), D = 1)

# run the CTT analysis
out <- ctt(data = dat)

# print the condensed report
print(out)
```

---

print.summary.ctt *Print the Full Report for a Combined CTT Analysis*

Description

Displays the complete classical test theory (CTT) report for an object of class `"summary.ctt"` produced by `summary.ctt()`: the function call, the full item-level statistics table (including flags, if present), the test-level reliability summary, and the total-score frequency distribution, following the sectioned reporting style used by `print.summary.est_irt()` in irtQ.

Usage

```
## S3 method for class 'summary.ctt'
print(x, digits = 3, ...)
```

Arguments

`x`	An object of class `"summary.ctt"`, as returned by `summary.ctt()`.
`digits`	Number of decimal places used when rounding numeric values for display. Default is 3. Note that the statistics bundled into `x` are already rounded to 3 decimal places upstream, so a `digits` value greater than 3 here cannot recover precision that was already discarded; `digits` is only useful for displaying the report at 3 or fewer decimal places.
`...`	Additional arguments passed to or from other methods (currently not used).

Value

`x`, invisibly.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

ctt(), summary.ctt()

Examples

```
# simulate the responses of 300 examinees to 15 dichotomous 3PLM items
set.seed(1)
x <- shape_df(
  par.drm = list(a = rep(1.5, 15), b = seq(-1.5, 1.5, length.out = 15),
                 g = rep(0.2, 15)),
  cats = 2, model = "3PLM"
)
dat <- simdat(x = x, theta = rnorm(300), D = 1)

# run the CTT analysis
out <- ctt(data = dat)

# print the full report with values rounded to 2 decimal places
print(summary(out), digits = 2)
```

---

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`. If the limit is reached while flagged items remain, a warning is issued and the `complete` element of the purification results is `FALSE`; for tests with many items or many DIF items, consider increasing `max.iter`.

`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 is assessed (Jung & Lim, 2026; Lim, Malatesta, & Lee, 2024). To evaluate global DIF for polytomous items, use `crdif()`.

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 numeric vector of the positions (rows of `x`) of the items flagged as DIF by the `purify.by` statistic across all purification iterations, sorted in ascending order.

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

Jung, H., & Lim, H. (2026, April). Detecting global and net DIF in polytomous items using RDIF. Paper presented at the annual meeting of the National Council on Measurement in Education, Los Angeles, CA.

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., Malatesta, J., & Lee, Y. (2024, July). Advancing polytomous DIF detection with the residual DIF framework. Paper presented at the annual International Meeting of the Psychometric Society, Prague, Czech Republic.

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 is theta = seq(-5, 5, 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 recursion of Lim et al. (2021) is built on inverse TCC ability estimates. At each stage, the sum score accumulated over all modules administered so far is converted to an ability estimate by inverse TCC scoring, this estimate is compared with the cut scores to route the test taker to the next module, and the final ability estimate is also the inverse TCC estimate of the total sum score. Accordingly, the function supports inverse TCC scoring with cut-score routing only. To evaluate other scoring or routing methods, use `run_mst()`, which runs a Monte Carlo simulation. With `route_method = NULL`, a `cut_score` list, and `route_score = list(method = "INV.TCC")`, `run_mst()` follows the same design that this function evaluates analytically.

All modules in the same stage must have the same maximum sum score (the sum of the maximum item scores), for example the same number of items when all items are dichotomous. The function stops with an error when this condition is not met.

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, scores from -0.5 up to but not including 0.5 to the second, and scores of 0.5 or above 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. These 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

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. doi:10.1111/jedm.12276.

Magis, D., Yan, D., & von Davier, A. A. (2017). *Computerized adaptive and multistage testing with R: Using packages catR and mstR*. Springer. doi:10.1007/9783319692180.

See Also

`shape_df()`, `est_score()`

Examples

```
## ------------------------------------------------------------------------------
# Evaluation of a 1-3-3 MST panel using simMST data.
# This panel was assembled using module-level target TIFs and design
# constraints similar to those used in Lim et al.'s (2021) simulation study
# (it is not the identical dataset from that 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(-7, 7)
  )

# 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 the vector should be the same as 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`. If the limit is reached while flagged items remain, a warning is issued and the `complete` element of the purification results is `FALSE`; for tests with many items or many DIF items, consider increasing `max.iter`.

`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

Theoretical Background: From RDIF to RIPD

The RIPD framework directly adapts the residual-based differential item functioning (RDIF) detection framework (Lim et al., 2022; Lim & Choe, 2023) to detect item parameter drift (IPD) in computerized adaptive testing (CAT). Each RIPD statistic ($RIPD_R$, $RIPD_S$, $RIPD_{RS}$) mirrors its RDIF counterpart ($RDIF_R$, $RDIF_S$, $RDIF_{RS}$) in both computation and asymptotic theory, with one key adaptation: whereas the original RDIF framework estimates item parameters once from the pooled data of both groups, the RIPD framework uses the *original, pre-calibrated* item parameters from the CAT pool directly, eliminating the need for item recalibration. This makes RIPD practically scalable for operational CAT programs, where the response data are sparse and recalibration is often impractical.

IPD in CAT can be viewed as a special case of DIF across time (Lord, 1980; Veerkamp & Glas, 2000): the focal group represents current test-takers whose responses may reflect drift, while the reference group represents a baseline under invariant item parameters.

Residual Definition

For examinee i and item j, the raw residual is defined as

$$e_{ij} = u_{ij} - P_j(\hat{\theta}_i),$$

where u_{ij} is the observed binary response (0 or 1) and $P_j(\hat{\theta}_i)$ is the model-predicted probability of a correct response under the *original* item parameters evaluated at the examinee's ability estimate $\hat{\theta}_i$. Because item responses are independent but not identically distributed (due to varying predicted probabilities across examinees with different ability levels), the asymptotic distributions of the RIPD statistics are established via Lyapunov's central limit theorem.

Three RIPD Statistics and Their Asymptotic Distributions

RIPD$_R$ (uniform drift): Measures the difference in mean raw residuals between the focal (F) and reference (R) groups,

$$RIPD_{R,j} = \frac{\sum_{i \in F} e_{ij}}{N_F} - \frac{\sum_{i \in R} e_{ij}}{N_R}.$$

$RIPD_{R,j}$ is most effective when the item response functions (IRFs) of the two groups differ primarily in location (e.g., drift in the difficulty b or guessing c parameter). Under the null hypothesis H_0 of no IPD, $RIPD_{R,j}$ asymptotically follows a normal distribution with mean zero and analytically derived variance $\sigma^2_{RIPD_R} = \frac{\sum_{i \in F} P_{ij}(1-P_{ij})}{N_F^2} + \frac{\sum_{i \in R} P_{ij}(1-P_{ij})}{N_R^2}$. A standard Z-test is then applied as $Z_R = RIPD_{R,j}/\sigma_{RIPD_R}$.

RIPD$_S$ (nonuniform drift): Measures the difference in mean squared residuals between groups,

$$RIPD_{S,j} = \frac{\sum_{i \in F} e_{ij}^2}{N_F} - \frac{\sum_{i \in R} e_{ij}^2}{N_R}.$$

$RIPD_{S,j}$ captures drift that varies across the ability continuum, such as a change in the discrimination parameter a, which $RIPD_{R,j}$ alone may not detect. Importantly, under H_0, the null mean of $RIPD_{S,j}$ is **not zero** in general; it equals

$$\mu_{RIPD_S} = \frac{\sum_{i \in F} P_{ij}(1 - P_{ij})}{N_F} - \frac{\sum_{i \in R} P_{ij}(1 - P_{ij})}{N_R},$$

which is a function of the predicted probabilities in both groups and vanishes only when their ability distributions are perfectly matched (as in the RIPD setup where focal $\hat{\theta}$ values are reused for the synthetic reference group). The asymptotic variance is also analytically derived. A standard Z-test is applied as $Z_S = (RIPD_{S,j} - \mu_{RIPD_S})/\sigma_{RIPD_S}$.

RIPD$_{RS}$ (combined): A Wald-type statistic that jointly tests for uniform and nonuniform drift within a single framework,

$$RIPD_{RS,j} = (\boldsymbol{\nu}_j - \boldsymbol{\mu}_j)^\top \hat{\boldsymbol{\Sigma}}_j^{-1} (\boldsymbol{\nu}_j - \boldsymbol{\mu}_j),$$

where $\boldsymbol{\nu}_j = (RIPD_{R,j},\ RIPD_{S,j})^\top$, $\boldsymbol{\mu}_j = (\mu_{RIPD_R},\ \mu_{RIPD_S})^\top$ is the vector of null means, and $\hat{\boldsymbol{\Sigma}}_j$ is the analytically derived 2×2 covariance matrix of $(RIPD_{R,j}, RIPD_{S,j})$. Under H_0, $RIPD_{RS,j}$ asymptotically follows a χ^2 distribution with 2 degrees of freedom. $RIPD_{RS}$ is the **recommended** statistic in practice because it is sensitive to both uniform and nonuniform drift and addresses potential inflation of the family-wise Type I error that arises from applying $RIPD_R$ and $RIPD_S$ separately to the same item.

Diagnosing the Nature of Drift

The pattern of flagging across the three statistics can help diagnose the type of drift. If an item is flagged by $RIPD_R$ (and possibly $RIPD_{RS}$) but not $RIPD_S$, uniform drift is indicated (e.g., difficulty shift). If flagged by $RIPD_S$ (and possibly $RIPD_{RS}$) but not $RIPD_R$, nonuniform drift is suggested (e.g., discrimination change). Items flagged by all statistics or by $RIPD_{RS}$ alone likely exhibit mixed drift.

CAT-Specific Workflow

The RIPD procedure for CAT consists of three steps (Lim & Han, 2026):

1. **Focal group CAT**: The current cohort of examinees takes the CAT using the operational (potentially drifted) item pool. Each examinee receives only a subset of items; the resulting response matrix is sparse. The ML ability estimates $\hat{\theta}$ from this step are retained.
2. **Synthetic reference group construction**: The focal group's $\hat{\theta}$ values are treated as true abilities. Item responses are generated from the *original* (pre-calibrated, drift-free) item parameters, and an independent CAT is administered. This creates a synthetic reference group that matches the focal group's ability distribution, so that any systematic differences in residuals can be attributed to IPD rather than ability confounds. The reference group size is controlled by a replication factor: e.g., 1F reuses the focal $\hat{\theta}$ values once (N_ref = N_foc), while kF replicates them k times. Larger reference groups (3F-8F) reduce sampling variability in reference residuals and improve detection power, with marginal gains typically diminishing beyond 5F.
3. **RIPD computation with purification**: Residuals are computed for both groups using the original item parameters, and the three RIPD statistics are evaluated item-by-item. Items whose Z-test (for $RIPD_R$ or $RIPD_S$) or chi-square test (for $RIPD_{RS}$) exceeds the critical value at level `alpha` are flagged as drifting.

Purification Procedure

When `purify = TRUE`, an iterative purification procedure adapted from Lim et al. (2022) is applied to mitigate the bias in ability estimates caused by drifted items–analogous to the contaminating effect of DIF items on matching variables. At each iteration:

1. The item with the most statistically significant drift statistic (smallest p-value) among currently unflagged items is identified and flagged as a potential IPD item.
2. Ability estimates are recomputed excluding all currently flagged items.
3. RIPD statistics are recalculated using the updated ability estimates for the remaining items.

The process continues until no additional items are flagged (convergence) or the maximum number of iterations (`max.iter`) is reached. The statistic used to drive purification is specified by `purify.by`; `"ripdrs"` ($RIPD_{RS}$) is recommended as it is sensitive to both types of drift.

Key Items and item.skip

In CAT contexts, many items in the pool receive few responses and thus lack sufficient data for reliable IPD analysis. The `item.skip` argument allows users to exclude such items (e.g., low-exposure non-key items) from the analysis. In practice, key items for RIPD evaluation are typically identified through a preliminary CAT simulation as those with the highest average exposure frequencies under the intended operational settings.

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 numeric vector of item indices (row positions in `x`) identified as IPD items across all purification iterations, sorted in ascending order.

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. (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., & Han, K. T. (2026). IRT residual-based approach to detecting item parameter drift in CAT. *Journal of Educational and Behavioral Statistics*. doi:10.3102/10769986261460852.

See Also

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

Examples

```
## --- RIPD Example: Detecting IPD in CAT ---------------------------------
##
## Background (Lim & Han, 2026):
##   In CAT-based IPD detection using RIPD, the reference group is
##   "synthetic" -- created by re-administering a CAT to examinees whose
##   true abilities are set equal to the focal group's ML theta estimates,
##   using the ORIGINAL (pre-drift) item parameters. This eliminates the
##   need for recalibration and makes RIPD directly applicable to
##   operational CAT settings.
##
## The simIPD dataset contains:
##   - foc_resp / foc_score : focal group CAT responses + ML theta estimates
##                            (IPD items: a and b each drifted by -0.5)
##   - ref_resp / ref_score : synthetic reference group CAT responses + ML
##                            theta estimates (original parameters, 1F size)
##   - item_par  : original (non-drifted) 360-item pool in irtQ format
##   - key_item  : indices of 90 key items (highly exposed in CAT)
##   - item.skip : indices of 270 non-key items (excluded from RIPD)
##   - ipd_item  : indices of 18 truly drifted items (ground truth)
## ------------------------------------------------------------------------

data(simIPD)

## Step 1. Combine focal and synthetic reference group data
##         (reference group first, then focal group -- as in the paper)
data  <- rbind(simIPD$ref_resp,  simIPD$foc_resp)
score <- c(simIPD$ref_score,     simIPD$foc_score)
group <- c(rep(0, nrow(simIPD$ref_resp)),   # 0 = reference
           rep(1, nrow(simIPD$foc_resp)))   # 1 = focal

## Step 2. Run RIPD with purification (recommended statistic: RIPD_RS)
##         item.skip excludes the 270 non-key items from analysis
ripd_result <- ripd(
  x          = simIPD$item_par,
  data       = data,
  score      = score,
  group      = group,
  focal.name = 1,           # focal group is coded as 1
  item.skip  = simIPD$item.skip,
  D          = 1.7,
  alpha      = 0.05,
  purify     = TRUE,
  purify.by  = "ripdrs",    # purify using RIPD_RS (combined statistic)
  max.iter   = 30,
  method     = "ML",
  range      = c(-5, 5)
)

## Step 3. Review RIPD_RS results (with purification)
print(ripd_result, what = "with_purify")
```

```
## Step 4. Compare detected items to ground truth
detected <- ripd_result$with_purify$ipd_item
cat("Truly drifted items (ground truth):", simIPD$ipd_item, "\n")
cat("RIPD-detected items:               ", detected, "\n")
cat("True positives:", sum(detected %in% simIPD$ipd_item), "of",
    length(simIPD$ipd_item), "\n")

## -- Note on reference group size ----------------------------------------
## This example uses a 1F reference group (n_ref = n_foc = 3,000).
## In practice, a larger reference group (3F-8F) substantially improves
## detection power. To create a 3F reference group, replicate the focal
## theta estimates and re-run the CAT simulation with original parameters:
##
##   theta_3F  <- rep(simIPD$foc_score, times = 3)   # 9,000 examinees
##   resp_3F   <- simdat(x = simIPD$item_par, theta = theta_3F, D = 1.7)
##   # ... then run CAT and call ripd() with the larger reference group
## ------------------------------------------------------------------------
```

`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)
```

`run_mst` *Multistage-Adaptive Test (MST) Simulation*

Description

Simulates an MST administration for a panel of examinees. At each stage, examinees are routed to a module based on their current ability estimate (or a set of cut scores), and a final ability estimate is computed from the accumulated responses across all administered stages. The function supports all IRT models available in **irtQ** (1PLM, 2PLM, 3PLM, GRM, GPCM) and all scoring methods available in `est_score`.

The overall structure - separate `route_score` and `final_score` list arguments, and the `route_map`/`module` matrix conventions - was inspired by the `randomMST()` function in the **mstR** package (Magis et al., 2017).

Usage

```
run_mst(
  x,
  route_map,
  module,
  theta = NULL,
  response = NULL,
  D = 1,
  ini_mod = NULL,
```

```
  route_method = "bmat",
  cut_score = NULL,
 route_score = list(method = "ML", range = c(-5, 5), norm.prior = c(0, 1), nquad = 41L,
  tol = 1e-04, max.iter = 100L, fence.a = 3, fence.b = NULL, intpol = TRUE, range.tcc =
    c(-7, 7), max.it = 500L),
 final_score = list(method = "ML", range = c(-5, 5), norm.prior = c(0, 1), nquad = 41L,
  tol = 1e-04, max.iter = 100L, fence.a = 3, fence.b = NULL, intpol = TRUE, range.tcc =
    c(-7, 7), max.it = 500L),
  se = TRUE,
  missing = NA,
  verbose = TRUE,
  return_full_resp = FALSE
)
```

Arguments

x	A data frame of item bank metadata following the irtQ format (columns: id, cats, model, par.1, par.2, ...). Use shape_df to create this frame or bring.flexmirt / bring.bilog / bring.parscale / bring.mirt to import from IRT software output. The function calls confirm_df(x) internally to validate and normalise the metadata.
route_map	A binary square matrix defining the MST transition structure. An entry of 1 in row *i* and column *j* means that a test taker can be routed from module *i* to module *j*. Equivalent to the transMatrix argument in randomMST() from **mstR** (Magis et al., 2017). See reval_mst for details.
module	A binary matrix mapping items (rows) to modules (columns). module[j, m] = 1 means item *j* belongs to module *m*. Equivalent to the modules argument in randomMST() from **mstR** (Magis et al., 2017). Must have nrow(module) == nrow(x) and ncol(module) == total number of modules in the panel.
theta	A numeric vector of true ability values, one per examinee (length *N*). Used to simulate item responses when response is NULL, and stored in the result for bias/RMSE computation. Either theta or response (or both) must be provided. If theta is omitted and only response is given, true.theta in the result will be NULL and bias/RMSE cannot be computed.
response	An optional *N* x *J* matrix of observed item responses, where *N* is the number of examinees and *J* is the total number of items in the item bank (nrow(x)). Responses for items not administered at a given stage may be NA. If NULL and theta is provided, responses are simulated internally via simdat. If both theta and response are provided, response is used as-is and theta is retained for evaluation (bias/RMSE computation) only. Providing a pre-generated response matrix (e.g., from simdat(x, theta, D)) is useful when you want to compare different routing methods or scoring options on the *same* set of item responses.
D	A numeric scaling constant for the IRT model. Use D = 1 (default) for the logistic metric or D = 1.702 to approximate the normal ogive.
ini_mod	Either NULL (default) or a single positive integer index into the stage-1 module list (panel_info(route_map)$config[[1]]). When NULL, each examinee is independently and randomly assigned to one of the stage-1 modules with equal

probability - appropriate for MST panels that use spiralling or booklet-based stage-1 assignment, or for panels with only a single stage-1 module (where random assignment is equivalent to a fixed assignment). When an integer is supplied, all examinees begin with that stage-1 module; use `ini_mod = 1` to force all examinees to the first (and typically only) stage-1 module.

`route_method` A character string specifying the adaptive routing method used to select the next module at each stage transition.

`"bmat"` B-matching: selects the module whose average item location (difficulty) is closest to the current ability estimate. For dichotomous items the *b* parameter is used; for polytomous items (GRM, GPCM) the mean of the threshold / step parameters is used.

`"mfi"` Maximum Fisher Information: selects the module that provides the highest test information function (TIF) value at the current ability estimate.

`NULL` Cut-score routing: uses the cut scores supplied in `cut_score` to route examinees to the next module. Each examinee's current routing score is compared against the cut scores to determine which of the reachable next-stage modules to administer.

Default is `"bmat"`.

`cut_score` A list of numeric vectors specifying the routing cut scores used when `route_method = NULL`. The list should have `n.stage - 1` elements, one per stage transition. Element *s* contains the cut scores for routing from stage *s* to stage *s*+1. For example, in a 1-3-3 MST, `cut_score = list(c(-0.5,0.5), c(-0.6, 0.6))` routes examinees whose stage-1 score is below -0.5 to the easiest stage-2 module, from -0.5 up to but not including 0.5 to the medium module, and 0.5 or above to the hardest module. Ignored when `route_method` is `"bmat"` or `"mfi"`. Default is `NULL`.

`route_score` A named list specifying the scoring method and options used to obtain intermediate ability estimates for routing decisions (applied after every stage except the last). Fields:

`method` Character. One of `"ML"` (maximum likelihood), `"WL"` (weighted likelihood; Warm, 1989), `"MLF"` (maximum likelihood with fences; Han, 2016), `"MAP"` (maximum a posteriori), `"EAP"` (expected a posteriori; Bock & Mislevy, 1982), `"EAP.SUM"` (EAP summed scoring; Thissen et al., 1995), or `"INV.TCC"` (inverse test characteristic curve scoring; Lim et al., 2021). For `"EAP.SUM"` and `"INV.TCC"`, a sum-score-to-theta lookup table is precomputed once per partial pathway (the modules of stages 1 to *s*) before the simulation loop; routing theta is then obtained by a single named-vector lookup, making the approach efficient for large N. Default: `"ML"`.

`range` Numeric vector of length 2: lower and upper bounds for the ability scale. Default: `c(-5, 5)`.

`norm.prior` Numeric vector of length 2: mean and SD of the normal prior for MAP/EAP. Default: `c(0, 1)`.

`nquad` Integer: number of quadrature points for EAP. Default: `41`.

`tol` Numeric: convergence tolerance for iterative methods. Default: `1e-4`.

`max.iter` Integer: maximum iterations. Default: `100`.

fence.a Numeric: discrimination parameter of fence items for MLF. Default: 3.0.

fence.b Numeric vector of length 2 or NULL: difficulty parameters of the lower and upper fence items. If NULL, range is used. Default: NULL.

intpol Logical: enable linear interpolation for "INV.TCC". Default: TRUE.

range.tcc Numeric vector of length 2: theta search range for "INV.TCC". Default: c(-7, 7).

max.it Integer: maximum bisection iterations for "INV.TCC". Default: 500L.

Unspecified fields take their defaults; the full default is list(method = "ML", range = c(-5, 5), norm.prior = c(0, 1), nquad = 41L, tol = 1e-4, max.iter = 100L, fence.a = 3.0, fence.b = NULL, intpol = TRUE, range.tcc = c(-7, 7), max.it = 500L). Example: route_score = list(method = "INV.TCC", range.tcc = c(-5, 5)).

final_score A named list specifying the scoring method and options for the final ability estimate (applied to all items accumulated across the entire administered pathway). It takes the same fields, methods, and defaults as route_score. The full default is equivalent to list(method = "ML", range = c(-5, 5), norm.prior = c(0, 1), nquad = 41L, tol = 1e-4, max.iter = 100L, fence.a = 3.0, fence.b = NULL, intpol = TRUE, range.tcc = c(-7, 7), max.it = 500L).

se Logical. Whether to compute and return the standard error of the final ability estimate. SE is computed for all final_score methods: for "ML", "WL", "MLF", "MAP", and "EAP" via the Fisher information / posterior variance; for "EAP.SUM" and "INV.TCC" via the posterior standard deviation stored in the pre-computed lookup table. Returns NA when se = FALSE, when all responses are missing, or when the sum score falls outside the estimable range (e.g. a perfect or zero score beyond range.tcc). Default is TRUE.

missing A scalar specifying how missing (not-administered) responses are represented in response. Default is NA.

verbose Logical. If TRUE, prints progress messages to the console during the simulation (approximately every 10\ Default is TRUE.

return_full_resp Logical. If TRUE, the full.resp element of the returned list is an N x J integer matrix, where N is the number of examinees and J = nrow(x) (all items in the item bank). Cell [i, j] contains examinee i's response to item j if that item was administered, or NA if it was not. Row names are "examinee.1", "examinee.2", etc.; column names are the item IDs from x$id. If FALSE (default), full.resp is NULL. Because this matrix can be large for large panels and many examinees, it is not populated by default.

Details

Panel structure: The function first calls panel_info to identify stages, module membership, and valid pathways from route_map. Module m's items are those where module[, m] == 1.

Response simulation: If response is NULL, item responses are simulated from theta using simdat.

Stage-1 module assignment: When ini_mod = NULL (default), each examinee is independently assigned to a stage-1 module by uniform random sampling from panel_info(route_map)$config[[1]].

For panels with a single stage-1 module (the most common design), this is equivalent to a fixed assignment. For panels with multiple stage-1 modules (e.g., a 2-3-3 or 3-2-3 design), random assignment approximates the spiralling or booklet-based allocation used in operational testing. When `ini_mod` is an integer, all examinees start at the specified stage-1 module.

Routing (stages 1 to n.stage - 1): After each non-final stage, an intermediate ability estimate is obtained from the responses to all modules administered so far using the `route_score` method. This estimate is used to select the next-stage module:

- `"bmat"`: the reachable module (via `route_map`) whose mean item location is closest to the routing estimate is selected. For dichotomous items, the location is the b parameter; for GRM/GPCM items, it is the mean of the threshold / step parameters.
- `"mfi"`: the reachable module with the highest test information at the routing estimate is selected.
- `NULL`: the routing estimate is compared against the cut scores in `cut_score[[s]]` (for the transition from stage s to stage s+1). The cut scores are tied to the modules of stage s+1 in order of module index: cut score k separates the k-th and the (k+1)-th module of the stage. When only some modules of the stage can be reached from the current module, only the cut scores that separate the reachable modules are used, as in `reval_mst`, and the module whose interval contains the estimate is administered. An estimate equal to a cut score is assigned to the higher module. When every module of the next stage is reachable, all cut scores are used.

Final scoring: Responses from all administered stages are concatenated, and the final ability estimate is computed using the `final_score` method. With `"EAP.SUM"` or `"INV.TCC"`, a missing response is counted as 0 in the sum score, as in `est_score`, but no warning is issued.

Relation to `reval_mst()`: With `route_method = NULL`, a `cut_score` list, and `route_score = list(method = "INV.TCC")`, the routing rule is the one that `reval_mst` evaluates analytically: cut scores are applied to the inverse TCC estimate of the cumulative sum score. With `final_score = list(method = "INV.TCC")` as well, the simulated bias and conditional standard error of measurement approach the `reval_mst()` values as the number of simulated examinees grows.

Value

An object of class `"run_mst"`, which is a named list containing:

`call` The matched function call.

`est.theta` A numeric vector of length N containing the final ability estimate for each examinee.

`se.theta` A numeric vector of length N containing the standard error of the final ability estimate. All seven `final_score` methods return SE; for `"EAP.SUM"` and `"INV.TCC"`, SE is the posterior standard deviation from the pre-computed lookup table. `NA` when `se = FALSE`, when all responses are missing, or when the sum score falls outside the estimable range.

`theta.route` An N x `n.stage` numeric matrix of ability estimates. Column s (s < `n.stage`) is the cumulative routing estimate after stage s, computed from the responses to all modules administered in stages 1 to s; column `n.stage` is the final estimate (equal to `est.theta`).

`path` An N x `n.stage` integer matrix of administered module indices.

`true.theta` The `theta` argument (true abilities), or `NULL` if not provided.

`panel` The output of `panel_info(route_map)`.

`D` The scaling constant used.

`route.method` The `route_method` argument.

`route.score` The fully-merged `route_score` list (with all defaults applied).

`final.score` The fully-merged `final_score` list (with all defaults applied).

`N` Number of examinees.

`full.resp` Always present in the returned list. When `return_full_resp = TRUE`, an N x J integer matrix of all item responses, where N is the number of examinees and J = `nrow(x)` (total items in the item bank). Row names are `"examinee.1"`, `"examinee.2"`, etc.; column names are the item IDs from `x$id`. `NA` for items not administered to a given examinee. When `return_full_resp = FALSE` (default), this element is `NULL`.

References

Bock, R. D., & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement, 6*(4), 431-444. doi:10.1177/014662168200600405.

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. doi:10.1177/0146621616631317.

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. doi:10.1111/jedm.12276.

Magis, D., Yan, D., & von Davier, A. A. (2017). *Computerized adaptive and multistage testing with R: Using packages catR and mstR*. Springer. doi:10.1007/9783319692180.

Thissen, D., Pommerich, M., Billeaud, K., & Williams, V. S. L. (1995). Item response theory for scores on tests including polytomous items with ordered responses. *Applied Psychological Measurement, 19*(1), 39-49. doi:10.1177/014662169501900105.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427-450. doi:10.1007/BF02294627.

See Also

`panel_info`, `reval_mst`, `est_score`, `simdat`, `simMST`

Examples

```
## ----------------------------------------------------------
## Example 1: bmat routing with ML scoring (default)
## Using the simMST 1-3-3 panel (7 modules, 8 items each)
## ----------------------------------------------------------
# Load item bank metadata
x         <- simMST$item_bank
module    <- simMST$module
route_map <- simMST$route_map

# True abilities for 500 examinees
set.seed(42)
theta_true <- rnorm(500, mean = 0, sd = 1)
```

```
# Run MST simulation with bmat routing and ML scoring
result_bmat <- run_mst(
  x          = x,
  route_map  = route_map,
  module     = module,
  theta      = theta_true,
  D          = 1.702,
  route_method = "bmat",
  route_score  = list(method = "ML", range = c(-4, 4)),
  final_score  = list(method = "ML", range = c(-4, 4)),
  se = TRUE
)

# Print simulation summary
print(result_bmat)

# Final ability estimates
head(result_bmat$est.theta)

# Module pathway taken by each examinee
head(result_bmat$path)

## ------------------------------------------------------------
## Example 2: Cut-score routing with EAP routing and ML final scoring
## EAP keeps the routing estimate finite even from a short Stage 1
## module (e.g., a perfect or zero score), while ML avoids the shrinkage
## that EAP's prior would otherwise introduce in the reported final score.
## ------------------------------------------------------------
cut_score <- simMST$cut_score   # list(c(-0.45, 0.46), c(-0.42, 0.45))

result_cut <- run_mst(
  x            = x,
  route_map    = route_map,
  module       = module,
  theta        = theta_true,
  D            = 1.702,
  route_method = NULL,
  cut_score    = cut_score,
  route_score  = list(method = "EAP", norm.prior = c(0, 1), nquad = 41),
  final_score  = list(method = "ML", range = c(-4, 4)),
  se = TRUE
)
print(result_cut)

## ------------------------------------------------------------
## Example 3: Using a pre-generated response matrix
## (useful when comparing multiple routing methods on the
##  same simulated responses, or when using real observed data)
## ------------------------------------------------------------
# Generate responses once from theta using simdat()
set.seed(123)
theta_true2 <- rnorm(500, mean = 0, sd = 1)
resp_matrix <- simdat(x = x, theta = theta_true2, D = 1.702)
```

```
# resp_matrix is an N x J matrix (N examinees x J total items in bank)

# Now run run_mst() twice with the SAME responses but different routing methods
# Method A: bmat routing
result_A <- run_mst(
  x            = x,
  route_map    = route_map,
  module       = module,
  theta        = theta_true2,   # kept for bias/RMSE evaluation
  response     = resp_matrix,   # pre-generated N x J response matrix
  D            = 1.702,
  route_method = "bmat",
  route_score  = list(method = "EAP", norm.prior = c(0, 1), nquad = 41),
  final_score  = list(method = "ML", range = c(-4, 4)),
  se = FALSE
)

# Method B: cut-score routing using the same responses
result_B <- run_mst(
  x            = x,
  route_map    = route_map,
  module       = module,
  theta        = theta_true2,
  response     = resp_matrix,   # identical response matrix -> fair comparison
  D            = 1.702,
  route_method = NULL,
  cut_score    = simMST$cut_score,
  route_score  = list(method = "EAP", norm.prior = c(0, 1), nquad = 41),
  final_score  = list(method = "ML", range = c(-4, 4)),
  se = FALSE
)

# Compare RMSE of the two routing strategies
rmse_A <- sqrt(mean((result_A$est.theta - theta_true2)^2))
rmse_B <- sqrt(mean((result_B$est.theta - theta_true2)^2))
cat(sprintf("RMSE (bmat): %.4f\n", rmse_A))
cat(sprintf("RMSE (cut-score): %.4f\n", rmse_B))

## ------------------------------------------------------------
## Example 4: TIF-based cut scores via find_cut()
## find_cut() computes principled cut scores from TIF crossings.
## This prevents anomalous routing under MFI-like heuristics.
## ------------------------------------------------------------
# Derive cut scores from TIF crossings between adjacent modules
cut_result <- find_cut(
  x         = x,
  module    = module,
  route_map = route_map,
  D         = 1.702
)

# Inspect the derived cut scores
print(cut_result)
```

```
# Use the derived cut scores for routing (route_method = NULL)
result_findcut <- run_mst(
  x            = x,
  route_map    = route_map,
  module       = module,
  theta        = theta_true,
  D            = 1.702,
  route_method = NULL,
  cut_score    = cut_result$cut_score,
  route_score  = list(method = "EAP", norm.prior = c(0, 1), nquad = 41),
  final_score  = list(method = "ML", range = c(-4, 4)),
  se = TRUE
)
print(result_findcut)

## Visualise the TIF-based cut scores used for routing
plot(cut_result)

## ------------------------------------------------------------
## Example 5: INV.TCC routing and final scoring
## (pre-computed lookup tables; efficient for large N)
## ------------------------------------------------------------
# INV.TCC can be used as both the routing and final scoring method.
# Before the examinee loop, run_mst() pre-computes a sum_score -> theta
# lookup table for every unique partial pathway (routing) and for every
# unique complete pathway (final scoring), consistent with the reval_mst()
# approach.
# SE is taken from the pre-computed lookup table.

result_inv <- run_mst(
  x            = x,
  route_map    = route_map,
  module       = module,
  theta        = theta_true,
  D            = 1.702,
  route_method = "bmat",
  route_score  = list(method = "INV.TCC", range.tcc = c(-7, 7)),
  final_score  = list(method = "INV.TCC", range.tcc = c(-7, 7)),
  se           = TRUE
)
print(result_inv)

# standard errors from the lookup table
head(result_inv$se.theta)

# Compare RMSE: INV.TCC routing vs. ML routing (result_bmat from Example 1)
rmse_inv <- sqrt(mean((result_inv$est.theta  - theta_true)^2, na.rm = TRUE))
rmse_ml  <- sqrt(mean((result_bmat$est.theta - theta_true)^2, na.rm = TRUE))
cat(sprintf("RMSE (INV.TCC routing + scoring): %.4f\n", rmse_inv))
cat(sprintf("RMSE (ML    routing + scoring):   %.4f\n", rmse_ml))
```

score_resp *Score Selected-Response Item Data Against an Answer Key*

Description

This function converts raw selected-response (e.g., multiple-choice) item data - coded as the selected option (an option number, e.g., 1-5 for a five-option item; a single-letter option label, e.g., `"A"`-`"E"`; or any other option label in a different script or notation, such as Korean syllable labels or Roman numerals), a missing-response indicator for an omitted response, or a comma-separated string (e.g., `"1,5"` or `"A,C"`) for a double-marked (multiple-option) response - into a dichotomously scored item-response matrix (0 = incorrect, 1 = correct) using a supplied answer key. The option coding scheme is detected independently for each item from its own `key` value, so a single test form may freely mix numerically coded, letter-coded, and other label-coded items. Letter (and other case-bearing) option labels are matched case-insensitively (e.g., a response of `"a"` matches a key of `"A"`). Omitted and double-marked responses are both scored as incorrect (0), and their frequencies are tabulated separately for reporting.

Usage

```
score_resp(data, key, missing = NA)
```

Arguments

`data`	A data frame or matrix of raw item responses, with examinees in rows and items in columns (in the same left-to-right order as `key`). Do not include non-item columns (e.g., group or examinee identifiers); subset those out before calling `score_resp()`. Each cell should contain either a single selected option - an option number (e.g., `2`), a single-letter option label (e.g., `"C"`), or any other option label (e.g., a Korean syllable label, or a Roman numeral such as `"II"`) - matching whichever coding scheme is used for that item's `key` value, a missing-response indicator (see `missing`), or a character string of comma-separated options (e.g., `"1,5"` or `"A,C"`) for a double-marked response. To keep double-marked responses as character strings rather than having them coerced to `NA` on import, read the source file with all item columns imported as text (e.g., `readxl::read_excel(..., col_types = "text")`) before passing `data` to this function.
`key`	Either (a) a vector of correct options in item order (length must equal `ncol(data)`), where each element is a numeric/integer option number (e.g., `4`), a single-letter option label (e.g., `"D"`), or any other non-blank option label (e.g., a Korean syllable label, or a Roman numeral such as `"II"`), or (b) a data frame with columns `item` and `key` giving the item number and its correct option, respectively (the `key` column may likewise mix option formats across rows). Each item's coding scheme is inferred independently from its own key value, so items using different option formats may be freely mixed within the same vector or data frame; letter (and other case-bearing) values are matched case-insensitively. When a data frame is supplied, it is internally sorted by `item` before use, so its row order

does not need to match the column order of `data`. The `item` column may be numeric, character, or factor; it is matched by item number.

`missing` A value indicating missing (omitted) responses in `data`, analogous to the `missing` argument in `est_irt()` and `est_score()`. Any cell equal to `missing` is recoded to `NA` before scoring. Default is `NA`, meaning `data` is assumed to already use `NA` (or an empty string) for omitted responses, and no recoding is performed. Set this to whatever sentinel value a particular data set uses for a missing response (e.g., `-9`, `"9"`) when it differs from `NA`.

Details

For each item, a response is scored as correct (1) only when it is a single, non-missing option that matches the corresponding value in `key`. All other cases - an omitted response, a double-marked response, or a single but incorrect option - are scored as incorrect (0). Missing and double-marked responses are tallied separately in `resp_summary` so that omission and double-marking rates can be reported independently, even though both are scored as 0.

Each item's option coding scheme is determined independently from its own `key` value, using one of three rules:

- If the key value parses as a number (e.g., `4`), responses to that item are compared numerically.
- If the key value consists of one or more Latin letters (e.g., `"D"`), responses are compared as letters, case-insensitively (`"d"` and `"D"` are treated as the same option), and a response must itself consist only of Latin letters to count as a single valid option (see below for what happens otherwise).
- Otherwise (e.g., a Korean syllable label, a Roman numeral written with non-Latin numeral characters, a circled-number symbol, or any other non-numeric, non-Latin-letter label), the key value is treated as a general option label: any non-blank, non-double-marked response to that item is accepted as a single valid option, and is compared to the key with a case-insensitive (`toupper()`-based, which only affects any embedded Latin characters) exact string match. There is no universal, script-independent rule for "a well-formed option label" analogous to the numeric or Latin-letter checks above, so this scheme cannot distinguish a genuinely incorrect option from a garbled response token; both are simply scored 0, and `n_invalid` is always 0 for items using this scheme (see below).

This per-item detection means a single test form may freely mix items using any of the three schemes - only the value supplied in `key` for a given item determines how that item's column in `data` is interpreted. A `key` value that is blank or `NA` causes `score_resp()` to stop with an error, since no correct option was actually supplied for that item.

A response cell is classified as double-marked when, after coercion to character, it contains a comma (e.g., `"1,5"`, `"B,D"`). A response cell is classified as blank/omitted when it equals `missing` (recoded to `NA` beforehand), or is otherwise `NA` or an empty/whitespace-only string. For numeric- and Latin-letter-coded items, any response that is none of blank, double-marked, or a single valid option in the coding scheme used for that item (e.g., a numeric token where a letter was expected for a letter-coded item, or stray text from a data-entry artifact) is classified as invalid; it is still scored 0, but tallied separately in `n_invalid` (with a `warning()` raised) so it is visible rather than silently merged into the "wrong" count. As noted above, general label-coded items never populate `n_invalid`, since any non-blank, non-double-marked token is accepted as a single valid (though possibly incorrect) option for that scheme. Note that `n_correct` + `n_wrong` always equals `n`, but `n_blank`, `n_double`,

and n_invalid are reported as separate diagnostic tallies, not as a four-way disjoint partition of n_wrong.

This function performs dichotomous (single-key, selected-response) scoring only, since a selected-response item has exactly one correct option by design and therefore only two possible score categories (correct/incorrect). It is not generalized to items with more than two score categories (e.g., partial-credit polytomous items). If such data become available, they should be supplied directly, already scored, to downstream item-analysis functions that support polytomous data, bypassing score_resp().

Value

A list with two elements:

scored	A data frame with the same dimensions as data, containing the dichotomously scored (0/1) item responses. Column names follow colnames(data) (or V1, V2, ... when data has no column names).
resp_summary	A data frame with one row per item, reporting the number and percentage of blank and double-marked responses, the number of unrecognized/invalid response tokens (n_invalid), and the number of examinees scored correct/incorrect, for each item.

Note

This function applies to **dichotomous, single-key, selected-response items only** (i.e., items with exactly one correct option, scored into exactly two categories: 0 = incorrect and 1 = correct). It does not generalize to items with more than two score categories (e.g., partial-credit polytomous items scored 0, 1, 2, ...), since a selected-response item cannot yield a partial-credit score by construction. If polytomous item response data are available, supply an already-scored response matrix (with categories 0, 1, 2, ..., cats - 1) directly to downstream item-analysis functions instead of using score_resp(). See **Details** for further discussion.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

ctt_distr(), est_irt(), est_score()

Examples

```
# A small 5-examinee, 3-item toy example with a blank and a double-mark
raw <- data.frame(
  V1 = c("1", "2", NA, "1", "3"),
  V2 = c("4", "4", "4", "2,4", "4"),
  V3 = c("2", "1", "3", "3", NA)
)
key <- c(1, 4, 3)
out <- score_resp(data = raw, key = key)
out$scored
```

```
out$resp_summary

# Same data, but omitted responses are coded as "9" instead of NA
raw2 <- data.frame(
  V1 = c("1", "2", "9", "1", "3"),
  V2 = c("4", "4", "4", "2,4", "4"),
  V3 = c("2", "1", "3", "3", "9")
)
out2 <- score_resp(data = raw2, key = key, missing = "9")
identical(out$scored, out2$scored)

# A letter-coded (A-E) example, including a double-marked response and a
# lowercase response ("a"), which is matched case-insensitively
raw3 <- data.frame(
  V1 = c("A", "B", NA, "a", "C"),
  V2 = c("D", "D", "D", "B,D", "D"),
  V3 = c("B", "A", "C", "C", NA)
)
key3 <- c("A", "D", "C")
out3 <- score_resp(data = raw3, key = key3)
out3$scored

# A mixed-format test: item 1 is numerically coded, items 2-3 are
# letter-coded; the coding scheme is inferred separately for each item
# from its own key value
raw4 <- data.frame(
  V1 = c("1", "2", "1", "1", "3"),
  V2 = c("D", "D", "D", "B,D", "D"),
  V3 = c("B", "A", "C", "C", NA)
)
key4 <- c(1, "D", "C")
out4 <- score_resp(data = raw4, key = key4)
out4$scored

# A full five-option (A-E) multiple-choice item, scored against key "C"
raw5 <- data.frame(
  V1 = c("A", "B", "C", "D", "E", "c")
)
key5 <- c("C")
out5 <- score_resp(data = raw5, key = key5)
out5$scored

# A general-label-coded example using non-Latin option labels: the Korean
# syllables romanized "ga"/"na"/"da"/"ra"/"ma", written here with \uXXXX
# escapes for ASCII portability. Because these labels are neither numbers
# nor Latin letters, they are handled by the general label-coding scheme
# described in Details
raw6 <- data.frame(
  V1 = c("\uAC00", "\uB098", "\uB2E4", "\uB2E4", NA) # ga, na, da, da, NA
)
key6 <- c("\uB2E4") # da
out6 <- score_resp(data = raw6, key = key6)
out6$scored
```

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.prm 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.

res.dat A matrix of item responses from 10,000 examinees (rows) to 100 items (columns). `NA` marks an item that was not administered to the examinee. The columns have no names; they follow the row order of `item.prm`, whose `id` values are `V1` to `V100`.

score A numeric vector of ability estimates for the 10,000 examinees.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

Examples

```
# structure of the data
str(simCAT_DC, max.level = 1)


# item fit of the first five items, using the ability estimates as scores
x <- simCAT_DC$item.prm[1:5, ]
data <- simCAT_DC$res.dat[, 1:5]
irtfit(
  x = x, score = simCAT_DC$score, data = data, group.method = "equal.freq",
  n.width = 10, loc.theta = "average", range.score = c(-4, 4), D = 1
```

```
)
```

---

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.prm A data frame in item metadata format consisting of 200 dichotomous items and 30 polytomous items.

- Dichotomous items: Calibrated using the IRT 2PL model.
- Polytomous items: Calibrated using the Generalized Partial Credit Model (GPCM), with four score categories (0, 1, 2, 3).

res.dat A matrix of item responses from 30,000 examinees (rows) to 230 items (columns). `NA` marks an item that was not administered to the examinee. The columns are named `Item.dc.1` to `Item.dc.200` and `Item.py.1` to `Item.py.30`, not by the `id` column of `item.prm` (`V1` to `V230`); the columns follow the row order of `item.prm`.

score A numeric vector of ability estimates for the 30,000 examinees.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

Examples

```
# structure of the data
str(simCAT_MX, max.level = 1)


# item fit of three dichotomous items and two polytomous items
loc <- c(1:3, 201:202)
x <- simCAT_MX$item.prm[loc, ]
data <- simCAT_MX$res.dat[, loc]
irtfit(
  x = x, score = simCAT_MX$score, data = data, group.method = "equal.freq",
```

```
  n.width = 10, loc.theta = "average", range.score = c(-4, 4), D = 1
)
```

---

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, 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()`.

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)
```

---

simIPD *Simulated CAT Data for Item Parameter Drift (IPD) Detection*

Description

A simulated dataset designed to illustrate the use of the `ripd` function for detecting item parameter drift (IPD) in computerized adaptive testing (CAT). The dataset represents one replication of a CAT-based IPD simulation study in which 5% of a 360-item pool (18 items) had both their discrimination (a) and difficulty (b) parameters drift downward by 0.5, under a 30-item adaptive test where all focal group examinees were exposed to the drifted items (100% exposure rate).

The data reflect a workflow of IPD detection using the residual-based IPD (RIPD) framework (Lim & Han, 2026): a focal group of examinees takes a CAT using a drifted item pool, and a synthetic reference group is created by re-running the CAT with the focal group's ability estimates as true abilities but with the original (non-drifted) item parameters. RIPD statistics are then used to detect which items have drifted between the two groups.

Usage

```
simIPD
```

Format

A named list with eight elements:

item_par A data frame with 360 rows and 6 columns containing the **original** (pre-drift) item parameters in `irtQ` format (columns: `id`, `cats`, `model`, `par.1`, `par.2`, `par.3`). All items follow the 3PLM.

key_item An integer vector of length 90 giving the row indices (in `item_par`) of the *key items* - items selected by a preliminary CAT simulation as highly exposed and therefore most relevant for IPD analysis.

item.skip An integer vector of length 270 giving the row indices of non-key items that should be excluded from RIPD analysis (i.e., the complement of `key_item` in `1:360`). Pass this vector to the `item.skip` argument of `ripd()`.

ipd_item An integer vector of length 18 giving the row indices of items that were subjected to IPD manipulation. These 18 items (5% of the 360-item pool) had both their discrimination (a) and difficulty (b) parameters decreased by 0.5. All 18 items are members of `key_item`.

foc_resp An integer matrix of dimensions 3000 x 360 containing the **focal group** CAT response data. Each row is one examinee; each column corresponds to an item in `item_par`. Because CAT administers only 30 items per examinee, approximately 92% of entries are `NA`. Responses were generated using the *drifted* item parameters (all focal examinees were exposed to IPD items; exposure rate = 100%).

foc_score A numeric vector of length 3000 containing the **focal group** final maximum likelihood (ML) theta estimates obtained from the CAT.

ref_resp An integer matrix of dimensions 3000 x 360 containing the **synthetic reference group** CAT response data (same sparsity structure as `foc_resp`). The reference group was constructed by: (1) using `foc_score` as true ability values (1F scaling, i.e., the reference group has the same size as the focal group); (2) generating item responses from the *original* (non-drifted) item parameters; and (3) running an independent CAT simulation. This synthetic reference group mirrors the construction described in Lim & Han (2026).

ref_score A numeric vector of length 3000 containing the **synthetic reference group** final ML theta estimates.

Details

Simulation conditions:

- Item pool: 360 three-parameter logistic model (3PLM) items
- Test length: 30 items per examinee
- Item selection: Maximum Fisher Information (MFI) with target exposure rate 0.30
- Focal group: $n = 3{,}000$; true abilities drawn from $N(0, 1)$
- Reference group: $n = 3{,}000$ (1F); true abilities = `foc_score`
- IPD items: 18 (5% of 360), randomly drawn from `key_item`
- IPD manipulation: both *a* and *b* decreased by 0.50
- Interim scoring: EAP; final scoring: ML
- Scaling constant: $D = 1.7$
- Random seed: 2024

Note on reference group size: A 1F reference group (same size as the focal group) is used here for compactness. In practice, larger synthetic reference groups (e.g., 3F - 8F) are recommended to improve RIPD detection power (Lim & Han, 2026). A larger reference group can be created by replicating the focal theta estimates: e.g., `rep(foc_score, times = 3)` for a 3F group, then re-running the CAT simulation with the original item parameters.

References

Lim, H., & Han, K. T. (2026). IRT residual-based approach to detecting item parameter drift in CAT. *Journal of Educational and Behavioral Statistics*. doi:10.3102/10769986261460852.

See Also

`ripd`, `pcd2`, `simCAT_DC`, `simCAT_MX`

Examples

```
data(simIPD)
str(simIPD, max.level = 1)

# Item parameter data frame (first 6 rows)
head(simIPD$item_par)

# Focal group response matrix (sparse)
dim(simIPD$foc_resp)
mean(is.na(simIPD$foc_resp))  # ~0.92 (92% NA due to CAT)
```

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>

Examples

```
# structure of the data
str(simMG, max.level = 1)


# multiple-group calibration of the three groups (see est_mg() for details)
x <- simMG$item.prm
fit <- est_mg(
  data = simMG$res.dat, group.name = simMG$group.name,
  model = list(x$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),
  D = 1, free.group = c(2, 3), use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 16)),
  group.mean = 0, group.var = 1, EmpHist = TRUE, Etol = 0.001, MaxE = 500,
  verbose = FALSE
)

# group means and variances
fit$group.par
```

`simMST` *Simulated 1-3-3 MST Panel Data*

Description

A simulated Multistage-Adaptive Test (MST) dataset representing a 1-3-3 panel structure, assembled using module-level target test information functions (TIFs) and design constraints similar to those 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 for the 56 items assigned to the panel (one row per item; columns follow the **irtQ** item metadata format: `id`, `cats`, `model`, `par.1`, `par.2`, `par.3`).

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, computed via `find_cut()` based on TIF-crossing points of the assembled modules.

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, drawn from 56 unique items (8 per module) with no item shared across modules. Each module contains 8 dichotomously scored items calibrated under the IRT 3-parameter logistic (3PL) model. The item parameters are on the D = 1.702 scale, so pass `D = 1.702` to `find_cut()`, `reval_mst()`, and `run_mst()` when using this panel.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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. doi:10.1111/jedm.12276.

Magis, D., Yan, D., & von Davier, A. A. (2017). *Computerized adaptive and multistage testing with R: Using packages catR and mstR*. Springer. doi:10.1007/9783319692180.

Examples

```
# structure of the data
str(simMST, max.level = 1)

# item metadata of the first three items
head(simMST$item_bank, 3)
```

---

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)
```

---

summary.ctt *Summarize a Combined CTT Analysis*

Description

Prepares the full, detailed report for an object of class `"ctt"` returned by `ctt()`. Following the convention used by `est_irt()`/`summary.est_irt()`, this method returns an object of class `"summary.ctt"` whose own `print.summary.ctt()` method displays the complete item-level table, test-level reliability summary, and total-score frequency distribution.

Usage

```
## S3 method for class 'ctt'
summary(object, ...)
```

Arguments

`object`	An object of class `"ctt"`, as returned by `ctt()`.
`...`	Additional arguments passed to or from other methods (currently not used).

Value

An object of class "summary.ctt": a list with the same item, crit, alpha, freq, and call elements as object (see ctt()'s **Value**), to be displayed by print.summary.ctt().

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

ctt(), print.ctt()

Examples

```
# simulate the responses of 300 examinees to 15 dichotomous 3PLM items
set.seed(1)
x <- shape_df(
  par.drm = list(a = rep(1.5, 15), b = seq(-1.5, 1.5, length.out = 15),
                 g = rep(0.2, 15)),
  cats = 2, model = "3PLM"
)
dat <- simdat(x = x, theta = rnorm(300), D = 1)

# run the CTT analysis
out <- ctt(data = dat)

# create the full report object and print it
summary(out)
```

---

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 matrices containing the category response probabilities for each item across the specified theta values. Each matrix 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. doi:10.1177/0146621620929431.

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

∗ **datasets**
LSAT6, 122
simCAT_DC, 186
simCAT_MX, 187
simIPD, 191
simMG, 194
simMST, 195

base::system(), *167*
bind.fill, 16
bisection, 17
bring.bilog, *170*
bring.bilog *(*bring.flexmirt*)*, 18
bring.bilog(), *19*
bring.flexmirt, 18, *170*
bring.flexmirt(), *19*, *185*
bring.mirt, *170*
bring.mirt *(*bring.flexmirt*)*, 18
bring.mirt(), *19*
bring.parscale, *170*
bring.parscale *(*bring.flexmirt*)*, 18
bring.parscale(), *19*

cac_lee, 21
cac_lee(), *25*
cac_rud, 24
cac_rud(), *23*
catsib, 27
catsib(), *28*, *30*
covirt, 34
covirt(), *97, 98*
crdif, 36
crdif(), *38*, *40*, *149*
ctt, 44
ctt(), *4*, *49–51*, *95, 96*, *142*, *144*, *197, 198*
ctt_alpha(), *49*
ctt_distr, 48
ctt_distr(), *4*, *49*, *180*
ctt_item(), *49*
cut, *94*

drm, 52
drm(), *145*, *190*

est_irt, *32*, 53
est_irt(), *6–8*, *21*, *24*, *28*, *34*, *36*, *38*, *41*, *44, 45*, *48, 49*, *59*, *68*, *71*, *73*, *78*, *81*, *87, 90, 91*, *96*, *98, 99*, *102*, *105*, *108, 109*, *111–113*, *115*, *118*, *120*, *123, 127*, *142*, *147*, *150, 151*, *160*, *165, 179, 180*, *183, 184*, *196, 197*, *199, 201–205*
est_item, 67
est_item(), *8*, *32*, *38*, *41*, *70*, *87*, *98, 99*, *101, 102*, *105*, *108, 109*, *111–113*, *115, 118*, *147*, *150, 151*, *160*, *165*, *196, 197*, *199*, *201–204*
est_mg, 72
est_mg(), *77, 78*, *98–100*, *102*, *196, 197*
est_score, 85, *169*, *173, 174*
est_score(), *18*, *23–25*, *28, 29*, *32*, *38, 39, 41*, *97, 98*, *105, 106*, *109*, *147, 148, 151*, *154*, *157*, *160*, *162*, *165*, *179, 180*

find_cut, 92, *132*, *195*
freq_score, 95
freq_score(), *4*, *46, 47*

gen.weight, 97
gen.weight(), *23*, *25*, *29*, *36*, *39*, *56*, *76*, *87, 91*, *106*, *127*, *148*, *161*, *200*
getirt, 98
getirt(), *60, 61*, *70, 71*, *80, 81*
ggplot2::geom_line(), *134*, *140*
ggplot2::ggplot(), *137*
grdif, 103
grdif(), *105–107*

info, *94*, 111
info(), *20*, *134*

irtfit, 114
irtfit(), *136–138*, *201*
irtQ *(*irtQ-package*)*, 3
irtQ-package, 3, *20*, *58*, *145*, *190*

llike_score, 120
LSAT6, 122
lwrc, 122

mirt::mirt(), *19*

panel_info, *94*, 124, *172–174*
parallel::makeCluster(), *87*
pcd2, 126, *193*
pcd2(), *128*
plot.find_cut, *94*, 131
plot.info, 133
plot.info(), *112, 113*
plot.irtfit, 135
plot.irtfit(), *117, 118*
plot.traceline, 139, *203*
plot.traceline(), *202*, *204*
print.ctt, 142
print.ctt(), *44*, *47*, *198*
print.summary.ctt, 143
prm, 144
prm(), *53*, *190*

rdif, 145
rdif(), *32*, *40, 41*, *109*, *147*, *149*, *165*
reval_mst, *94*, *125*, 153, *170*, *173, 174*, *195*
reval_mst(), *155*
ripd, 158, *192, 193*
ripd(), *160*
run_flexmirt, 167
run_mst, *92–94*, *125*, *132*, 169, *195*
run_mst(), *155*

score_resp, 178
score_resp(), *4*, *45*, *48–51*, *96*
shape_df, *92*, *170*, 182
shape_df(), *21*, *24*, *28*, *32*, *34*, *36*, *38*, *44, 45*, *48*, *54*, *58*, *61*, *68*, *71*, *81*, *87*, *91*, *105*, *109*, *111*, *113*, *115*, *118*, *120*, *123*, *127*, *147*, *151*, *154*, *157*, *160*, *165*, *185*, *188*, *190*, *199*, *201, 202*, *205*
shape_df_fipc, 184
shape_df_fipc(), *7*, *59*, *61*, *81*, *183*

simCAT_DC, 186, *193*
simCAT_MX, 187, *193*
simdat, *170*, *172*, *174*, 188
simdat(), *8*, *20, 21*, *24*, *28*, *32*, *34*, *36*, *38*, *68*, *73*, *87*, *91*, *105*, *109*, *111, 112*, *115*, *120*, *123*, *127*, *147*, *151*, *160*, *165*, *183*, *199*, *201, 202*, *205*
simIPD, 191
simMG, 194
simMST, *174*, 195
stats::dbeta(), *55*, *69*, *75*
stats::dlnorm(), *55*, *69*, *75*
stats::dnorm(), *55*, *69*, *75*
stats::nlminb(), *57*, *69, 70*, *76*
summary, 196
summary.ctt, 197
summary.ctt(), *44*, *47*, *142–144*
sx2_fit, 198
sx2_fit(), *97, 98*, *200*

traceline, 202
traceline(), *139*, *141*

uniroot, *93*

write.flexmirt, 204