

Package: irtQ (via r-universe)

October 24, 2026

Type Package

Title Unidimensional Item Response Theory Modeling

Version 1.4.0

Date 2026-10-16

Description Fits unidimensional item response theory (IRT) models to test data that include dichotomous and polytomous items, calibrates pretest items with fixed item and fixed ability parameter calibration, performs multiple-group calibration, estimates the abilities of examinees, and evaluates model-data fit at the item level. It also provides tools for detecting differential item functioning and item parameter drift, for computing classification accuracy and consistency, and for evaluating and simulating multistage-adaptive test panels, together with classical test theory functions for item- and test-level statistics (e.g., item difficulty, item-total correlation, and coefficient alpha) and for scoring 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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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

Roxygen list(markdown = TRUE)

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

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

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libicu-dev

Repository <https://hwangq.r-universe.dev>

Date/Publication 2026-10-24 05:14:20 UTC

RemoteUrl <https://github.com/hwangq/irtq>

RemoteRef HEAD

RemoteSha 5ad30cd2aa3a7e6a74964df3162e70b0352e7ebc

Contents

irtQ-package	3
bind.fill	16
bisection	17
bring.flexmirt	19
cac_lee	22
cac_rud	26
catsib	29
covirt	39
crdif	42
ctt	51
ctt_distr	55
drm	60
est_irt	61
est_item	78
est_mg	84
est_score	100
find_cut	108
freq_score	112
gen.weight	113
getirt	115
grdif	120
info	131
irtfit	134

llike_score	141
LSAT6	143
lwrc	143
panel_info	145
pcd2	147
plot.find_cut	153
plot.info	155
plot.irtfit	157
plot.traceline	161
print.ctt	164
print.summary.ctt	165
prm	166
rdif	168
reval_mst	177
ripd	182
run_flexmirt	193
run_mst	195
score_resp	205
shape_df	209
shape_df_fipc	211
simCAT_DC	213
simCAT_MX	214
simdat	215
simIPD	218
simMG	221
simMST	222
summary	223
summary.ctt	225
sx2_fit	226
traceline	231
write.flexmirt	233

Index	236
--------------	------------

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 (`est_mg()`; Bock & Zimowski, 1997).

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 with item fit statistics (`irtfit()`, `sx2_fit()`)
- Detecting DIF with RDIF, RDIF-CR, GRDIF (multiple groups), and CATSIB (`rdif()`, `crdif()`, `grdif()`, `catsib()`)
- Detecting item parameter drift (IPD) with the residual-based RIPD framework and the pseudo-count D2 statistic (`ripd()`, `pcd2()`)
- Computing classification accuracy and consistency indices (`cac_lee()`, `cac_rud()`)
- Evaluating and simulating multistage-adaptive test (MST) panels (`panel_info()`, `reval_mst()`, `run_mst()`, `find_cut()`)
- Importing item or ability parameters from BILOG-MG, PARSCALE, flexMIRT, and the **mirt** package (`bring.flexmirt()`), and writing flexMIRT parameter files (`write.flexmirt()`)
- Running flexMIRT (Cai, 2017) from R on Windows (`run.flexmirt()`)
- 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

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: 1.4.0
Date: 2026-10-16
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:

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

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. With $D = 1$ (the default in **irtQ**), the model is on the logistic metric; $D = 1.702$ makes the logistic function approximate the normal ogive function. 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 ($k = 0, 1, \dots, K - 1$) is given by:

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

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

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

where $P^*(Y \geq k | \theta)$ 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:

$$\begin{aligned} P(Y = 0 | \theta) &= 1 - P^*(Y \geq 1 | \theta), \\ P(Y = K - 1 | \theta) &= P^*(Y \geq K - 1 | \theta). \end{aligned}$$

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

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

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.

A GRM or GPCM item with $K = 2$ score categories is equivalent to a 2PLM item. In the item metadata it has one threshold in `par.2` and NA in `par.3`, and `info()` and `traceline()` compute it with the same values as the 2PLM.

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 from the cross-product approximation of the information matrix (Meilijson, 1989), to which the information of any item parameter priors is added.

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. Use the default $D = 1$ for the logistic metric, or $D = 1.702$ to approximate 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.

Missing responses coded as NA need no further setting; when another code is used (e.g., -9), give it in 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 FIPC

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 FAPC

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. Use the default $D = 1$ for the logistic metric, or $D = 1.702$ to approximate the normal ogive model.

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

Example 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. doi:10.1080/15366367.2018.1511199.
- Chalmers, R. P. (2012). mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, 48(6), 1-29. doi:10.18637/jss.v048.i06
- 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. doi:10.18637/jss.v059.i07.
- 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* (CASMA Research Report No. 49). Center for Advanced Studies in Measurement and Assessment, University of Iowa.
- 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. doi:10.1111/j.17453992.2010.00179.x.
- Laplace, P. S. (1820). *Theorie analytique des probabilités* (3rd ed., in French). Paris: 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. H., & Lissitz, R. W. (2004). Applications of the analytically derived asymptotic standard errors of item response theory item parameter estimates. *Journal of Educational Measurement*, 41(2), 85-117. doi:10.1111/j.17453984.2004.tb01109.x.
- 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. doi:10.1111/jedm.12415.
- Lord, F. M. (1980). Applications of item response theory to practical testing problems. Lawrence Erlbaum Associates.
- Lord, F. M., & Wingersky, M. S. (1984). Comparison of IRT true-score and equipercentile observed-score "equatings." *Applied Psychological Measurement*, 8(4), 453-461. doi:10.1177/014662168400800409.
- Magis, D., & Barrada, J. R. (2017). Computerized adaptive testing with R: Recent updates of the package catR. *Journal of Statistical Software*, 76(Code Snippet 1), 1-19. doi:10.18637/jss.v076.c01.
- 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.
- Moore, D. S. (1977). Generalized inverses, Wald's method, and the construction of chi-squared tests of fit. *Journal of the American Statistical Association*, 72(357), 131-137. doi:10.1080/01621459.1977.10479921.
- 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. doi:10.3102/10769986021004365.
- 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. doi:10.1111/j.1745-3984.2000.tb01076.x.
- 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(4), 397-412. doi:10.1007/BF02293705.
- Veerkamp, W. J. J., & Glas, C. A. W. (2000). Detection of known items in adaptive testing with a statistical quality control method. *Journal of Educational and Behavioral Statistics*, 25(4), 373-389. doi:10.3102/10769986025004373.
- 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

<code>list</code>	A list containing numeric vectors of possibly different lengths.
<code>type</code>	A character string indicating the type of binding to perform. Options are "rbind" or "cbind". Default is "rbind".
<code>fill</code>	A value used to fill missing elements when aligning the vectors. For <code>type = "cbind"</code> , this fills missing rows in shorter columns; for <code>type = "rbind"</code> , 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. The matrix has no row or column names.

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

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

Details

The bisection method is a root-finding algorithm for a continuous function whose values at the lower (`lb`) and upper (`ub`) bounds have opposite signs. The method repeatedly halves the interval until its width is no greater than `tol` or the maximum number of iterations (`max.it`) is reached, and returns the midpoint of the final interval. A warning is issued when the function values at the bounds do not have opposite signs or when `max.it` is reached before the interval is no wider than `tol`.

Value

A list with the following components:

- `root`: The estimated root of the function.
- `iter`: The number of iterations performed.
- `delta`: The width of the final search interval.

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 parameter estimates from the output files of BILOG-MG 3, PARSCALE 4, and flexMIRT. `bring.mirt()` imports the item parameters, together with the latent mean and variance, of a model fitted with 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

<code>file</code>	The path to the output file that contains the item or ability parameter estimates.
<code>type</code>	A character string giving the type of output file: "par" for an item parameter file or "sco" for an ability (score) file. The case is ignored. Default is "par". Not used by <code>bring.mirt()</code> .
<code>rePar</code>	Logical. Used only by <code>bring.flexmirt()</code> with <code>type = "par"</code> . If TRUE (default), the intercepts of dichotomous and graded response model (GRM) items are converted to difficulty (threshold) parameters and the logits of the guessing parameters to guessing parameters, so that <code>full_df</code> follows the irtQ item metadata format. This requires a unidimensional model. If FALSE, the slope-intercept values in the file are kept, and <code>full_df</code> cannot then be used directly by other irtQ functions.
<code>rePar.gpc</code>	Logical. Used only by <code>bring.flexmirt()</code> with <code>type = "par"</code> . If TRUE (default), the nominal model parameters of generalized partial credit model (GPCM) items are converted to a slope and step parameters. The function stops when the file contains a nominal model item or a multidimensional GPCM item; use <code>rePar.gpc = FALSE</code> to read such a file, in which case the scoring function and intercept values are kept and nominal items get the model name "NOMM".

n.factor	An integer giving the number of latent traits (factors). Used only by <code>bring.flexmirt()</code> with <code>type = "sco"</code> . Default is 1.
x	A fitted model object returned by <code>mirt::mirt()</code> for a single group and a unidimensional model. The mirt package, listed in Suggests, must be installed to use <code>bring.mirt()</code> .

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 functions read the layout of the file, not its name. The usual files are the ".PAR" and ".SCO" files of BILOG-MG and PARSCALE and the "-prm.txt" and "-sco.txt" files of flexMIRT; other flexMIRT output files (e.g., "-irt.txt") cannot be read. For **mirt**, the fitted model object is passed directly.

`bring.flexmirt()` can also read the files of a multidimensional model: item parameter files with `rePar = FALSE` (and `rePar.gpc = FALSE` when GPCM items are present), and ability files with `n.factor` set to the number of factors. The other functions of **irtQ** work only with unidimensional models.

For polytomous items, `bring.flexmirt()`, `bring.parscale()`, and `bring.mirt()` import the parameters of the graded response model (GRM) and the generalized partial credit model (GPCM); a polytomous **mirt** item of type "Rasch" is imported as a GPCM item with a slope of 1. `bring.bilog()` reads dichotomous items only. `bring.parscale()` does not support the model codes 2, 4, 6, and 8 of PARSCALE for polytomous items and stops with an error when the file has them.

`bring.mirt()` accepts the item types "Rasch", "2PL", and "3PL" for dichotomous items and "Rasch", "graded", "gpcm", and "gpcmIRT" for polytomous items. A two-category item of a mixed-format test may also be fitted with "graded", "gpcm", or "gpcmIRT". A model fitted to multiple groups is not supported, and the function stops with an error; extract one group first with `mirt::extract.group()`.

In `full_df`, `bring.flexmirt()` labels dichotomous items "3PLM" or "2PLM", while `bring.bilog()`, `bring.parscale()`, and `bring.mirt()` label them "DRM", with a guessing parameter of 0 for items without guessing. For `bring.mirt()`, this includes two-category items fitted with the "graded", "gpcm", or "gpcmIRT" model. Change the model column if another model is wanted in `est_irt()`, which fits "DRM" items as 3PLM items. The parameters are kept on the scale of the program, and the files do not record the scaling constant; pass the value of D used in the calibration (e.g., 1 for flexMIRT and **mirt**) to the other **irtQ** functions.

Value

- `bring.flexmirt()`: a named list with one element per group (Group1, Group2, ...). With `type = "par"`, each element is a list with `id`, `factor` (number of dimensions), `cats`, `model`, `par` (a matrix of item parameters), `full_df` (the item metadata), `mean` and `cov` (the latent mean vector and covariance matrix), and `emphist` (a data frame with `theta` and `weight`, or NULL when no empirical histogram was estimated). With `type = "sco"`, each element is a list with `theta` and `se`, plus `iter` (the number of iterations) or `imputation` (the imputation number) when the file contains that column. `se` is NULL for multidimensional models and for imputed values.

- `bring.bilog()`: with `type = "par"`, a list with `par` and `full_df`; with `type = "sco"`, a list with one element per group code in the file, each a list with `theta`, `se`, and `full_df` (a data frame with `group`, `id`, `nitem`, `ncorrect`, `p`, `theta`, and `se`).
- `bring.parscale()`: with `type = "par"`, a list with `id`, `factor`, `cats`, `model`, `par_parscale` (the parameters and standard errors as reported by PARSCALE), `par`, and `full_df`; with `type = "sco"`, a data frame with `id`, `theta`, and `se` (a value of 999 becomes NA).
- `bring.mirt()`: a list with `id`, `factor`, `cats`, `model` (a list with the **mirt** item types, `old`, and the **irtQ** model names, `new`), `emphist` (a matrix of quadrature points and weights), `par_mirt` (the **mirt** parameters), `par`, `full_df`, and `ability` (the latent mean and variance).

Sample Output Files

Two sample item parameter files are included in the package: the PARSCALE 4 file "parscale_sample.PAR" and the flexMIRT file "flexmirt_sample-prm.txt". Both contain the estimates for the same mixed-format test of 55 items: 50 dichotomous items fit with the 3PL model and five items with five score categories fit with the GRM (items 39, 40, and 53 to 55). The examples below import these files.

Note

For an item parameter file, the element `full_df` of the result is the item metadata (item IDs, numbers of score categories, model names, and item parameters) used by the other functions of **irtQ**. See `info()` or `simdat()` for the metadata format.

For GPCM items, `full_df` contains the step parameters $b_v = \beta - \tau_v$, computed from the overall location β and the category parameters τ_v reported by the program (PARSCALE) or from the nominal model intercepts (flexMIRT). See the **IRT Models** section in [irtQ-package](#) for 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. doi:10.18637/jss.v048.i06
- 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 using the item response theory (IRT) approach of Lee (2010). Tests may contain dichotomous items, polytomous items, or both.

Usage

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

Arguments

<code>x</code>	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function.
<code>cutscore</code>	A numeric vector of finite cut scores in strictly ascending order. The $K - 1$ cut scores divide examinees into K performance levels. With <code>cut.obs = TRUE</code> , the cut scores are on the observed summed score metric. With <code>cut.obs = FALSE</code> , they are on the ability (theta) metric. A score equal to a cut score is assigned to the higher level.
<code>theta</code>	A numeric vector of ability estimates, one per examinee, used by the P method. It is ignored when <code>weights</code> is supplied. Missing values (NA) are excluded with a warning. Default is NULL.
<code>weights</code>	A two-column data frame or matrix used by the D method. The first column holds the quadrature points (nodes), and the second column holds the corresponding weights, which should sum to 1; weights that do not sum to 1 are rescaled to sum to 1 with a warning (for example, frequencies can be given as weights). The weights must be nonnegative and finite and have a positive sum. <code>gen.weight()</code> creates such a data frame. When <code>weights</code> is supplied, <code>theta</code> is ignored. Default is NULL.
<code>D</code>	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1.
<code>cut.obs</code>	Logical. If TRUE, <code>cutscore</code> is on the observed summed score metric. If FALSE, <code>cutscore</code> is on the ability (theta) metric and is converted to the summed score metric with the test characteristic curve (TCC). 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.

For each ability value (a quadrature point in the D method, or an ability estimate in the P method), the conditional distribution of the observed summed score is computed with the Lord-Wingersky recursion (Lord & Wingersky, 1984) and its extension to polytomous items (see `lwrc()`). Summing this distribution over the score ranges defined by the cut scores gives the probability of being classified into each level. The true level of an ability value is found by comparing its expected summed score (the TCC value) with the same cut scores, so the observed cut scores also serve as the true cut scores. The conditional classification accuracy is the probability of being classified into the true level, and the conditional classification consistency is the sum of the squared level probabilities, which is the probability of the same classification on two independent administrations (Lee, 2010).

In the D method (weights supplied), the marginal indices are the weighted sums of the conditional indices over the quadrature points. In the P method (theta supplied), each examinee receives the weight $1/N$, where N is the number of ability estimates (after the missing values are excluded), so the marginal indices are the averages of the conditional indices.

Value

A list with the following elements:

- **confusion**: A $K \times K$ matrix of expected proportions, with the true levels in rows (dimension name `True`) and the expected (observed) levels in columns (dimension name `Expected`). The entries are rounded to 7 decimal places and sum to 1 up to rounding.
- **marginal**: A data frame with the columns `level`, `accuracy`, and `consistency`. The rows for levels 1 to K give the contribution of the ability values whose true level is that level, that is, the weighted sums of their conditional indices. These rows add up to the last row, `marginal`, which holds the marginal classification accuracy and consistency.
- **conditional**: A data frame with one row per ability value and the columns `theta`, `weights`, `true.score` (expected summed score), `level` (true level, a factor), `accuracy`, and `consistency`.
- **prob.level**: A data frame with the columns `theta`, `weights`, `true.score`, and `level`, followed by `p.level.1` to `p.level.K`, the probabilities of being classified into each level.
- **cutscore**: The cut scores used, on the observed summed score metric. When `cut.obs = FALSE`, these are the converted cut scores.

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.
- Lord, F. M., & Wingersky, M. S. (1984). Comparison of IRT true-score and equipercentile observed-score "equatings." *Applied Psychological Measurement*, 8(4), 453-461. doi:10.1177/014662168400800409.

See Also

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

Examples

```
## -----
## 1. When the ability distribution is given by quadrature points and
##    weights (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)

# Marginal indices and confusion matrix
cac_2$marginal
cac_2$confusion

## -----
## 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
)

# Marginal indices and confusion matrix
cac_3$marginal
cac_3$confusion

```

cac_rud	<i>Classification Accuracy and Consistency Based on Rudner's (2001, 2005) Approach</i>
---------	--

Description

This function computes classification accuracy and consistency indices with the approach of Rudner (2001, 2005), which assumes that an ability estimate is normally distributed around the true ability with a standard deviation equal to its standard error. The indices can be computed over an ability distribution given by quadrature points and weights, or over individual ability estimates.

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 of finite cut scores on the ability (theta) metric, in strictly ascending order. The K - 1 cut scores divide examinees into K performance levels. An ability value equal to a cut score is assigned to the higher level.
theta	A numeric vector of ability estimates, one per examinee. It is ignored when weights is supplied. An examinee with a missing theta (or a missing se) is excluded with a warning. Default is NULL.
se	A numeric vector of the same length as theta representing the standard errors associated with each ability estimate. All standard errors must be positive. If NULL and x is supplied, standard errors are computed using the test information function. See the Details section for more information. When weights is supplied, se must have one value per quadrature point. 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	A two-column data frame or matrix. The first column holds the quadrature points (nodes), and the second column holds the corresponding weights, which should sum to 1; weights that do not sum to 1 are rescaled to sum to 1 with a warning (for example, frequencies can be given as weights). The weights must be nonnegative and finite and have a positive sum. gen.weight() creates such a data frame. When weights is supplied, theta is ignored. 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 1.702 is commonly used for this purpose. Default is 1. It is used only when se is computed from x.

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`. The conditional classification accuracy is the probability of being classified into the level that contains the ability value itself (Rudner, 2001, 2005). The conditional classification consistency is the sum of the squared level probabilities, as in Lee (2010); Rudner (2001, 2005) defines only accuracy indices. When individual ability estimates are used, each estimate is treated as the true ability, and each examinee receives the weight $1/N$, where N is the number of ability estimates after the examinees with a missing `theta` or `se` are excluded.

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

Value

A list with the following elements:

- `confusion`: A $K \times K$ matrix of expected proportions, with the true levels in rows (dimension name `True`) and the expected (observed) levels in columns (dimension name `Expected`). The entries are rounded to 7 decimal places and sum to 1 up to rounding.
- `marginal`: A data frame with the columns `level`, `accuracy`, and `consistency`. The rows for levels 1 to K give the contribution of the ability values whose true level is that level, that is, the weighted sums of their conditional indices. These rows add up to the last row, `marginal`, which holds the marginal classification accuracy and consistency.
- `conditional`: A data frame with one row per ability value and the columns `theta`, `weights`, `level` (true level, an integer), `accuracy`, and `consistency`.
- `prob.level`: A data frame with the columns `theta`, `weights`, and `level`, followed by `p.level.1` to `p.level.K`, the probabilities of being classified into each level.
- `cutscore`: The cut scores used, on the ability (`theta`) metric.

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.
- 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 a population ability distribution (D method)
## -----

# 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 (P method)
## -----

# 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

# Replace the SEs of the estimates at a bound of range (set to 99.9999)
# with the SEs from the test information function
```

```

at_bound <- se == 99.9999
se[at_bound] <- 1 / sqrt(info(x = x, theta = theta_hat[at_bound], D = 1, tif = TRUE)$tif)

# 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 & Rousos, 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,
rho.screen = TRUE,
...
)

## S3 method for class 'catsib'
print(x, digits = max(2L, getOption("digits") - 5L), ...)

```

Arguments

<code>x</code>	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. <code>x</code> is needed to estimate the abilities internally, that is, when <code>score = NULL</code> or <code>purify = TRUE</code> . It can be <code>NULL</code> when <code>score</code> and <code>se</code> are supplied and <code>purify = FALSE</code> ; the items are then labeled <code>item.1</code> , <code>item.2</code> , and so on. Default is <code>NULL</code> .
<code>data</code>	A matrix of examinees' scored responses (0 or 1). Rows represent examinees and columns represent items, in the order of the rows of <code>x</code> when <code>x</code> is supplied.
<code>score</code>	A numeric vector of examinees' ability estimates (theta values), in the same row order as <code>data</code> , with their standard errors in <code>se</code> . The estimates should be based on the item parameters in <code>x</code> . If <code>NULL</code> , abilities and standard errors are estimated internally with est_score() using <code>method</code> , <code>range</code> , <code>norm.prior</code> , <code>nquad</code> , <code>weights</code> , and <code>ncore</code> , and <code>x</code> must be supplied. When <code>purify = TRUE</code> , abilities and standard errors are re-estimated internally at every purification iteration, so a supplied score is used only in the initial analysis. A missing value in <code>score</code> excludes the examinee from the analysis, also during purification, with a warning. Default is <code>NULL</code> .
<code>se</code>	A numeric vector of the standard errors of the ability estimates in <code>score</code> , in the same order. The standard errors are used to estimate the reliability of the ability estimates in the regression correction (see Details). It must be supplied when <code>score</code> is supplied, and it is ignored when <code>score = NULL</code> . Default is <code>NULL</code> .
<code>group</code>	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.
<code>focal.name</code>	A single numeric or character value specifying the focal group. For instance, given <code>group = c(0, 1, 0, 1, 1)</code> and <code>'1'</code> indicating the focal group, set <code>focal.name = 1</code> .
<code>item.skip</code>	A numeric vector of item positions (columns of <code>data</code> , or rows of <code>x</code>), or a logical vector with one value per item, identifying the items to exclude from the

analysis. Skipped items still contribute to the ability estimates, but their results (beta, se, z.beta, p, and the contingency table) are set to NA, and they are never flagged or removed during purification; their sample sizes (n.ref, n.foc, and n.total) are still reported, as in `rdif()` and `crdif()`. If NULL, all items are analyzed. Default is NULL.

D	A scaling factor in IRT models, used only when abilities are estimated internally. Default is 1.
n.bin	A numeric vector of two positive integers giving the maximum and minimum numbers of equal-width bins (intervals) of the corrected ability scale, in this order (the maximum must not be smaller than the minimum). For each item, the number of bins starts at the maximum and decreases by one until the condition set by <code>max.del</code> is met; if it is not met with the minimum number, the minimum number is used. Default is <code>c(80, 10)</code> . See Details .
min.binsize	An integer of 2 or more specifying the minimum number of examinees of each group in a bin. A bin is used in the computation of $\hat{\beta}$ only if it contains at least <code>min.binsize</code> examinees of both the reference and focal groups who responded to the item. Default is 3, the value used by Nandakumar and Roussos (2004). See Details .
max.del	A numeric value specifying the maximum proportion of the examinees of each group who responded to the item that may be excluded because they fall in bins that are not used (see <code>min.binsize</code>). The number of bins is reduced until the excluded proportion is at most <code>max.del</code> in both groups. It must be at least 0 and less than 1. Default is 0.075. See Details .
weight.group	A character string specifying the target ability distribution used to weight the bins in $\hat{\beta}$ and its standard error: "comb" for the combined distribution of the reference and focal groups, "foc" for the focal group, or "ref" for the reference group. The weight of a bin is the proportion of the examinees of the target distribution in the bins used who fall in that bin. Default is "comb".
alpha	A numeric value specifying the significance level (α) of the two-sided test of the CATSIB statistic. An item is flagged when its p-value is less than or equal to alpha. The comparison uses unrounded p-values, although the p-values reported in <code>dif_stat</code> are rounded to four decimal places. 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. Each iteration removes one item, so <code>max.iter</code> should be at least as large as the number of items expected to be flagged. If the limit is reached while flagged items remain, a warning is issued, the complete element of the purification results is FALSE, and the items flagged in the last iteration are added to the flagged items. This argument is not passed to <code>est_score()</code> .
min.resp	A single nonnegative number, usually a positive integer, specifying the minimum number of item responses that an examinee must have to be used in the analysis. All responses of examinees with fewer than <code>min.resp</code> (but at least one) responses are set to NA before the ability estimation, in the initial analysis and at every purification iteration, also when score is supplied, and these

examinees are excluded from the analysis. A warning reports the number of examinees excluded in the initial analysis, and one warning at the end of the purification reports the examinees excluded during it. A large `min.resp` can leave a group with fewer than two ability estimates within range in a later purification iteration, which stops the purification with a warning. If `NULL`, no minimum is applied. Default is `NULL`. See **Details** for more information.

method	A character string indicating the scoring method used to estimate the abilities when <code>score = NULL</code> and in every purification iteration. Available options are: • "ML": Maximum likelihood estimation • "MLF": Maximum likelihood estimation with fences • "WL": Weighted likelihood estimation (Warm, 1989) • "MAP": Maximum a posteriori estimation (Hambleton et al., 1991) • "EAP": Expected a posteriori estimation (Bock & Mislevy, 1982) The methods "EAP.SUM" and "INV.TCC" of <code>est_score()</code> are not supported, because they do not give an ability estimate for each response pattern. "ML" or "WL" is recommended, because the regression correction assumes unbiased ability estimates (see Details). Default is "ML".
range	A numeric vector of length two specifying the lower and upper bounds of the ability scale. It is used when abilities are estimated internally with the scoring methods "ML", "MLF", "WL", and "MAP". In addition, ability estimates at or beyond these bounds are excluded when the mean, the variance, and the error variance of the ability estimates are computed for the regression correction (see Details); this also applies when <code>score</code> is supplied, so <code>range</code> should match the bounds used to obtain <code>score</code>. Default is <code>c(-5, 5)</code>.
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 <code>method</code> is "ML" or "WL". Default is <code>c(0, 1)</code>.
nquad	An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. Used only when <code>method</code> 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 <code>gen.weight()</code>. If <code>NULL</code> and <code>method = "EAP"</code>, default quadrature values are generated based on the <code>norm.prior</code> and <code>nquad</code> arguments. Ignored if <code>method</code> is "ML", "WL", or "MAP".
ncore	An integer specifying the number of logical CPU cores used by <code>est_score()</code> when abilities are estimated internally. The test statistics themselves are not computed in parallel. Default is 1.
verbose	Logical. If <code>TRUE</code>, the progress of the purification procedure is printed to the console. Default is <code>TRUE</code>.
rho.screen	Logical. If <code>TRUE</code>, the ability estimates whose squared standard error is equal to or greater than the variance of the ability estimates of their group are left out

of the computation of the mean, the variance, and the error variance that define $\hat{\rho}_G^2$ in the regression correction. These examinees are still matched and used to compute $\hat{\beta}$. If FALSE, $\hat{\rho}_G^2$ is computed from all ability estimates within range, as in Nandakumar and Roussos (2004, Equation 7). The screening is not part of the original procedure and is added in irtQ. Default is TRUE. See **Details**.

... Additional arguments passed to `est_score()` when abilities are estimated internally, for example `tol`. The `est_score()` arguments `se`, `max.iter`, and `missing` cannot be passed this way, because `catsib()` uses arguments with these names for its own purposes.

digits Not used; kept for consistency with the other print methods.

Details

CATSIB (Nandakumar & Roussos, 2004) estimates the amount of DIF in an item as $\hat{\beta} = \sum_k (\hat{P}_{R,k} - \hat{P}_{F,k}) \hat{p}_k$, where $\hat{P}_{R,k}$ and $\hat{P}_{F,k}$ are the proportions of correct responses of the reference and focal groups in bin k of the corrected ability scale, and $\hat{p}_k$ is the weight of bin k (see `weight.group`). A positive $\hat{\beta}$ indicates DIF in favor of the reference group. The standard error of $\hat{\beta}$ is estimated from the sample variances of the item responses in each bin and group (Nandakumar & Roussos, 2004, Equation 10), and the test statistic $\hat{\beta}/SE(\hat{\beta})$ (`z.beta`) is compared with the standard normal distribution in a two-sided test. Examinees are matched on $\hat{\theta}^*$, an estimate of the expected value of θ given $\hat{\theta}$ obtained by the regression correction described below. The range of $\hat{\theta}^*$ is divided into K equal-width bins, and examinees are classified into the bins by their $\hat{\theta}^*$. A bin is used only if it contains at least `min.binsize` examinees of both groups who responded to the item; Nandakumar and Roussos (2004) required at least three, which is the default.

To choose the number of bins K , `catsib()` follows the rule of Nandakumar and Roussos (2004). For each item, K starts at a large value (80 by default), and if more than 7.5% of the examinees of either group who responded to the item would be excluded because they fall in bins that are not used, K is reduced by one, until at most 7.5% of each group is excluded. Nandakumar and Roussos (2004) did not allow fewer than 20 bins, but they reported that a minimum of 10 bins reduced the estimation bias caused by excluding examinees from sparse bins. `catsib()` therefore uses a minimum of 10 bins by default; if the condition is not met with the minimum number, the minimum number is used. The maximum and minimum numbers of bins are set by `n.bin`, and the maximum proportion of excluded examinees by `max.del`.

Li and Stout (1996) and Nandakumar and Roussos (2004) used the combined distribution of the two groups as the target ability distribution, which is the default of `weight.group`. See Nandakumar and Roussos (2004) for further details about the CATSIB method.

Because the ability distributions of the reference and focal groups can differ (impact), matching examinees directly on $\hat{\theta}$ would bias $\hat{\beta}$ toward DIF in favor of the group with the higher mean ability. CATSIB therefore applies a regression correction separately in each group G (Nandakumar & Roussos, 2004, Equation 7): $\hat{\theta}^* = \bar{\hat{\theta}}_G + \hat{\rho}_G^2 (\hat{\theta} - \bar{\hat{\theta}}_G)$, where $\bar{\hat{\theta}}_G$ is the mean of the ability estimates in group G and $\hat{\rho}_G^2 = 1 - \hat{\sigma}_{e,G}^2 / \hat{\sigma}_{\hat{\theta},G}^2$ is the estimated reliability of the ability estimates in group G . Here $\hat{\sigma}_{e,G}^2$ is the mean of the squared standard errors (`se`) and $\hat{\sigma}_{\hat{\theta},G}^2$ is the variance of the ability estimates in group G . Ability estimates at or beyond a bound of `range` are excluded when these quantities are computed, because their standard errors are not meaningful; this also applies when `score` is supplied, so `range` should match the bounds used to obtain `score`.

Because $\hat{\sigma}_{e,G}^2$ is a mean of squared standard errors, a few examinees with very large standard errors, typically examinees with very low or very high scores whose abilities are poorly determined, can make $\hat{\rho}_G^2$ much smaller than the reliability of the other ability estimates, or even negative. The corrected ability estimates then shrink too much toward the group mean, which distorts the matching and can inflate the Type I error rate, particularly during purification. When `rho.screen = TRUE` (the default), the examinees whose squared standard error is greater than or equal to $\hat{\sigma}_{\theta,G}^2$ of their group are left out of the computation of the mean, the variance, and the error variance that define $\hat{\rho}_G^2$, unless fewer than two estimates would be left. For these examinees the individual reliability $1 - SE^2/\hat{\sigma}_{\theta,G}^2$ is zero or negative, that is, their ability estimates carry no information about their abilities beyond the spread of the group. They still receive corrected ability estimates and are used in the matching and in the computation of $\hat{\beta}$. This screening is not part of the original procedure (Nandakumar & Roussos, 2004) and is added in `irtQ`; set `rho.screen = FALSE` to compute $\hat{\rho}_G^2$ from all ability estimates within range as in the original procedure. Even with the screening, the Type I error rate of CATSIB can be higher than the nominal level, and in data without DIF the purification can increase the number of false positives. In addition, $\hat{\rho}_G^2$ is bounded to the interval $[0.05, 1]$, and a warning is issued when its unbounded value is below 0.05. A negative value would reverse the order of the ability estimates within a group, and a value near zero would place almost all corrected ability estimates of a group in one bin. This bound is not part of the original procedure and is added in `irtQ`. Even with the bound, the corrected ability estimates of the two groups may not overlap when the groups differ in ability; then no bin contains enough examinees of both groups for an item, $\hat{\beta}$, its standard error, `z.beta`, and the p-value are NA, one warning names the items, and the items are not flagged.

The regression correction assumes that the ability estimates are unbiased, that is, that $\hat{\theta}$ equals θ plus an error with mean zero. The Bayesian estimates ("MAP" and "EAP") are already shrunk toward the prior mean, so the correction shrinks them a second time. Use "ML" or "WL", especially when the abilities are re-estimated during the purification.

CATSIB, like SIBTEST (Shealy & Stout, 1993), was designed and evaluated for uniform DIF (Nandakumar & Roussos, 2004). Because $\hat{\beta}$ sums signed differences across bins, differences of opposite sign can cancel, so its sensitivity to nonuniform or mixed DIF is limited, and results for items suspected of nonuniform DIF should be interpreted with caution.

Nandakumar and Roussos (2004) did not propose a purification procedure for CATSIB, but `catsib()` can apply an iterative purification procedure in the manner of Lim et al. (2022). It requires `x`. When `purify = TRUE`, at each iteration the flagged item with the smallest p-value is removed (the p-values are compared on the log scale), the abilities and their standard errors are re-estimated from the remaining items with the scoring method specified in `method`, and the CATSIB statistics of the remaining items are recomputed. Supplied `score` and `se` are therefore used only in the initial analysis. The procedure stops when no remaining item is flagged or when `max.iter` iterations have been performed. If no item is flagged in the initial analysis, no iteration is performed. If every item is flagged, the procedure stops with a warning, because no item is left to estimate the abilities, and `complete` is FALSE. The procedure also stops with a warning and `complete = FALSE` when, after an item is removed, a group has fewer than two ability estimates within range, which the regression correction requires, or no bin contains enough examinees of both groups for any remaining item; the remaining items then keep the results of the previous iteration, and the items flagged in that iteration are added to the flagged items.

Scoring based on a small number of item responses can lead to large standard errors, potentially reducing the accuracy of DIF detection with CATSIB. The `min.resp` argument excludes such examinees. For example, if `min.resp = 5`, examinees who responded to fewer than five items (but

at least one) have all their responses treated as missing (i.e., NA) before the abilities are estimated, in the initial analysis and at every purification iteration, and a warning reports their number in the initial analysis. As a result, their ability estimates are missing, and they are not used in the computation of the CATSIB statistics. When score is supplied, the scores of these examinees are also treated as missing. During purification, the count is based on the items that remain after the flagged items are removed, and an examinee excluded once stays excluded. If `min.resp = NULL`, a score is computed for any examinee with at least one valid item response.

Examinees whose ability estimate is NA, such as examinees without any item response or examinees with NA in a supplied score, are excluded from the analysis, and a warning reports the number of examinees with item responses who are excluded because of a missing value in score; these examinees stay excluded during purification. During purification, an examinee whose remaining responses are all removed with the flagged items, or who has fewer than `min.resp` responses to the remaining items, has no ability estimate and is excluded, and one warning at the end reports the number of such examinees. The sample sizes `n.ref` and `n.foc` count only the examinees in the bins used to compute $\hat{\beta}$.

The `group` argument should contain exactly two distinct values, one for the reference group and one for the focal group identified by `focal.name`, and its length must equal the number of rows in data. Examinees with NA in `group` are excluded from the analysis. The function stops when the reference group or the focal group has fewer than two ability estimates within range, because the regression correction cannot be computed.

The print method of this class has an argument `digits` that is not used; the statistics are printed rounded to three decimal places.

Value

This function returns an object of class "catsib", which is a list with the following five components:

- no_purify** A list containing the results of the DIF analysis without applying a purification procedure. This list includes:
 - dif_stat** A data frame with one row per item and the columns `id`, `beta`, `se`, `z.beta`, `p`, `n.ref`, `n.foc`, and `n.total`: the item ID, the CATSIB statistic $\hat{\beta}$, its standard error, the standardized statistic, the two-sided p-value, and the numbers of examinees of the reference and focal groups in the bins used to compute $\hat{\beta}$, and their sum. A positive $\hat{\beta}$ indicates DIF in favor of the reference group. All values are rounded to four decimal places.
 - dif_item** A numeric vector of the positions (columns of data) of the flagged items, or NULL if no item is flagged.
 - contingency** A list with one element per item, named by item ID. Each element is a data frame with one row per bin used to compute $\hat{\beta}$ and the columns `bin` (the interval of corrected ability estimates), `n.ref`, `prop.ref`, `var.ref`, `n.foc`, `prop.foc`, and `var.foc` (the number of examinees, the proportion of correct responses, and the sample variance of the item responses of each group in the bin), `n.total`, `weight` (the weight of the bin), `beta` (the weighted difference in the proportions of correct responses), and `var.beta` (the contribution of the bin to the squared standard error), followed by a row whose `bin` is `Total`, with the totals of `n.ref`, `n.foc`, `n.total`, `beta`, and `var.beta`. The element is NA for a skipped item.

purify	A logical value indicating whether purification was requested (the <code>purify</code> argument).
with_purify	A list with the results of the purification procedure. All elements are NULL when <code>purify = FALSE</code> . The elements are: dif_stat A data frame with the same columns as <code>no_purify\$dif_stat</code> and an additional column <code>n.iter</code>. For an item removed during purification, the row reports the statistics from the iteration in which the item was removed, and <code>n.iter</code> is that iteration minus 1 (0 for an item removed on the basis of the initial analysis). For the other items, the row reports the statistics from the last iteration, and <code>n.iter</code> is the number of that iteration. If no item is flagged in the initial analysis, this is <code>no_purify\$dif_stat</code> with <code>n.iter = 0</code>. dif_item A numeric vector of the positions (rows of <code>x</code>) of the items flagged by the CATSIB statistic, sorted in ascending order. It contains the items removed during purification and, if the purification stops before it is complete (see <code>complete</code>) with items left, the items flagged in the last iteration (iteration <code>n.iter</code>). It contains all items if every item is flagged. NULL if no item is flagged in the initial analysis. n.iter The number of purification iterations performed (0 if no item is flagged in the initial analysis). An iteration after which the statistics cannot be recomputed (see Details) is not counted. complete A logical value. TRUE if the procedure stopped because no remaining item was flagged (including the case in which no item is flagged in the initial analysis), and FALSE if it stopped because <code>max.iter</code> was reached, because every item was flagged, because a group had fewer than two ability estimates within range, or because no remaining item had a usable bin. contingency A list with the same structure as <code>no_purify\$contingency</code>. For an item removed during purification, it holds the table from the iteration in which the item was removed; for the other items, it holds the table from the last iteration. If no item is flagged in the initial analysis, this is <code>no_purify\$contingency</code>.
alpha	A numeric value indicating the significance level (α) used for the tests.
call	The matched function call.

Methods (by generic)

- `print(catsib)`: Print method for an object of class `catsib` (given as `x`). It prints the flagged items and the CATSIB statistics of the analyses without and with purification, rounded to three decimal places.

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.

- Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. Newbury Park, CA: Sage.
- 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.
- Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, 54(3), 427-450. doi:10.1007/BF02294627.

See Also

`rdif()`, `grdif()`, `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 = rep(0.15, 4)),
    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))

# (1) Compute the CATSIB statistic using provided scores and standard errors,
# without purification (x can be NULL in this case)
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)

# (2) Compute the CATSIB statistic with purification (x is required; not
# run: it takes about 7 seconds because the abilities of the examinees
# are re-estimated at every iteration)
#
# 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 examinee response data. For each item, the matrix is the inverse of the expected (Fisher) information of the item parameters, which is integrated over an assumed ability distribution and multiplied by the calibration sample size (Thissen & Wainer, 1982; Li & Lissitz, 2004). The square roots of the diagonal elements are 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 (item IDs, numbers of score categories, IRT models, and item parameters). The item parameters are treated as the true values at which the information is evaluated; for 1PLM items and for PCM items specified in <code>pcm.loc</code> , the slope in x is held fixed. See est_irt() or simdat() for the format of the item metadata, which can be created with shape_df() . |
| D | A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1. |
| nstd | An integer, or a vector of integers with one value per item in x, giving the number of examinees used to calibrate the items. The information of each item is multiplied by its sample size, so the standard errors are proportional to $1 / \sqrt{nstd}$. Default is 1000. See Details . |

pcm.loc	A vector of integers giving the positions (rows of <code>x</code>) of GPCM items that were calibrated with the partial credit model (PCM). For these items, the slope is held at the value in <code>x</code> and the variance-covariance matrices are computed only for the threshold parameters. Positions of items that are not GPCM items are ignored. Default is NULL. See Details .
norm.prior	A numeric vector of length two specifying the mean and standard deviation of the normal ability distribution over which the information is integrated. Used to generate the Gaussian quadrature points and weights when <code>weights = NULL</code> . Default is <code>c(0, 1)</code> .
nquad	An integer specifying the number of Gaussian quadrature points generated from the normal distribution given in <code>norm.prior</code> . Used only when <code>weights = NULL</code> . Default is 41.
weights	An optional two-column data frame or matrix of quadrature points (first column) and weights (second column) that defines the ability distribution. The weights should sum to 1, because they are not normalized. If <code>weights</code> is supplied, <code>norm.prior</code> and <code>nquad</code> are ignored. Such a data frame can be created with <code>gen.weight()</code> . Default is NULL.

Details

For each item, the asymptotic variance-covariance matrix is the inverse of the expected (Fisher) information matrix of the item parameters, which is integrated over the ability distribution with quadrature and multiplied by the sample size (Thissen & Wainer, 1982; Li & Lissitz, 2004). For an item with K score categories, the element of the information matrix for a pair of item parameters ξ_s and ξ_t is

$$I(\xi_s, \xi_t) = N \sum_{q=1}^Q A(X_q) \sum_{k=1}^K \frac{1}{P_k(X_q)} \frac{\partial P_k(X_q)}{\partial \xi_s} \frac{\partial P_k(X_q)}{\partial \xi_t},$$

where N is the sample size (`nstd`), X_q and $A(X_q)$ are the Q quadrature points and weights that represent the ability distribution (`norm.prior` and `nquad`, or `weights`), and $P_k(X_q)$ is the probability of a response in category k at X_q . This is the information of a categorical (multinomial) response. For a dichotomous item, it reduces to the familiar form in which each term is divided by $P(X_q)[1 - P(X_q)]$ (Li & Lissitz, 2004, Equation A4).

For polytomous items, the information above differs from Equation A8 of Li and Lissitz (2004). Their Equation A6 writes the likelihood of a response (and Equation A7 its logarithm) as $\prod_k P_k^{u_k} (1 - P_k)^{1-u_k}$, where u_k is 1 for the observed category and 0 otherwise. The K category indicators are thus treated as independent Bernoulli variables, and Equation A8 accordingly divides each term by $P_k(1 - P_k)$ instead of P_k . However, the indicators of one response sum to 1, so the response follows a multinomial distribution with likelihood $\prod_k P_k^{u_k}$, whose information divides each term by P_k only. Because $1 - P_k < 1$, Equation A8 overstates the information and understates the standard errors. For example, for an item with two categories, Equation A8 gives exactly twice the information of their Equation A4 for dichotomous items. This function uses the multinomial information.

The multinomial information was checked by a simulation that matches the setting of this function: abilities drawn from $N(0, 1)$ and treated as known, maximum likelihood estimation, a sample size of 1,000, and 1,000 replications for GRM and GPCM items with three to five categories. The standard

errors from the multinomial information agreed with the standard deviations of the estimates across replications (ratios of 0.96 to 1.08), whereas those from Equation A8 were about 10 to 23 percent smaller (ratios of 0.77 to 0.90). For a dichotomous (2PLM) item, the two forms give the same results.

The matrices are derived under the assumptions that the IRT model fits the data, that the item parameters are estimated by maximum likelihood, and that the ability distribution is known (Li & Lissitz, 2004). Because the abilities are treated as known, the uncertainty in the abilities is ignored, and prior distributions on the item parameters are not taken into account. The resulting standard errors are therefore generally considered lower bounds of the standard errors of an actual calibration (Thissen & Wainer, 1982). They may serve as approximations for evaluating the precision of item parameter estimates when empirical standard errors are not reported.

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. For these items, the slope is not treated as an estimated parameter: it is held at the value in `x`, and only the threshold parameters enter the information matrix. For instance, if the last two of five polytomous items were calibrated using the PCM, specify `pcm.loc = c(4, 5)`. A GRM or GPCM item with two score categories is treated as a 2PLM item, and a PCM item with two score categories as a 1PLM item with a fixed slope.

Value

A named list with the following two components:

- `cov`: A named list of variance-covariance matrices, one per item. Each matrix contains only the estimated parameters of the item, and its row and column names are the parameter columns of `x`: `par.2` (difficulty) for the 1PLM; `par.1` and `par.2` (slope and difficulty) for the 2PLM; `par.1` to `par.3` (slope, difficulty, and guessing) for the 3PLM; `par.1` to `par.K` (slope and $K - 1$ thresholds) for GRM and GPCM items with K score categories; and `par.2` to `par.K` (thresholds only) for PCM items specified in `pcm.loc`. Covariances between parameters of different items are not computed. If the information matrix of an item is singular, a warning is issued and the matrix is filled with NA.
- `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. H., & Lissitz, R. W. (2004). Applications of the analytically derived asymptotic standard errors of item response theory item parameter estimates. *Journal of Educational Measurement*, 41(2), 85-117. doi:10.1111/j.17453984.2004.tb01109.x.
- Thissen, D., & Wainer, H. (1982). Some standard errors in item response theory. *Psychometrika*, 47(4), 397-412. doi:10.1007/BF02293705.

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	<i>Residual-Based DIF Detection Framework Using Categorical Residuals (RDIF-CR)</i>
-------	---

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 vector of IRT model-predicted probabilities 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,
...
)

## S3 method for class 'crdif'
print(x, digits = max(2L, getOption("digits") - 5L), ...)

```

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.), or an object of class <code>est_irt</code> or <code>est_item</code> obtained from <code>est_irt()</code> or <code>est_item()</code> . See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function.
...	Additional arguments passed to <code>est_score()</code> when abilities are estimated internally, for example <code>tol</code> or <code>se</code> . The <code>est_score()</code> arguments <code>max.iter</code> and <code>missing</code> cannot be passed this way, because the function uses arguments with these names for its own purposes.
data	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in x. Responses to an item with K score categories must be integers from 0 to K - 1. Missing responses are coded as NA or, in functions that have a missing argument, as the value given in <code>missing</code> .
score	A numeric vector of examinees' ability estimates (theta values), in the same row order as data. The estimates should be based on the item parameters in x (see Details). If NULL, abilities are estimated internally with <code>est_score()</code> using <code>method</code> , <code>range</code> , <code>norm.prior</code> , <code>nquad</code> , <code>weights</code> , and <code>ncore</code> . When <code>purify = TRUE</code> , abilities are re-estimated internally at every purification iteration, so a supplied score is used only in the initial analysis. A missing value in score excludes the examinee from the analysis, also during purification, with a warning. The <code>est_item</code> method has no score argument; it uses the abilities stored in the object. 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 <code>group = c(0, 1, 0, 1, 1)</code> and '1' indicating the focal group, set <code>focal.name = 1</code> .

<code>item.skip</code>	A numeric vector of item positions (row numbers of x), or a logical vector with one value per item, identifying the items to exclude from the analysis. Skipped items still contribute to the ability estimates, but their results (statistics, p-values, and moments) are set to NA, and they are never flagged or removed during purification. If NULL, all items are analyzed. Default is NULL.
<code>D</code>	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1.
<code>alpha</code>	A numeric value specifying the significance level (α) of the tests. An item is flagged by a statistic when its p-value is less than or equal to alpha. The comparison uses unrounded p-values, although the p-values reported in <code>dif_stat</code> are rounded to four decimal places. Default is 0.05.
<code>missing</code>	A value indicating missing responses in the data set. Default is NA.
<code>purify</code>	Logical. Indicates whether to apply a purification procedure. Default is FALSE.
<code>purify.by</code>	A character string specifying the RDIF-CR statistic used for purification: "crdifrs" for $RDIF_{RS} - CR$, "crdifr" for $RDIF_R - CR$, or "crdifs" for $RDIF_S - CR$. Used only when <code>purify = TRUE</code> . Default is "crdifrs".
<code>max.iter</code>	A positive integer specifying the maximum number of iterations allowed for the purification process. Default is 10. Each iteration removes one item, so <code>max.iter</code> should be at least as large as the number of items expected to be flagged. If the limit is reached while flagged items remain, a warning is issued, the complete element of the purification results is FALSE, and the items flagged in the last iteration are added to the flagged items. This argument is not passed to <code>est_score()</code> .
<code>min.resp</code>	A single nonnegative number, usually a positive integer, specifying the minimum number of item responses that an examinee must have to be used in the analysis. All responses of examinees with fewer than <code>min.resp</code> (but at least one) responses are set to NA before the ability estimation, in the initial analysis and at every purification iteration, also when score is supplied, and these examinees are excluded from the analysis. A warning reports the number of examinees excluded in the initial analysis, and one warning at the end of the purification reports the examinees excluded during it. A large <code>min.resp</code> can leave a group without any ability estimate in a later purification iteration, which stops the purification with a warning. If NULL, no minimum is applied. Default is NULL. See Details for more information.
<code>method</code>	A character string indicating the scoring method used to estimate the abilities when <code>score = NULL</code> and in every purification iteration. Available options are: • "ML": Maximum likelihood estimation • "MLF": Maximum likelihood estimation with fences • "WL": Weighted likelihood estimation (Warm, 1989) • "MAP": Maximum a posteriori estimation (Hambleton et al., 1991) • "EAP": Expected a posteriori estimation (Bock & Mislevy, 1982) The methods "EAP.SUM" and "INV.TCC" of <code>est_score()</code> are not supported, because they do not give an ability estimate for each response pattern. 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 <code>c(-5, 5)</code> .
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 <code>c(0, 1)</code> .
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 <code>gen.weight()</code> . If NULL and method = "EAP", default quadrature values are generated based on the <code>norm.prior</code> and <code>nquad</code> arguments. Ignored if method is "ML", "WL", or "MAP".
ncore	An integer specifying the number of logical CPU cores used by <code>est.score()</code> when abilities are estimated internally. The test statistics themselves are not computed in parallel. Default is 1.
verbose	Logical. If TRUE, the progress of the purification procedure is printed to the console. Default is TRUE.
digits	Not used; kept for consistency with the other print methods.

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 predicts the category probabilities as $(0.1, 0.2, 0.4, 0.25, 0.05)^T$, so that the expected item score is $0(0.1) + 1(0.2) + 2(0.4) + 3(0.25) + 4(0.05) = 1.95$. 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, net DIF is assessed with a single item score residual, the observed item score minus the expected item score. In this example, the residual would be $2 - 1.95 = 0.05$. To detect net DIF, use `rdif()`.

$RDIF_R - CR$ is the quadratic form of the vector of category-level differences in mean raw residuals between the focal and reference groups, centered at its null mean, in a generalized inverse of its null covariance matrix. $RDIF_S - CR$ is the corresponding statistic for the mean squared residuals, and $RDIF_{RS} - CR$ uses the two vectors combined. Under the null hypothesis of no DIF, each statistic asymptotically follows a chi-square distribution whose degrees of freedom equal the rank of the covariance matrix (Moore, 1977). For an item with K score categories (K of 3 or more), the K category residuals of each examinee sum to zero, so the elements of the $RDIF_R - CR$ vector also sum to zero and its covariance matrix has rank $K - 1$. $RDIF_R - CR$ therefore has $K - 1$ degrees of freedom. It is computed after dropping one score category, which gives the same value as the Moore-Penrose generalized inverse of the full covariance matrix, and the same category is dropped from the raw residual part of $RDIF_{RS} - CR$. The squared residuals are not subject to a constraint that holds for all examinees, so $RDIF_S - CR$ has K degrees of freedom, and $RDIF_{RS} - CR$ has $2K - 1$. For a dichotomous item, the two category residuals are the negatives of each other, so only the residual for score category 1 is used, and the three statistics have 1, 1, and 2 degrees of freedom; then $RDIF_R - CR$ and $RDIF_S - CR$ equal the squares of the standardized $RDIF_R$ and $RDIF_S$ statistics of `rdif()`, $RDIF_{RS} - CR$ equals $RDIF_{RS}$, and the p-values and flagged items are identical to those of `rdif()`.

In rare cases, for example when all examinees who responded to an item have the same category probabilities, a covariance matrix has an even lower rank. Then the statistic is computed with the Moore-Penrose generalized inverse and compared with a chi-square distribution whose degrees of freedom equal the rank, and a warning names the item. The columns `df.crdifr`, `df.crdifs`, and `df.crdifrs` of `dif_stat` report the degrees of freedom used.

The purification procedure, the `min.resp` argument, the handling of examinees with NA ability estimates, the coding of group, and the NaN results of an item to which no examinee of one group responded follow `rdif()` (see its Details). The p-value, not the statistic, is used to choose the item to remove because the degrees of freedom of the RDIF-CR statistics differ between items with different numbers of score categories.

The print method of this class has an argument `digits` that is not used; the statistics are printed rounded to three decimal places.

Value

This function returns an object of class "crdif", which is a list with the following five 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 with one row per item and the columns `id`, `crdifr`, `df.crdifr`, `crdifs`, `df.crdifs`, `crdifrs`, `df.crdifrs`, `p.crdifr`, `p.crdifs`, `p.crdifrs`, `n.ref`, `n.foc`, and `n.total`: the item ID, each RDIF-CR statistic followed by the degrees of freedom used, the p-values of the three statistics, the numbers of examinees in the reference and focal groups who responded to the item, and their sum. Statistics and p-values are rounded to four decimal places.
 - moments** A list with the elements `mu.crdifr`, `mu.crdifs`, `mu.crdifrs`, `cov.crdifr`, `cov.crdifs`, and `cov.crdifrs`. Each element is a list named by item ID that holds, for each item, the null mean vector or the null covariance matrix of the category-level statistics. For an item with K score categories (K of 3 or more), the vectors have lengths K , K , and $2K$, and the matrices have

	dimensions K by K, K by K, and 2K by 2K. For a dichotomous item, only score category 1 is used, so the lengths are 1, 1, and 2.
	dif_item A list with the elements <code>crdifr</code> , <code>crdifs</code> , and <code>crdifrs</code> , each giving the positions (rows of <code>x</code>) of the items flagged by the corresponding statistic, or NULL if no item is flagged.
	score A numeric vector of the ability estimates used in the initial analysis: the supplied score, or the internal estimates when <code>score = NULL</code> .
<code>purify</code>	A logical value indicating whether purification was requested (the <code>purify</code> argument).
<code>with_purify</code>	A list with the results of the purification procedure. All elements are NULL when <code>purify = FALSE</code> . The elements are:
	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 with the same columns as <code>no_purify\$dif_stat</code> and an additional column <code>n.iter</code> . For an item removed during purification, the row reports the statistics from the iteration in which the item was removed, and <code>n.iter</code> is that iteration minus 1 (0 for an item removed on the basis of the initial analysis). For the other items, the row reports the statistics from the last iteration, and <code>n.iter</code> is the number of that iteration. If no item is flagged in the initial analysis, this is <code>no_purify\$dif_stat</code> with <code>n.iter = 0</code> .
	moments A list with the same structure as <code>no_purify\$moments</code> . For an item removed during purification, it holds the moments from the iteration in which the item was removed; for the other items, it holds the moments from the last iteration. If no item is flagged in the initial analysis, this is <code>no_purify\$moments</code> .
	dif_item A numeric vector of the positions (rows of <code>x</code>) of the items flagged by the <code>purify.by</code> statistic, sorted in ascending order. It contains the items removed during purification and, if <code>max.iter</code> is reached or the purification stops because a group has no ability estimate left, the items flagged in the last iteration (iteration <code>n.iter</code>). It contains all items if every item is flagged. NULL if no item is flagged in the initial analysis.
	n.iter The number of purification iterations performed (0 if no item is flagged in the initial analysis). The iteration that removes the last item, or after which a group has no ability estimate left, is not counted, because the statistics are not recomputed.
	score A numeric vector of the ability estimates from the last iteration counted in <code>n.iter</code> . NULL if no item is flagged in the initial analysis, because purification is not carried out; the initial estimates are then in <code>no_purify\$score</code> .
	complete A logical value. TRUE if the procedure stopped because no remaining item was flagged (including the case in which no item is flagged in the initial analysis), and FALSE if it stopped because <code>max.iter</code> was reached, because every item was flagged, or because a group had no examinee with an ability estimate left.
<code>alpha</code>	A numeric value indicating the significance level (α) used for the tests.
<code>call</code>	The matched function call.

Methods (by class)

- `crdif(default)`: Default method for a data frame `x` containing item metadata.
- `crdif(est_irt)`: Method for an object of class `est_irt` created by `est_irt()`. The response data and the scaling factor `D` are taken from the object.
- `crdif(est_item)`: Method for an object of class `est_item` created by `est_item()`. The response data, ability estimates, and the scaling factor `D` are taken from the object.

Methods (by generic)

- `print(crdif)`: Print method for an object of class `crdif` (given as `x`). It prints the flagged items and the RDIF-CR statistics of the analyses without and with purification, rounded to three decimal places.

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.
- Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. Newbury Park, CA: Sage.
- 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.
- Moore, D. S. (1977). Generalized inverses, Wald's method, and the construction of chi-squared tests of fit. *Journal of the American Statistical Association*, 72(357), 131-137. doi:10.1080/01621459.1977.10479921.
- 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.
- Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, 54(3), 427-450. doi:10.1007/BF02294627.

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_{RS}-CR (not run: it takes
#       about 15 seconds because the abilities of the examinees are
#       re-estimated at every iteration)
#
#   dif_puri <- 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)
#
#   purify.by = "crdifr" or "crdifs" can be used in the same way.

```

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 <code>cats[j] - 1</code> for each item <code>j</code> (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 <code>cats[j] - 1</code> ; a score within 1e-8 of a whole number is treated as that whole number.
item.id	A character vector of item identifiers, in the same order as the columns of <code>data</code> . If <code>NULL</code> (default), item IDs are generated automatically as <code>paste0("V", 1:ncol(data))</code> , following the convention used elsewhere in <code>irtQ</code> (e.g., shape_df() , est_irt()). Note that this generated ID does not fall back to <code>colnames(data)</code> ; pass <code>item.id</code> explicitly to label items using the column names of <code>data</code> 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 <code>cats</code> convention used elsewhere in <code>irtQ</code> (see, e.g., shape_df()). If <code>NULL</code> (default), the number of categories for each item is inferred from the observed maximum score in <code>data</code> plus one, with a minimum of two (i.e., <code>max(max(data[, j]) + 1, 2)</code>); supply <code>cats</code> 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 item-total correlation (the item score correlated with a total score that includes the item) and the corrected item-total correlation (corre-

	lated with a total score that excludes the item) are always computed and reported as separate columns. This argument only selects which of the two is used for flagging: the corrected value if TRUE, the raw value if FALSE (default).
missing	A value indicating missing responses in data, analogous to the missing argument in <code>est_irt()</code> and <code>score_resp()</code> . 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 <code>crit.p</code> and <code>crit.dis</code> , or when their discrimination is undefined.
crit.p	A numeric vector of length two giving the lower and upper difficulty bounds used for flagging: an item with difficulty below the first value is flagged as too difficult ("difficulty too low"), and an item with difficulty above the second value is flagged as too easy ("difficulty too high"). Default is <code>c(0.10, 0.95)</code> .
crit.dis	A single numeric value giving the minimum acceptable discrimination (item-total correlation); an item whose discrimination is strictly below this value is flagged as poorly discriminating ("discrimination too low"). An item whose discrimination is undefined (a constant item, or a constant reference total score) is flagged "discrimination undefined". Default is <code>0.20</code> .

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. A correlation is NA when the item score or the reference total score is constant. `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 unweighted total score obtained by summing the item scores, which is the score most tests use for reporting and decisions. Standardized alpha is raw alpha computed after standardizing every item with nonzero variance to unit variance. When no item is constant, it equals

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

where $\bar{r}$ is the average pairwise correlation among the items. A constant item cannot be standardized, so it stays in the item count k with zero variance, as in raw alpha, and $\bar{r}$ is the average correlation among the items that vary. When only one item varies, standardized alpha is 0, as is raw alpha; it is NA when no item varies or the standardized total score is constant. 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 by `freq_score()` from the row sums of data after the same missing recoding and listwise deletion, so it covers the same examinees as the other statistics. `freq_score()` can also be called directly on any vector of integer total scores.

Value

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

item	A data frame with one row per item and the columns item (item ID), cats (number of score categories), difficulty, discrimination_raw (raw item-total correlation), discrimination_corrected (corrected item-total correlation), alpha_removed (alpha with the item removed), and, if flag = TRUE, flag (the criteria the item meets, among "difficulty too low", "difficulty too high", "discrimination too low", and "discrimination undefined", separated by "; ", or "" when none). Statistics are rounded to three decimal places.
crit	A list with elements crit.p, crit.dis, and correct, the thresholds and the correlation used for flagging (correct = TRUE for the corrected item-total correlation, FALSE for the raw one).
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_corrected. Both the raw (alpha) and standardized (alpha_std) forms of Cronbach's alpha are always included; standardized alpha standardizes every item with nonzero variance 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 <code>freq_score()</code> (score, freq, pct, cum_pct).
call	The matched call, as used by the <code>print()/summary()</code> 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](https://doi.org/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](https://doi.org/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)
```

Description

Computes, for each item, the number and percentage of examinees choosing each response option (or, for already-scored data, each score category), together with each option's or category's point-biserial correlation with the total score. When an answer key is supplied, the function operates in "selected-response" mode: it treats data as raw, unscored option responses (as used by [score_resp\(\)](#)), also reports omission and double-marking rates, and flags distractors (incorrect options) whose correlation with the total score exceeds `crit.distractor` (0 by default), a common sign of a miskeyed or poorly written item. When no key is supplied, the function operates in "scored-category" mode: it treats data as an already-scored response matrix (0, 1, 2, ..., `cats - 1`) and reports each score category's response proportion and category-total correlation, with no correct option and therefore no flagging.

Usage

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

Arguments

<code>data</code>	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, 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 <code>cats[j] - 1</code> for each item <code>j</code> ; an error is raised when a score is not a whole number between 0 and <code>cats[j] - 1</code> , and a score within 1e-8 of a whole number is treated as that whole number. Either way, examinees are rows and items are columns.
<code>item.id</code>	A character vector of item identifiers, in the same order as the columns of data. If NULL (default), item IDs are generated automatically as <code>paste0("V", 1:ncol(data))</code> , following the convention used elsewhere in <code>irtQ</code> (e.g., shape_df() , est_irt()). Note that this generated ID does not fall back to <code>colnames(data)</code> ; pass <code>item.id</code> explicitly to label items using the column names of data or any other identifier scheme.
<code>key</code>	NULL (default) for scored-category mode. For selected-response mode, either a vector of correct options in item order or a data frame with columns <code>item</code> and <code>key</code> , 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 <code>ctt()</code> . 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 <code>score_resp()</code> in selected-response mode, or as <code>rowSums(data)</code> in scored-category mode (see Details for how total interacts with missing/listwise deletion in scored-category mode). A supplied total must be numeric with no missing values for the examinees analyzed, and it should include the item scores, since <code>pb_corrected</code> subtracts each item's score from it.
correct	Logical. Both the raw item-total correlation (the item score, or in <code>ctt_distr()</code> an option or category indicator, correlated with a total score that includes the item) and the corrected item-total correlation (correlated with a total score that excludes the item) are always computed and reported as separate columns. This argument only selects which of the two is used for flagging: the corrected value if TRUE, the raw value if FALSE (default). It drives the flag column of <code>ctt()</code> and the <code>crit.distractor</code> check of <code>ctt_distr()</code> .
missing	A value indicating missing responses, analogous to the missing argument in <code>est_irt()</code> and <code>score_resp()</code> . Its effect differs by mode (see Details): in selected-response mode, a cell equal to missing after trimming surrounding spaces 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 <code>ctt()</code> . 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 written with digits only (e.g., 4, optionally with a decimal part) 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 with non-Latin numeral characters, 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 in a single call is not supported, since the shared opt cannot represent two different option sets at once; for example, mixing numeric-coded items with letter- or label-coded items stops with an error. Analyze such items in separate calls.

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 in scored-category mode, it must align with the *original* rows of data (its length must equal the original `nrow(data)`); it is subset with the same completeness mask before use.

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 a total score. As with the item-level discrimination statistics computed by `ctt()`, two versions are always computed and returned as separate columns: `pb_raw` uses the total score, and `pb_corrected` uses the total score minus the item's own score. `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 or category was chosen by zero or all examinees (a constant indicator) or the total score is constant.

Value

A list with these elements:

<code>distr</code>	A data frame in long format with one row per item x option (selected-response mode) or item x category (scored-category mode), containing <code>item</code> , <code>option</code> (the option number, option label, or score category; numeric for a numeric-coded or scored-category test form, character and upper-cased for a letter- or general-label-coded test form), <code>is_key</code> (logical; whether this option is the item's correct answer, or NA in scored-category mode), <code>freq</code> (the number of examinees choosing this option or category), <code>pct</code> (<code>freq</code> as a percentage of all examinees analyzed, rounded to two decimal places), <code>pb_raw</code> and <code>pb_corrected</code> (raw and corrected point-biserial correlations, rounded to three decimal places), and <code>flag</code> (in selected-response mode, "distractor more attractive to high scorers" for a distractor whose point-biserial correlation, chosen by correct, exceeds <code>crit.distractor</code> ; "" otherwise).
<code>omit</code>	In selected-response mode, a data frame with one row per item and the columns <code>item</code> , <code>pct_blank</code> , and <code>pct_double</code> (the percentages of blank and double-marked responses). 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. NA values are treated as 0.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 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). When prior distributions are specified for item parameters, the estimates are the modes of the marginal posterior distribution (Bayes modal estimates). The function 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 (item IDs, numbers of score categories, IRT models, and item parameters); see Details for its format. You can create an empty item metadata frame using the function <code>shape_df()</code> . The item parameters in x are used as starting values when <code>use.startval = TRUE</code> and are otherwise ignored, except for the fixed items in FIPC, whose parameters are held at the values in x. If x = NULL, the models and numbers of score categories are taken from <code>model</code> and <code>cats</code> . When <code>fipc = TRUE</code> , x must be supplied. Default is NULL. |
| data | A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in x. Responses to an item with K score categories must be integers from 0 to K - 1. Missing responses are coded as NA or, in functions that have a <code>missing</code> argument, as the value given in <code>missing</code> . |
| D | A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1. |
| model | A character vector specifying the IRT model to fit each item. Available values are: <ul style="list-style-type: none"> • "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" is a general label for dichotomous items; in this function, items labeled "DRM" are fitted with the 3PLM, and a warning

	is issued. If a single model name is provided, it is recycled for all items. This argument is only used when <code>x = NULL</code> and <code>fipc = FALSE</code> . Default is <code>NULL</code> .
<code>cats</code>	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 <code>cats = NULL</code> and all models specified in the <code>model</code> argument are dichotomous ("1PLM", "2PLM", "3PLM", or "DRM"), the function defaults to 2 categories per item. This argument is used only when <code>x = NULL</code> and <code>fipc = FALSE</code> . Default is <code>NULL</code> .
<code>item.id</code>	Character vector of item identifiers. If <code>NULL</code> , the IDs in <code>x</code> are used, or IDs are generated automatically when <code>x = NULL</code> . A provided <code>item.id</code> replaces any IDs in <code>x</code> , with or without FIPC, and must have one value per item. With <code>fipc = TRUE</code> , <code>fix.id</code> then refers to the new IDs. Default is <code>NULL</code> .
<code>fix.a.1pl</code>	Logical. If <code>TRUE</code> , the slope parameters of all 1PLM items are fixed to <code>a.val.1pl</code> ; otherwise, they are constrained to be equal and estimated. Default is <code>FALSE</code> .
<code>fix.a.gpcm</code>	Logical. If <code>TRUE</code> , GPCM items are calibrated as PCM with slopes fixed to <code>a.val.gpcm</code> ; otherwise, each item's slope is estimated. Default is <code>FALSE</code> .
<code>fix.g</code>	Logical. If <code>TRUE</code> , all 3PLM guessing parameters are fixed to <code>g.val</code> ; otherwise, each guessing parameter is estimated. The prior in <code>gprior</code> is then not used. Default is <code>FALSE</code> .
<code>a.val.1pl</code>	Numeric. Value to which the slope parameters of 1PLM items are fixed when <code>fix.a.1pl = TRUE</code> . Default is 1.
<code>a.val.gpcm</code>	Numeric. Value to which the slope parameters of GPCM items are fixed when <code>fix.a.gpcm = TRUE</code> . Default is 1.
<code>g.val</code>	Numeric. Value to which the guessing parameters of 3PLM items are fixed when <code>fix.g = TRUE</code> . Default is 0.2.
<code>use.aprior</code>	Logical. If <code>TRUE</code> , applies a prior distribution to all item discrimination (slope) parameters during calibration. Default is <code>FALSE</code> .
<code>use.bprior</code>	Logical. If <code>TRUE</code> , applies a prior distribution to all item difficulty (or threshold) parameters during calibration. Default is <code>FALSE</code> .
<code>use.gprior</code>	Logical. If <code>TRUE</code> , applies the prior distribution in <code>gprior</code> to the guessing parameters of 3PLM items when they are estimated; the prior is not used when <code>fix.g = TRUE</code> . Default is <code>TRUE</code> .
<code>aprior, bprior, gprior</code>	A list specifying the prior distribution for all item discrimination (slope), difficulty (or threshold), and guessing parameters. The slope prior is placed on D times the slope, that is, the density in <code>aprior</code> is evaluated at $D * a$; with $D = 1$ (the default), this is the slope itself. The priors also enter the information matrix from which the standard errors are computed. Three distributions are supported: Beta, Log-normal, and Normal. The list must have two elements: • <code>dist</code>: A character string, one of "beta", "lnorm", or "norm". • <code>params</code>: A numeric vector of length two giving the distribution's parameters. For details on each parameterization, see <code>stats::dbeta()</code>, <code>stats::dlnorm()</code>, and <code>stats::dnorm()</code>.

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, in addition to NA, which is always treated as missing. Missing responses are omitted from the likelihood, which assumes that they are missing at random. 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, <code>c(49, 6)</code> specifies 49 evenly spaced points from -6 to 6. The points are used to approximate the integral over the latent ability distribution in the E-step of the EM algorithm. Quadrature is ignored when weights is supplied. Default is <code>c(49, 6)</code>.
weights	A two-column matrix or data frame of quadrature points (first column) and weights (second column) that defines the latent ability distribution; it can be created with <code>gen.weight()</code>. The weights should sum to 1; weights that do not sum to 1 are rescaled to sum to 1 with a warning. The number of points may differ from <code>Quadrature[1]</code>. If supplied, <code>Quadrature</code>, <code>group.mean</code>, and <code>group.var</code> are not used. Without FIPC, the mean and variance of this distribution fix the scale: the distribution is used as it is when <code>EmpHist = FALSE</code>, and its shape is updated but rescaled to its mean and variance when <code>EmpHist = TRUE</code>, in which case the quadrature points must be equally spaced, because the histogram is rescaled by interpolation on an equally spaced grid. With FIPC, it is only the initial distribution of the first EM cycle, which is then updated (see Details). If NULL, a normal distribution with mean <code>group.mean</code> and variance <code>group.var</code> at the points given by <code>Quadrature</code> is used. Default is NULL.
group.mean	A numeric value specifying the mean of the normal latent ability distribution when <code>weights = NULL</code> . Without FIPC, the mean is fixed at this value to identify the scale of the item parameters, also when <code>EmpHist = TRUE</code> . With FIPC, it is the mean of the initial distribution, and the mean is then estimated. Default is 0.
group.var	A positive numeric value specifying the variance of the normal latent ability distribution when <code>weights = NULL</code> . Without FIPC, the variance is fixed at this value to identify the scale of the item parameters, also when <code>EmpHist = TRUE</code> . With FIPC, it is the variance of the initial distribution, and the variance is then estimated. Default is 1.
EmpHist	Logical. If TRUE, the latent ability distribution is estimated as an empirical histogram on the quadrature points together with the item parameters (Woods, 2007). Item calibration is conducted relative to the estimated empirical histogram. See Details. Default is FALSE.
use.startval	Logical. If TRUE, the item parameters provided in the item metadata (i.e., the <code>x</code> argument) are used as starting values for the items whose parameters are estimated; <code>x</code> must then be supplied. Otherwise, internally generated starting values are used. Default is FALSE.

Etol	A positive numeric value specifying the convergence criterion of the EM algorithm. The EM cycles stop when the largest absolute change in the estimated item parameters between two consecutive cycles is less than or equal to Etol. With FIPC, only the parameters of the items that are not fixed are compared. The mean and variance of the latent ability distribution estimated with FIPC, and the weights of an empirical histogram (EmpHist = TRUE), are not part of the criterion, so they can be less converged than the item parameters; a smaller Etol gives more precise estimates of them. In particular, an estimated histogram can keep changing very slowly over many cycles (toward a spiky distribution) while the log-likelihood barely changes, so group.par and weights then depend on Etol; with small samples, the item parameters can then also need many cycles, or more than MaxE, to meet Etol. When all items are fixed, the criterion is applied to the largest absolute change in the mean and variance of the latent ability distribution. Etol is not used when fipc.method = "OEM" with items to calibrate. The printed output of the fitted object refers to this criterion as the convergence criterion of the E-step. Default is 1e-4.
MaxE	A positive integer specifying the maximum number of EM cycles. It is set to 1 when fipc.method = "OEM". The printed output of the fitted object refers to the cycles as E-step cycles. Default is 500.
control	A named list of options passed directly to <code>stats::nlminb()</code>, which maximizes the M-step objective of each item. The default is <code>list(eval.max = 500, iter.max = 200, x.tol = 1e-4)</code>, where • <code>eval.max</code> is the maximum number of objective function evaluations, • <code>iter.max</code> is the maximum number of optimizer iterations, and • <code>x.tol</code> is the X-convergence tolerance of <code>stats::nlminb()</code>. A partial list may be supplied; options that are not supplied keep their default values. Other options of <code>stats::nlminb()</code>, such as <code>rel.tol</code> or <code>trace</code>, can also be supplied.
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 <code>fix.loc</code> or <code>fix.id</code>. See Details.
fipc.method	A character string specifying the FIPC method, used only when fipc = TRUE. 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", a single EM cycle is run regardless of MaxE. See Details. Default is "MEM".
fix.loc	A vector of positive integers giving the row positions in <code>x</code> of the items to be fixed when fipc = TRUE. For example, suppose that five items located in the 1st, 2nd, 4th, 7th, and 9th rows of <code>x</code> should be fixed. Then use <code>fix.loc = c(1, 2, 4, 7, 9)</code>. It must not contain duplicated positions. If <code>fix.id</code> is also supplied, <code>fix.loc</code> is ignored with a warning. See Details.

<code>fix.id</code>	A character vector of the IDs of the items to be fixed when <code>fipc = TRUE</code> . The IDs are matched to the first column (<code>id</code>) of <code>x</code> , or to <code>item.id</code> when it is supplied. For example, use <code>fix.id = c("CMC1", "CMC2", "CMC3", "CMC4", "CMC5")</code> to fix the items with these IDs. If <code>fix.id</code> is not <code>NULL</code> , <code>fix.loc</code> is ignored. See Details.
<code>se</code>	Logical. If <code>FALSE</code> , the standard errors and the variance-covariance matrix of the item parameter estimates are not computed; <code>se.est</code> is then filled with <code>NA</code> and covariance is <code>NULL</code> . Default is <code>TRUE</code> .
<code>verbose</code>	Logical. If <code>FALSE</code> , all progress messages, including information about the EM algorithm process, are suppressed. Default is <code>TRUE</code> .

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 score categories of 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. In `est_irt()`, items labeled "DRM" are calibrated with the 3PLM.

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 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 a GPCM item with K score categories has $K - 1$ threshold parameters, because the term for the lowest score category (score 0) is fixed at 0 and is not included in the metadata. 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:

id	cats	model	par.1	par.2	par.3
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:

id	cats	model	par.1	par.2	par.3	par.4	par.5
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` is supplied, `model` and `cats` are ignored. If `x = NULL`, the function uses `model` and `cats`; `cats` may be omitted when all items are dichotomous.

Item parameters are estimated with the EM algorithm of Bock and Aitkin (1981). In the E-step, the posterior distribution of each examinee over the quadrature points is computed from the current item parameters and the current latent distribution, and the expected numbers of examinees and of responses in each score category are obtained at each quadrature point. In the M-step, the expected complete-data log-likelihood of each item, plus the log prior densities when priors are used, is maximized with `stats::nlminb()` using analytic gradients and Hessians; the 1PLM items with a common slope are estimated jointly. The cycles are repeated until the convergence criterion given by `Etol` is met or `MaxE` cycles are completed. Missing responses are omitted from the likelihood. The slopes of dichotomous items are not bounded during estimation, whereas those of GRM and GPCM items are kept positive.

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 item parameters are estimated with the initial latent distribution (given by `group.mean` and `group.var`,

or by weights). The latent distribution returned in `group.par` and `weights` is computed once afterward from the posterior distributions based on the fixed items, and `EmpHist` affects only this returned distribution. When all items are fixed, the function always warns that the convergence criteria are not satisfied.

The second method is "MWU-MEM", which runs multiple EM cycles and updates the weights of the latent ability distribution in every cycle (Kim, 2006; Kim & Kolen, 2016). The first cycle is the same as in NWU-OEM: the posterior distribution of each examinee is computed from the responses to the fixed items only. From the second cycle onward, the posterior distribution is computed from the responses to all items, using the fixed parameters and the current estimates of the other items, and the latent distribution is updated from the expected numbers of examinees at the quadrature points. With `EmpHist = TRUE`, the updated distribution is this empirical histogram, as in Kim (2006); with `EmpHist = FALSE`, it is a normal distribution with the mean and variance of the histogram. This method can be implemented by setting `fipc.method = "MEM"`.

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 latent distribution is an empirical histogram: in each EM cycle, the weight at each quadrature point is updated to the average of the posterior probabilities of the examinees at that point (Woods, 2007). Without FIPC, the updated histogram is then rescaled so that its mean and variance equal `group.mean` and `group.var` (or the mean and variance of weights, if supplied): the quadrature points are standardized and transformed to the target scale, the weights are interpolated back to the original points (and extrapolated at the ends), and the weights are renormalized. With FIPC, the histogram is not rescaled, so its mean and variance are estimated. When `EmpHist = FALSE`, a normal distribution is used. With FIPC, its mean and variance are updated in each cycle to those of the histogram; otherwise, they are fixed at `group.mean` and `group.var`.

A maximum likelihood estimate does not exist for an item with a score category that has no responses, which includes an item whose responses are all the same. The estimates of such an item can be extreme or remain near their starting values, and their standard errors are unreliable (a standard error of 99999 means that it could not be obtained). Before the estimation, one warning lists the IDs of these items; the items fixed in FIPC are not checked. A parameter with a prior distribution, such as the guessing parameter of a 3PLM item with `use.gprior = TRUE`, can still be stable.

When prior distributions are used, the item parameters are Bayes modal estimates, and loglikelihood is the log-likelihood at these estimates without the log prior densities.

The information criteria are $aic = -2 * \text{loglikelihood} + 2 * \text{npar.est}$ and $bic = -2 * \text{loglikelihood} + \text{npar.est} * \log(\text{ncase})$. Here `loglikelihood` is the marginal log-likelihood without the prior densities of the item parameters, and `ncase` is the number of examinees. `npar.est` is the number of freely estimated parameters, which consists of (a) the item parameters that are estimated (a slope shared by 1PLM items is counted once; fixed parameters and the items fixed in FIPC are not counted), (b) the mean and variance of the latent distribution when they are estimated, which is the case with FIPC and `EmpHist = FALSE` (2 values), and (c) the weights of the empirical histogram when `EmpHist = TRUE`. Without FIPC, the histogram is rescaled to a fixed mean and variance (given by `group.mean` and `group.var`, or by weights), so its Q weights, where Q is the number of quadrature points, have $Q - 3$ free values (the sum of the weights and the two moments are fixed). With

FIPC, the histogram is not rescaled, so the weights have $Q - 1$ free values. The counts by type are returned in `npar.detail`.

The print method of this class has an argument `digits` that is not used. It prints the log-likelihood with the number of significant digits given by `getOption("digits")` (7 by default).

Value

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

<code>estimates</code>	A data frame with the item ID, number of score categories, and model of each item, followed by the item parameter estimates and their standard errors in alternating columns (<code>par.1</code> , <code>se.1</code> , <code>par.2</code> , <code>se.2</code> , ...).
<code>par.est</code>	A data frame of item parameter estimates, structured according to the item meta-data format.
<code>se.est</code>	A data frame of standard errors for the item parameter estimates, computed from the inverse of the information matrix. The information matrix is the cross-product (empirical) information matrix of the marginal log-likelihood evaluated at the final estimates (Meilijson, 1989), plus the prior information (the negative second derivatives of the log prior densities) for the parameters that have prior distributions. With FIPC, the latent ability distribution is treated as known in this matrix, so the uncertainty of the estimated distribution is not reflected in these standard errors. Standard errors are NA for parameters that are not estimated (for example, fixed slopes or guessing parameters, and the items fixed in FIPC) and for all parameters when <code>se = FALSE</code> . The standard error of a common slope of 1PLM items appears only in the row of the first 1PLM item. A value of 99999 indicates that the standard error could not be obtained.
<code>pos.par</code>	A data frame indicating, for each estimated item parameter, its row (and column) position in covariance. Parameters that are not estimated are NA, and the position of a common slope of 1PLM items is given only in the row of the first 1PLM item. NULL when FIPC is used with all items fixed.
<code>covariance</code>	The variance-covariance matrix of the item parameter estimates (the inverse of the information matrix described in <code>se.est</code>), without row or column names; the order of the parameters is given by <code>pos.par</code> . NULL when <code>se = FALSE</code> or when no item parameter is estimated.
<code>loglikelihood</code>	The marginal log-likelihood of the response data at the final estimates: the sum over examinees of the logarithm of the probability of each examinee's response pattern, integrated over the latent ability distribution with the quadrature points and weights. With FIPC, the responses to all items, including the fixed items, are used. The log prior densities of the item parameters are not included.
<code>aic</code>	Akaike information criterion, $-2 * \text{loglikelihood} + 2 * \text{npar.est}$.
<code>bic</code>	Bayesian information criterion, $-2 * \text{loglikelihood} + \text{npar.est} * \log(\text{n case})$.
<code>group.par</code>	A data frame with columns <code>mu</code> , <code>sigma2</code> , and <code>sigma</code> (mean, variance, and standard deviation of the latent ability distribution) and rows <code>estimates</code> and <code>se</code> . Without FIPC, the values are fixed at <code>group.mean</code> and <code>group.var</code> (or at the mean and variance of weights, if supplied) and <code>se</code> is NA. When <code>Emphist =</code>

	TRUE, the empirical histogram is rescaled in each cycle to have this mean and variance. Because the rescaling interpolates on the quadrature points, the mean and variance computed from the returned weights can differ slightly from the reported values. With FIPC, the values are the mean and variance of the final latent distribution (these are the mean and variance of the returned weights), and se gives approximate standard errors, $\sigma / \sqrt{N}$ for the mean and $\sigma^2 * \sqrt{2 / (N - 1)}$ for the variance, with the delta method for sigma, where N is the number of examinees; these treat the abilities as observed and are therefore usually too small.
weights	A data frame with columns theta and weight giving the quadrature points and the weights of the final latent ability distribution. Without FIPC and with EmpHist = FALSE, it is the distribution given by group.mean and group.var, or weights as supplied (rescaled to sum to 1 if needed). With fipc.method = "OEM", it is the distribution updated once after the item parameters were estimated.
posterior.dist	A matrix of the posterior probabilities of the quadrature points for each examinee, computed at the final estimates, with one row per examinee and one column per quadrature point; each row sums to 1.
data	A matrix of the examinees' response data, with the missing values coded as NA.
scale.D	The scaling factor used in the IRT model.
ncase	The number of examinees (rows of data).
nitem	The total number of items in the response data.
Etol	The convergence criterion of the EM algorithm.
MaxE	The maximum number of EM cycles (1 when fipc.method = "OEM").
aprior	The prior distribution specified for the slope parameters when use.aprior = TRUE; NULL otherwise.
bprior	The prior distribution specified for the difficulty (or threshold) parameters when use.bprior = TRUE; NULL otherwise.
gprior	The prior distribution specified for the guessing parameters when use.gprior = TRUE; NULL otherwise.
npar.est	The number of freely estimated parameters used for aic and bic: the estimated item parameters, the mean and variance of the latent distribution when they are estimated, and the weights of the empirical histogram when EmpHist = TRUE. See Details.
niter	The number of completed EM cycles.
maxpar.diff	The largest absolute change in the estimated item parameters in the last EM cycle (for FIPC with all items fixed, in the mean and variance of the latent ability distribution). With fipc.method = "OEM", it is the change from the starting values.
EMtime	Time (in seconds) spent on EM cycles.
SEtime	Time (in seconds) spent computing standard errors; NULL when they are not computed.
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	A message indicating whether the information matrix used for the standard errors (see se.est) is positive definite. Printed results label it "Second-order test", but because the cross-product information matrix is positive semidefinite by construction, this check mainly detects a singular (rank-deficient) information matrix, for example due to unidentified parameters; it is not a second-order test of the log-likelihood surface. The message reports a possible local maximum only when the first-order test is also satisfied. When FIPC is used with all items fixed, no information matrix is computed, and the message says so.
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 FIPC method used; NULL when fipc = FALSE.
fix.loc	An integer vector of the row positions of the fixed items in x, also when the items were specified with fix.id; NULL when fipc = FALSE.
call	The matched call.
npar.detail	An integer vector with the elements item (free item parameters), dist (mean and variance of the latent distribution), and weight (weights of the empirical histogram), whose sum is npar.est. It is the last element of the object; when it is absent, summary() shows the total only.

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.
- Kim, S., & Kolen, M. J. (2016). *Multiple group IRT fixed-parameter estimation for maintaining an established ability scale* (CASMA Research Report No. 49). Center for Advanced Studies in Measurement and Assessment, University of Iowa.
- 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.

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 EM algorithm
(mod.2pl.hist <- est_irt(data = LSAT6, D = 1, model = "2PLM", cats = 2,
                        EmphHist = 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)))

# 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 to 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,
  Eto1 = 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,
  Eto1 = 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,
  Eto1 = 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 estimates item parameters by maximum likelihood given ability estimates (Baker & Kim, 2004; Ban et al., 2001; Stocking, 1988). When prior distributions are specified for item parameters (by default, for the guessing parameters of 3PLM items), the estimates are the modes of the posterior distribution (Bayes modal estimates). 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 <code>use.startval = TRUE</code>, the item parameters specified in the metadata will be used as starting values for parameter estimation; otherwise they are ignored. If <code>x = NULL</code>, the models and numbers of score categories are taken from the <code>model</code> and <code>cats</code> arguments. See est_irt() or simdat() for more details about the item metadata. Default is <code>NULL</code>.
data	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in <code>x</code>. Responses to an item with <code>K</code> score categories must be integers from 0 to <code>K - 1</code>. Missing responses are coded as <code>NA</code> or, in functions that have a <code>missing</code> argument, as the value given in <code>missing</code>.
score	A numeric vector of examinees' ability estimates (theta values), which are treated as known. It must have one finite value per row of <code>data</code>, in the same order as the rows.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 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" is a general label for dichotomous items; items labeled "DRM" are fitted with the 3PLM, and a warning is issued. If a single model name is provided, it is recycled for all items. This argument is only used when <code>x = NULL</code>. Default is <code>NULL</code>.
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 <code>cats = NULL</code> and all models specified in the <code>model</code> argument are dichotomous ("1PLM", "2PLM", "3PLM", or "DRM"), the function defaults to 2 categories per item. This argument is used only when <code>x = NULL</code>. Default is <code>NULL</code>.
item.id	Character vector of item identifiers. If <code>NULL</code>, the IDs in <code>x</code> are used, or IDs are generated automatically when <code>x = NULL</code>. A provided <code>item.id</code> replaces any IDs in <code>x</code> and must have one value per item. Default is <code>NULL</code>.

<code>fix.a.1pl</code>	Logical. If TRUE, the slope parameters of all 1PLM items are fixed to <code>a.val.1pl</code> ; otherwise, they are constrained to be equal and estimated. Default is FALSE.
<code>fix.a.gpcm</code>	Logical. If TRUE, GPCM items are calibrated as PCM with slopes fixed to <code>a.val.gpcm</code> ; otherwise, each item's slope is estimated. Default is FALSE.
<code>fix.g</code>	Logical. If TRUE, all 3PLM guessing parameters are fixed to <code>g.val</code> ; otherwise, each guessing parameter is estimated. The prior in <code>gprior</code> is then not used. Default is FALSE.
<code>a.val.1pl</code>	Numeric. Value to which the slope parameters of 1PLM items are fixed when <code>fix.a.1pl = TRUE</code> . Default is 1.
<code>a.val.gpcm</code>	Numeric. Value to which the slope parameters of GPCM items are fixed when <code>fix.a.gpcm = TRUE</code> . Default is 1.
<code>g.val</code>	Numeric. Value to which the guessing parameters of 3PLM items are fixed when <code>fix.g = TRUE</code> . Default is 0.2.
<code>use.aprior</code>	Logical. If TRUE, applies a prior distribution to all item discrimination (slope) parameters during calibration. Default is FALSE.
<code>use.bprior</code>	Logical. If TRUE, applies a prior distribution to all item difficulty (or threshold) parameters during calibration. Default is FALSE.
<code>use.gprior</code>	Logical. If TRUE, applies the prior distribution in <code>gprior</code> to the guessing parameters of 3PLM items when they are estimated; the prior is not used when <code>fix.g = TRUE</code> . Default is TRUE.
<code>aprior, bprior, gprior</code>	A list specifying the prior distribution for all item discrimination (slope), difficulty (or threshold), and guessing parameters. The slope prior is placed on D times the slope, that is, the density in <code>aprior</code> is evaluated at $D * a$; with $D = 1$ (the default), this is the slope itself. The priors also enter the information matrix from which the standard errors are computed. Three distributions are supported: Beta, Log-normal, and Normal. The list must have two elements: • <code>dist</code>: A character string, one of "beta", "lnorm", or "norm". • <code>params</code>: A numeric vector of length two giving the distribution's parameters. For details on each parameterization, see <code>stats::dbeta()</code>, <code>stats::dlnorm()</code>, and <code>stats::dnorm()</code>. Defaults are: • <code>aprior = list(dist = "lnorm", params = c(0.0, 0.5))</code> • <code>bprior = list(dist = "norm", params = c(0.0, 1.0))</code> • <code>gprior = list(dist = "beta", params = c(5, 16))</code> for discrimination, difficulty, and guessing parameters, respectively.
<code>missing</code>	A value indicating missing responses in the data set, in addition to NA, which is always treated as missing. Missing responses are omitted from the likelihood, which assumes that they are missing at random. Default is NA.
<code>use.startval</code>	Logical. If TRUE, the item parameters provided in the item metadata (i.e., the <code>x</code> argument) are used as starting values for the items whose parameters are estimated; <code>x</code> must then be supplied. Otherwise, internally generated starting values are used. Default is FALSE.

control	A named list of control options for <code>stats::nlminb()</code>, which maximizes the objective of each item (see Details). The default is <code>list(eval.max = 500, iter.max = 200, x.tol = 1e-4)</code>, where • <code>eval.max</code> is the maximum number of objective function evaluations, • <code>iter.max</code> is the maximum number of optimizer iterations, and • <code>x.tol</code> is the X-convergence tolerance of <code>stats::nlminb()</code>. A partial list may be supplied; options that are not supplied keep their default values. Other options of <code>stats::nlminb()</code>, such as <code>rel.tol</code> or <code>trace</code>, can also be supplied.
verbose	Logical. If FALSE, all progress messages are suppressed. Default is TRUE.

Details

The parameters of each item are estimated separately by maximizing the log-likelihood of its responses given the abilities in `score` (plus the log prior densities when priors are used), except that the 1PLM items share one slope, estimated jointly with their difficulties, when `fix.a.1pl = FALSE`. Missing responses are omitted. Items without any response are excluded from the estimation with a warning.

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`.

A maximum likelihood estimate does not exist for an item with a score category that has no responses, which includes an item whose responses are all the same. The estimates of such an item can be extreme or remain near their starting values, and their standard errors are unreliable (a standard error of 99999 means that it could not be obtained). After the items without any response are excluded, one warning lists the IDs of the remaining items of this kind. A parameter with a prior distribution, such as the guessing parameter of a 3PLM item with `use.gprior = TRUE`, can still be stable.

When prior distributions are used, the item parameters are Bayes modal estimates, and loglikelihood is the log-likelihood at these estimates without the log prior densities.

The print method of this class has an argument `digits` that is not used. It prints the log-likelihood with the number of significant digits given by `getOption("digits")` (7 by default).

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 meta-data format.

se.est	A data frame of standard errors for the item parameter estimates, computed from the inverse of the observed information matrix (the Hessian of the negative log-likelihood, plus the negative second derivatives of the log prior densities when priors are used). The abilities in score are treated as known, so their estimation error is not reflected in the standard errors.
pos.par	A data frame of the positions of the estimated item parameters in covariance. The rows of the excluded items are NA, and a slope shared by 1PLM items appears only in the row of the first 1PLM item.
covariance	A variance-covariance matrix of the item parameter estimates. Items are estimated separately, so the covariances between the parameters of different items are 0, except among the 1PLM items that share a common slope.
loglikelihood	The log-likelihood of the observed responses given the abilities in score, summed over the items, at the final estimates. The log prior densities of the item parameters are not included. Printed results report -2 times this value as "-2loglikelihood".
group.par	A named numeric vector with the mean (μ) and the standard deviation (σ) of the abilities in score.
data	A data frame of the response data used in the estimation, with missing responses as NA and without the excluded items (see <code>deleted.item</code>).
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 number of items in data as supplied, including the excluded items.
deleted.item	A numeric vector of the positions of the items without any response, which are excluded from the estimation (empty if there are none). Their rows in <code>par.est</code> and <code>se.est</code> are NA.
npar.est	The number of estimated item parameters.
n.response	A numeric vector of the numbers of non-missing responses to the items, with 0 for the excluded items.
TotalTime	Total computation time in seconds.
call	The matched call.

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(mod3, 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). When prior distributions are specified for item parameters, the estimates are the modes of the marginal posterior distribution (Bayes modal estimates). 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 | <p>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 <code>group.name</code> argument.</p> <p>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.</p> <p>When <code>use.startval = TRUE</code>, the item parameters specified in the metadata will be used as starting values for parameter estimation; otherwise they are ignored, except for the fixed items in MG-FIPC, whose parameters are held at the values in x. A common item must have the same model and number of score categories in every group that includes it. If <code>x = NULL</code>, the <code>model</code>, <code>cats</code>, and <code>item.id</code> arguments are used. Note that when <code>fipc = TRUE</code> to implement MG-FIPC, the <code>x</code> argument must be specified and cannot be <code>NULL</code>. Default is <code>NULL</code>.</p> |
| data | <p>A list containing the item response data of all groups to be analyzed. For example, if five groups are analyzed, the list should include five elements, each representing the response data of one group. The elements in the list must be ordered to match the group names specified in the <code>group.name</code> argument. Each element is a matrix or data frame whose rows represent the examinees of the group and whose columns represent the items of the group's form, in the same</p> |

order as in the group's item metadata. Responses to an item with K score categories must be integers from 0 to K - 1, and missing responses are coded as NA or as the value given in missing. Items are matched across groups by their IDs, and the items that are not in a group's form are treated as missing for the examinees of that group.

group.name A character vector indicating the names of the groups, in the order of the elements of data. For example, if five groups are analyzed, use `group.name = c("G1", "G2", "G3", "G4", "G5")`. Group names can be any valid character strings and label the group-level elements of the returned object. If NULL, the names "g1", "g2", and so on are used. 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 1.702 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" is a general label for dichotomous items; items labeled "DRM" are fitted with the 3PLM, and a warning is issued. 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. Items with the same ID in different groups are treated as common items. A provided `item.id` replaces any IDs in `x`, with or without FIPC, and each element must have one value per item of its group. With `fipc = TRUE`, `fix.id` then refers to the new IDs. It is required when `x = NULL`. Default is NULL.

<code>free.group</code>	A numeric or character vector of the positions or names of the groups whose latent ability distributions are freely estimated: their means and variances, or their empirical histograms when <code>EmpHist = TRUE</code>. The distributions of the remaining groups (reference groups, those not specified in this argument) are fixed using the values provided in the <code>group.mean</code> and <code>group.var</code> arguments, or from the <code>weights</code> argument. Each group in <code>free.group</code> must share common items with another group or, with FIPC, include fixed items. For example, suppose that five groups are analyzed with group names "G1", "G2", "G3", "G4", and "G5". To freely estimate the distributions of groups 2 through 5, set <code>free.group = c(2, 3, 4, 5)</code> or <code>free.group = c("G2", "G3", "G4", "G5")</code>. In this case, the first group ("G1") will have a fixed distribution (e.g., a mean of 0 and variance of 1 when <code>group.mean = 0</code>, <code>group.var = 1</code>, and <code>weights = NULL</code>). Without FIPC, at least one group must be a reference group to identify the scale; if <code>free.group = NULL</code>, all groups are reference groups. With FIPC, <code>free.group</code> must be supplied, and it usually includes all groups so that every distribution is estimated on the scale of the fixed items (see Details). Default is <code>NULL</code>.
<code>fix.a.1pl</code>	Logical. If <code>TRUE</code> , the slope parameters of all 1PLM items are fixed to <code>a.val.1pl</code> ; otherwise, they are constrained to be equal and estimated. Default is <code>FALSE</code> .
<code>fix.a.gpcm</code>	Logical. If <code>TRUE</code> , GPCM items are calibrated as PCM with slopes fixed to <code>a.val.gpcm</code> ; otherwise, each item's slope is estimated. Default is <code>FALSE</code> .
<code>fix.g</code>	Logical. If <code>TRUE</code> , all 3PLM guessing parameters are fixed to <code>g.val</code> ; otherwise, each guessing parameter is estimated. The prior in <code>gprior</code> is then not used. Default is <code>FALSE</code> .
<code>a.val.1pl</code>	Numeric. Value to which the slope parameters of 1PLM items are fixed when <code>fix.a.1pl = TRUE</code> . Default is 1.
<code>a.val.gpcm</code>	Numeric. Value to which the slope parameters of GPCM items are fixed when <code>fix.a.gpcm = TRUE</code> . Default is 1.
<code>g.val</code>	Numeric. Value to which the guessing parameters of 3PLM items are fixed when <code>fix.g = TRUE</code> . Default is 0.2.
<code>use.aprior</code>	Logical. If <code>TRUE</code> , applies a prior distribution to all item discrimination (slope) parameters during calibration. Default is <code>FALSE</code> .
<code>use.bprior</code>	Logical. If <code>TRUE</code> , applies a prior distribution to all item difficulty (or threshold) parameters during calibration. Default is <code>FALSE</code> .
<code>use.gprior</code>	Logical. If <code>TRUE</code> , applies the prior distribution in <code>gprior</code> to the guessing parameters of 3PLM items when they are estimated; the prior is not used when <code>fix.g = TRUE</code> . Default is <code>TRUE</code> .
<code>aprior, bprior, gprior</code>	A list specifying the prior distribution for all item discrimination (slope), difficulty (or threshold), and guessing parameters. The slope prior is placed on D times the slope, that is, the density in <code>aprior</code> is evaluated at $D * a$; with $D = 1$ (the default), this is the slope itself. The priors also enter the information matrix from which the standard errors are computed. Three distributions are supported: Beta, Log-normal, and Normal. The list must have two elements: <code>dist</code>: 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.

<code>missing</code>	A value indicating missing responses in the data set, in addition to NA, which is always treated as missing. Missing responses are omitted from the likelihood, which assumes that they are missing at random. Default is NA.
<code>Quadrature</code>	A numeric vector of length two: • first element: number of quadrature points • second element: symmetric bound (absolute value) for those points For example, <code>c(49, 6)</code> specifies 49 evenly spaced points from -6 to 6. The points are used to approximate the integral over the latent ability distribution in the E-step of the EM algorithm. Quadrature is ignored when weights is supplied. Default is <code>c(49, 6)</code>.
<code>weights</code>	A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) that define the initial latent ability distribution of every group. The weights and points can be conveniently generated using the function <code>gen.weight()</code>. The weights should sum to 1; weights that do not sum to 1 are rescaled to sum to 1 with a warning. The number of points may differ from <code>Quadrature[1]</code>. If supplied, <code>Quadrature</code>, <code>group.mean</code>, and <code>group.var</code> are not used. For the groups not specified in <code>free.group</code>, the mean and variance of this distribution fix the scale: the distribution is used as it is when <code>EmpHist = FALSE</code>, and its shape is updated but rescaled to this mean and variance when <code>EmpHist = TRUE</code>, in which case the quadrature points must be equally spaced, because the histogram is rescaled by interpolation on an equally spaced grid. For the groups in <code>free.group</code>, it is only the initial distribution. If NULL, a normal prior density is used instead, based on the information provided in the <code>Quadrature</code>, <code>group.mean</code>, and <code>group.var</code> arguments. Default is NULL.
<code>group.mean</code>	A numeric value specifying the mean of the normal latent ability distribution when <code>weights = NULL</code> . Default is 0. The means of the groups not specified in the <code>free.group</code> argument are fixed at this value to resolve the indeterminacy of the item parameter scale, also when <code>EmpHist = TRUE</code> . For the groups in <code>free.group</code> , it is the mean of the initial distribution.
<code>group.var</code>	A positive numeric value specifying the variance of the normal latent ability distribution when <code>weights = NULL</code> . Default is 1. The variances of the groups not specified in the <code>free.group</code> argument are fixed at this value to resolve the indeterminacy of the item parameter scale, also when <code>EmpHist = TRUE</code> . For the groups in <code>free.group</code> , it is the variance of the initial distribution.

EmpHist	Logical. If TRUE, the latent ability distribution of each group is estimated as an empirical histogram on the quadrature points together with the item parameters, using the approach proposed by Woods (2007). The histograms of the groups not in <code>free.group</code> are rescaled to the fixed mean and variance in every EM cycle, and those of the groups in <code>free.group</code> are not rescaled. If FALSE, the distribution of each group in <code>free.group</code> is a normal distribution whose mean and variance are updated in every EM cycle. Default is FALSE.
use.startval	Logical. If TRUE, the item parameters provided in the item metadata (i.e., the <code>x</code> argument) are used as starting values for the items whose parameters are estimated; <code>x</code> must then be supplied. Otherwise, internally generated starting values are used. Default is FALSE.
Etol	A positive numeric value specifying the convergence criterion of the EM algorithm. The EM cycles stop when the largest absolute change in the estimated item parameters between two consecutive cycles is less than or equal to <code>Etol</code> . With MG-FIPC, only the parameters of the items that are not fixed are compared. The means and variances of the latent ability distributions of the groups in <code>free.group</code> , and the weights of the empirical histograms (<code>EmpHist = TRUE</code>), are not part of the criterion, so they can be less converged than the item parameters; a smaller <code>Etol</code> gives more precise estimates of them. In particular, an estimated histogram can keep changing very slowly over many cycles (toward a spiky distribution) while the log-likelihood barely changes, so <code>group.par</code> and <code>weights</code> then depend on <code>Etol</code> ; with small samples, the item parameters can then also need many cycles, or more than <code>MaxE</code> , to meet <code>Etol</code> . When all items are fixed, the criterion is applied to the largest absolute change in the means and variances of the latent ability distributions of all groups; those of the groups not in <code>free.group</code> stay at their fixed values, apart from small changes due to the rescaling of their histograms when <code>EmpHist = TRUE</code> . <code>Etol</code> is not used when <code>fipc.method = "OEM"</code> with items to calibrate. The printed output of the fitted object refers to this criterion as the convergence criterion of the E-step. Default is $1e-3$, which is larger than the default of <code>est_irt()</code> .
MaxE	A positive integer specifying the maximum number of EM cycles. It is set to 1 when <code>fipc.method = "OEM"</code> . The printed output of the fitted object refers to the cycles as E-step cycles. Default is 500.
control	A named list of options passed directly to <code>stats::nlminb()</code> , which maximizes the M-step objective of each item. The default is <code>list(eval.max = 500, iter.max = 200, x.tol = 1e-4)</code> , where • <code>eval.max</code> is the maximum number of objective function evaluations, • <code>iter.max</code> is the maximum number of optimizer iterations, and • <code>x.tol</code> is the X-convergence tolerance of <code>stats::nlminb()</code>. A partial list may be supplied; options that are not supplied keep their default values. Other options of <code>stats::nlminb()</code> , such as <code>rel.tol</code> or <code>trace</code> , can also be supplied.
fipc	Logical. If TRUE, multiple-group fixed item parameter calibration (MG-FIPC) is applied during item parameter estimation. When <code>fipc = TRUE</code> , the information on which items are fixed must be provided via either <code>fix.loc</code> or <code>fix.id</code> . See Details.

<code>fixpc.method</code>	A character string specifying the FIPC method, used only when <code>fixpc = TRUE</code>. 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 <code>fixpc.method = "OEM"</code>, a single EM cycle is run regardless of <code>MaxE</code>. See Details. Default is "MEM".
<code>fix.loc</code>	A list of positive integer vectors. Each internal vector specifies the positions of the items to be fixed in the item metadata (i.e., <code>x</code>) for each group when MG-FIPC is implemented (i.e., <code>fixpc = TRUE</code>). The internal objects in the list must follow the same order as the group names provided in the <code>group.name</code> argument. Use <code>NULL</code> for a group without fixed items. An item fixed in one group is fixed in every group that includes it. 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 <code>fix.loc = list(c(1, 3, 5), c(2, 3, 4, 7), c(1, 2, 6))</code>. Note that if the <code>fix.id</code> argument is not <code>NULL</code>, the information in <code>fix.loc</code> will be ignored. See Details.
<code>fix.id</code>	A vector of character strings specifying the IDs of items to be fixed when MG-FIPC is implemented (i.e., <code>fixpc = TRUE</code>). It is a single character vector for all groups, not a list. 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 (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 <code>fix.id = c("G1I1", "C1I1", "C1I2", "C2I1", "C2I2", "G3I1")</code>. Note that if the <code>fix.id</code> argument is not <code>NULL</code>, the information provided in <code>fix.loc</code> is ignored. See Details.
<code>se</code>	Logical. If <code>FALSE</code>, the standard errors and the variance-covariance matrix of the item parameter estimates are not computed; <code>se.est</code> is then filled with <code>NA</code> and covariance is <code>NULL</code>. Default is <code>TRUE</code>.
<code>verbose</code>	Logical. If <code>FALSE</code>, all progress messages, including information about the EM algorithm process, are suppressed. Default is <code>TRUE</code>.

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. The `est_mg()` function treats items with the same item ID in different groups as common items and constrains them to have 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 the arguments `x`, `data`, `model`, `cats`, `item.id`, and `fix.loc` accept list objects containing elements for each group to be analyzed, in the order of the group names in `group.name`. In contrast, `fix.id` is a single character vector of the IDs of the fixed items across all groups.

Additionally, `est_mg()` introduces two new arguments: `group.name` and `free.group`. The `group.name` argument assigns a unique identifier to each group.

The `free.group` argument indicates the groups whose latent ability distributions are freely estimated; the distributions of the other groups are fixed using the values provided in the `group.mean` and `group.var` arguments or from the `weights` argument. When no item parameters are fixed (i.e., `fipc = FALSE`), at least one group must have a fixed latent ability distribution (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.

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`.

Two methods are available for MG-FIPC through the `fipc.method` argument, as in `est_irt()`. The method "MEM" runs multiple EM cycles. The posterior distributions of the first EM cycle are computed from the responses to the fixed items only, and those of the later cycles from the responses to all items, as in Kim and Kolen (2016, Equations 13 to 20). The latent ability distribution of each group in `free.group` is updated in every EM cycle without rescaling, so that the item parameters and the distributions are placed on the scale of the fixed items. With `EmpHist = TRUE`, the distribution is the histogram of the expected numbers of examinees at the quadrature points, as in Kim and Kolen (2016); with `EmpHist = FALSE`, it is a normal distribution with the mean and variance of that histogram. The distributions of the groups not in `free.group` remain fixed also in MG-FIPC, whereas Kim and Kolen (2016) estimate the distributions of all groups; to follow their method, include all groups in `free.group`. The method "OEM" (Wainer & Mislevy, 1990) runs a single EM cycle: the posterior distributions are computed from the fixed items and the initial distributions, the parameters of the other items are estimated once, and the returned distributions are updated once afterward.

A maximum likelihood estimate does not exist for an item with a score category that has no responses, which includes an item whose responses are all the same. The estimates of such an item can be extreme or remain near their starting values, and their standard errors are unreliable (a standard error of 99999 means that it could not be obtained). Before the estimation, one warning lists the IDs of these items. Because the parameters of an item are common to all the groups that include it, the responses of an item are pooled over these groups; an item that is constant in one group but varies in another is not flagged. The items fixed in MG-FIPC are not checked. A parameter with a prior distribution, such as the guessing parameter of a 3PLM item with `use.gprior = TRUE`, can still be stable.

When prior distributions are used, the item parameters are Bayes modal estimates, and loglikelihood is the log-likelihood at these estimates without the log prior densities.

The information criteria are computed as $aic = -2 * \text{loglikelihood}\$overall + 2 * \text{npar.est}$ and $bic = -2 * \text{loglikelihood}\$overall + \text{npar.est} * \log(\text{ncase}\$overall)$, where `loglikelihood$overall` is the overall marginal log-likelihood without the prior densities of the item parameters and `ncase$overall` is the overall number of examinees. `npar.est` is the number of freely estimated parameters, which consists of (a) the item parameters that are estimated across all unique items (a slope shared by 1PLM items is counted once; fixed parameters and the items fixed in FIPC are not counted), (b) the mean and variance of each group in `free.group` when `EmpHist = FALSE` (2 values per group), and (c) the weights of the empirical histograms when `EmpHist = TRUE`. The histogram of a group in `free.group` is not rescaled, so its Q weights, where Q is the number of quadrature points, have Q - 1 free values (the sum of the weights is fixed). The histogram of any other group is rescaled to a fixed mean and variance, so its weights have Q - 3 free values. The counts by type are returned in `npar.detail`.

The print method of this class has an argument `digits` that is not used. It prints the log-likelihood with the number of significant digits given by `getOption("digits")` (7 by default).

Value

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

<code>estimates</code>	A list containing two internal elements: <code>overall</code> and <code>group</code> . The <code>overall</code> 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 <code>group</code> element is a list of group-specific data frames, each containing item parameter estimates and standard errors for that particular group.
<code>par.est</code>	A list with the same structure as <code>estimates</code> , containing only the item parameter estimates (excluding standard errors), formatted according to the item metadata structure.
<code>se.est</code>	A list with the same structure as <code>estimates</code> , but containing only the standard errors of the item parameter estimates. Note that the standard errors are computed from the inverse of the information matrix, which is the cross-product (empirical) information matrix of the marginal log-likelihood evaluated at the final estimates (Meilijson, 1989), plus the prior information (the negative second derivatives of the log prior densities) for the parameters that have prior distributions. The latent ability distributions of the groups are treated as known in this matrix, so the uncertainty of the estimated distributions is not reflected in these standard errors. Parameters that are not estimated, including those of the items fixed in FIPC, have NA standard errors, and all are NA when <code>se = FALSE</code> .
<code>pos.par</code>	A data frame of the positions of the estimated item parameters in covariance, for the unique items across all groups (the <code>overall</code> element of <code>estimates</code>). Parameters that are not estimated, including those of the items fixed in FIPC, are NA. It is NULL when all items are fixed in FIPC.
<code>covariance</code>	The variance-covariance matrix of the estimated item parameters of the unique items across all groups, in the order given by <code>pos.par</code> . It is NULL when <code>se = FALSE</code> or when all items are fixed in FIPC.
<code>loglikelihood</code>	A list with two elements. <code>overall</code> is the marginal log-likelihood of the response data at the final estimates: the sum over all examinees of the logarithm of the probability of each examinee's response pattern, integrated over the latent ability distribution of the examinee's group. <code>group</code> is a named list of the same sums within the groups, which add up to <code>overall</code> . The log prior densities of the item parameters are not included.
<code>aic</code>	Akaike information criterion, $-2 * \text{loglikelihood}\$overall + 2 * \text{npar.est}$, calculated from the overall log-likelihood.
<code>bic</code>	Bayesian information criterion, $-2 * \text{loglikelihood}\$overall + \text{npar.est} * \log(\text{ncase}\$overall)$, calculated from the overall log-likelihood and the overall number of examinees.
<code>group.par</code>	A named list with one data frame per group, with rows <code>estimates</code> and <code>se</code> and columns <code>mu</code> , <code>sigma2</code> , and <code>sigma</code> (the mean, variance, and standard deviation of the latent ability distribution). For a group not in <code>free.group</code> , the estimates are the fixed values given by <code>group.mean</code> and <code>group.var</code> (or by weights), and the standard errors are NA; with <code>EmpHist = TRUE</code> , the histogram of such a group is rescaled in each cycle to have this mean and variance, but because the rescaling

	interpolates on the quadrature points, the mean and variance computed from the returned weights can differ slightly from the reported values. For a group in <code>free.group</code> , the estimates are the mean and variance of the final distribution (the same as those computed from the returned weights), and the standard errors are approximations that treat the abilities as observed and are therefore usually too small: $\sigma / \sqrt{N}$, $\sigma^2 * \sqrt{2 / (N - 1)}$, and $\text{se}(\sigma^2) / (2 * \sigma)$, where N is the number of examinees in the group.
<code>weights</code>	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.
<code>posterior.dist</code>	A matrix of the posterior probabilities of the quadrature points for each examinee, computed with the final estimates and the latent distribution of the examinee's group. Rows represent the examinees of all groups, stacked in the order of <code>group.name</code> , and columns represent quadrature points; each row sums to 1.
<code>data</code>	A list with two elements. <code>overall</code> is a matrix of the responses of all examinees to all unique items, with NA for the items not in a group's form and for missing responses. <code>group</code> is the list of response data sets named by <code>group.name</code> , in which the missing responses are also recoded as NA.
<code>scale.D</code>	The scaling factor used in the IRT model.
<code>ncase</code>	A list with two elements: <code>overall</code> , the total number of examinees, and <code>group</code> , a named vector of the numbers of examinees in the groups.
<code>nitem</code>	A list with two elements: <code>overall</code> , the number of unique items across all groups, and <code>group</code> , a named vector of the numbers of items in the groups.
<code>Etol</code>	The convergence criterion of the EM algorithm; see <code>Etol</code> in Arguments.
<code>MaxE</code>	The maximum number of EM cycles (1 when <code>fipc.method = "OEM"</code>).
<code>aprior</code>	A list describing the prior distribution used for discrimination parameters. It is NULL when <code>use.aprior = FALSE</code> .
<code>bprior</code>	A list describing the prior distribution used for difficulty parameters. It is NULL when <code>use.bprior = FALSE</code> .
<code>gprior</code>	A list describing the prior distribution used for guessing parameters. It is NULL when <code>use.gprior = FALSE</code> .
<code>npar.est</code>	The number of freely estimated parameters used for <code>aic</code> and <code>bic</code> : the estimated item parameters of all unique items, the means and variances of the free groups when they have normal distributions, and the weights of the empirical histograms when <code>Emphist = TRUE</code> . See Details.
<code>niter</code>	The number of completed EM cycles.
<code>maxpar.diff</code>	The largest absolute change in the estimated item parameters in the last EM cycle (for MG-FIPC with all items fixed, in the means and variances of the latent ability distributions of all groups; see <code>Etol</code>). With <code>fipc.method = "OEM"</code> , it is the change from the starting values.
<code>EMtime</code>	Time (in seconds) spent on EM cycles.
<code>SEtime</code>	Time (in seconds) spent computing standard errors; NULL when they are not computed.

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 Eto1. When fipc.method = "OEM" with new items to calibrate, the single EM cycle is not judged by Eto1 and only the M-step check applies. For FIPC with all items fixed, see Eto1.
test.2	A message indicating whether the information matrix used for the standard errors (see se.est) is positive definite. Printed results label it "Second-order test", but because the cross-product information matrix is positive semidefinite by construction, this check mainly detects a singular (rank-deficient) information matrix, for example due to unidentified parameters; it is not a second-order test of the log-likelihood surface. The message reports a possible local maximum only when the first-order test is also satisfied. When FIPC is used with all items fixed, no information matrix is computed, and the message says so.
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, or NULL when fipc = FALSE.
fix.loc	A list with two elements: overall, the positions of the fixed items among the unique items across all groups (the rows of par.est\$overall), and group, a named list of the positions of the fixed items in each group's form. Both are NULL when fipc = FALSE.
call	The matched call.
npar.detail	An integer vector with the elements item (free item parameters), dist (means and variances of the latent distributions of all groups), and weight (weights of the empirical histograms of all groups), whose sum is npar.est. It is the last element of the object; when it is absent, summary() shows the total only.

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* (CASMA Research Report No. 49). Center for Advanced Studies in Measurement and Assessment, University of Iowa.

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.

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

`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,
  ...
)

## S3 method for class 'est_item'
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 <code>est_irt</code> or <code>est_item</code> obtained from <code>est_irt()</code> or <code>est_item()</code>. See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function.
...	Additional arguments passed to <code>parallel::makeCluster()</code> when <code>ncore > 1</code> .
data	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in <code>x</code>. Responses to an item with <code>K</code> score categories must be integers from 0 to <code>K - 1</code>. Missing responses are coded as <code>NA</code> or, in functions that have a <code>missing</code> argument, as the value given in <code>missing</code>.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 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 for the "ML", "MLF", "WL", and "MAP" methods. Estimates outside these bounds are set to the nearest bound. The bounds also define the search grid for <code>stval.opt = 1</code> and the default fence locations for "MLF". Default is <code>c(-5, 5)</code>.

norm.prior	A numeric vector of length two specifying the mean and standard deviation of the normal prior distribution. For "MAP", it defines the prior. For "EAP" and "EAP.SUM", it is used to generate the Gaussian quadrature points and weights when weights = NULL. Ignored for "ML", "MLF", "WL", and "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" and weights = NULL. 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 <code>gen.weight()</code>. 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 slope (a) parameter of the two fence items used in MLF. See Details below. Default is 3.0.
fence.b	A numeric vector of length two specifying the difficulty (b) parameters of the lower and upper fence items used in MLF. If NULL, the values in range are used. Default is NULL.
tol	A positive numeric value specifying the convergence tolerance. For "ML", "MLF", "WL", and "MAP", the Fisher scoring iterations stop when the absolute update is less than tol, and tol is also the tolerance of the root search used when the iterations do not converge. For "INV.TCC", the bisection search stops when the search interval is no wider than tol. Default is $1e-4$.
max.iter	A positive integer specifying the maximum number of Fisher scoring iterations for "ML", "MLF", "WL", and "MAP". Default is 100. See Details for the case where the iterations do not converge.
se	Logical. If TRUE, standard errors of ability estimates are computed. If FALSE, the standard errors are returned as NA. For "EAP.SUM" and "INV.TCC", standard errors are always computed. 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: Grid 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", ability estimates are assigned to sum scores that cannot be mapped through the TCC (sum scores less than or equal to the sum of the guessing parameters, and the maximum possible sum score). Default is TRUE. See Details below.
range.tcc	A numeric vector of length two giving the ability estimates assigned to the lowest and the maximum possible sum scores when method = "INV.TCC" and intpol = TRUE. 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 used for parallel scoring with "ML", "MLF", "WL", "MAP", and "EAP". Default is 1. See Details below.

Details

For "MAP", the prior is the normal distribution given by `norm.prior`.

Each response must be an integer from 0 to the number of categories of the item minus 1, and data must have one column per item in `x`. Missing responses must be coded as NA or declared with the `missing` argument. For "ML", "MLF", "WL", "MAP", and "EAP", missing responses are excluded from the likelihood, and examinees with all responses missing receive NA with a warning. For "EAP.SUM" and "INV.TCC", missing responses are recoded as 0 with a warning.

In maximum likelihood estimation with fences (MLF; Han, 2016), two imaginary 2PL items are added to every response pattern: a lower fence item answered correctly and an upper fence item answered incorrectly. The lower fence should have a b -parameter below, and the upper fence a b -parameter above, all item difficulties in the test. Both should have steep slopes (large a -parameters). If `fence.b = NULL`, the fences are placed at the bounds given in `range`. See Han (2016) for details.

For "INV.TCC", the ability estimate for a sum score X is the root of $TCC(\theta) = X$, found by the bisection method (Howard, 2017). No root exists for the maximum possible sum score or for sum scores less than or equal to the sum of the guessing parameters, which is only the score 0 when there are no guessing parameters. With `intpol = TRUE`, these scores receive estimates as follows. Let θ_{min} and θ_{max} be the first and second values of `range.tcc`, and let θ_X be the estimate for the smallest sum score X that is greater than the sum of the guessing parameters. Scores below X are mapped onto the line through $(x = \theta_{min}, y = 0)$ and $(x = \theta_X, y = X)$, and the maximum possible sum score receives θ_{max} . If θ_{min} is above θ_X or θ_{max} is below the largest root, a warning is issued and no values are assigned. With `intpol = FALSE`, these scores receive NA.

For "INV.TCC", the standard error for a sum score is the standard deviation of the inverse TCC estimates over the conditional sum score distribution at the estimate for that score, computed with the Lord-Wingersky recursion (Lim et al., 2021; see `lwrc()`). When some sum scores have no estimate (`intpol = FALSE`, or the warning described above), the distribution is restricted to the sum scores with an estimate and rescaled to sum to one. The implementation is based on a modified version of `SNSequate::irt.eq.tse()` 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), the log-likelihood (log-posterior for "MAP") is evaluated on a grid from `range[1]` to `range[2]` in steps of 0.1. The grid point at the highest local maximum is the starting value. If there is no local maximum on the grid, the starting value is 0.
- When `stval.opt = 2`, the starting value is the log-odds of the observed sum score `obs.score` relative to the maximum possible score `max.score`, both computed over the non-missing items: $\log(\text{obs.score} / (\text{max.score} - \text{obs.score}))$.
 - If `obs.score = 0`, the starting value is $\log(1 / \text{max.score})$.
 - If `obs.score = max.score`, the starting value is $\log(\text{max.score})$.
- When `stval.opt = 3`, the starting value is fixed at 0.

For the ML, MLF, WL, and MAP scoring methods, the estimate is found by Fisher scoring. This is a Newton-type iteration that divides the first derivative of the log-likelihood (of the log-posterior for "MAP", or Warm's weighted score function for "WL") by the expected Fisher information instead of the observed second derivative, and a single update is capped at 1 in absolute value. The iterations stop when the absolute update is less than `tol`. If they have not converged after `max.iter` iterations (for example, when the iterations alternate between two values for a low-scoring examinee on a test with 3PLM items), the estimate is found as follows. Among the values visited during the iterations, the function takes the closest pair in which the score function (the derivative above) is positive at the lower value and negative at the upper value, so that the pair encloses a local maximum, and finds the root of the score function between them with `stats::uniroot()` using the tolerance `tol`. If there is no such pair, the last value of the iterations is used. This search is not carried out when the iterations converge. In all cases, the estimate is truncated to range, and the standard error is computed at the final estimate.

For "ML", "MLF", "WL", "MAP", and "EAP", examinees can be scored in parallel by setting `ncore` greater than 1. A warning is issued when `ncore > 1` and there are fewer than 5,000 examinees, because the parallel overhead then exceeds the computation time.

For "ML", "MLF", "WL", and "MAP", the standard error is the inverse square root of the expected Fisher information at the estimate. The information includes the two fence items for "MLF" and the prior information $1/\sigma^2$ for "MAP". For "EAP" and "EAP.SUM", the standard error is the posterior standard deviation. For "EAP.SUM", the posterior for each sum score is computed with the Lord-Wingersky recursion (see `lwrc()`).

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

For `method = "ML", "MLF", "WL", "MAP", or "EAP"`, a data frame with one row per examinee and two columns:

- `est.theta`: Ability estimates.
- `se.theta`: Standard errors of the ability estimates (NA when `se = FALSE`).

For "ML", "MLF", "WL", and "MAP", the standard error is set to 99.9999 when the estimate equals a bound of range. Examinees with all responses missing receive NA in both columns.

For `method = "EAP.SUM" or "INV.TCC"`, a list with two data frames:

- `est.par`: One row per examinee with the columns `sum.score`, `est.theta`, and `se.theta`.
- `score.table`: One row per possible sum score, from 0 to the maximum possible score, with the same three columns.

For "INV.TCC", sum scores without an estimate (see **Details**) have NA in `est.theta` and `se.theta`.

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)`: Method for an object of class `est_irt`. The item parameter estimates, response data, and scaling constant `D` are taken from `x`, so supplying data or `D` stops with an error.

- `est_score(est_item)`: Method for an object of class `est_item`. The item parameter estimates, response data, and scaling constant D are taken from `x`, so supplying data or D stops with an error.

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). SNSequat: Standard and nonstandard statistical models and methods for test equating. *Journal of Statistical Software*, 59(7), 1-30. doi:10.18637/jss.v059.i07.
- 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. doi:10.1111/j.17453992.2010.00179.x.
- 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(Code Snippet 1), 1-19. doi:10.18637/jss.v076.c01.
- Stocking, M. L. (1996). An alternative method for scoring adaptive tests. *Journal of Educational and Behavioral Statistics*, 21(4), 365-389. doi:10.3102/10769986021004365.
- 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),
  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`), which avoids the path reversals that maximum Fisher information (MFI) routing can produce (see [Details](#)).

Usage

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

## S3 method for class 'find_cut'
print(x, digits = 4L, ...)
```

Arguments

<code>x</code>	A data frame of item metadata in the standard irtQ format (columns: <code>id</code> , <code>cats</code> , <code>model</code> , <code>par.1</code> , <code>par.2</code> , ...). See shape_df for details.
<code>module</code>	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 <code>module</code> argument used in run_mst .
<code>route_map</code>	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 <code>route_map</code> argument used in run_mst .
<code>D</code>	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 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 <code>c(-6, 6)</code> .
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. It must be at least 100. 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 <code>ref_theta</code> is selected. Default is 0.
digits	The number of decimal places used for the mean item locations, the cut scores, and the crossing points. Default is 4. The selected cut score of each module pair is always printed with four decimal places.
...	Additional arguments, currently not used.

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, which is the opposite of what the MST is designed to do. This 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 modules that are adjacent in module index (which should follow the order of difficulty; see *Module index vs. difficulty order*), the function finds the theta value at which the higher-index module's TIF first overtakes the lower-index module's TIF *from below* as theta increases. This crossing point is the natural boundary between the two modules: below it, the lower-index (easier) module is more informative; above it, the higher-index (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 $\text{TIF}_{\text{right}}(\theta) - \text{TIF}_{\text{left}}(\theta)$, where the right module is the higher-index (intended harder) module of the pair, 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, which is 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 $TIF_{right}(\theta) - TIF_{left}(\theta)$ 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 cut scores of each stage are returned in ascending order. When the index order and the difficulty order agree, the k -th cut score is the crossing between the k -th and the $(k+1)$ -th module of the stage, which is how `run_mst` and `reval_mst` use it.

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. The elements are named "stage.2", "stage.3", etc., after the stage being entered. Stages with only one module have an empty numeric vector (`numeric(0)`).

`details` A list with one element per stage from stage 2 (named "stage.2", "stage.3", etc.). Each element contains `modules` (module indices), `mean_locs` (mean item locations, named "mod2", ...), `sorted_order` (modules in ascending order of mean location), and `pairs`, a list with one element per adjacent pair (named, e.g., "mod2_vs_mod3") holding `left_module`, `right_module`, `proper_crossings`, `anomalous_crossings`, and `selected_cut`. For a stage with a single module, `modules` holds that module, `mean_locs`, `sorted_order`, and `pairs` are NULL, and a message is added.

`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 visualize TIF curves and crossing points.

Methods (by generic)

- `print(find_cut)`: Print method for an object of class `find_cut` (given as `x`). It prints, for each stage, the mean item locations of the modules, the proper and anomalous crossing points, and the selected cut score of each pair of adjacent modules.

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)

## Visualize 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 that can be passed to 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 agree up to numerical precision
simMST$cut_score
all.equal(cut_result$cut_score, 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.

set.seed(1)
theta_true <- rnorm(500)
result <- 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,  
verbose = FALSE  
)
```

freq_score	<i>Frequency Distribution Table for Total Scores</i>
------------	--

Description

Computes a frequency distribution table for a vector of total (raw) scores: the frequency, percentage, and cumulative percentage of each score value, as commonly reported with classical test theory (CTT) item analysis results. `ctt()` calls this function to build the total-score frequency distribution in its output, and it can also be used on its own for any vector of integer total scores.

Usage

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

Arguments

- | | |
|---------|---|
| score | A numeric vector of integer total (raw) scores, one value per examinee. A factor is converted through its labels (for example, the level "12" becomes the score 12), not through its level codes. |
| missing | A value indicating missing scores in score, analogous to the missing argument in <code>est_irt()</code> and <code>score_resp()</code> . 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 rounding error does not accumulate and the last cumulative percentage is 100.

This function requires integer scores, as is standard for a raw total score. It does not bin or group values, and it stops with an error when a score is not a whole number. A score within 1e-8 of a whole number, such as the result of a floating-point sum, is counted as that whole number.

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:

<code>score</code>	the score value.
<code>freq</code>	the number of examinees with that score.
<code>pct</code>	the percentage of examinees with that score.
<code>cum_pct</code>	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

<code>n</code>	An integer specifying the number of theta (node) values for which weights are to be generated. Default is 41.
<code>dist</code>	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. Default is "norm". • If <code>dist = "norm"</code>, either <code>n</code> or <code>theta</code> must be provided. • If <code>dist = "unif"</code>, only <code>n</code> is used; <code>theta</code> must not be supplied. • If <code>dist = "emp"</code>, <code>theta</code> is required.

mu, sigma	Mean and standard deviation of the normal distribution (used when <code>dist = "norm"</code>). Defaults are 0 and 1.
l, u	Lower and upper bounds of the uniform distribution (used when <code>dist = "unif"</code>). Defaults are -4 and 4.
theta	A numeric vector of theta (node) values for which weights are generated. Used when <code>dist = "norm"</code> or <code>dist = "emp"</code> .

Details

If theta is not specified, n 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

<code>x</code>	An object of class <code>est_irt</code> , <code>est_mg</code> , or <code>est_item</code> as returned by <code>est_irt()</code> , <code>est_mg()</code> , or <code>est_item()</code> , respectively.
<code>...</code>	Additional arguments passed to or from other methods.
<code>what</code>	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, computed from the inverse of the information matrix: the cross-product (empirical) information matrix of the marginal log-likelihood evaluated at the final estimates (Meilijson, 1989), plus the prior information (the negative second derivatives of the log prior densities) for the parameters that have prior distributions.
- 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 marginal log-likelihood of the response data at the final estimates: the sum over examinees of the logarithm of the probability of each examinee's response pattern, integrated over the latent ability distribution. The log prior densities of the item parameters are not included.
- 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 the posterior probabilities of the quadrature points for each examinee. Rows represent examinees, and columns represent quadrature points; each row sums to 1.
- data** A matrix of the examinee response dataset used in estimation.
- scale.D** The scaling constant (usually 1 or 1.702) used in the IRT model.
- ncase** The number of examinees.
- nitem** The number of items included in the dataset.
- Etol** The convergence criterion of the EM algorithm: the largest absolute change in the estimated item parameters between consecutive cycles. For FIPC with all items fixed, it applies to the change in the mean and variance of the latent ability distribution; see [est_irt\(\)](#).
- MaxE** The maximum number of EM cycles allowed during estimation (1 when `fipc.method = "OEM"`).
- 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 number of freely estimated parameters used for `aic` and `bic`; see [est_irt\(\)](#).
- niter** The number of EM cycles completed.
- maxpar.diff** The largest absolute change in the estimated item parameters at the last EM cycle (for FIPC with all items fixed, in the mean and variance of the latent ability distribution).
- 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 A message indicating whether the information matrix used for the standard errors was positive definite. Printed results label it "Second-order test", but because the cross-product information matrix is positive semidefinite by construction, this check mainly detects a singular (rank-deficient) information matrix; it is not a second-order test of the log-likelihood surface. When FIPC is used with all items fixed, no information matrix is computed, and the message says so.

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 of the positions of the fixed items in FIPC, or NULL without FIPC.

npar.detail The numbers of free parameters by type; see [est_irt\(\)](#).

call The matched call.

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 from the inverse of the information matrix: the cross-product (empirical) information matrix of the marginal log-likelihood evaluated at the final estimates (Meilijson, 1989), plus the prior information (the negative second derivatives of the log prior densities) for the parameters that have prior distributions.

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. It is NULL when all items are fixed in FIPC.

covariance A variance-covariance matrix for the item parameter estimates based on the combined data from all groups. It is NULL when `se = FALSE` or when all items are fixed in FIPC.

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

- `overall`: The marginal log-likelihood of the response data of all groups at the final estimates (summed over examinees, without the log prior densities).
- `group`: The same sums within the groups.

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 data frames with the mean, variance, and standard deviation of the latent ability distribution and their standard errors; see [est_mg\(\)](#).

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 the posterior probabilities of the quadrature points for each examinee. Rows represent the examinees of all groups in the order of `group.name`, and columns represent quadrature points; each row sums to 1.
- 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 1.702).
- ncase** A list with `overall` and `group` components indicating the numbers of examinees.
- nitem** A list with `overall` and `group` components indicating the number of items in the respective response sets.
- Etol** The convergence criterion of the EM algorithm: the largest absolute change in the estimated item parameters between consecutive cycles. For FIPC with all items fixed, it applies to the change in the means and variances of the latent ability distributions of all groups; see `est_mg()`.
- MaxE** The maximum number of EM cycles allowed during estimation (1 when `fipc.method = "OEM"`).
- 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** The number of freely estimated parameters used for `aic` and `bic`; see `est_mg()`.
- niter** Number of EM cycles completed.
- maxpar.diff** The largest absolute change in the estimated item parameters at the last EM cycle (for FIPC with all items fixed, in the means and variances of the latent ability distributions of all groups; see `Etol` in `est_mg()`).
- 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** A message indicating whether the information matrix used for the standard errors was positive definite. Printed results label it "Second-order test", but because the cross-product information matrix is positive semidefinite by construction, this check mainly detects a singular (rank-deficient) information matrix; it is not a second-order test of the log-likelihood surface. When FIPC is used with all items fixed, no information matrix is computed, and the message says so.
- 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.
- npar.detail** The numbers of free parameters by type; see `est_mg()`.
- call** The matched call.

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 from the inverse of the observed information matrix (the Hessian of the negative log-likelihood, plus the negative second derivatives of the log prior densities when priors are used). The abilities in score are treated as known, so their estimation error is not reflected in the standard errors.

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 log-likelihood of the observed responses given the abilities in score, summed over the items, at the final estimates, without the log prior densities of the item parameters.

group.par A named numeric vector with the mean (μ) and the standard deviation (σ) of the abilities in score.

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 1.702) used in the IRT model.

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

nitem The number of items in the response data as supplied, including the excluded items.

deleted.item The positions of the items that contained no response data and were excluded from estimation.

npar.est The total number of estimated item parameters.

n.response A numeric vector of the numbers of responses to the items (0 for the excluded items).

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

call The matched call.

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 designed mainly to detect 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(-5, 5),
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(-5, 5),
  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(-5, 5),
  norm.prior = c(0, 1),
```

```

    nquad = 41,
    weights = NULL,
    ncore = 1,
    verbose = TRUE,
    ...
)

## S3 method for class 'grdif'
print(x, digits = max(2L, getOption("digits") - 5L), ...)

```

Arguments

<code>x</code>	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.), or an object of class <code>est_irt</code> or <code>est_item</code> obtained from <code>est_irt()</code> or <code>est_item()</code>. All items must be dichotomous ("1PLM", "2PLM", "3PLM", or "DRM"); the function stops when any item is polytomous. See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function.
<code>...</code>	Additional arguments passed to <code>est_score()</code> when abilities are estimated internally, for example <code>tol</code> or <code>se</code>. The <code>est_score()</code> arguments <code>max.iter</code> and <code>missing</code> cannot be passed this way, because the function uses arguments with these names for its own purposes.
<code>data</code>	A matrix of examinees' scored responses (0 or 1) to the dichotomous items specified in the <code>x</code> argument. Rows represent examinees and columns represent items.
<code>score</code>	A numeric vector of examinees' ability estimates (theta values), in the same row order as <code>data</code>. The estimates should be based on the item parameters in <code>x</code> (see Details). If <code>NULL</code>, abilities are estimated internally with <code>est_score()</code> using <code>method</code>, <code>range</code>, <code>norm.prior</code>, <code>nquad</code>, <code>weights</code>, and <code>ncore</code>. When <code>purify = TRUE</code>, abilities are re-estimated internally at every purification iteration, so a supplied <code>score</code> is used only in the initial analysis. A missing value in <code>score</code> excludes the examinee from the analysis, also during purification, with a warning. The <code>est_item</code> method has no <code>score</code> argument; it uses the abilities stored in the object. Default is <code>NULL</code>.
<code>group</code>	A numeric or character vector of examinees' group membership, with one element per row of <code>data</code>. It should contain the value of the reference group and the values in <code>focal.name</code>. Examinees whose value is not in <code>focal.name</code> form the reference group, and the function stops when they have more than one value. Examinees with <code>NA</code> in <code>group</code> are excluded from the analysis.
<code>focal.name</code>	A numeric or character vector of the values in <code>group</code> that identify the focal groups, with at least one value. For example, given <code>group = c(0, 0, 1, 2, 2, 3, 3)</code>, where '0' indicates the reference group and '1', '2', and '3' indicate three focal groups, set <code>focal.name = c(1, 2, 3)</code>. The order of <code>focal.name</code> determines the order of the focal groups in the results (for example, the columns <code>n.foc1</code>, <code>n.foc2</code>, and <code>n.foc3</code>); the GRDIF statistics do not depend on it. All examinees whose <code>group</code> value is not in <code>focal.name</code> form the reference group (see Details). Each value must be found in <code>group</code>, and the values must be distinct.

D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1.
alpha	A numeric value specifying the significance level (α) of the tests. An item is flagged by a statistic when its p-value is less than or equal to alpha. The comparison uses unrounded p-values, although the p-values reported in dif_stat are rounded to four decimal places. The same level is used in the post-hoc pairwise analyses, without an adjustment for multiple comparisons. 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 the GRDIF statistic used for purification: "grdifrs" for $GRDIF_{RS}$, "grdifr" for $GRDIF_R$, or "grdifs" for $GRDIF_S$. Used only when <code>purify = TRUE</code> . Default is "grdifrs".
max.iter	A positive integer specifying the maximum number of iterations allowed for the purification process. Default is 10. Each iteration removes one item, so max.iter should be at least as large as the number of items expected to be flagged. If the limit is reached while flagged items remain, a warning is issued, the complete element of the purification results is FALSE, and the items flagged in the last iteration are added to the flagged items. This argument is not passed to <code>est_score()</code> .
min.resp	A single nonnegative number, usually a positive integer, specifying the minimum number of item responses that an examinee must have to be used in the analysis. All responses of examinees with fewer than min.resp (but at least one) responses are set to NA before the ability estimation, in the initial analysis and at every purification iteration, also when score is supplied, and these examinees are excluded from the analysis. A warning reports the number of examinees excluded in the initial analysis, and one warning at the end of the purification reports the examinees excluded during it. A large min.resp can leave a group without any ability estimate in a later purification iteration, which stops the purification with a warning. If NULL, no minimum is applied. Default is NULL. See Details for more information.
post.hoc	Logical. If TRUE, post-hoc pairwise RDIF analyses are performed, for every pair of groups, on the items flagged by each GRDIF statistic. The p-values of the post-hoc analyses are not adjusted for multiple comparisons. See Details . Default is TRUE.
method	A character string indicating the scoring method used to estimate the abilities when <code>score = NULL</code> and in every purification iteration. Available options are: • "ML": Maximum likelihood estimation • "MLF": Maximum likelihood estimation with fences • "WL": Weighted likelihood estimation (Warm, 1989) • "MAP": Maximum a posteriori estimation (Hambleton et al., 1991) • "EAP": Expected a posteriori estimation (Bock & Mislevy, 1982) The methods "EAP.SUM" and "INV.TCC" of <code>est_score()</code> are not supported, because they do not give an ability estimate for each response pattern. Default is "ML".

range	A numeric vector of length two specifying the lower and upper bounds of the ability scale used when abilities are estimated internally with the scoring methods "ML", "MLF", "WL", and "MAP". Default is <code>c(-5, 5)</code> , the default of the other DIF functions.
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 <code>c(0, 1)</code> .
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 <code>gen.weight()</code> . 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 used by <code>est_score()</code> when abilities are estimated internally. The test statistics themselves are not computed in parallel. Default is 1.
verbose	Logical. If TRUE, the progress of the purification procedure is printed to the console. Default is TRUE.
digits	Not used; kept for consistency with the other print methods.

Details

The GRDIF framework (Lim et al., 2024) generalizes the RDIF framework (Lim & Choe, 2023; Lim et al., 2022) to more than two groups. For each item and group, the mean raw residual (MRR) and the mean squared residual (MSR) are computed, where a residual is the observed item score minus the model-expected item score evaluated at the examinee's ability estimate. Under the null hypothesis that the item has no DIF between any pair of groups, the MRRs and MSRs asymptotically follow normal distributions whose means, variances, and covariances are computed analytically from the model-predicted probabilities (Lim et al., 2024, Equations 2 to 6). $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$ are quadratic forms of the contrasts between the focal groups and the reference group of the MRRs, of the MSRs, and of both, centered at their null means, in the inverse of their null covariance matrices (Lim et al., 2024, Equation 10). Under the null hypothesis, they asymptotically follow χ^2 distributions with $G-1$, $G-1$, and $2(G-1)$ degrees of freedom, respectively, where G is the number of groups. The statistics do not depend on which group is the reference group. For two groups, $GRDIF_R$ and $GRDIF_S$ equal the squares of the standardized $RDIF_R$ and $RDIF_S$ statistics of `rdif()`, $GRDIF_{RS}$ equals $RDIF_{RS}$, and the p-values are the same.

`grdif()` supports dichotomous items only, because the GRDIF framework has been developed and evaluated for dichotomous items (Lim et al., 2024). The function stops with an error when `x` includes any item with three or more score categories or any item of the "GRM" or "GPCM" model.

To compute the GRDIF statistics, `grdif()` 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`), `grdif()` estimates them with the scoring method specified in the `method` argument (e.g., `method = "ML"`). When `x` is an object of class `est_irt`, the item parameter estimates, the response data, and the scaling constant D are taken from the object. When `x` is an object of class `est_item`, the ability values stored in the object are also used as `score`.

In rare cases, the covariance matrix of the contrasts that define $GRDIF_R$, $GRDIF_S$, or $GRDIF_{RS}$ is singular or nearly singular (its reciprocal condition number is $1e-10$ or less), for example when all examinees of a group who responded to an item have the same probability of a correct response, so that the MSR is a linear function of the MRR. Then the statistic is computed with the Moore-Penrose generalized inverse of the covariance matrix and compared with a chi-square distribution whose degrees of freedom equal the rank of the matrix (Moore, 1977), and one warning names the items. This handling is not part of the original method (Lim et al., 2024) and is added in `irtQ`. In most data the matrix is not singular, and the degrees of freedom are $G-1$, $G-1$, and $2(G-1)$.

The `group` argument contains the group membership of each examinee as numeric or character values, and its length must equal the number of rows in `data`. The values listed in `focal.name` identify the focal groups, and all other examinees form the reference group. The `group` vector should therefore contain one value for the reference group in addition to the values in `focal.name`, and the function stops when it contains more than one such value. Examinees with NA in `group` are excluded from the analysis.

As in the RDIF framework, an iterative purification procedure can be applied (Lim et al., 2022, 2024). When `purify = TRUE`, the statistic specified in `purify.by` drives the procedure (e.g., `purify.by = "grdifrs"`). At each iteration, the flagged item with the smallest p-value of the `purify.by` statistic is removed (the p-values are compared on the log scale), the abilities are re-estimated from the remaining items with the scoring method specified in `method`, and the GRDIF statistics of the remaining items are recomputed. A supplied score is therefore used only in the initial analysis. The procedure stops when no remaining item is flagged or when `max.iter` iterations have been performed. If no item is flagged in the initial analysis, no iteration is performed. If every item is flagged, the procedure stops with a warning, because no item is left to estimate the abilities, and `complete` is FALSE. The procedure also stops with a warning and `complete = FALSE` when, after an item is removed, no examinee of the reference group or of a focal group has an ability estimate left (see `min.resp` below); the remaining items then keep the results and the scores of the previous iteration, and the items flagged in that iteration are added to the flagged items.

Scoring based on a small number of item responses can lead to large standard errors, potentially reducing the accuracy of DIF detection in the GRDIF framework. The `min.resp` argument excludes such examinees. For example, if `min.resp = 5`, examinees who responded to fewer than five items (but at least one) have all their responses treated as missing (i.e., NA) before the abilities are estimated, in the initial analysis and at every purification iteration, and a warning reports their number in the initial analysis. As a result, their ability estimates are missing, and they are not used in the computation of the GRDIF statistics. When `score` is supplied, the scores of these examinees are also treated as missing. During purification, the count is based on the items that remain after the flagged items are removed, and an examinee excluded once stays excluded. If `min.resp = NULL`, a score is computed for any examinee with at least one valid item response.

Examinees whose ability estimate is NA, such as examinees without any item response or examinees with NA in a supplied score, are excluded from the computation of the GRDIF statistics and from the post-hoc analyses. A warning reports the number of examinees with item responses who are

excluded because of a missing value in score; these examinees stay excluded during purification. During purification, an examinee whose remaining responses are all removed with the flagged items, or who has fewer than `min.resp` responses to the remaining items, has no ability estimate and is excluded, and one warning at the end reports the number of such examinees. The sample sizes `n.ref`, `n.foc1`, `n.foc2`, and so on count only the examinees who are used.

If no examinee of one of the groups responded to an item, the GRDIF statistics and p-values of the item are NaN, and the item is not flagged.

The GRDIF statistics test only whether DIF is present between at least one pair of groups (Lim et al., 2024). When `post.hoc = TRUE`, the three RDIF statistics of `rdif()` are computed for every pair of distinct values in `group`, for each item flagged by $GRDIF_R$, $GRDIF_S$, or $GRDIF_{RS}$. For example, with four groups A, B, C, and D, the pairs A-B, A-C, A-D, B-C, B-D, and C-D are compared. Each pairwise analysis uses the responses of the examinees in the two groups, their ability estimates from the GRDIF analysis, and the item parameters in `x`; the abilities are not re-estimated for the pair. In a pair labeled "a & b" in the column `group.pair`, a is the reference group and b is the focal group of the pair, so $RDIF_R$ and $RDIF_S$ are the residual means of group b minus those of group a. The reference group of the GRDIF analysis is the reference group of every pair that includes it (Lim et al., 2024, Equation 14), and for a pair of focal groups, the group listed earlier in `focal.name` is the reference group. The p-values are not adjusted for multiple comparisons; to control the familywise error rate, compare them with a corrected level, for example alpha divided by the number of pairs (a Bonferroni correction). When `purify = TRUE`, the post-hoc results in `with_purify` are computed for each item when it is removed during purification, with the ability estimates of that iteration; the items added to the flagged items when the purification ends before it is complete (`max.iter` reached or no ability estimate left in a group) are not analyzed.

The print method of this class has an argument `digits` that is not used; the statistics are printed rounded to three decimal places.

Value

This function returns an object of class "grdif", which is a list with the following five 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 with one row per item and the columns `id`, `grdifr`, `grdifs`, `grdifrs`, `p.grdifr`, `p.grdifs`, `p.grdifrs`, `n.ref`, `n.foc1`, `n.foc2`, ..., and `n.total`: the item ID, the three GRDIF statistics, their p-values, the numbers of examinees in the reference group and in each focal group (in the order of `focal.name`) who responded to the item, and their sum. Statistics and p-values are rounded to four decimal places.
 - moments** A list of three data frames, `mu`, `var`, and `covariance`, with the null moments of the MRR and MSR of each group, computed analytically under the null hypothesis of no DIF (Lim et al., 2024, Equations 3 to 5). `mu` and `var` have the columns `id`, `mrr.ref`, `msr.ref`, `mrr.foc1`, `msr.foc1`, and so on, with the null means and null variances of the MRR and MSR of the reference group and of each focal group. `covariance` has the columns `id`, `ref`, `foc1`, and so on, with the null covariance between the MRR and MSR of each group.

- dif_item** A list with the elements `grdifr`, `grdifs`, and `grdifrs`, each giving the positions (rows of `x`) of the items flagged by the corresponding statistic, or `NULL` if no item is flagged.
- score** A numeric vector of the ability estimates used in the initial analysis: the supplied `score`, or the internal estimates when `score = NULL`.
- post.hoc** A list with the elements `by.grdifr`, `by.grdifs`, and `by.grdifrs`. Each element is a data frame with the post-hoc results for the items flagged by the corresponding GRDIF statistic, with one row per item and pair of groups and the columns `id`, `group.pair`, `rdifr`, `z.rdifr`, `rdifs`, `z.rdifs`, `rdifrs`, `p.rdifr`, `p.rdifs`, `p.rdifrs`, `n.ref`, `n.foc`, and `n.total`, defined as in the `dif_stat` element of `rdif()`; `n.ref` and `n.foc` refer to the first and second groups in `group.pair`. An element is `NULL` if no item is flagged by the statistic or if `post.hoc = FALSE`.
- purify** A logical value indicating whether purification was requested (the `purify` argument).
- with_purify** A list with the results of the purification procedure. All elements are `NULL` when `purify = FALSE`. The elements are:
- 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 with the same columns as `no_purify$dif_stat` and an additional column `n.iter`. For an item removed during purification, the row reports the statistics from the iteration in which the item was removed, and `n.iter` is that iteration minus 1 (0 for an item removed on the basis of the initial analysis). For the other items, the row reports the statistics from the last iteration, and `n.iter` is the number of that iteration. If no item is flagged in the initial analysis, this is `no_purify$dif_stat` with `n.iter = 0`.
- moments** A list of three data frames with the same columns as in `no_purify$moments` and an additional column `n.iter`, defined in the same way as in `dif_stat`.
- dif_item** A numeric vector of the positions (rows of `x`) of the items flagged by the `purify.by` statistic, sorted in ascending order. It contains the items removed during purification and, if `max.iter` is reached or the purification stops because a group has no ability estimate left, the items flagged in the last iteration (iteration `n.iter`). It contains all items if every item is flagged. `NULL` if no item is flagged in the initial analysis.
- n.iter** The number of purification iterations performed (0 if no item is flagged in the initial analysis). The iteration that removes the last item, or after which a group has no ability estimate left, is not counted, because the statistics are not recomputed.
- score** A numeric vector of the ability estimates from the last iteration counted in `n.iter`. `NULL` if no item is flagged in the initial analysis, because purification is not carried out; the initial estimates are then in `no_purify$score`.
- post.hoc** A data frame with the same columns as the elements of `no_purify$post.hoc` and an additional column `n_iter`. It contains the post-hoc results of each item removed during purification, computed with the ability estimates of the iteration in which the item was removed; `n_iter` is defined as `n.iter`

in `dif_stat`. NULL if `post.hoc = FALSE` or if no item is flagged in the initial analysis.

complete A logical value. TRUE if the procedure stopped because no remaining item was flagged (including the case in which no item is flagged in the initial analysis), and FALSE if it stopped because `max.iter` was reached, because every item was flagged, or because a group had no examinee with an ability estimate left.

`alpha` A numeric value indicating the significance level (α) used for the tests.

`call` The matched function call.

Methods (by class)

- `grdif(default)`: Default method for a data frame `x` containing item metadata.
- `grdif(est_irt)`: Method for an object of class `est_irt` created by `est_irt()`. The response data and the scaling constant `D` are taken from the object.
- `grdif(est_item)`: Method for an object of class `est_item` created by `est_item()`. The response data, the ability estimates, and the scaling constant `D` are taken from the object.

Methods (by generic)

- `print(grdif)`: Print method for an object of class `grdif` (given as `x`). It prints the flagged items and the GRDIF statistics of the analyses without and with purification, rounded to three decimal places.

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.
- Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. Newbury Park, CA: Sage.
- 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. doi:10.1111/jedm.12415.
- Moore, D. S. (1977). Generalized inverses, Wald's method, and the construction of chi-squared tests of fit. *Journal of the American Statistical Association*, 72(357), 131-137. doi:10.1080/01621459.1977.10479921.
- Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, 54(3), 427-450. doi:10.1007/BF02294627.

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 = rep(0.15, 4)),
    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, each of the first four items has uniform DIF in two of the three
# 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 (not run: it takes about 10 seconds
#     because the abilities of the examinees are re-estimated at every
#     iteration)
#
# 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 after purification
# print(dif_puri_r$with_purify$post.hoc)
#
# purify.by = "grdifs" or "grdifrs" can be used in the same way.
```

info

Item and Test Information Functions

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 <code>est_irt</code> obtained from <code>est_irt()</code> , or <code>est_item</code> from <code>est_item()</code> . See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function. An item with <code>cats = 2</code> under the "GRM" or "GPCM" model has one threshold in <code>par.2</code> , must have NA in <code>par.3</code> , and is computed as a 2PLM item.
...	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 1.702 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. For an `est_irt` or `est_item` object, the item parameter estimates and the scaling constant `D` are taken from the object, and a `D` supplied through `...` is ignored.

Item information is computed under each item's IRT model and summed over items to give the TIF when `tif = TRUE`. The TIF shows where on the theta scale the test measures most precisely.

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:

<code>iif</code>	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", ...).
<code>tif</code>	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 NULL when <code>tif = FALSE</code> .
<code>theta</code>	A numeric vector of theta values at which the item and test information functions are evaluated. This matches the user-supplied <code>theta</code> argument.

The structure is the same for all input types (`data.frame`, `est_item`, and `est_irt`).

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,
  ...
)

## S3 method for class 'irtfit'
print(x, ...)
```

Arguments

<code>x</code>	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.), or an object of class <code>est_irt</code> or <code>est_item</code> obtained from <code>est_irt()</code> or <code>est_item()</code> . See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function.
<code>...</code>	Further arguments passed to or from other methods.
<code>score</code>	A numeric vector of examinees' ability estimates (theta values). Examinees with a missing ability estimate are excluded. Not used by the <code>est_item</code> method, which takes the ability estimates stored in <code>x</code> .
<code>data</code>	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in <code>x</code> . Responses to an item with <code>K</code> score categories must be integers from 0 to <code>K - 1</code> . Missing responses are coded as <code>NA</code> or, in functions that have a missing argument, as the value given in <code>missing</code> .

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 range into intervals of equal width. • "equal.freq": Divides the examinees into groups of approximately equal size. Default is "equal.width". For each item, the intervals span the minimum and maximum ability estimates (after truncation by range.score) of the examinees who responded to the item. The number of intervals is set by n.width.
n.width	An integer specifying the number of intervals (groups) into which the ability scale is divided. Duplicate cut points and empty intervals are dropped, so an item can have fewer groups. 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 giving the lower and upper bounds of the ability scale. Ability estimates outside these bounds are set to the nearest bound before grouping and before computing infit and outfit. The bounds also set the theta range of the plots drawn by <code>plot.irtfit()</code>. If NULL, no truncation is applied, and the plot range is $\pm$ the largest absolute ability estimate rounded up to an integer. 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 1.702 is commonly used for this purpose. Default is 1.
alpha	A numeric value specifying the significance level (α) for the tests of the χ^2 and G^2 item fit statistics. It also sets the confidence level, $1 - \alpha$, of the intervals drawn by <code>plot.irtfit()</code>. Default is 0.05.
missing	A value indicating missing responses in data. Default is NA. Missing responses are excluded item by item.
overSR	A positive numeric threshold for the absolute standardized residuals. The proportion of cells (ability groups by score categories), before collapsing, whose absolute standardized residuals exceed this value is reported in overSR.prop. Default is 2.
min.collapse	A numeric value giving the minimum expected frequency per cell. When computing χ^2 and G^2, adjacent ability groups are merged until every expected cell frequency is at least this value. Default is 1.
pcm.loc	An integer vector giving the positions (rows of x) of partial credit model (PCM) items whose slope parameters are fixed. It is used only to count the item parameters for the degrees of freedom of χ^2. Default is NULL.

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 by intervals of equal width along 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).

Use `n.width` to change the number of groups (default 10). For reference:

- Yen (1981) used 10 groups of approximately equal size,
- Bock (1960) allowed for flexibility in the number of groups.

With `loc.theta = "average"`, the expected probabilities of each group are evaluated at the average ability estimate of the group. The probability at this point approximates the average of the model probabilities of the examinees in the group (e.g., Yen, 1981).

Regarding degrees of freedom (*df*), let G be the number of ability groups after collapsing, K the number of score categories, and m the number of item parameters:

- The χ^2 statistic is approximately chi-square distributed with $G(K-1)-m$ degrees of freedom, which is $G-m$ for a dichotomous item (Ames & Penfield, 2015).
- The G^2 statistic is approximately chi-square distributed with $G(K-1)$ degrees of freedom, which is G for a dichotomous item (Ames & Penfield, 2015; Muraki & Bock, 2003).

The number of item parameters m is determined by the IRT model of each item: 1 for the 1PLM, 2 for the 2PLM, 3 for the 3PLM, and the number of score categories K for the GRM and GPCM; an item listed in `pcm.loc` has $m = K - 1$, and an item labeled "DRM" is counted as a 3PLM item. This also applies to `est_irt` and `est_item` objects: parameters fixed during estimation (for example, with `fix.g = TRUE`, or the items fixed in FIPC) are still counted, and GPCM items estimated with `fix.a.gpcm = TRUE` are counted as PCM items only when they are given in `pcm.loc`. When no degrees of freedom remain for χ^2 after collapsing (`df.X2` of 0 or less), `crit.val.X2` and `p.X2` are NA, and a warning names the item.

Responses in data must be integer scores from 0 to the number of score categories minus 1, with one column per item. Character and factor responses are read as numbers, logical responses as 0 and 1, and other responses stop the function with an error.

For ability group j with N_j examinees, let O_{jk} and E_{jk} be the observed proportion and the model-expected probability of score category k at the theta point of the group (see `loc.theta`). The statistics are $\chi^2 = \sum_j \sum_k N_j (O_{jk} - E_{jk})^2 / E_{jk}$ and $G^2 = 2 \sum_j \sum_k N_j O_{jk} \log(O_{jk} / E_{jk})$, where

cells with $O_{jk} = 0$ add nothing to G^2 . For a dichotomous item, χ^2 has the form of Yen's (1981) Q_1 statistic $\sum_j N_j (O_j - E_j)^2 / [E_j(1 - E_j)]$. The standardized residual of a cell before collapsing is $(O_{jk} - E_{jk}) / \sqrt{E_{jk}(1 - E_{jk}) / N_j}$.

Infit and outfit use the category scores $0, \dots, K - 1$. For examinee n , the residual is $r_n = x_n - E(X | \theta_n)$ and the variance is $W_n = \sum_k (k - E(X | \theta_n))^2 P_k(\theta_n)$, where θ_n is the ability estimate in score. Outfit is the unweighted mean $\sum_n (r_n^2 / W_n) / N$, and infit is the variance-weighted mean $\sum_n r_n^2 / \sum_n W_n$.

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

Value

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

<code>fit_stat</code>	A data frame with one row per item and the columns <code>id</code> , <code>X2</code> (χ^2), <code>G2</code> (G^2), <code>df.X2</code> , <code>df.G2</code> , <code>crit.val.X2</code> and <code>crit.val.G2</code> (critical values at α), <code>p.X2</code> , <code>p.G2</code> , <code>outfit</code> , <code>infit</code> , <code>N</code> (the number of examinees who responded to the item and have an ability estimate), and <code>overSR.prop</code> (the proportion of cells, before collapsing, whose absolute standardized residuals exceed <code>overSR</code>).
<code>contingency.fitstat</code>	A list of contingency tables, one per item, used to compute χ^2 and G^2 . Adjacent ability groups are merged until every expected frequency is at least <code>min.collapse</code> , the rows are numbered, and the interval labels are dropped. The columns are <code>total</code> and, for each score category, <code>obs.freq.*</code> , <code>exp.freq.*</code> , <code>obs.prop.*</code> , <code>exp.prop.*</code> , and <code>raw.rsd.*</code> .
<code>contingency.plot</code>	A list of contingency tables, one per item, used by <code>plot.irtfit()</code> to draw raw and standardized residual plots (Hambleton et al., 1991). The tables use the uncollapsed groups and contain <code>interval</code> , <code>point</code> (the theta point of the group), <code>total</code> , and, for each score category, <code>obs.freq.*</code> , <code>obs.prop.*</code> , <code>exp.prop.*</code> , <code>raw.rsd.*</code> , <code>se.*</code> , and <code>std.rsd.*</code> .
<code>item_df</code>	The item metadata used in the analysis. Items with fewer than two responses are removed.
<code>individual.info</code>	A list of data frames, one per item, with the residual (<code>resid</code>) and variance (<code>Var</code>) for each examinee who responded to the item. These values are used to compute infit and outfit.
<code>ancillary</code>	A list with <code>range.score</code> , <code>alpha</code> , <code>overSR</code> , and <code>scale.D</code> , used by <code>plot.irtfit()</code> .
<code>call</code>	The matched call.

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)`: Method for an object of class `est_item` created by `est_item()`. The item parameter estimates, response data, ability estimates, and scaling constant D are taken from the object.
- `irtfit(est_irt)`: Method for an object of class `est_irt` created by `est_irt()`. The item parameter estimates, response data, and scaling constant D are taken from the object, and score must be supplied.

Methods (by generic)

- `print(irtfit)`: Print method for an object of class `irtfit` (given as `x`). It prints the significance level and the table of item fit statistics (`fit_stat`) as stored in the object.

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>
- 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 whose summed item scores exceed 10,000
over10000 <- which(colSums(simCAT_MX$res.dat, na.rm = TRUE) > 10000)

# Select these items
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	<i>Log-Likelihood of Ability Parameters</i>
-------------	---

Description

This function computes the log-likelihood values for a set of ability values, 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 <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function.
data	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in x. Responses to an item with K score categories must be integers from 0 to K - 1. Missing responses are coded as NA or, in functions that have a missing argument, as the value given in missing.
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 1.702 is commonly used for this purpose. Default is 1.
method	A character string specifying the estimation method. Available options are: • "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).

fence.a	A numeric value specifying the slope (<i>a</i>) parameter of the two fence items used in MLF. See Details below. Default is 3.0.
fence.b	A numeric vector of length two specifying the difficulty (<i>b</i>) parameters of the lower and upper fence items. If NULL, <code>c(-5, 5)</code> is used. Used only when method = "MLF". Default is NULL.
missing	A value indicating missing responses in the data set. Default is NA.

Details

This function evaluates the log-likelihood at each value of theta for one or more examinees, based on item parameters (*x*) and item response data (*data*). For method = "MAP", the log density of the normal prior is added, so the values are log-posterior values.

If method = "MLF", two fence items with slope `fence.a` and difficulties `fence.b` are appended, with the lower fence answered correctly and the upper fence answered incorrectly. The fences give the likelihood a finite maximum for all-correct and all-incorrect patterns. See [est_score\(\)](#) for details.

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 (log-posterior values for method = "MAP").

- Each **row** corresponds to a value of theta.
- Each **column** corresponds to an examinee and is named `Resp.1`, `Resp.2`, and so on. Examinees without any observed response have NA in their column.

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 (ML)
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

An integer matrix with 1,000 rows (examinees) and 5 columns (items item.1 to item.5). Each entry is a dichotomously scored response (0 = incorrect, 1 = correct). The 1,000 rows contain 30 distinct response patterns.

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 summed (observed) scores with the Lord and Wingersky (1984) recursion formula, given either the category response probabilities of each item or item metadata and a set of theta values.

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 <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function. If x is supplied, prob and cats are ignored. Default is NULL.
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. Used only when x is NULL. Default is NULL.
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 1.702 is commonly used for this purpose. Default is 1.

Details

The Lord and Wingersky recursion efficiently computes the probabilities of all possible summed scores on a test under IRT models, which gives the model-based distribution of observed scores. It applies to both dichotomous and polytomous items.

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 prob is used, a named numeric vector (score.0, score.1, ...) of the probabilities of all possible summed scores.

When x is used, a matrix of conditional probabilities with one row per summed score (score.0, score.1, ...) and one column per theta value (theta.1, theta.2, ...).

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. M., & Wingersky, M. S. (1984). Comparison of IRT true-score and equipercentile observed-score "equatings." *Applied Psychological Measurement*, 8(4), 453-461. doi:10.1177/014662168400800409.

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)
```

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.

The route map must be a square matrix of 0s and 1s, and the modules do not need to be numbered in the order of the stages. The function stops with an error when the route map defines a single stage, contains a cycle, has a module with no transition to or from another module, has a module that cannot be reached from a stage-1 module or that belongs to more than one stage, or has a module before the last stage without a transition to a module of the next stage. A logical route map is read as a 0/1 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., and column names are "stage.1", "stage.2", etc. Each entry is the index of the module administered at that 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 for Item Parameter Drift Detection

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). The current version supports dichotomous items only.

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

<code>x</code>	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.). See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function. Only dichotomous items are supported; the function stops if <code>x</code> contains GRM or GPCM items.
<code>data</code>	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in <code>x</code> . Responses to an item with <code>K</code> score categories must be integers from 0 to <code>K - 1</code> . Missing responses are coded as <code>NA</code> or, in functions that have a <code>missing</code> argument, as the value given in <code>missing</code> .
<code>D</code>	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1.
<code>item.skip</code>	A numeric vector of item positions (row numbers of <code>x</code>), or a logical vector with one value per item, identifying the items to exclude from the IPD analysis. Skipped items still contribute to the posterior ability distributions; only their <code>pcd2</code> values are set to <code>NA</code> , and they are never flagged. If <code>NULL</code> , all items are analyzed. Default is <code>NULL</code> .
<code>missing</code>	A value indicating missing responses in the data set. Default is <code>NA</code> .
<code>Quadrature</code>	A numeric vector of length two giving the number of quadrature points and the symmetric bound (absolute value) of the points. For example, <code>c(49, 6)</code> specifies 49 evenly spaced points from -6 to 6. These points are used to compute each examinee's posterior ability distribution when <code>weights = NULL</code> . Default is <code>c(49, 6)</code> .
<code>weights</code>	A two-column matrix or data frame of quadrature points (first column) and weights (second column) defining the prior ability distribution used to compute the posterior distributions. It can be created with <code>gen.weight()</code> . If <code>NULL</code> , a normal prior defined by <code>Quadrature</code> , <code>group.mean</code> , and <code>group.var</code> is used. Default is <code>NULL</code> .
<code>group.mean</code>	A numeric value specifying the mean of the normal prior ability distribution when <code>weights = NULL</code> . Default is 0.
<code>group.var</code>	A positive numeric value specifying the variance of the normal prior ability distribution when <code>weights = NULL</code> . Default is 1.
<code>crit.val</code>	A numeric critical value for the pseudo-count D^2 statistic. Items whose statistic is greater than or equal to <code>crit.val</code> are flagged. Because the statistic has no known null distribution, the critical value is usually obtained by bootstrapping (see Examples). If <code>NULL</code> (default), the statistic is reported but no item is flagged and purification is not performed.

<code>min.resp</code>	A single nonnegative number, usually a positive integer, specifying the minimum number of responses an item must have to be analyzed. All responses of items with fewer than <code>min.resp</code> (but at least one) responses are set to NA, so their <code>pcd2</code> value is NaN and <code>N</code> is 0, and a warning lists those items. If NULL, no minimum is applied. Default is NULL.
<code>purify</code>	Logical. Whether to apply the iterative purification procedure. Purification is performed only when <code>crit.val</code> is also supplied. Default is FALSE.
<code>max.iter</code>	A positive integer specifying the maximum number of purification iterations. Default is 10. If the limit is reached while flagged items remain, a warning is issued, the items flagged at the last iteration are added to the flagged set, and the complete element of <code>with_purify</code> is FALSE. Because one item is removed per iteration, <code>max.iter</code> should be at least the number of items expected to drift.
<code>verbose</code>	Logical. If TRUE, the progress of the purification procedure is printed to the console. Default is TRUE.

Details

The pseudo-count D^2 statistic quantifies item parameter drift (IPD) by computing, across quadrature points, the weighted squared differences between the observed pseudo-proportion of correct responses and the proportion expected under the item bank parameters, weighted by the pseudo-counts. The observed pseudo-counts are obtained from the posterior distribution of each examinee's ability, and the expected proportions are the probabilities of a correct response computed from the item bank parameters (Stone, 2000).

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

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

where Q is the number of quadrature points, r_{0k} and r_{1k} are the pseudo-counts of incorrect and correct responses at quadrature point k (the sums of the examinees' posterior probabilities at that point over the examinees with each response), E_{1k} is the probability of a correct response at point k under the item bank parameters, and N is the number of examinees who responded to the 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 is greater than or equal to this threshold, the item is flagged. 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`.
- If `max.iter` is reached before convergence, a warning is issued, the items flagged at the last iteration are added to the flagged set, and `complete` is set to `FALSE`.
- If every item is flagged, the procedure stops with a warning, because no item is left to analyze, all items are returned as flagged, and `complete` is `FALSE`.
- All flagged items and statistics are saved, and convergence status is reported.

This process reduces the influence of already-flagged items on the posterior distributions used for the remaining items.

Value

This function returns a list with the following elements:

<code>no_purify</code>	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: <code>id</code> (item ID), <code>pcd2</code> (the computed D^2 value), and <code>N</code> (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 <code>crit.val</code>. If no items are flagged or <code>crit.val = NULL</code>, this is <code>NULL</code>.
<code>purify</code>	The value of the <code>purify</code> argument. Purification is actually performed only when <code>crit.val</code> is also supplied.
<code>with_purify</code>	A list containing the results of pseudo-count D^2 analysis after applying the purification procedure. This list is populated only when both <code>purify = TRUE</code> and <code>crit.val</code> is not <code>NULL</code>; otherwise, all of its elements are <code>NULL</code>. It includes: ipd_stat A data frame of the pseudo-count D^2 statistics for all items, taken for each item from the analysis given in <code>n.iter</code>. Columns include: <code>id</code> (item ID), <code>pcd2</code> (the computed D^2 value), <code>N</code> (the number of valid responses), and <code>n.iter</code> (for a flagged item, the iteration at which it was flagged, where 0 is the initial analysis; for the other items, the final iteration). ipd_item A numeric vector of item indices (row positions in <code>x</code>) flagged as IPD items during purification, sorted in ascending order. The iteration at which each item was flagged is given in the <code>n.iter</code> column of <code>ipd_stat</code>. It contains all items if every item is flagged. It is <code>NULL</code> when no item is flagged in the initial analysis. n.iter An integer indicating the number of purification iterations completed. The iteration that removes the last item is not counted, because the statistics are not recomputed. complete A logical value indicating whether the purification procedure converged before reaching the maximum number of iterations (<code>max.iter</code>). <code>FALSE</code> if the iteration limit was reached before convergence or if every item was flagged.

<code>crit.val</code>	A numeric value indicating the critical threshold used to flag items for parameter drift. If not specified by the user, this will be NULL.
<code>call</code>	The matched function call.

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. doi:10.1080/15366367.2018.1511199.
- 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.
- 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. doi:10.1111/j.1745-3984.2000.tb01076.x.

See Also

[ripd\(\)](#), [simIPD](#), [gen.weight\(\)](#)

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
# crit.val = 0.002 is an arbitrary value for illustration; no drift was
# simulated, so any flagged item is a false positive.
```

```

# 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 Cappaert et al. (2018), the critical value is estimated
## empirically via bootstrap:
##   (1) Select items that are not suspected of IPD (anchor items).
##   (2) Compute D2 for the whole item pool from all responses, and take
##       the D2 values of the anchor items as the null distribution.
##   (3) Resample the null distribution with replacement, take the 95th
##       percentile of each bootstrap sample, and average the percentiles
##       to obtain the critical value.
##   (4) Run pcd2() with this 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 -----
## Select items that (a) have at least min_resp_anchor observed responses
## and (b) are not suspected of IPD. In practice, exclude the items you
## suspect to have drifted; here the ground truth (simIPD$ipd_item) is
## excluded for illustration.
## The minimum response count follows Lim & Han (2026); it is a practical
## addition to the procedure of Cappaert et al. (2018), who did not set such
## a criterion. In a CAT pool with uneven exposure, items with few responses
## have large D2 values that inflate the tail of the null distribution. For
## the simIPD data, the critical value computed below is about 0.034 when
## all items with at least one response are used, about 0.008 with at least
## 50 responses, about 0.006 with at least 100, and about 0.004 with at
## least 300.
min_resp_anchor <- 300
n_resp          <- colSums(!is.na(simIPD$foc_resp)) # observed responses per item
anchor_items    <- setdiff(which(n_resp >= min_resp_anchor), simIPD$ipd_item)

## -- Step 2. Compute D2 for the whole pool and keep the anchor items -----
## D2 is computed once from all items and responses, so that the posterior
## ability distributions are the same as in the final analysis (Step 4).
## A warning lists the pool items that were never administered.
pcd2_all <- pcd2(
  x    = simIPD$item_par,
  data = simIPD$foc_resp,
  D    = 1.7

```

```

)
pcd2_null <- pcd2_all$no_purify$ipd_stat$pcd2[anchor_items]

## -- Step 3. Bootstrap critical value (Cappaert et al., 2018) -----
## For each bootstrap iteration: resample the null D2 values with replacement
## (the sample size equals the number of anchor items) and take the 95th
## percentile. The critical value is the mean of these percentiles.
## 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 <- mean(vapply(
  X = seq_len(n_boots),
  FUN = function(b) {
    stats::quantile(sample(x = pcd2_null, size = length(pcd2_null), replace = TRUE),
                      probs = 0.95)
  },
  FUN.VALUE = numeric(1)
))

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

## -- Step 4. Run PCD2 with the bootstrap critical value + purification --
## The statistics of the first analysis (pcd2_result$no_purify) are the same
## as those of pcd2_all for the analyzed items, since the same responses are
## used. The same warning about the items that were never administered
## appears.
pcd2_result <- pcd2(
  x = simIPD$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")

```

Description

Produces a **ggplot2** visualization 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

<code>x</code>	An object of class "find_cut" returned by find_cut .
<code>theta_range</code>	A numeric vector of length 2 specifying the theta range shown on the x-axis. The TIF curves, the vertical lines, and the dots outside this range are cut off. Defaults to the full range stored in <code>x\$tif_data</code> .
<code>show_anomalous</code>	Logical. If TRUE (default), anomalous TIF crossings (negative-slope crossings excluded from the cut scores) are shown as dashed red vertical lines.
<code>label_cuts</code>	Logical. If TRUE (default), each selected cut score is annotated with its theta value above the crossing point.
<code>layout</code>	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).
<code>...</code>	Currently unused. Reserved for future arguments.

Value

A **ggplot** object. The object is printed automatically when called interactively. It can be further customized 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

<code>x</code>	An object of class <code>info</code> obtained from <code>info()</code> .
<code>item.loc</code>	A numeric vector indicating which item information functions to plot, specified by item position (e.g., 1 for the first item). If <code>NULL</code> (default), the test information function for the entire test form is plotted.
<code>overlap</code>	Logical. If <code>TRUE</code> , multiple item information functions are plotted in a single panel. If <code>FALSE</code> (default), each item information function is displayed in a separate panel.
<code>csee</code>	Logical. If <code>TRUE</code> , 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 <code>FALSE</code> (default), item or test information functions are plotted.
<code>xlab.text, ylab.text</code>	Character strings specifying the labels for the x and y axes, respectively. If missing, default labels are used.
<code>main.text</code>	Character string specifying the overall title of the plot. If missing, a default title is used.
<code>lab.size</code>	Numeric value specifying the font size of axis titles. Default is 15.
<code>main.size</code>	Numeric value specifying the font size of the plot title. Default is 15.
<code>axis.size</code>	Numeric value specifying the font size of axis tick labels. Default is 15.
<code>line.color</code>	A character string specifying the color of the plot lines. Default is <code>"#F8766D"</code> . Not used when curves are overlaid (<code>overlap = TRUE</code> with <code>item.loc</code> specified), where colors are assigned automatically. See http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/ for available color names.
<code>line.size</code>	Numeric value specifying the thickness of plot lines. Default is 1.
<code>layout.col</code>	Integer. Number of columns to use when faceting multiple item information functions. Used only when <code>overlap = FALSE</code> . Default is 4.
<code>strip.size</code>	Numeric value specifying the font size of the panel (facet) labels. Used only when <code>overlap = FALSE</code> . Default is 12.
<code>...</code>	Additional arguments passed to <code>ggplot2::geom_line()</code> 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

A **ggplot2** object of the item or test information functions, or of the test-level CSEE when `csee = TRUE`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also[info\(\)](#)**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(-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)
```

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

<code>x</code>	An object of class <code>irtfit</code> obtained from <code>irtfit()</code> .
<code>item.loc</code>	An integer giving the position of the item to plot in the <code>item_df</code> element of <code>x</code> . See Details below.
<code>type</code>	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".
<code>ci.method</code>	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.
<code>show.table</code>	A logical value indicating whether to return the contingency table used to draw the residual plots of the item, taken from the <code>contingency.plot</code> element of <code>x</code> . Default is TRUE.

layout.col	An integer specifying the number of columns in the panel layout for a polytomous item when type is "icc" or "sr". With type = "both", each plot uses one column. 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 limits of the standardized residual plot are set to $\pm$ the largest absolute standardized residual of the item plus 1, rounded to an integer. If FALSE, ylim.sr is used. 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 <code>ggplot2::ggplot()</code> from the ggplot2 package.

Details

All plots are generated using the **ggplot2** package.

The object of class `irtfit` returned by `irtfit()` is used to draw raw and standardized residual plots. The plots are based mainly on the `contingency.plot` element of the object.

Residual plots are drawn for one item at a time. Set `item.loc` to the position of the item in the `item_df` element of the `irtfit` object; for example, `item.loc = 3` selects the third item. Items with fewer than two responses are removed by `irtfit()`, so positions can differ from those in the original item metadata.

For the raw residual plot, the `ci.method` argument sets the method used to compute the confidence intervals of the observed proportions:

- "wald": Wald-type interval based on the normal approximation (Laplace, 1820), $\hat{p} \pm z_{1-\alpha/2} \sqrt{P(1-P)/N}$, where $\hat{p}$ is the observed proportion, P is the model-expected probability, and N is the group size. The limits are truncated to the interval from 0 to 1.
- "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. The confidence level is $1 - \alpha$, where α is the alpha argument of `irtfit()`.

In the standardized residual plot, residuals whose absolute values exceed the `overSR` argument of `irtfit()` are drawn as red circles, and the others as blue crosses. Dashed lines mark $\pm$ `overSR`.

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 probabilités* (3rd ed., in French). Paris: 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

<code>x</code>	An object of class <code>traceline</code> obtained from <code>traceline()</code> .
<code>item.loc</code>	A numeric vector specifying the position(s) of the item(s) to plot. If <code>NULL</code> (default), the test characteristic curve (TCC) for the entire test form is plotted.
<code>score.curve</code>	Logical. If <code>TRUE</code> , plots the item score curve, which is the expected item score (the sum of category scores weighted by their probabilities) as a function of θ . If <code>FALSE</code> , plots the item characteristic curves (ICCs) of all score categories. For dichotomous items, the item score curve equals the ICC for score category 1. Ignored when <code>item.loc = NULL</code> . Default is <code>FALSE</code> .
<code>overlap</code>	Logical. Determines how multiple curves are displayed when plotting ICCs or item score curves. If <code>TRUE</code> , curves are overlaid in different colors: the category ICCs of each item share one panel per item when <code>score.curve = FALSE</code> , and the score curves of all items share a single panel when <code>score.curve = TRUE</code> . If <code>FALSE</code> , each curve is drawn in its own panel: one panel per score category of a single item when <code>score.curve = FALSE</code> , or one panel per item when <code>score.curve = TRUE</code> . Default is <code>FALSE</code> .
<code>layout.col</code>	An integer value indicating the number of panel columns when multiple panels are drawn. Not used when <code>overlap = TRUE</code> and <code>score.curve = TRUE</code> . Default is 2.
<code>xlab.text, ylab.text</code>	Character strings specifying the labels for the x and y axes, respectively. If missing, default labels are used.
<code>main.text</code>	Character string specifying the overall title of the plot. If missing, a default title is used.
<code>lab.size</code>	Numeric value specifying the font size of axis titles. Default is 15.
<code>main.size</code>	Numeric value specifying the font size of the plot title. Default is 15.
<code>axis.size</code>	Numeric value specifying the font size of axis tick labels. Default is 15.
<code>line.color</code>	A character string specifying the color of the plot lines. Default is <code>"#F8766D"</code> . Not used when curves are overlaid (<code>overlap = TRUE</code> with <code>item.loc</code> specified), where colors are assigned automatically. See http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/ for available color names.
<code>line.size</code>	Numeric value specifying the thickness of plot lines. Default is 1.
<code>strip.size</code>	Numeric. Font size of the panel (facet) labels. Default is 12.
<code>...</code>	Additional arguments passed to <code>ggplot2::geom_line()</code> from the ggplot2 package.

Details

All plots are drawn with the **ggplot2** package. If `item.loc = NULL`, the test characteristic curve (TCC) of the entire test form is plotted. Otherwise, `item.loc` gives the positions of the items to plot; for example, `item.loc = 1:3` plots the first three items.

Value

This method prints the plot of item characteristic curves (ICCs), item score curves, or the test characteristic curve (TCC), and invisibly returns the **ggplot2** object.

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

<code>x</code>	An object of class "ctt", as returned by <code>ctt()</code> .
<code>digits</code>	Number of decimal places used for display. Default is 3. The statistics in <code>x</code> are already rounded to three decimal places by <code>ctt()</code> , so a value above 3 adds no precision.
<code>...</code>	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

<code>x</code>	An object of class "summary.ctt", as returned by summary.ctt() .
<code>digits</code>	Number of decimal places used for display. Default is 3. The values in <code>x</code> are already rounded by ctt() (item and test statistics to three decimal places, percentages to two), so a larger value adds no precision.
<code>...</code>	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 value of the item discrimination (slope) parameter.
d	A numeric vector of the threshold parameters of the item. For the GRM, the thresholds must be in increasing order.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1.

`pr.model` A character string specifying the polytomous IRT model: "GRM" for the graded response model or "GPCM" for the (generalized) partial credit model. Default is "GRM".

Details

When computing category probabilities using the partial credit model (PCM), set $a = 1$.

For `pr.model = "GPCM"`, `d` contains the threshold parameters $b_v = \beta - \tau_v$, that is, the overall item location minus the threshold of each score category. An item with K score categories requires $K - 1$ threshold parameters. The term for the lowest score category is fixed at 0 and is not supplied. For example, a GPCM item with five categories requires four threshold parameters.

The GRM probabilities are bounded to the range from $1e-10$ to $1 - 1e-10$, whereas the GPCM probabilities are bounded below only by the smallest positive normalized double value (about $2.2e-308$). The category probabilities in `traceline()` and `info()` are bounded below by $1e-10$ for both models.

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,
  ...
)

## S3 method for class 'rdif'
print(x, digits = max(2L, getOption("digits") - 5L), ...)

```

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` or `est_item` obtained from `est_irt()` or `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 <code>est_score()</code> when abilities are estimated internally, for example <code>tol</code> or <code>se</code> . The <code>est_score()</code> arguments <code>max.iter</code> and <code>missing</code> cannot be passed this way, because the function uses arguments with these names for its own purposes.
<code>data</code>	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in <code>x</code> . Responses to an item with <code>K</code> score categories must be integers from 0 to <code>K - 1</code> . Missing responses are coded as <code>NA</code> or, in functions that have a <code>missing</code> argument, as the value given in <code>missing</code> .
<code>score</code>	A numeric vector of examinees' ability estimates (theta values), in the same row order as <code>data</code> . The estimates should be based on the item parameters in <code>x</code> (see Details). If <code>NULL</code> , abilities are estimated internally with <code>est_score()</code> using method, range, norm.prior, nquad, weights, and ncore. When <code>purify = TRUE</code> , abilities are re-estimated internally at every purification iteration, so a supplied score is used only in the initial analysis. A missing value in <code>score</code> excludes the examinee from the analysis, also during purification, with a warning. The <code>est_item</code> method has no <code>score</code> argument; it uses the abilities stored in the object. Default is <code>NULL</code> .
<code>group</code>	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.
<code>focal.name</code>	A single numeric or character value specifying the focal group. For instance, given <code>group = c(0, 1, 0, 1, 1)</code> and <code>'1'</code> indicating the focal group, set <code>focal.name = 1</code> .
<code>item.skip</code>	A numeric vector of item positions (row numbers of <code>x</code>), or a logical vector with one value per item, identifying the items to exclude from the analysis. Skipped items still contribute to the ability estimates, but their results (statistics, p-values, and moments) are set to <code>NA</code> , and they are never flagged or removed during purification. If <code>NULL</code> , all items are analyzed. Default is <code>NULL</code> .
<code>D</code>	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1.
<code>alpha</code>	A numeric value specifying the significance level (α) of the tests. An item is flagged by a statistic when its p-value is less than or equal to <code>alpha</code> . The comparison uses unrounded p-values, although the p-values reported in <code>dif_stat</code> are rounded to four decimal places. Default is <code>0.05</code> .
<code>missing</code>	A value indicating missing responses in the data set. Default is <code>NA</code> .
<code>purify</code>	Logical. Indicates whether to apply a purification procedure. Default is <code>FALSE</code> .
<code>purify.by</code>	A character string specifying the RDIF statistic used for purification: <code>"rdifrs"</code> for $RDIF_{RS}$, <code>"rdifr"</code> for $RDIF_R$, or <code>"rdifs"</code> for $RDIF_S$. Used only when <code>purify = TRUE</code> . Default is <code>"rdifrs"</code> .
<code>max.iter</code>	A positive integer specifying the maximum number of iterations allowed for the purification process. Default is 10. Each iteration removes one item, so <code>max.iter</code> should be at least as large as the number of items expected to be flagged. If the limit is reached while flagged items remain, a warning is issued, the complete element of the purification results is <code>FALSE</code> , and the items flagged

in the last iteration are added to the flagged items. This argument is not passed to `est_score()`.

<code>min.resp</code>	A single nonnegative number, usually a positive integer, specifying the minimum number of item responses that an examinee must have to be used in the analysis. All responses of examinees with fewer than <code>min.resp</code> (but at least one) responses are set to NA before the ability estimation, in the initial analysis and at every purification iteration, also when <code>score</code> is supplied, and these examinees are excluded from the analysis. A warning reports the number of examinees excluded in the initial analysis, and one warning at the end of the purification reports the examinees excluded during it. A large <code>min.resp</code> can leave a group without any ability estimate in a later purification iteration, which stops the purification with a warning. If NULL, no minimum is applied. Default is NULL. See Details for more information.
<code>method</code>	A character string indicating the scoring method used to estimate the abilities when <code>score = NULL</code> and in every purification iteration. Available options are: • "ML": Maximum likelihood estimation • "MLF": Maximum likelihood estimation with fences • "WL": Weighted likelihood estimation (Warm, 1989) • "MAP": Maximum a posteriori estimation (Hambleton et al., 1991) • "EAP": Expected a posteriori estimation (Bock & Mislevy, 1982) The methods "EAP.SUM" and "INV.TCC" of <code>est_score()</code> are not supported, because they do not give an ability estimate for each response pattern. Default is "ML".
<code>range</code>	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 <code>c(-5, 5)</code> .
<code>norm.prior</code>	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 <code>method</code> is "ML" or "WL". Default is <code>c(0, 1)</code> .
<code>nquad</code>	An integer indicating the number of Gaussian quadrature points to be generated from the normal prior distribution. Used only when <code>method</code> is "EAP". Ignored for "ML", "WL", and "MAP". Default is 41.
<code>weights</code>	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 <code>gen.weight()</code>. If NULL and <code>method = "EAP"</code>, default quadrature values are generated based on the <code>norm.prior</code> and <code>nquad</code> arguments. Ignored if <code>method</code> is "ML", "WL", or "MAP".
<code>ncore</code>	An integer specifying the number of logical CPU cores used by <code>est_score()</code> when abilities are estimated internally. The test statistics themselves are not computed in parallel. Default is 1.
<code>verbose</code>	Logical. If TRUE, the progress of the purification procedure is printed to the console. Default is TRUE.
<code>digits</code>	Not used; kept for consistency with the other print methods.

Details

The RDIF framework (Lim & Choe, 2023; Lim et al., 2022) consists of three IRT residual-based statistics: $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$. For each item, a residual is the observed item score minus the model-expected item score evaluated at the examinee's ability estimate. $RDIF_R$ is the difference between the focal and reference groups in the mean raw residual, and $RDIF_S$ is the difference in the mean squared residual. Under the null hypothesis that the item has no DIF, $RDIF_R$ and $RDIF_S$ asymptotically follow normal distributions whose means and variances are computed analytically from the model-predicted probabilities (Lim et al., 2022, Equations 9, 10, 12, and 13). The null mean of $RDIF_R$ is zero, whereas the null mean of $RDIF_S$ is generally not zero. Each statistic is standardized by subtracting its null mean and dividing by its null standard deviation ($z.rdifr$ and $z.rdif_s$), and a two-sided p-value is obtained from the standard normal distribution. $RDIF_{RS}$ is the quadratic form of the vector of $RDIF_R$ and $RDIF_S$, centered at their null means, in the inverse of their null covariance matrix. Under the null hypothesis, $RDIF_{RS}$ asymptotically follows a χ^2 distribution with 2 degrees of freedom (Lim et al., 2022, Equation 20).

In rare cases, the covariance matrix of $RDIF_R$ and $RDIF_S$ is singular or nearly singular (its reciprocal condition number is 1e-10 or less), for example when all examinees who responded to an item have the same probability of a correct response, so that $RDIF_S$ is a linear function of $RDIF_R$. Then $RDIF_{RS}$ is computed with the Moore-Penrose generalized inverse of the covariance matrix and compared with a chi-square distribution whose degrees of freedom equal the rank of the matrix (Moore, 1977), and a warning names the item. This handling is not part of the original method (Lim et al., 2022) and is added in `irtQ`. In most data the matrix is not singular, and $RDIF_{RS}$ has two degrees of freedom.

`rdif()` accepts both dichotomous and polytomous items. For a polytomous item, the residual is the observed item score minus the model-expected item score, and the null means, variances, and covariance are computed from the model-predicted category probabilities in the same way, so the three statistics assess net DIF (Jung & Lim, 2026; Lim, Malatesta, & Lee, 2024). To evaluate global DIF in 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"`). When `x` is an object of class `est_irt`, the item parameter estimates, the response data, and the scaling constant `D` are taken from the object. When `x` is an object of class `est_item`, the ability values stored in the object are also used as `score`.

The `group` argument should be a vector containing exactly two distinct values (either numeric or character), representing the reference and focal groups, and the function stops when it contains more than two. 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. Use `grdif()` to compare more than two groups. Examinees with NA in `group` are excluded from both groups.

Examinees whose ability estimate is NA, such as examinees without any item response or examinees with NA in a supplied `score`, are excluded from the computation of the RDIF statistics. A warning

reports the number of examinees with item responses who are excluded because of a missing value in `score`; these examinees stay excluded during purification. During purification, an examinee whose remaining responses are all removed with the flagged items, or who has fewer than `min.resp` responses to the remaining items, has no ability estimate and is excluded, and one warning at the end reports the number of such examinees. The sample sizes `n.ref` and `n.foc` count only the examinees who are used.

If no examinee of one of the two groups responded to an item, the statistics and p-values of the item are `NaN`, and the item is not flagged.

Similar to other DIF detection approaches, the RDIF framework supports an iterative purification procedure (Lim et al., 2022). When `purify = TRUE`, the statistic specified in `purify.by` drives the procedure (e.g., `purify.by = "rdifrs"`). At each iteration, the flagged item with the smallest p-value of the `purify.by` statistic is removed (the p-values are compared on the log scale), the abilities are re-estimated from the remaining items with the scoring method specified in `method`, and the RDIF statistics of the remaining items are recomputed. A supplied score is therefore used only in the initial analysis. The procedure stops when no remaining item is flagged or when `max.iter` iterations have been performed, where `max.iter` must be a single whole number of at least 1. If no item is flagged in the initial analysis, no iteration is performed. If every item is flagged, the procedure stops with a warning, because no item is left to estimate the abilities, and `complete` is `FALSE`. The procedure also stops with a warning and `complete = FALSE` when, after an item is removed, no examinee of the reference group or of the focal group has an ability estimate left (see `min.resp` below); the remaining items then keep the results of the previous iteration, and the items flagged in that iteration are added to the flagged items.

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 excludes such examinees. For example, if `min.resp = 5`, examinees who responded to fewer than five items (but at least one) have all their responses treated as missing (i.e., `NA`) before the abilities are estimated, in the initial analysis and at every purification iteration. As a result, their ability estimates are missing, and they are not used in the computation of the RDIF statistics. When `score` is supplied, the scores of these examinees are also treated as missing in the initial analysis. During purification, the count is based on the items that remain after the flagged items are removed, and an examinee excluded once stays excluded. Because the count is based on the remaining items, a large `min.resp` can leave a group without any ability estimate in a later iteration, and the purification then stops with a warning. If `min.resp = NULL`, a score is computed for any examinee with at least one valid item response.

The print method of this class has an argument `digits` that is not used; the statistics are printed rounded to three decimal places.

Value

This function returns an object of class `"rdif"`, which is a list with the following five 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 with one row per item and the columns `id`, `rdifr`, `z.rdifr`, `rdifs`, `z.rdifs`, `rdifrs`, `p.rdifr`, `p.rdifs`, `p.rdifrs`, `n.ref`, `n.foc`, and `n.total`: the item ID, the $RDIF_R$ statistic and its standardized value, the $RDIF_S$ statistic and its standardized value, the $RDIF_{RS}$ statistic, the

p-values of the three statistics (two-sided for $RDIF_R$ and $RDIF_S$), the numbers of examinees in the reference and focal groups who responded to the item, and their sum. Statistics and p-values are rounded to four decimal places. $RDIF_{RS}$ has no standardized value because it is a χ^2 -based statistic.

moments A data frame with the columns `id`, `mu.rdifr`, `sigma.rdifr`, `mu.rdifs`, `sigma.rdifs`, and `covariance`: the item ID, the null mean and standard deviation of $RDIF_R$, the null mean and standard deviation of $RDIF_S$, and the null covariance between $RDIF_R$ and $RDIF_S$. These moments are computed analytically under the null hypothesis of no DIF and are used to standardize the statistics.

dif_item A list with the elements `rdifr`, `rdifs`, and `rdifrs`, each giving the positions (rows of x) of the items flagged by the corresponding statistic, or NULL if no item is flagged.

score A numeric vector of the ability estimates used in the initial analysis: the supplied score, or the internal estimates when `score = NULL`.

purify A logical value indicating whether purification was requested (the `purify` argument).

with_purify A list with the results of the purification procedure. All elements are NULL when `purify = FALSE`. The elements are:

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 with the same columns as `no_purify$dif_stat` and an additional column `n.iter`. For an item removed during purification, the row reports the statistics from the iteration in which the item was removed, and `n.iter` is that iteration minus 1 (0 for an item removed on the basis of the initial analysis). For the other items, the row reports the statistics from the last iteration, and `n.iter` is the number of that iteration. If no item is flagged in the initial analysis, this is `no_purify$dif_stat` with `n.iter = 0`.

moments A data frame with the same columns as `no_purify$moments` and an additional column `n.iter`, defined in the same way as in `dif_stat`.

dif_item A numeric vector of the positions (rows of x) of the items flagged by the `purify.by` statistic, sorted in ascending order. It contains the items removed during purification and, if `max.iter` is reached or the purification stops because a group has no ability estimate left, the items flagged in the last iteration (iteration `n.iter`). It contains all items if every item is flagged. NULL if no item is flagged in the initial analysis.

n.iter The number of purification iterations performed (0 if no item is flagged in the initial analysis). The iteration that removes the last item, or after which a group has no ability estimate left, is not counted, because the statistics are not recomputed.

score A numeric vector of the ability estimates from the last iteration counted in `n.iter`. NULL if no item is flagged in the initial analysis, because purification is not carried out; the initial estimates are then in `no_purify$score`.

	complete	A logical value. TRUE if the procedure stopped because no remaining item was flagged (including the case in which no item is flagged in the initial analysis), and FALSE if it stopped because <code>max.iter</code> was reached, because every item was flagged, or because a group had no examinee with an ability estimate left.
alpha		A numeric value indicating the significance level (α) used for the tests.
call		The matched function call.

Methods (by class)

- `rdif(default)`: Default method for a data frame `x` containing item metadata.
- `rdif(est_irt)`: Method for an object of class `est_irt` created by `est_irt()`. The response data and the scaling factor `D` are taken from the object.
- `rdif(est_item)`: Method for an object of class `est_item` created by `est_item()`. The response data, ability estimates, and the scaling factor `D` are taken from the object.

Methods (by generic)

- `print(rdif)`: Print method for an object of class `rdif` (given as `x`). It prints the flagged items and the RDIF statistics of the analyses without and with purification, rounded to three decimal places.

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.
- Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. Newbury Park, CA: Sage.
- 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.
- Moore, D. S. (1977). Generalized inverses, Wald's method, and the construction of chi-squared tests of fit. *Journal of the American Statistical Association*, 72(357), 131-137. doi:10.1080/01621459.1977.10479921.
- 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()`, `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 = rep(0.15, 4)),
    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
# Therefore, 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) 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)

# purify.by = "rdifr" or "rdifs" can be used in the same way.

```

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 item metadata of the item bank (e.g., item parameters, number of categories, and IRT models). Each row is one item, and the rows correspond to the rows of module. See shape_df() for creating item metadata. |
| D | A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1. |
| route_map | A binary square matrix defining the MST structure. A value of 1 at row i and column j means that test takers can be routed from module i to module j . This is the same convention as the <code>transMatrix</code> argument of <code>randomMST()</code> in the mstR package (Magis et al., 2017). The stages are found from the transitions (see panel_info()), and the first stage must contain a single routing module. |
| module | A binary matrix that assigns the items in <code>x</code> to modules. It has one row per item (in the order of <code>x</code>) and one column per module, and a value of 1 at row j and column m means that item j belongs to module m . This is the same convention as the <code>modules</code> argument of <code>randomMST()</code> in the mstR package (Magis et al., 2017). |
| cut_score | A list with one numeric vector per stage transition (the number of stages minus one). Element s holds the cut scores on the ability (θ) metric for routing from stage s to stage $s + 1$, finite and in strictly ascending order, with one fewer value than the number of modules in stage $s + 1$; cut score k separates the k -th and the $(k + 1)$ -th module of that stage in the order of the module indices. A NULL element is read as an empty vector for a stage with a single module. For example, in a 1-3-3 panel, <code>cut_score = list(c(-0.5, 0.5), c(-0.6, 0.6))</code> . See Details . |

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 <code>theta = seq(-5, 5, 1)</code> .
intpol	Logical. If TRUE, inverse TCC scoring assigns ability estimates to the sum scores that cannot be mapped through the TCC (sum scores less than or equal to the sum of the guessing parameters, and the maximum possible sum score), as in <code>est_score()</code> . The low scores are mapped by linear interpolation (Lim et al., 2021), and the maximum sum score receives the second value of <code>range.tcc</code> . With <code>intpol = FALSE</code> , some sum scores have no ability estimate, and the function stops with an error. Default is TRUE.
range.tcc	A numeric vector of length two giving the ability estimates assigned to the lowest and the maximum possible sum scores when <code>intpol = TRUE</code> (see <code>est_score()</code>). The range must be wide enough to cover the ability estimates of the sum scores, or the function stops with an error. Default is <code>c(-7, 7)</code> .
tol	A positive number giving the tolerance of the bisection search used by inverse TCC scoring; the search stops when the search interval is no wider than <code>tol</code> . Default is <code>1e-4</code> .

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.

The first stage must have a single routing module, whatever its module index, and the function stops with an error when stage 1 has more than one module. `run_mst()` also runs panels with several stage-1 modules, so the two functions differ in the panels they support.

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 defines the possible transitions between modules, following the `transMatrix` argument of `randomMST()` in the **mstR** package (Magis et al., 2017). A value of 1 at row i and column j means that test takers can move from module i directly to module j . `panel_info()` uses this matrix to enumerate all pathways, and `reval_mst()` evaluates the routing along these pathways analytically, without simulation.

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 routing module. Upon completion, the inverse TCC ability estimate of the sum score over all modules taken so far determines the next module: estimates below -0.5 route to the first module of the next stage, estimates from -0.5 up to but not including 0.5 to the second, and estimates of 0.5 or above to the third. When a module can reach only some modules of the next stage, only the cut scores that separate the reachable modules are used. For example, a test taker in a stage-2 module that can reach only the second and third stage-3 modules is routed by the second cut score alone.

Value

A list with the following seven elements:

<code>panel.info</code>	The output of <code>panel_info()</code> : <code>config</code> , <code>pathway</code> , <code>n.module</code> , and <code>n.stage</code> .
<code>item.by.mod</code>	A list of item metadata data frames, one per module, named <code>m.1</code> , <code>m.2</code> , etc. The item metadata are checked and completed as in <code>run_mst()</code> , so these are plain data frames with upper-case model names and without parameter columns that are all NA, whatever the class of <code>x</code> . The same holds for <code>item.by.path</code> .
<code>item.by.path</code>	A list with one element per stage (<code>stage.1</code> , <code>stage.2</code> , ...). Each element is a list of item metadata data frames, one per distinct partial pathway up to that stage (<code>path.1</code> , <code>path.2</code> , ...), holding the items of all modules on the partial pathway.
<code>eq.theta</code>	A list with one element per stage. Each element is a matrix with one row per possible cumulative sum score (0 to the maximum) and one column per partial pathway up to that stage (<code>path.1</code> , <code>path.2</code> , ...), holding the inverse TCC ability estimates. The columns of the last stage correspond to the complete pathways.
<code>cdist.by.mod</code>	A list with one element per value of <code>theta</code> (named by the values). Each element is a matrix with one row per sum score and one column per module (<code>m.1</code> , <code>m.2</code> , ...), holding the conditional sum score distribution of each module; rows beyond a module's maximum score are 0.
<code>jdists.by.path</code>	A list with one element per stage. Each element is a list with one element per value of <code>theta</code> , holding a matrix with one row per cumulative sum score and one column per partial pathway, which gives the joint probability of the sum score and the pathway.
<code>eval.tb</code>	A data frame with the columns <code>theta</code> (true ability), <code>mu</code> and <code>sigma2</code> (conditional mean and variance of the final ability estimate), <code>bias</code> (<code>mu - theta</code>), and <code>csem</code> (square root of <code>sigma2</code>).

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

`run_mst()`, `panel_info()`, `find_cut()`, `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_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,
...
)

## S3 method for class 'ripd'
print(
  x,
  digits = max(2L, getOption("digits") - 5L),
  what = c("all", "no_purify", "with_purify"),
  ...
)

```

Arguments

<code>x</code>	A data frame containing item metadata (e.g., item parameters, number of categories, IRT model types, etc.), or an object of class <code>est_irt</code> or <code>est_item</code> obtained from <code>est_irt()</code> or <code>est_item()</code> . See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function.
<code>...</code>	Additional arguments passed to <code>est_score()</code> when abilities are estimated internally, for example <code>tol</code> or <code>se</code> . The <code>est_score()</code> arguments <code>max.iter</code> and <code>missing</code> cannot be passed this way, because the function uses arguments with these names for its own purposes.
<code>data</code>	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in <code>x</code> . Responses to an item with <code>K</code> score categories must be integers from 0 to <code>K - 1</code> . Missing responses are coded as <code>NA</code> or, in functions that have a <code>missing</code> argument, as the value given in <code>missing</code> .
<code>score</code>	A numeric vector of examinees' ability estimates (theta values) in the same row order as <code>data</code> . If <code>NULL</code> , the ability estimates are computed internally with <code>est_score()</code> using <code>method</code> , <code>range</code> , <code>norm.prior</code> , <code>nquad</code> , <code>weights</code> , and <code>ncore</code> . When <code>purify = TRUE</code> , the ability estimates of all examinees in both groups are re-estimated internally at every purification iteration with these settings, so a supplied <code>score</code> is used only for the initial (non-purified) analysis. A missing value in <code>score</code> excludes the examinee from the analysis, also during purification, with a warning. Default is <code>NULL</code> .
<code>group</code>	A numeric or character vector indicating the group membership of examinees. Its length must equal the number of rows in <code>data</code> . Examinees whose value

	equals focal.name form the focal group; all other examinees are pooled into the reference group (for example, the synthetic reference group described in Details). Examinees with NA in group are excluded from both groups.
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 positions (row numbers of x), or a logical vector with one value per item, identifying the items to exclude from the IPD analysis. If NULL, all items are analyzed. Skipped items still contribute to the ability estimates; only their RIPD statistics and moments are set to NA, and they are never flagged. In CAT, this is typically used to skip low-exposure items (see Details). 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 1.702 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 $RIPD_S$. Used only when <code>purify = TRUE</code> . Default is "ripdrs".
max.iter	A positive integer specifying the maximum number of purification iterations. Default is 10. If the limit is reached while flagged items remain, a warning is issued, the items flagged at the last iteration are added to the flagged set, and the complete element of <code>with_purify</code> is FALSE. Because one item is removed per iteration, <code>max.iter</code> should be at least the number of items expected to drift (Lim & Han, 2026, used 80).
min.resp	A single nonnegative number, usually a positive integer, specifying the minimum number of valid item responses required from a focal group examinee for the examinee's responses to be used. All responses of focal group examinees with fewer than <code>min.resp</code> (but at least one) responses are set to NA before the ability estimation, in the initial analysis and at every purification iteration, and these examinees are excluded from the analysis. A warning reports the number of examinees excluded in the initial analysis, and one warning at the end of the purification reports the examinees excluded during it. A large <code>min.resp</code> can leave the focal group without any ability estimate in a later purification iteration, which stops the purification with a warning. Reference group examinees are not affected. If NULL, no minimum is applied. Default is NULL.
method	A character string indicating the scoring method used to estimate the abilities when <code>score = NULL</code> and in every purification iteration. Available options are: • "ML": Maximum likelihood estimation • "MLF": Maximum likelihood estimation with fences • "WL": Weighted likelihood estimation • "MAP": Maximum a posteriori estimation

	• "EAP": Expected a posteriori estimation The methods "EAP.SUM" and "INV.TCC" of <code>est_score()</code> are not supported, because they do not give an ability estimate for each response pattern. 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 <code>c(-5, 5)</code> .
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 <code>c(0, 1)</code> .
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 <code>gen.weight()</code>. If NULL and method = "EAP", default quadrature values are generated based on the <code>norm.prior</code> and <code>nquad</code> arguments. Ignored if method is "ML", "WL", or "MAP".
ncore	An integer specifying the number of logical CPU cores used by <code>est_score()</code> when abilities are estimated internally. The test statistics themselves are not computed in parallel. Default is 1.
verbose	Logical. If TRUE, the progress of the purification procedure is printed to the console. Default is TRUE.
digits	Not used; kept for consistency with the other print methods.
what	A character string selecting the results to print: "all" (default), "no_purify", or "with_purify".

Details

The current version supports dichotomous items only. The function stops if `x` contains GRM or GPCM items or if data contain response values other than 0 and 1.

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 two key adaptations. First, 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. Second, the reference group is a synthetic, drift-free group simulated from the focal group's ability estimates (see CAT-Specific Workflow). 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. Below, $P_{ij} = P_j(\hat{\theta}_i)$, and N_F and N_R denote the numbers of focal and reference group examinees who responded to item j ; all sums run over these examinees only.

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 $\sigma_{RIPD_R}^2 = \frac{\sum_{i \in F} P_{ij}(1-P_{ij})}{N_F^2} + \frac{\sum_{i \in R} P_{ij}(1-P_{ij})}{N_R^2}$. 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 depends on the predicted probabilities of the examinees in each group who responded to item j . Even with a synthetic reference group, the ability estimates and the administered items of the reference group differ from those of the focal group, so this null mean is generally nonzero and is computed item by item. 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.

In rare cases, the covariance matrix of $RIPD_R$ and $RIPD_S$ is singular or nearly singular (its reciprocal condition number is 1e-10 or less), for example when all examinees who responded to an item have the same probability of a correct response, so that $RIPD_S$ is a linear function of $RIPD_R$. Then $RIPD_{RS}$ is computed with the Moore-Penrose generalized inverse of the covariance matrix and compared with a chi-square distribution whose degrees of freedom equal the rank of the matrix (Moore, 1977), and a warning names the item. This handling is not part of the original method (Lim et al., 2022; Lim & Han, 2026) and is added in irtQ. In most data the matrix is not singular, and $RIPD_{RS}$ has two degrees of freedom.

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 final ability estimates $\hat{\theta}$ (e.g., ML estimates) from this step are retained.
2. **Synthetic reference group construction:** The focal group's $\hat{\theta}$ values are treated as true abilities. A CAT simulation with the same item pool and algorithm settings as in step 1 is run, generating item responses from the *original* (pre-calibrated, drift-free) item parameters. 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 ($N_{\text{ref}} = N_{\text{foc}}$), while kF replicates them k times. Larger reference groups reduce sampling variability in the reference residuals and improve detection power. In the simulation study of Lim and Han (2026), power gains diminished beyond 5F, and a reference group of at least 8 to 10 times the focal group size (8F to 10F) was suggested as a practical guideline. Note that `ripd()` does not simulate the reference group; this step must be carried out with a CAT simulation program outside irtQ, and the simulated responses and ability estimates are then combined with the focal group data.
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. This is analogous to the contaminating effect of DIF items on matching variables. At each iteration:

1. Among the remaining items flagged by the `purify.by` statistic, the item with the smallest p-value (compared on the log scale) is removed and added to the flagged set. For $RIPD_R$ and $RIPD_S$, this is the item with the largest absolute z value.
2. Ability estimates of all examinees in both groups are recomputed with `est_score`, excluding all currently flagged items and using the `method`, `range`, `norm.prior`, `nquad`, `weights`, and `ncore` arguments.
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 `max.iter` is reached. In the latter case, a warning is issued, the items flagged at the last iteration are added to the flagged set, and `complete` is set to `FALSE`. If every item is flagged, the procedure stops with a warning, because no item is left to estimate the abilities, all items are returned as flagged, and `complete` is `FALSE`. The procedure also stops with a warning and `complete = FALSE` when, after an item is removed, no examinee of the reference group or of the focal group has an ability estimate left (see `min.resp`); the remaining items then keep the results of the previous iteration, and the items flagged in that iteration are added to the flagged items. Examinees without any response have no ability estimate and are excluded. During purification, an examinee whose remaining responses are all removed with the flagged items, or who has fewer than `min.resp` responses to the remaining items, is also excluded, and one warning at the end reports the number of such examinees. 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 excludes such items (e.g., low-exposure non-key items) from the IPD tests. Skipped items are still used to compute the ability estimates. 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.

Printing

The `print()` method of a `"ripd"` object shows the flagged items and the RIPD statistics of the analyses without and with purification. Its argument `what` selects `"no_purify"` or `"with_purify"` (default `"all"`). Items without statistics (skipped items and items without responses in a group) are not shown in the tables.

Value

This function returns an object of class `"ripd"`, a list with the following elements:

- | | |
|------------------------|--|
| <code>no_purify</code> | A list of sub-objects containing the results of IPD analysis without applying a purification procedure. The sub-objects include: |
| <code>ipd_stat</code> | A data frame of RIPD results for all items with the columns <code>id</code> , <code>ripdr</code> , <code>z.ripdr</code> (standardized $RIPD_R$), <code>ripds</code> , <code>z.ripds</code> (standardized $RIPD_S$), <code>ripdrs</code> , <code>p.ripdr</code> , <code>p.ripds</code> , <code>p.ripdrs</code> , <code>n.ref</code> , <code>n.foc</code> , and <code>n.total</code> . <code>n.ref</code> and <code>n.foc</code> are the numbers of examinees in each group who responded to |

the item and have an ability estimate. Statistics and p-values are rounded to four decimal places, and items are flagged using the unrounded p-values; they are NA for items in `item.skip`. $RIPD_{RS}$ has no standardized value because it is a χ^2 -based statistic.

moments A data frame of the null means and standard deviations of the RIPD statistics with the columns `id`, `mu.ripdr`, `sigma.ripdr`, `mu.ripds`, `sigma.ripds`, and covariance (the covariance between $RIPD_R$ and $RIPD_S$). Values are NA for items in `item.skip`.

ipd_item A list with the elements `ripdr`, `ripds`, and `ripdrs`, each giving the indices (row positions in `x`) of the items flagged by the corresponding statistic at level `alpha`. An element is NULL when no item is flagged.

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. All elements are NULL when `purify = FALSE`. 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 of the RIPD results for all items with the same columns as in `no_purify` plus `n.iter`, the analysis from which each row is taken. For a flagged item, `n.iter` is the iteration at which it was flagged (0 is the initial analysis); for the other items, it is the final iteration.

moments A data frame of the null moments for all items with the same columns as in `no_purify` plus `n.iter`, taken from the same analysis as the corresponding row of `ipd_stat`.

ipd_item A numeric vector of item indices (row positions in `x`) identified as IPD items across all purification iterations, sorted in ascending order. If `max.iter` is reached before convergence or the purification stops because a group has no ability estimate left, the items flagged at the last iteration (iteration `n.iter`) are also included. It contains all items if every item is flagged. NULL if no item is flagged in the initial analysis.

n.iter An integer indicating the total number of iterations performed during the purification process. The iteration that removes the last item, or after which a group has no ability estimate left, is not counted, because the statistics are not recomputed.

score A numeric vector of purified ability estimates used to compute the final RIPD statistics, from the last iteration counted in `n.iter`. NULL when no item is flagged in the first analysis, because purification is not carried out.

complete A logical value indicating whether the purification process converged. FALSE if the procedure stopped because the maximum number of iterations was reached, because every item was flagged, or because a group had no examinee with an ability estimate left.

`alpha`

A numeric value indicating the significance level (α) used in hypothesis testing for RIPD statistics.

`call`

The matched function call.

Methods (by class)

- `ripd(default)`: Default method for a data frame `x` containing item metadata.
- `ripd(est_irt)`: Method for an object of class `est_irt` created by `est_irt()`. The response data and the scaling factor `D` are taken from the object.
- `ripd(est_item)`: Method for an object of class `est_item` created by `est_item()`. The response data, ability estimates, and the scaling factor `D` are taken from the object.

Methods (by generic)

- `print(ripd)`: Print method for an object of class `ripd`. It prints the flagged items and the RIPD statistics of the analyses without and with purification (see **Printing** in Details).

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.
- Lord, F. M. (1980). Applications of item response theory to practical testing problems. Lawrence Erlbaum Associates.
- Moore, D. S. (1977). Generalized inverses, Wald's method, and the construction of chi-squared tests of fit. *Journal of the American Statistical Association*, 72(357), 131-137. doi:10.1080/01621459.1977.10479921.
- Veerkamp, W. J. J., & Glas, C. A. W. (2000). Detection of known items in adaptive testing with a statistical quality control method. *Journal of Educational and Behavioral Statistics*, 25(4), 373-389. doi:10.3102/10769986025004373.

See Also

`rdif()`, `est_irt()`, `est_item()`, `simdat()`, `shape_df()`, `est_score()`, `pcd2()`, `simIPD`

Examples

```
## --- RIPD Example: Detecting IPD in CAT -----
##
## Background (Lim & Han, 2026):
##   In CAT-based IPD detection using RIPD, the reference group is
##   "synthetic": it is 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
## (the row order does not matter as long as group matches the rows)
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 without purification
## 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 = FALSE,
  method = "ML",
  range  = c(-5, 5)
)

## Step 3. Compare the items flagged by RIPD_RS to the ground truth
detected <- ripd_result$no_purify$ipd_item$ripdrs
cat("Truly drifted items (ground truth):", simIPD$ipd_item, "\n")
cat("RIPD_RS-flagged items:", detected, "\n")
cat("True positives:", sum(detected %in% simIPD$ipd_item), "of",
    length(simIPD$ipd_item), "\n")

## Step 4. Purification (not run: it takes several minutes because the
##          abilities of the examinees are re-estimated at every iteration)
##
## ripd_purified <- ripd(
##   x = simIPD$item_par, data = data, score = score, group = group,
##   focal.name = 1, item.skip = simIPD$item.skip, D = 1.7, alpha = 0.05,
##   purify = TRUE, purify.by = "ripdrs", max.iter = 30,
##   method = "ML", range = c(-5, 5)
## )

```

```
## )
## print(ripd_purified, what = "with_purify")

## -- Note on reference group size -----
## This example uses a 1F reference group (n_ref = n_foc = 3,000).
## In practice, a larger reference group improves detection power;
## Lim and Han (2026) suggest at least 8F to 10F. To create, for example,
## an 8F reference group, replicate the focal ability estimates and run
## the same CAT simulation with the original item parameters using a CAT
## simulation program (irtQ does not provide one):
##
## theta_8F <- rep(simIPD$foc_score, times = 8) # 24,000 examinees
## # simulate a CAT for theta_8F with simIPD$item_par, then call ripd()
## # with the simulated responses and final ability estimates
## -----
```

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). flexMIRT runs on Windows only, so this function works only on Windows. 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.flexmirt".
dir.flex	A character string giving the folder where flexMIRT is installed (e.g., "C:/Program Files/flexMIRT3"). If NULL (default), the function looks in "C:/Program Files" for a folder whose name contains "flexMIRT"; give the folder explicitly when there is no such folder or more than one.
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

The syntax files "2PLM_example.flexmirt" and "3PLM_example.flexmirt" in the `extdata` folder of the package fit the 2PL and 3PL models to the responses of 2,844 examinees to 12 items in "g341-19.dat". The folder also contains the resulting "-prm.txt" and "-irt.txt" files.

Value

Called for its side effect. flexMIRT writes its output files (e.g., "-prm.txt", "-irt.txt") to the folder of each syntax file, and they can be read with `bring.flexmirt()`. The value returned to R is not meant to be used.

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
# Copy an example syntax file and its data file to a temporary folder
ext <- system.file("extdata", package = "irtQ")
file.copy(file.path(ext, c("2PLM_example.flexmirt", "g341-19.dat")),
  tempdir())
file.syntax <- file.path(tempdir(), "2PLM_example.flexmirt")

# Run flexMIRT to estimate item parameters for the 2PL model
run_flexmirt(file.syntax = file.syntax, dir.flex = NULL,
  show.output.on.console = TRUE)

# Read the item parameter file written by flexMIRT
out.file <- file.path(tempdir(), "2PLM_example-prm.txt")
bring.flexmirt(out.file, type = "par")

# (2) Run multiple syntax files
# Copy two example syntax files and the data file to a temporary folder
file.copy(file.path(ext, c("3PLM_example.flexmirt", "g341-19.dat")),
  tempdir())
file.syntax1 <- file.path(tempdir(), "2PLM_example.flexmirt")
file.syntax2 <- file.path(tempdir(), "3PLM_example.flexmirt")

# Run flexMIRT to estimate item parameters for both models
run_flexmirt(file.syntax = c(file.syntax1, file.syntax2), dir.flex = NULL,
  show.output.on.console = FALSE)

# Read the item parameter files written by flexMIRT
out.file1 <- file.path(tempdir(), "2PLM_example-prm.txt")
out.file2 <- file.path(tempdir(), "3PLM_example-prm.txt")
bring.flexmirt(out.file1, type = "par")
bring.flexmirt(out.file2, type = "par")

## End(Not run)
```

Description

Simulates an MST administration for a set 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, with separate `route_score` and `final_score` list arguments and the `route_map` and 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
)

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

Arguments

<code>x</code>	A data frame of item bank metadata following the irtQ format (columns: <code>id</code> , <code>cats</code> , <code>model</code> , <code>par.1</code> , <code>par.2</code> , ...). Use shape_df to create this frame or bring.flexmirt / bring.bilog / bring.parscale / bring.mirt to import from IRT software output. The item metadata are checked and completed internally before the simulation.
<code>route_map</code>	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 <code>transMatrix</code> argument in <code>randomMST()</code> from mstR (Magis et al., 2017). See reval_mst for details.
<code>module</code>	A binary matrix mapping items (rows) to modules (columns). <code>module[j, m] = 1</code> means item j belongs to module m . Equivalent to the <code>modules</code> argument in <code>randomMST()</code> from mstR (Magis et al., 2017). Must have <code>nrow(module) == nrow(x)</code> and <code>ncol(module) == total number of modules in the panel</code> .

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 $N \times J$ matrix of observed item responses, where N is the number of examinees and J is the total number of items in the item bank (<code>nrow(x)</code>). Only the responses to the items of the administered modules are used, so the entries for the other items are ignored and may be NA. A missing response to an administered item (NA or the value given in <code>missing</code>) is excluded from the likelihood for "ML", "WL", "MLF", "MAP", and "EAP", and counted as 0 in the sum score for "EAP.SUM" and "INV.TCC". If NULL and theta is provided, responses are simulated internally via <code>simdat</code> . 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 <code>simdat(x, theta, D)</code>) is useful when you want to compare different routing methods or scoring options on the <i>same</i> set of item responses.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 is commonly used for this purpose. Default is 1.
ini_mod	Either NULL (default) or a single positive integer index into the stage-1 module list (<code>panel_info(route_map)\$config[[1]]</code>). When NULL, each examinee is independently and randomly assigned to one of the stage-1 modules with equal probability. This is appropriate for panels that use spiraling or booklet-based stage-1 assignment, and it is equivalent to a fixed assignment for panels with a single stage-1 module. When an integer is supplied, all examinees begin with that stage-1 module; use <code>ini_mod = 1</code> 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 <i>b</i> 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 <code>cut_score</code> 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 <code>route_method = NULL</code> . The list should have <code>n.stage - 1</code> elements, one per stage transition. Element <i>s</i> contains the cut scores for routing from stage <i>s</i> to stage <i>s</i> +1. For example, in a 1-3-3 MST, <code>cut_score = list(c(-0.5, 0.5), c(-0.6, 0.6))</code> 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 (the modules numbered from easiest to hardest). Each element must contain finite values in strictly ascending order and have one fewer value than the number of modules in stage $s+1$; cut score k separates the k -th and the $(k+1)$ -th module of that stage in the order of the module indices. A NULL element is read as an empty vector for a stage with a single module. The output element `cut_score` of `find_cut` can be passed directly. 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; see `est_score`). For "EAP.SUM" and "INV.TCC", a sum-score-to-theta lookup table is pre-computed 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 of the ability estimates for "ML", "WL", "MLF", and "MAP"; also the default fence locations for "MLF". Default: `c(-5, 5)`.

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

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

`tol` A single positive number: convergence tolerance of the Fisher scoring iterations for "ML", "WL", "MLF", and "MAP", and of the bisection search for "INV.TCC". Default: `1e-4`.

`max.iter` A single positive whole number: maximum number of Fisher scoring iterations for "ML", "WL", "MLF", and "MAP". 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: ability estimates assigned to the lowest and the maximum possible sum scores for "INV.TCC" when `intpol` = TRUE (see `est_score`). Default: `c(-7, 7)`.

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

As in `est_score`, the function stops with an error when `tol` or `max.iter` is not valid for the chosen method. 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 <code>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)</code> .
se	Logical. Whether to return the standard error of the final ability estimate. The standard errors are computed as in est_score : from the test information for "ML", "WL", "MLF", and "MAP", as the posterior standard deviation for "EAP" and "EAP.SUM", and, for "INV.TCC", as the standard deviation of the inverse TCC estimates over the conditional sum score distribution at the estimate. For "ML", "WL", "MLF", and "MAP", the standard error is set to 99.9999 when the estimate equals a bound of range. The standard error is NA when <code>se = FALSE</code> , when all responses are missing, or, for "INV.TCC", when the sum score has no ability estimate (for example, with <code>intpol = FALSE</code>). Default is TRUE.
missing	A scalar specifying the value that marks a missing response in response. It is used only when response is supplied. 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 <code>full.resp</code> element of the returned list is an $N \times J$ integer matrix, where N is the number of examinees and $J = \text{nrow}(x)$ (all items in the item bank). Cell <code>[i, j]</code> 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 <code>x\$id</code> . If FALSE (default), <code>full.resp</code> is NULL. Because this matrix can be large for large panels and many examinees, it is not populated by default.
digits	The number of decimal places used for the summary of the ability estimates and for the bias and RMSE. Default is 3.
...	Additional arguments, currently not used.

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 spiraling 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. The same rule applies to the routing estimates. When all responses up to a non-final stage are missing, the routing estimate is 0 after stage 1 and the previous routing estimate after a later stage. An examinee with no observed response to the administered items receives NA as the final estimate, and one warning reports the number of such examinees.

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 errors of the final ability estimates (see the se argument).

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). Column names are "stage.1", "stage.2", etc.

path An *N* x *n.stage* integer matrix of administered module indices. Column names are "stage.1", "stage.2", etc.

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 $N \times J$ integer matrix of all item responses, where N is the number of examinees and $J = \text{nrow}(x)$ (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.

Methods (by generic)

- `print(run_mst)`: Print method for an object of class `run_mst` (given as `x`). It prints the panel structure, the routing and scoring methods, a summary of the final ability estimates, the bias and RMSE when the true abilities are available, and the module frequencies by stage.

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() derives the cut scores from the TIF crossings of adjacent
## modules; cut-score routing with these values avoids the path
## reversals that MFI routing can produce.
## -----
# 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)

## Visualize 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 to theta
# lookup table for every partial pathway (routing) and every complete
# pathway (final scoring). This example uses bmat routing; the design
# that reval_mst() evaluates analytically uses cut-score routing
# (route_method = NULL) with INV.TCC routing and final scores.
# 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 and scoring vs. ML routing and scoring
# (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	<i>Score Selected-Response Item Data Against an Answer Key</i>
------------	--

Description

This function converts raw selected-response (e.g., multiple-choice) item data into a dichotomously scored item response matrix (0 = incorrect, 1 = correct) using a supplied answer key. Each response is coded as the selected option (an option number, e.g., 1-5 for a five-option item; a Latin-letter option label, e.g., "A"- "E"; or any other option label, such as a Korean syllable label), as a missing-response indicator for an omitted response, or as a comma-separated string (e.g., "1,5" or "A,C") for a double-marked (multiple-option) response. 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 a single selected option in the coding scheme of that item's key value (an option number such as 2, a Latin-letter label such as "C", or any other option label such as a Korean syllable label), 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 an option number (e.g., 4), a Latin-letter option label (e.g., "D"), or any other non-blank option label (e.g., a Korean syllable label), or (b) a data frame with columns item and key giving the item number and its correct option, respectively. Each item's coding scheme is inferred independently from its own key value, so items using different option formats may be mixed within the same vector or data frame; letter (and other case-bearing) values are

matched case-insensitively. When a data frame is supplied, its `item` column (numeric, character, or factor) must contain each of the integers 1 to `ncol(data)` exactly once; the rows are sorted by item number before use, so their order need not match the column order of data. A key value that contains a comma names more than one option, and `score_resp()` stops with an error because every item must have exactly one correct option.

`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`, after trimming surrounding spaces, 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 responses (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 is written with digits only (e.g., 4, optionally with a decimal part), 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) exact string match, which leaves caseless scripts such as Hangul unchanged. 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` after trimming surrounding spaces (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. `n_correct + n_wrong` always equals `n`. `n_blank`, `n_double`, and `n_invalid` count disjoint subsets of `n_wrong`; the remaining wrong responses are single valid but incorrect options, which are not counted separately.

Value

A list with two elements:

<code>scored</code>	A data frame with the same dimensions as <code>data</code> , containing the dichotomously scored (0/1) item responses. Column names follow <code>colnames(data)</code> (or <code>V1</code> , <code>V2</code> , ... when <code>data</code> has no column names).
<code>resp_summary</code>	A data frame with one row per item and the columns <code>item</code> (item label), <code>key</code> (the correct option, as trimmed text), <code>n</code> (number of examinees), <code>n_correct</code> , <code>n_wrong</code> , <code>n_blank</code> , <code>n_double</code> , <code>n_invalid</code> (number of unrecognized response tokens), <code>pct_blank</code> , and <code>pct_double</code> (percentages of blank and double-marked responses, rounded to two decimal places).

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 such as `ctt()` or `ctt_distr()` instead of using `score_resp()`.

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 (item parameters, numbers of score categories, and IRT models) for use in IRT analyses with 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), each with one value per item with cats = 2. A single value of g is used for every dichotomous item. If g is omitted or NULL, the guessing parameters are set to 0. |
| par.prm | A list containing polytomous item parameters. The list must include a numeric vector a of slope parameters and a list d of numeric vectors of threshold parameters, with one slope and one vector of cats - 1 thresholds per item with cats > 2. 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 is set to NA. If `cats` or `model` has length 1, it is recycled 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 the threshold parameters for each polytomous item

For GPCM items, the threshold parameters are $b_v = \beta - \tau_v$, the overall item location minus the threshold of each score category. An item with K score categories requires $K - 1$ threshold parameters, because the term for the lowest score category is fixed at 0 and is not supplied. For GRM items, the thresholds must be in increasing order.

Value

A data frame of item metadata with columns `id`, `cats`, `model`, and `par.1`, `par.2`, ..., one row per item. The number of parameter columns is the larger of 3 and the maximum of `cats`; unused cells are NA. 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(-1.9, 0.0, 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 item metadata frame with default parameters for five dichotomous 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

<code>x</code>	A data.frame of metadata for items whose parameters remain fixed (e.g., output from <code>shape_df()</code>).
<code>fix.loc</code>	An integer vector specifying the row positions in the final output where the fixed items are placed, in the row order of <code>x</code> . This argument must be specified.
<code>item.id</code>	A character vector of IDs for the new items whose parameters are estimated. If NULL, default IDs ("V1", "V2", ...) are assigned.
<code>cats</code>	An integer vector giving the number of score categories of each new item, in the order of <code>item.id</code> . A single value is recycled across all new items.
<code>model</code>	A character vector of IRT model names for the new items: "1PLM", "2PLM", "3PLM", or "DRM" for dichotomous items, and "GRM" or "GPCM" for polytomous items. A single value is recycled across all new items.

Details

First, prepare a metadata frame `x` that contains only the fixed items, created by `shape_df()` or imported from external software (e.g., with `bring.flexmirt()`). It must include the columns `id`, `cats`, `model`, and the parameter columns (`par.1`, `par.2`, ...). Then use `fix.loc` to give the row positions of these items in the final test form. The length of `fix.loc` must equal the number of rows in `x`, and the *i*-th element of `fix.loc` is the position of the *i*-th row of `x`.

Next, provide `item.id`, `cats`, and `model` for the new items, whose number equals the total form length minus the length of `fix.loc`. A single value of `cats` or `model` is used for all new items. If `item.id` is NULL, default IDs ("V1", "V2", ...) are assigned.

Value

A data frame of item metadata for all items (fixed and new), ordered by test position, with columns `id`, `cats`, `model`, and `par.1`, `par.2`, ... New items receive default parameters as in `shape_df()` with `default.par = TRUE`. Fixed "DRM" items are relabeled "3PLM".

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 of item metadata for 100 dichotomous items (2PLM; par. 3 is NA). The `id` column is a factor (V1 to V100).

- 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 numeric matrix of item responses (0 or 1) from 10,000 examinees (rows) to 100 items (columns). Each examinee answered 40 items; 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 of item metadata (columns `id` to `par.4`; `id` is a factor) 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 numeric matrix of item responses from 30,000 examinees (rows) to 230 items (columns). Each examinee answered 45 items (40 dichotomous and 5 polytomous items). 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

Simulate 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 (e.g., number of score categories, IRT model, and item parameters). It can be created with shape_df() . See 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 single value is used for all dichotomous items. |
| 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 IRT model of each polytomous item, in the order the polytomous items appear in cats. Each element is either "GRM" (graded response model) or "GPCM" (generalized partial credit model). A single value is used for all polytomous items.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 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, the sixth column is ignored and can be NA.

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 of a GPCM item are $b_v = \beta - \tau_v$, the overall item location minus the threshold of each score category. A GPCM item with K unique score categories requires $K - 1$ threshold parameters, because the term for the lowest score category is fixed at 0 and is not supplied. For example, a GPCM item with five score categories requires four threshold parameters.

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 specify the item parameters directly instead of `x` (see Examples). In this case, `theta` and `cats` are required, along with `a.drm` and `b.drm` for dichotomous items and `a.prm`, `d.prm`, and `pr.model` for polytomous items.

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 dichotomous 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 numeric matrix of simulated item responses with one row per examinee (`theta` value) and one column per item, in the item order of `x` or `cats`. Responses range from 0 to `K - 1` for an item with `K` score categories. The matrix has no row or column names.

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 = g.drm,
  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)

```

Description

A simulated dataset designed to illustrate the use of the `ripd` and `pcd2` functions 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 in which the drift affected all focal group examinees (the 100% drift exposure condition of Lim & Han, 2026).

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 meta-data (columns: `id`, `cats`, `model`, `par.1`, `par.2`, `par.3`). All items follow the 3PLM.

key_item A numeric vector of length 90 giving the row indices (in `item_par`) of the *key items*, that is, the items identified by a preliminary CAT simulation as the most frequently administered 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 the IPD analysis (i.e., the complement of `key_item` in `1:360`). Pass this vector to the `item.skip` argument of `ripd()` or `pcd2()`.

ipd_item A numeric 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 A numeric 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` (the column names are the item IDs). Because CAT administers only 30 items per examinee, approximately 92% of entries are NA. Responses were generated using the *drifted* item parameters (the drifted parameters were used for every focal group examinee who received an IPD item).

foc_score A numeric vector of length 3000 containing the **focal group** final maximum likelihood (ML) ability estimates obtained from the CAT, bounded to the range -5 to 5.

ref_resp A numeric matrix of dimensions 3000 x 360 containing the **synthetic reference group** CAT response data (each examinee also has 30 observed responses and about 92% of the entries are NA). The reference group was constructed by: (1) using `foc_score` as true ability values (1F condition, 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 ability estimates, bounded to the range -5 to 5.

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 the fade-away exposure control method (maximum exposure rate 0.30) and initial abilities drawn from $U(-1, 1)$
- 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 factor: $D = 1.7$
- 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 improve RIPD detection power; Lim and Han (2026) suggest a reference group at least 8 to 10 times the focal group size (8F to 10F). A larger reference group can be created by replicating the focal ability estimates, e.g., `rep(foc_score, times = 8)` for an 8F group, and re-running the CAT simulation with the original item parameters using a CAT simulation program.

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**

A simulated data set for three groups that take three test forms linked by common items, for illustrating multiple-group calibration.

Usage

```
simMG
```

Format

A list with three elements:

item.prm A list of three data frames of item metadata, named Group1, Group2, and Group3. The forms of Groups 1 and 2 have 50 items (47 3PLM items and three GRM items with five score categories), and the form of Group 3 has 38 items (37 3PLM items and one GRM item). Forms 1 and 2 share 12 common items (C1I1 to C1I12), forms 2 and 3 share 10 common items (C2I1 to C2I10), and forms 1 and 3 share no item. The item parameters are the true values used to simulate the responses, on the logistic metric ($D = 1$).

res.dat A list of three item response matrices, named Group1 to Group3, with 2,000 rows (examinees) and 50, 50, and 38 columns (items in the order of `item.prm`). The abilities were drawn from $N(0, 1)$ (Group 1), $N(0.5, 0.8^2)$ (Group 2), and $N(-0.3, 1.3^2)$ (Group 3).

group.name A character vector of the group names, "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 integer matrix (56 x 7) 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 integer matrix (7 x 7) that defines the MST transition structure, showing module pathways across stages. Module 1 routes to modules 2 to 4, module 2 to modules 5 and 6, module 3 to modules 5 to 7, and module 4 to modules 6 and 7. This corresponds to the `transMatrix` argument in the `randomMST()` function in the **mstR** package.

cut_score A list of two numeric vectors, `stage.2` and `stage.3`, giving the routing cut scores into stages 2 and 3. They are the TIF-crossing points in the `cut_score` element returned by `find_cut(x = simMST$item_bank, module = simMST$module, route_map = simMST$route_map, D = 1.702)`.

theta A numeric vector of 81 ability values from -4 to 4 in steps of 0.1, used to evaluate the measurement precision of the panel 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	<i>Summary of Item Calibration Results</i>
---------	--

Description

Summarizes the item calibration results in an object of class `est_irt`, `est_mg`, or `est_item`, returned by `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, ...)

## S3 method for class 'est_irt'
print(x, digits = max(2L, getOption("digits") - 5L), ...)

## S3 method for class 'est_mg'
print(x, digits = max(2L, getOption("digits") - 5L), ...)

## S3 method for class 'est_item'
print(x, digits = max(2L, getOption("digits") - 5L), ...)

## S3 method for class 'summary.est_irt'
print(x, digits = max(2L, getOption("digits") - 5L), ...)
```

```
## S3 method for class 'summary.est_mg'
print(x, digits = max(2L, getOption("digits") - 5L), ...)

## S3 method for class 'summary.est_item'
print(x, digits = max(2L, getOption("digits") - 5L), ...)
```

Arguments

object	An object of class <code>est_irt</code> , <code>est_mg</code> , or <code>est_item</code> .
...	Additional arguments passed to or from other methods (currently not used).
x	An object of class <code>est_irt</code> , <code>est_mg</code> , or <code>est_item</code> , or the summary of one of them, to be printed.
digits	For a summary object, the number of decimal places used to round the tables of item and group parameter estimates. Not used when an <code>est_irt</code> , <code>est_mg</code> , or <code>est_item</code> object itself is printed.

Details

irtQ exports its own generic `summary()`, which calls the method of the class of object and therefore works as `base::summary()` for other objects. Printing a summary object rounds the tables of item and group parameter estimates to the number of decimal places given in the `digits` argument of the print method (default `max(2L, getOption("digits") - 5L)`).

Value

An object of class `summary.est_irt`, `summary.est_mg`, or `summary.est_item`: a list of selected components of object (the call, the numbers of items and examinees, the estimation settings and convergence messages, the log-likelihood, AIC and BIC (not for `est_item`), and the item and group parameter estimates). Printing the object displays an overview of the calibration results.

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()`.

Functions

- `print(est_irt)`: Print method for an object of class `est_irt` (given as `x`). It prints the number of EM cycles, the number of quadrature points, the two convergence checks, a note on the computation of the variance-covariance matrix, and the log-likelihood.
- `print(est_mg)`: Print method for an object of class `est_mg` (given as `x`). It prints the number of EM cycles, the number of quadrature points, the two convergence checks, a note on the computation of the variance-covariance matrix, and the log-likelihood.
- `print(est_item)`: Print method for an object of class `est_item` (given as `x`). It prints the convergence message of the fixed ability parameter calibration and the log-likelihood.

- `print(summary.est_irt)`: Print method for the summary of an `est_irt` object (given as `x`). It prints the full calibration summary with the tables rounded to digits decimal places.
- `print(summary.est_mg)`: Print method for the summary of an `est_mg` object (given as `x`). It prints the full calibration summary with the tables rounded to digits decimal places.
- `print(summary.est_item)`: Print method for the summary of an `est_item` object (given as `x`). It prints the full calibration summary with the tables rounded to digits decimal places.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

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

Examples

```
# Fit the 1PL model to the LSAT6 data, constraining the slopes to be equal
fit.1pl <- est_irt(data = LSAT6, D = 1, model = "1PLM", cats = 2,
                  fix.a.1pl = FALSE, verbose = 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

<code>object</code>	An object of class "ctt", as returned by <code>ctt()</code> .
<code>...</code>	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 <code>est_irt</code> or <code>est_item</code> obtained from <code>est_irt()</code> or <code>est_item()</code> . See <code>est_irt()</code> or <code>simdat()</code> for more details about the item metadata. This data frame can be easily created using the <code>shape_df()</code> function.
...	Additional arguments passed to or from other methods.
data	A matrix of examinees' item responses, with rows for examinees and columns for items in the same order as the items in x. Responses to an item with K score categories must be integers from 0 to K - 1. Missing responses must be coded as NA; they are replaced with 0 before the statistics are computed, with a warning.
D	A scaling constant used in IRT models to make the logistic function closely approximate the normal ogive function. A value of 1.702 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	A numeric value giving the minimum expected frequency per cell. Cells with smaller expected frequencies are merged with adjacent cells. Default is 1. See Details .
norm.prior	A numeric vector of length two giving the mean and standard deviation of the normal latent ability distribution used to generate the quadrature points and weights. Ignored when weights is supplied. For an <code>est_irt</code> object, see weights. Default is <code>c(0, 1)</code> .
nquad	An integer specifying the number of Gaussian quadrature points used to approximate the normal latent ability distribution. Ignored when weights is supplied. For an <code>est_irt</code> object, see weights. 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 <code>gen.weight()</code> according to the <code>norm.prior</code> and <code>nquad</code> arguments. For an <code>est_irt</code> object, when weights, <code>norm.prior</code>, and <code>nquad</code> are all omitted, the latent distribution stored in the object (<code>x\$weights</code>), which was used or estimated in the calibration, is used. Otherwise, weights is used if given, and a normal distribution given by <code>norm.prior</code> and <code>nquad</code> is used if not. For an <code>est_item</code> object, which stores no latent distribution, the normal distribution is used when weights is omitted.
pcm.loc	An optional integer vector giving the row indices of partial credit model (PCM) items whose slope parameters are fixed. It is used only to count the item parameters for the degrees of freedom. Default is <code>NULL</code> .

Details

For item i and summed score group s with N_s examinees, let O_{isk} and E_{isk} be the observed and expected proportions of score category k . The statistic is $S-X_i^2 = \sum_s \sum_k N_s (O_{isk} - E_{isk})^2 / E_{isk}$, which has the form of Orlando and Thissen (2000) for a dichotomous item. The expected proportions are $E_{isk} = \sum_q P_{ik}(\theta_q) f^{*i}(s - k | \theta_q) A(\theta_q) / \sum_q f(s | \theta_q) A(\theta_q)$, where f and f^{*i} are the summed score likelihoods with and without item i , computed by the Lord-Wingersky recursion, and θ_q and $A(\theta_q)$ are the quadrature points and weights.

The latent ability distribution used to compute the expected proportions is given by weights, or by a normal distribution generated from `norm.prior` and `nquad` when weights is omitted. For an `est_irt` object, the latent distribution stored in the object is used when weights, `norm.prior`, and `nquad` are all omitted.

The lowest and highest possible summed scores are excluded. For an item with K score categories, the $K - 1$ lowest remaining summed scores are pooled into one group, and so are the $K - 1$ highest. When the test is so short that the two pooled ranges overlap, each summed score is counted once, in the lower group.

The accuracy of the χ^2 approximation can be compromised when expected cell frequencies are too small (Orlando & Thissen, 2000). For dichotomous items, Orlando and Thissen (2000) merged adjacent summed score groups so that every expected frequency is at least 1. For polytomous items, this approach can discard too much information, so Kang and Chen (2008) instead merged adjacent

score categories *within* each summed score group. `sx2_fit()` follows both strategies, working from the ends of the table toward the middle, and `min.collapse` sets the minimum expected frequency.

The degrees of freedom are $G(K - 1) - m - C$, where G is the number of summed score groups after collapsing, m is the number of item parameters, and C is the number of category cells removed by merging. The number of item parameters m is determined by the IRT model of each item: 1 for the 1PLM, 2 for the 2PLM, 3 for the 3PLM, and the number of score categories K for the GRM and GPCM; an item listed in `pcm.loc` has $m = K - 1$, and an item labeled "DRM" is counted as a 3PLM item. This also applies to `est_irt` and `est_item` objects: parameters fixed during estimation (for example, with `fix.g = TRUE`, or the items fixed in FIPC) are still counted, and GPCM items estimated with `fix.a.gpcm = TRUE` are counted as PCM items only when they are given in `pcm.loc`. When no degrees of freedom remain after collapsing (df of 0 or less), `crit.val` and `p` are NA, and a warning names the item.

The test must contain at least two items. Responses in data must be integer scores from 0 to the number of score categories minus 1, with one column per item. Character and factor responses are read as numbers, logical responses as 0 and 1, and other responses stop the function with an error. Missing responses in data are replaced with 0 (the lowest score category), and a warning is issued.

Value

A list with the following components:

<code>fit_stat</code>	A data frame with one row per item and the columns <code>id</code> , <code>chisq</code> (the $S\text{-}X^2$ statistic), <code>df</code> , <code>crit.val</code> (the critical value at alpha), and <code>p</code> (the p-value).
<code>item_df</code>	The item metadata used in the analysis.
<code>exp_freq</code>	A list of collapsed expected frequency tables, one per item, with summed score groups in rows and score categories in columns.
<code>obs_freq</code>	A list of collapsed observed frequency tables in the same layout as <code>exp_freq</code> .
<code>exp_prob</code>	A list of matrices of expected proportions within each summed score group.
<code>obs_prop</code>	A list of matrices of observed proportions within each summed score group.

For a polytomous item, categories merged within a summed score group are stored in the leftmost columns of that row, and the remaining cells are NA.

Methods (by class)

- `sx2_fit(default)`: Default method for computing $S\text{-}X^2$ fit statistics from a data frame `x` containing item metadata.
- `sx2_fit(est_item)`: Method for an object of class `est_item` created by `est_item()`. The item parameter estimates, response data, and `D` are taken from the object.
- `sx2_fit(est_irt)`: Method for an object of class `est_irt` created by `est_irt()`. The item parameter estimates, response data, and `D` are taken from the object. When `weights`, `norm.prior`, and `nquad` are all omitted, the latent distribution stored in the object (`x$weights`) is used.

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 <code>est_irt</code> obtained from est_irt() , or <code>est_item</code> 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. An item with <code>cats = 2</code> under the "GRM" or "GPCM" model has one threshold in <code>par.2</code> , must have NA in <code>par.3</code> , and is computed as a 2PLM item.
...	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 1.702 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 `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, and a `D` supplied through `...` is ignored. 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:

<code>prob.cats</code>	A list of matrices of category response probabilities, one per item and named by item ID. Each matrix has one row per theta value and one column per score category of the item ("resp.0", "resp.1", ...).
<code>icc</code>	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.
<code>tcc</code>	A numeric vector representing the TCC, computed as the sum of expected item scores across all items at each theta value.
<code>theta</code>	A numeric vector of theta values at which the item and test characteristic 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 supports unidimensional IRT models only. 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

- | | |
|-------------------|---|
| <code>x</code> | 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() . |
| <code>file</code> | A character string giving the path of the file to be written. It must be supplied. flexMIRT parameter files usually end in "-prm.txt". |

norm.pop	A numeric vector of length two giving the mean and the variance of the normal ability distribution of a single group, or a list of such vectors, one per group (e.g., <code>norm.pop = list(c(0, 1), c(0, 0.64), c(0.5, 1.44))</code> for three groups). The second value is written to the variance field of the file. If <code>mgroup = TRUE</code> and a single vector is given, it is used for all groups. Default is <code>c(0, 1)</code> .
rePar	Logical. Indicates how the parameters in <code>x</code> are given. If <code>TRUE</code> (default), <code>x</code> is taken to hold flexMIRT slope-intercept values: slopes, intercepts, and logits of the guessing parameters, and for GPCM items the scoring function and intercept values of the nominal model; they are written as they are. If <code>FALSE</code> , <code>x</code> is taken to hold irtQ item metadata (slopes, difficulty or step parameters, and guessing parameters), which is converted to the flexMIRT parameterization. Item metadata from irtQ (for example, from <code>shape_df()</code> or <code>est_irt()</code>) must be written with <code>rePar = FALSE</code> . Note that the meaning is the reverse of the <code>rePar</code> argument of <code>bring.flexmirt()</code> , where <code>TRUE</code> converts the values of the file to irtQ metadata.
mgroup	A logical value indicating whether the file includes multiple groups. Default is <code>FALSE</code> .
group.name	A character vector of group names. If <code>NULL</code> , group names are automatically generated (e.g., "Group1", "Group2", ...).

Details

The function writes items of the models "1PLM", "2PLM", "3PLM", "DRM", "GRM", and "GPCM". Dichotomous items are written as flexMIRT 3PL ("3PLM" and "DRM") or graded ("1PLM" and "2PLM") items, so `bring.flexmirt()` reads them back as "3PLM" or "2PLM" items. Every item and every group is written as one line, whatever the width of the console. The parameters are written without rescaling. flexMIRT uses the logistic metric ($D = 1$); when `x` was calibrated with another scaling constant, multiply the slopes by that constant before writing.

Value

Called for its side effect of writing the file to `file`. The value is returned invisibly and is not meant to be used.

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)

# 1-(4) Read the file back and compare the item parameters
x_back <- bring.flexmirt(file = temp_prm, type = "par")$Group1$full_df
head(x_back)
max(abs(as.matrix(x[, 4:7]) - as.matrix(x_back[, 4:7])), na.rm = TRUE)

## 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, giving the mean and variance of each group
write.flexmirt(x,
  file = temp_prm, norm.pop = list(c(0, 1), c(0.5, 0.64), c(-0.3, 1.69)),
  rePar = FALSE, mggroup = 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, mggroup = TRUE, group.name = NULL
)
```

Index

* datasets

LSAT6, 143
simCAT_DC, 213
simCAT_MX, 214
simIPD, 218
simMG, 221
simMST, 222

base::summary(), 224
base::system(), 193
bind.fill, 16
bisection, 17
bring.bilog, 196
bring.bilog(bring.flexmirt), 19
bring.bilog(), 20
bring.flexmirt, 19, 196
bring.flexmirt(), 4, 20, 194, 212, 234
bring.mirt, 196
bring.mirt(bring.flexmirt), 19
bring.mirt(), 20
bring.parscale, 196
bring.parscale(bring.flexmirt), 19
bring.parscale(), 20

cac_lee, 22
cac_lee(), 4, 28
cac_rud, 26
cac_rud(), 4, 24
catsib, 29
catsib(), 4, 33, 34
covirt, 39
covirt(), 113, 114
crdif, 42
crdif(), 4, 31, 46, 172
ctt, 51
ctt(), 4, 57–59, 112, 113, 164–166, 207, 225, 226
ctt_distr, 55
ctt_distr(), 4, 57, 207

drm, 60
drm(), 167, 217

est_irt, 61
est_irt(), 6–8, 20, 23, 26, 30, 37, 39, 42, 44, 49, 52, 53, 56, 57, 67, 79, 83, 85, 89, 91, 92, 96, 102, 106, 112, 115–117, 119, 120, 122, 128, 129, 131–133, 135, 139, 141, 144, 148, 164, 169, 175, 176, 184, 191, 206, 207, 210, 211, 223–225, 227, 229–234

est_item, 78
est_item(), 8, 9, 37, 44, 49, 81, 102, 115, 119, 120, 122, 128, 129, 131–133, 135, 139, 169, 175, 176, 184, 191, 223–225, 227, 229–232

est_mg, 84
est_mg(), 4, 91, 115, 117–120, 223–225
est_score, 100, 189, 196, 198–201
est_score(), 18, 24, 26, 28, 30–33, 37, 44–46, 49, 113, 114, 122–124, 129, 142, 170, 171, 176, 179, 181, 184, 186, 191, 206, 207

find_cut, 108, 154, 198, 222
find_cut(), 4, 181
freq_score, 112
freq_score(), 4, 54, 55

gen.weight, 113
gen.weight(), 23, 24, 26, 28, 32, 40, 42, 46, 64, 88, 103, 106, 124, 148, 151, 171, 186, 228

getirt, 115
getirt(), 71, 72, 82, 83, 95, 96
ggplot2::geom_line(), 156, 162
ggplot2::ggplot(), 159
grdif, 120
grdif(), 4, 37, 124, 125, 172

info, 110, 131

- info(), 6, 21, 156, 157, 167
- irtfit, 134
- irtfit(), 4, 158–160, 230
- irtQ (irtQ-package), 3
- irtQ-package, 3, 21, 22, 67, 167, 217
- llike_score, 141
- LSAT6, 143
- lwrc, 143
- lwrc(), 23, 104, 105
- mirt::mirt(), 20
- panel_info, 110, 145, 199–201
- panel_info(), 4, 178, 180, 181
- parallel::makeCluster(), 102
- pcd2, 147, 219, 220
- pcd2(), 4, 149, 191
- plot.find_cut, 110, 153
- plot.info, 155
- plot.info(), 132, 133
- plot.irtfit, 157
- plot.irtfit(), 136, 138, 139
- plot.traceline, 161
- plot.traceline(), 231, 232
- print.catsib (catsib), 29
- print.crdif (crdif), 42
- print.ctt, 164
- print.ctt(), 52, 55, 226
- print.est_irt (summary), 223
- print.est_item (summary), 223
- print.est_mg (summary), 223
- print.find_cut (find_cut), 108
- print.grdif (grdif), 120
- print.irtfit (irtfit), 134
- print.rdif (rdif), 168
- print.ripd (ripd), 182
- print.run_mst (run_mst), 195
- print.summary.ctt, 165
- print.summary.est_irt (summary), 223
- print.summary.est_item (summary), 223
- print.summary.est_mg (summary), 223
- prm, 166
- prm(), 61, 217
- rdif, 168
- rdif(), 4, 31, 37, 46, 47, 49, 124, 126, 127, 129, 172, 191
- reval_mst, 110, 146, 147, 177, 196, 200, 201, 222
- reval_mst(), 4, 179, 180
- ripd, 182, 219, 220
- ripd(), 4, 151
- run_flexmirt, 193
- run_flexmirt(), 4
- run_mst, 108–110, 147, 154, 195, 222
- run_mst(), 4, 179–181
- score_resp, 205
- score_resp(), 4, 53, 56–59, 112
- shape_df, 108, 196, 209
- shape_df(), 23, 26, 30, 37, 39, 42, 44, 52, 56, 62, 67, 72, 79, 83, 96, 102, 106, 122, 129, 131, 133, 135, 139, 141, 144, 148, 170, 176, 178, 181, 184, 191, 212, 215, 217, 227, 230, 231, 233, 234
- shape_df_fipc, 211
- shape_df_fipc(), 7, 67, 72, 96, 210
- simCAT_DC, 213, 220
- simCAT_MX, 214, 220
- simdat, 197, 199, 201, 215
- simdat(), 9, 21, 23, 26, 30, 37, 39, 42, 44, 79, 85, 102, 106, 122, 129, 131, 132, 135, 141, 144, 148, 169, 176, 184, 191, 210, 227, 230, 231, 233
- simIPD, 151, 191, 218
- simMG, 221
- simMST, 201, 222
- stats::dbeta(), 63, 80, 88
- stats::dlnorm(), 63, 80, 88
- stats::dnorm(), 63, 80, 88
- stats::nlminb(), 65, 67, 81, 89
- stats::uniroot(), 105
- summary, 223
- summary.ctt, 225
- summary.ctt(), 52, 55, 165, 166
- sx2_fit, 226
- sx2_fit(), 4, 113, 114, 229
- traceline, 231
- traceline(), 6, 162, 163, 167
- uniroot, 110
- write.flexmirt, 233
- write.flexmirt(), 4