Package 'irtQ'

August 25, 2024

Type Package

Title Unidimensional Item Response Theory Modeling

Version 0.2.1

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

Depends R (>= 4.1)

License GPL (>= 2)

Encoding UTF-8

LazyData true

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

RoxygenNote 7.3.2

NeedsCompilation no

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

Jorge González [ctb],
David Magis [ctb]

Maintainer Hwanggyu Lim <hglim83@gmail.com>

Repository CRAN

Date/Publication 2024-08-25 03:50:02 UTC

Contents

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

Description

Fit unidimensional item response theory (IRT) models to a mixture of dichotomous and polytomous data, calibrate online item parameters (i.e., pretest and operational items), estimate examinees' abilities, and provide useful functions related to unidimensional IRT such as IRT model-data fit evaluation and differential item functioning analysis.

For the item parameter estimation, the marginal maximum likelihood estimation via the expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981) is used. Also, the fixed item parameter calibration (FIPC) method (Kim, 2006) and the fixed ability parameter calibration (FAPC) method, (Ban, Hanson, Wang, Yi, & Harris, 2001; stocking, 1988), often called Stocking's Method A, are provided. For the ability estimation, several popular scoring methods (e.g., ML, EAP, and MAP) are implemented.

In addition, there are many useful functions related to IRT analyses such as evaluating IRT model-data fit, analyzing differential item functioning (DIF), importing item and/or ability parameters from popular IRT software, running flexMIRT (Cai, 2017) through R, generating simulated data, computing the conditional distribution of observed scores using the Lord-Wingersky recursion formula, computing item and test information functions, computing item and test characteristic curve functions, and plotting item and test characteristic curves and item and test information functions.

Package:	irtQ
Version:	0.2.1
Date:	2024-08-23
Depends:	R (>= 4.1)
License:	GPL (>= 2)

Details

Following five sections describe a) how to implement the online item calibration using FIPC, a) how to implement the online item calibration using Method A, b) two illustrations of the online calibration, and c) IRT Models used in **irtQ** package.

Online item calibration with the fixed item parameter calibration method (e.g., Kim, 2006)

The fixed item parameter calibration (FIPC) is one of useful online item calibration methods for computerized adaptive testing (CAT) to put the parameter estimates of pretest items on the same scale of operational item parameter estimates without post hoc linking/scaling (Ban, Hanson, Wang, Yi, & Harris, 2001; Chen & Wang, 2016). In FIPC, the operational item parameters are fixed to estimate the characteristic of the underlying latent variable prior distribution when calibrating the pretest items. More specifically, the underlying latent variable prior distribution of the operational items is estimated during the calibration of the pretest items to put the item parameters of the pretest items on the scale of the operational item parameters (Kim, 2006). In the **irtQ** package, FIPC is implemented with two main steps:

1. Prepare a response data set and the item metadata of the fixed (or operational) items.
2. Implement FIPC to estimate the item parameters of pretest items using the `est_irt` function.

1. Preparing a data set To run the `est_irt` function, it requires two data sets:

1. Item metadata set (i.e., model, score category, and item parameters. see the desciption of the argument `x` in the function `est_irt`).
2. Examinees' response data set for the items. It should be a matrix format where a row and column indicate the examinees and the items, respectively. The order of the columns in the response data set must be exactly the same as the order of rows of the item metadata.

2. Estimating the pretest item parameters When FIPC is implemented in `est_irt` function, the pretest item parameters are estimated by fixing the operational item parameters. To estimate the item parameters, you need to provide the item metadata in the argument `x` and the response data in the argument `data`.

It is worthwhile to explain about how to prepare the item metadata set in the argument `x`. A specific form of a data frame should be used for the argument `x`. The first column should have item IDs, the second column should contain the number of score categories of the items, and the third column should include IRT models. The available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous items, and "GRM" and "GPCM" for polytomous items. Note that "DRM" covers all dichotomous IRT models (i.e, "1PLM", "2PLM", and "3PLM") and "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. From the fourth column, item parameters should be included. For dichotomous items, the fourth, fifth, and sixth columns represent the item discrimination (or slope), item difficulty, and item guessing parameters, respectively. When "1PLM" or "2PLM" is specified for any items in the third column, NAs should be inserted for the item guessing parameters. For polytomous items, the item discrimination (or slope) parameters should be contained in the fourth column and the item threshold (or step) parameters should be included from the fifth to the last columns. When the number of categories differs between items, the empty cells of item parameters should be filled with NAs. See 'est_irt' for more details about the item metadata.

Also, you should specify in the argument `fipc = TRUE` and a specific FIPC method in the argument `fipc.method`. Finally, you should provide a vector of the location of the items to be fixed in the argument `fix.loc`. For more details about implementing FIPC, see the description of the function `est_irt`.

When implementing FIPC, you can estimate both the emprical histogram and the scale of latent variable prior distribution by setting `EmpHist = TRUE`. If `EmpHist = FALSE`, the normal prior distribution is used during the item parameter estimation and the scale of the normal prior distribution is updated during the EM cycle.

The `est_item` function requires a vector of the number of score categories for the items in the argument `cats`. For example, a dichotomous item has two score categories. If a single numeric value is specified, that value will be recycled across all items. If NULL and all items are binary items (i.e., dichotomous items), it assumes that all items have two score categories.

If necessary, you need to specify whether prior distributions of item slope and guessing parameters (only for the IRT 3PL model) are used in the arguments of `use.aprior` and `use.gprior`, respectively. If you decide to use the prior distributions, you should specify what distributions will be used for the prior distributions in the arguments of `aprior` and `gprior`, respectively. Currently three probability distributions of Beta, Log-normal, and Normal distributions are available.

In addition, if the response data include missing values, you must indicate the missing value in argument `missing`.

Once the `est_irt` function has been implemented, you'll get a list of several internal objects such as the item parameter estimates, standard error of the parameter estimates.

Online item calibration with the fixed ability parameter calibration method (e.g., Stocking, 1988)

In CAT, the fixed ability parameter calibration (FAPC), often called Stocking's Method A, is the relatively simplest and most straightforward online calibration method, which is the maximum likelihood estimation of the item parameters given the proficiency estimates. In CAT, FAPC can be used to put the parameter estimates of pretest items on the same scale of operational item parameter estimates and recalibrate the operational items to evaluate the parameter drifts of the operational items (Chen & Wang, 2016; Stocking, 1988). Also, FAPC is known to result in accurate, unbiased item parameters calibration when items are randomly rather than adaptively administered to examinees, which occurs most commonly with pretest items (Ban, Hanson, Wang, Yi, & Harris, 2001; Chen & Wang, 2016). Using **irtQ** package, the FAPC is implemented to calibrate the items with two main steps:

1. Prepare a data set for the calibration of item parameters (i.e., item response data and ability estimates).
2. Implement the FAPC to estimate the item parameters using the `est_item` function.

1. Preparing a data set To run the `est_item` function, it requires two data sets:

1. Examinees' ability (or proficiency) estimates. It should be in the format of a numeric vector.
2. response data set for the items. It should be in the format of matrix where a row and column indicate the examinees and the items, respectively. The order of the examinees in the response data set must be exactly the same as that of the examinees' ability estimates.

2. Estimating the pretest item parameters The `est_item` function estimates the pretest item parameters given the proficiency estimates. To estimate the item parameters, you need to provide the response data in the argument `data` and the ability estimates in the argument `score`.

Also, you should provide a string vector of the IRT models in the argument `model` to indicate what IRT model is used to calibrate each item. Available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous items, and "GRM" and "GPCM" for polytomous items. "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. Note that "DRM" is considered as "3PLM" in this function. If a single character of the IRT model is specified, that model will be recycled across all items.

The `est_item` function requires a vector of the number of score categories for the items in the argument `cats`. For example, a dichotomous item has two score categories. If a single numeric value is specified, that value will be recycled across all items. If NULL and all items are binary items (i.e., dichotomous items), it assumes that all items have two score categories.

If necessary, you need to specify whether prior distributions of item slope and guessing parameters (only for the IRT 3PL model) are used in the arguments of `use.aprior` and `use.gprior`, respectively. If you decide to use the prior distributions, you should specify what distributions will be used for the prior distributions in the arguments of `aprior` and `gprior`, respectively. Currently three probability distributions of Beta, Log-normal, and Normal distributions are available.

In addition, if the response data include missing values, you must indicate the missing value in argument `missing`.

Once the `est_item` function has been implemented, you'll get a list of several internal objects such as the item parameter estimates, standard error of the parameter estimates.

Three examples of R script

The example code below shows how to implement the online calibration and how to evaluate the IRT model-data fit:

```
##---------------------------------------------------------------
# Attach the packages
library(irtQ)

##-------------------------------------------------------------------------------
# 1. The example code below shows how to prepare the data sets and how to
#    implement the fixed item parameter calibration (FIPC):
##-------------------------------------------------------------------------------

## Step 1: prepare a data set
## In this example, we generated examinees' true proficiency parameters and simulated
## the item response data using the function "simdat".

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

# simulate the response data
sim.dat <- simdat(x=x, theta=score, D=1)

## Step 2: Estimate the item parameters
# fit the 3PL model to all dichotmous items, fit the GRM model to all polytomous data,
# fix the five 3PL items (1st - 5th items) and three GRM items (53th to 55th items)
# also, estimate the empirical histogram of latent variable
fix.loc <- c(1:5, 53:55)
(mod.fix1 <- est_irt(x=x, data=sim.dat, 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))
summary(mod.fix1)

# plot the estimated empirical histogram of latent variable prior distribution
(emphist <- getirt(mod.fix1, what="weights"))
plot(emphist$weight ~ emphist$theta, xlab="Theta", ylab="Density")
```

```
##-----------------------------------------------------------------------------
# 2. The example code below shows how to prepare the data sets and how to estimate
#    the item parameters using the fixed abilit parameter calibration (FAPC):
##-----------------------------------------------------------------------------

## Step 1: prepare a data set
## In this example, we generated examinees' true proficiency parameters and simulated
## the item response data using the function "simdat". Because, the true
## proficiency parameters are not known in reality, however, the true proficiencies
## would be replaced with the proficiency estimates for the calibration.

# 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

# 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 the response data
data <- simdat(x=x, theta=score, D=1)

## Step 2: Estimate the item parameters
# 1) item parameter estimation: constrain the slope parameters of the 1PLM to be equal
(mod1 <- est_item(x, data, score, D=1, fix.a.1pl=FALSE, use.gprior=TRUE,
                  gprior=list(dist="beta", params=c(5, 17)), use.startval=FALSE))
summary(mod1)

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

# 3) item parameter estimation: fix the guessing parameters of the 3PLM 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)
```

IRT Models

In the **irtQ** package, both dichotomous and polytomous IRT models are available. For dichotomous items, IRT one-, two-, and three-parameter logistic models (1PLM, 2PLM, and 3PLM) are used. For polytomous items, the graded response model (GRM) and the (generalized) partial credit model (GPCM) are used. Note that the item discrimination (or slope) parameters should be fixed to 1 when the partial credit model is fit to data.

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

IRT 1-3PL models For the IRT 1-3PL models, the probability that an examinee with θ provides a correct answer for an item is given by,

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

where a is the item discrimination (or slope) parameter, b represents the item difficulty parameter, g refers to the item guessing parameter. D is a scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function when $D = 1.702$. When the 1PLM is used, a is either fixed to a constant value (e.g., $a = 1$) or constrained to have the same value across all 1PLM item data. When the IRT 1PLM or 2PLM is fit to data, $g = 0$ is set to 0.

GRM For the GRM, the probability that an examinee with latent ability θ responds to score category k ($k = 0, 1, ..., K-1$) of an item is a given by,

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

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

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

where $P^*(Y \geq k|\theta$ refers to the category boundary (threshold) function for score category k of an item and its formula is analogous to that of 2PLM. b_k is the difficulty (or threshold) parameter for category boundary k of an item. Note that $P(Y = 0|\theta) = 1 - P^*(Y \geq 1|\theta)$ and $P(Y = K-1|\theta) = P^*(Y \geq K-1|\theta)$.

GPCM For the GPCM, the probability that an examinee with latent ability θ responds to score category k ($k = 0, 1, ..., K-1$) of an item is a given by,

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

where b_v is the difficulty parameter for category boundary v of an item. In other contexts, the difficulty parameter b_v can also be parameterized as $b_v = \beta - \tau_v$, where β refers to the location (or overall difficulty) parameter and τ_{jv} represents a threshold parameter for score category v of an item. In the **irtQ** package, $K-1$ difficulty parameters are necessary when an item has K unique score categories because $b_0 = 0$. When a partial credit model is fit to data, a is fixed to 1.

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.

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. *Psychometrika, 35*, 179-198.

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

Cappaert, K. J., Wen, Y., & Chang, Y. F. (2018). Evaluating CAT-adjusted approaches for suspected item parameter drift detection. *Measurement: Interdisciplinary Research and Perspectives, 16*(4), 226-238.

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

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

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

Hambleton, R. K., & Swaminathan, H. (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.

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

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.

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

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

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

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

Lim, H., & Choe, E. M. (2023). Detecting differential item functioning in CAT using IRT residual DIF approach. *Journal of Educational Measurement*. 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. (2023, April). *Detecting differential item functioning among multiple groups using IRT residual DIF framework*. Paper presented at the Annual Meeting of the National Council on Measurement in Education. Chicago, IL.

Lim, H., Davey, T., & Wells, C. S. (2020). A recursion-based analytical approach to evaluate the performance of MST. *Journal of Educational Measurement, 58*(2), 154-178.

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

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

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

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

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

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

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.

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.

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

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

Stone, C. A. (2000). Monte Carlo based null distribution for an alternative goodness-of-fit test statistic in IRT models. *Journal of educational measurement, 37*(1), 58-75.

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

Thissen, D. & Wainer, H. (1982). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427-450.

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

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.

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.

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

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 Program]. Chicago, IL: Scientific Software International. URL http://www.ssicentral.com

---

`bind.fill` *Bind Fill*

Description

This function creates a cbind matrix or rbind matrix using a list containing different length of numeric vectors.

Usage

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

Arguments

List	A list containing different length of numeric vectors
type	A character string specifying whether rbind is used or cbind is used.
fill	The value used to fill in missing data when aligning datasets. For type = "cbind", this fills missing rows in shorter columns. For type = "rbind", this fills missing columns in shorter rows. Accepts any R object (e.g., numeric, character, logical). Defaults to NA.

Value

A matrix.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

Examples

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

# examples
# 1) create a rbind with the sample score list
bind.fill(score_list, type="rbind")

# 2) create a cbind with the sample score list
bind.fill(score_list, type="cbind")

# 3) create a cbind with the sample score list,
#    and fill missing data with 0s.
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 bisetion function in the **cmna** R package (Howard, 2017) to find a root of the function .funs with respect to its first argument. Unlike the bisetion of the **cmna**, this bisetion function accepts additional arguments of the function .fun.

Usage

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

Arguments

.fun	A function for which the root is searched.
...	Additional arguments to be passed to .fun.
lb	A lower bound of the interval to be searched.
ub	An upper bound of the interval to be searched.
tol	The tolerance of error. Default is 1e-4.
max.it	The maximum number of iterations. Default is 100.

Details

The bisection method is a well-known root finding numerical algorithm that works for any continuous function when the lower (lb) and upper (ub) bounds with opposite signs are provided. This method repeatedly bisects the defined interval by two values with opposite signs until the absolute difference of two values becomes less than the error tolerance (tol) or the maximum number of iterations (max.it) is reached.

Value

A list with three internal objects. The first object is the root found, the second object is the number of iterations used, and the third object is the approximate accuracy of the root (i.e., absolute difference between the final two values with opposite signs).

References

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

See Also

est_score

Examples

```
## example: find a theta corresponding to the probability of
## correct answer using the item response function of 2PLM
## (a = 1, b = 0.2)

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

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

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

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

Description

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

Usage

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

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

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

bring.mirt(x)
```

Arguments

`file`	A file name (including a directory) containing the item or ability parameters.
`type`	A character string indicating a type of output file. Available types are "par" for a file containing item parameter estimates and "sco" for a file containing ability parameter estimates.
`rePar`	A logical value. If TRUE and when the IRT dichotomous model (e.g., 3PLM) or GRM is fit to data, the item intercept and logit of item guessing parameters are reparameterized into the item difficulty and item guessing parameters, respectively. Default is TRUE.
`rePar.gpc`	A logical value. If TRUE and when (G)PCM is fit to data, the nominal model parameters in the flexMIRT parameter output file are reparameterized into the (G)PCM slope/difficulty parameters. Default is TRUE.
`n.factor`	A numeric value indicating the number of estimated factors. This argument should be specified when `type = "sco"`. Default is 1.
`x`	An output object obtained from the function `mirt`.

Details

The `bring.flexmirt` was written by modifying the function `read.flexmirt` (Pritikin & Falk, 2020). The functions `bring.bilog` and `bring.parscale` were written by modifying the functions `read.bilog` and `read.parscale` (Weeks, 2010), respectively.

The file extensions for item parameter and ability files, respectively, are: ".par" and ".sco" for BILOG-MG and PARSCALE, and "-prm.txt" and "-sco.txt" for flexMIRT. For mirt, the name of the output object is specified by the user.

Although `bring.flexmirt` is able to extract multidimensional item and ability parameter estimates, this package only deals with unidimensional IRT methods.

For polytomous item parameters, `bring.flexmirt` and `bring.mirt` are able to import the item parameters of the graded response model and the (generalized) partial credit model.

Value

These functions return a list including several objects. Only for the output of flexMIRT, the results of multiple group analysis can be returned. In that case, each element of the list contains the estimation results for each group.

Sample Output Files of IRT software

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

Among the two output files, one of them is from PARSCALE 4 with a file extension of ".PAR" (i.e., "parscale_sample.PAR") and another one is from flexMIRT with a file extension of "-prm.txt" (i.e., "flexmirt_sample-prm.txt").

For the two item parameter estimate output files, both are mixed-format tests with 55 items consisting of fifty dichotomous items following the IRT 3PL model and five polytomous items with five categories following the graded response model. The examples below show how to import those output files.

Note

Regarding the item parameter files for any IRT software, only the internal object "full_df" in the returned list is necessary for the IRT linking. The object "full_df" is a data frame containing the item metadata in a test form (e.g., item parameters, number of categories, models). See `info` or `simdat` for more details about the item metadata.

Also, when item parameters are estimated using the partial credit or the generalized partial credit model, item step parameters are returned in the object "full_df". Item step parameters are the overall item difficulty (or location) parameter subtracted by the difficulty (or threshold) parameter for each category. See `irtfit` for more details about the parameterization of the (generalized) partial credit model.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

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

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

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

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

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

See Also

irtfit

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 transform them to item meta data
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 transform them to item meta data
bring.parscale(file = pscale_sam, "par")$full_df
```

---

cac_lee *Classification accuracy and consistency using Lee's (2010) approach.*

Description

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

Usage

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

Arguments

x	A data frame containing the item metadata (e.g., item parameters, number of score categories, models ...). See est_irt, irtfit, info or simdat for more detail about the item metadata.
cutscore	A numeric vector specifying the cut scores for classification. Cut scores are the points that separate different performance categories (e.g., pass vs. fail, or different grades).
theta	A numeric vector of ability estimates. Ability estimates (theta values) are the individual proficiency estimates obtained from the IRT model. The theta parameter is optional and can be NULL.
weights	An optional two-column data frame or matrix where the first column is the quadrature points (nodes) and the second column is the corresponding weights. This is typically used in quadrature-based IRT analysis.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
cut.obs	A logical value. If TRUE, it indicates the cutscores on the observed-summed score metric. If FALSE, it indicates they are on the IRT theta score metric. Default is TRUE.

Details

The function works by first checking the provided inputs. If both theta and weights are NULL, the function will stop and return an error message. Depending on the provided inputs, the function will then compute the classification accuracy and consistency indices using either the quadrature points and corresponding weights (D method) or individual ability estimates (P method). The function returns a list containing the confusion matrix, marginal and conditional classification accuracy and consistency indices, probabilities of being assigned to each level category, and cut scores.

Value

A list containing the following elements:

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

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

See Also

gen.weight, est_score, cac_rud

Examples

```
## ------------------------------------------------------------------------------
# 1. When the empirical ability distribution of the population is available
#    (D method)
## ------------------------------------------------------------------------------
## import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

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

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

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

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

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

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

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

## ------------------------------------------------------------------------------
# 3. When individual ability estimates are available,
#    but, the cutscores are on the IRT theta score metric
## ------------------------------------------------------------------------------
# set the theta cut scures
```

```
cutscore <- c(-2, -0.4, 0.2, 1.0)

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

---

cac_rud *Classification accuracy and consistency using Rudner's (2001, 2005) approach.*

Description

This function computes the classification accuracy and consistency based on the methods proposed by Rudner (2001, 2005). It can handle both situations where the empirical ability distribution of the population is available and when individual ability estimates are available.

Usage

```
cac_rud(cutscore, theta = NULL, se, weights = NULL)
```

Arguments

cutscore	A numeric vector specifying the cut scores for classification. Cut scores are the points that separate different performance categories (e.g., pass vs. fail, or different grades).
theta	A numeric vector of ability estimates. Ability estimates (theta values) are the individual proficiency estimates obtained from the IRT model. The theta parameter is optional and can be NULL.
se	A numeric vector of the same length as theta indicating the standard errors of the ability estimates.
weights	An optional two-column data frame or matrix where the first column is the quadrature points (nodes) and the second column is the corresponding weights. This is typically used in quadrature-based IRT analysis.

Details

The function first checks the provided inputs for correctness. It then computes the probabilities that an examinee with a specific ability is assigned to each level category, and calculates the conditional classification accuracy and consistency for each theta value. Finally, it computes the marginal accuracy and consistency.

Value

A list containing the following elements:

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

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

See Also

gen.weight, est_score, cac_lee

Examples

```
## ------------------------------------------------------------------------------
# 1. When the empirical ability distribution of the population is available
## ------------------------------------------------------------------------------
## import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

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

# set the cut scores on the theta score metric
cutscore <- c(-2, -0.5, 0.8)

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

# compute the conditional standard errors across all quadrature points
tif <- info(x = x, theta = node, D = 1, tif = TRUE)$tif
se <- 1 / sqrt(tif)

# calculate the classification accuracy and consistency
cac_1 <- cac_rud(cutscore = cutscore, se = se, weights = weights)
```

```
print(cac_1)

## ------------------------------------------------------------------------------
# 2. When individual ability estimates are available
## ------------------------------------------------------------------------------
# randomly select the true abilities from N(0, 1)
set.seed(12)
theta <- rnorm(n = 1000, mean = 0, sd = 1)

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

# estimate the ability parameters and standard errors using the 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

# calculate the classification accuracy and consistency
cac_2 <- cac_rud(cutscore = cutscore, theta = theta_hat, se = se)
print(cac_2)
```

`catsib` *CATSIB DIF detection procedure*

Description

This function analyzes DIF on an item using CATSIB procedure (Nandakumar & Roussos, 2004), which is a modified version of SIBTEST (Shealy & Stout, 1993). The CATSIB procedure can be applied to a computerized adaptive testing (CAT) environment for differential item functioning (DIF) detection. In CATSIB, examinees are matched on IRT-based ability estimates adjusted by employing a regression correction method (Shealy & Stout, 1993) to reduce a statistical bias of the CATSIB statistic due to impact.

Usage

```
catsib(
  x = NULL,
  data,
  score = NULL,
  se = NULL,
  group,
  focal.name,
  D = 1,
  n.bin = c(80, 10),
```

```
  min.binsize = 3,
  max.del = 0.075,
  weight.group = c("comb", "foc", "ref"),
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...). x should to be provided to estimate latent ability parameters when score = NULL or purify = TRUE. Default is NULL. See est_irt, irtfit, info or simdat for more detail about the item metadata.
data	A matrix containing examinees' response data of the items in the argument x. A row and column indicate the examinees and items, respectively.
score	A vector of examinees' ability estimates. If the abilities are not provided (i.e., score = NULL), catsib computes the ability estimates before computing the CATSIB statistics. See est_score for more detail about scoring methods. Default is NULL.
se	A vector of the standard errors of the ability estimates. The standard errors should be ordered in accordance with the order of the ability estimates specified in the score argument. Default is NULL.
group	A numeric or character vector indicating group membership of examinees. The length of vector should be the same with the number of rows in the response data matrix.
focal.name	A single numeric or character scalar representing the level associated with the focal group. For instance, given group = c(0, 1, 0, 1, 1) and '1' indicating the focal group, set focal.name = 1.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
n.bin	A vector of two positive integers to set the maximum and minimum numbers of bins (or intervals) on the ability scale. The first and second values indicate the maximum and minimum numbers of the bins, respectively. See below for more detail.
min.binsize	A positive integer value to set the minimum size of each bin. To ensure stable statistical estimation, each bin is required to have a certain number of examinees

(e.g, 3), at least, from both reference and focal groups if it was to be included in calculation of $\hat{\beta}$. All bins with fewer than the minimum number are not used for the computation. Default is 3. See below for more detail.

`max.del` A numerical value to set the maximum permissible proportion of examinees to be deleted from either reference group or focal group when automatically determining the number of bins on the ability scale. Default is 0.075. See below for more detail.

`weight.group` A single character string to specify a target ability distribution over which the expectation of DIF measure, called $\hat{\beta}$, and the corresponding standard error are computed. Available options are "comb" for the combined ability distribution from both the reference and focal groups, "foc" for the ability distribution of the focal group, and "ref" for the ability distribution of the reference group. Defulat is "comb". See below for more detail.

`alpha` A numeric value to specify significance α-level of the hypothesis test using the CATSIB statistics. Default is .05.

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

`purify` A logical value indicating whether a purification process will be implemented or not. Default is FALSE. See below for more detail.

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

`min.resp` A positive integer value specifying the minimum number of item responses for an examinee required to compute the ability estimate. Default is NULL. See details below for more information.

`method` A character string indicating a scoring method. Available methods are "ML" for the maximum likelihood estimation, "WL" for the weighted likelihood estimation, "MAP" for the maximum a posteriori estimation, and "EAP" for the expected a posteriori estimation. Default method is "ML".

`range` A numeric vector of two components to restrict the range of ability scale for the ML, WL, MLF, and MAP scoring methods. Default is c(-5, 5).

`norm.prior` A numeric vector of two components specifying a mean and standard deviation of the normal prior distribution. These two parameters are used to obtain the gaussian quadrature points and the corresponding weights from the normal distribution. Default is c(0,1). Ignored if `method` is "ML" or "WL".

`nquad` An integer value specifying the number of gaussian quadrature points from the normal prior distribution. Default is 41. Ignored if `method` is "ML", "WL", or "MAP".

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the latent variable prior distribution. The weights and quadrature points can be easily obtained using the function `gen.weight`. If NULL and `method` is "EAP", default values are used (see the arguments of `norm.prior` and `nquad`). Ignored if `method` is "ML", "WL", or "MAP".

`ncore` The number of logical CPU cores to use. Default is 1. See `est_score` for details.

verbose	A logical value. If TRUE, the progress messages of purification procedure are suppressed. Default is TRUE.
...	Additional arguments that will be forwarded to the est_score function.

Details

In CATSIB procedure (Nandakumar & Roussos, 2004), because $\hat{\beta}^*$, which is the expected θ regressed on $\hat{\beta}$, is a continuous variable, the range of $\hat{\beta}^*$ is divided into K equal intervals and examinees are classified into one of K intervals on the basis of their $\hat{\beta}^*$.Then, any intervals that contain less than three examinees in either reference or focal groups were excluded from the computation of $\hat{\beta}$, which is a measure of the amount of DIF, to ensure stable statistical estimation. According to Nandakumar and Roussos (2004), a default minimum size of each bin is set to 3 in `min.binsize`.

To carefully choose the number of intervals (K), the `catsib` automatically determines it by gradually decreasing K from a larger to smaller numbers based the rule used in Nandakumar and Roussos (2004). Specifically, beginning with an arbitrary large number (e.g., 80), if more than a certain permissible percentage, let's say 7.5%, of examinees in either the reference or focal groups were removed, the `catsib` automatically decreases the number of bins by one unit until a total number of examinees in each group reaches to more than or equal to 92.5%. However, Nandakumar and Roussos (2004) recommended setting the minimum K to 10 to avoid a situation that extremely a few intervals are left, even if the number of remaining examinees in each group is less than 92.5%. Thus, the maximum and minimum number of bins are set to 80 and 10, respectively, as default in `n.bin`. Also, a default maximum permissible proportion of examinees to be deleted from either reference group or focal group is set to 0.075 in `max.del`.

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

Although Nandakumar and Roussos (2004) did not propose a purification procedure for DIF analysis using CATSIB, the `catsib` can implement an iterative purification process in a similar way as in Lim, Choe, and Han (2022). Simply, at each iterative purification, examinees' latent abilities are computed using purified items and scoring method specified in the `method` argument. The iterative purification process stops when no further DIF items are found or the process reaches a predetermined limit of iteration, which can be specified in the `max.iter` argument. See Lim et al. (2022) for more details about the purification procedure.

Scoring with a limited number of items can result in large standard errors, which may impact the effectiveness of DIF detection within the CATSIB procedure. The `min.resp` argument can be employed to avoid using scores with significant standard errors when calculating the CATSIB statistic, particularly during the purification process. For instance, if `min.resp` is not NULL (e.g., `min.resp=5`), item responses from examinees whose total item responses fall below the specified minimum number are treated as missing values (i.e., NA). Consequently, their ability estimates become missing values and are not utilized in computing the CATSIB statistic. If `min.resp=NULL`, an examinee's score will be computed as long as there is at least one item response for the examinee.

Value

This function returns a list of four internal objects. The four objects are:

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

- **dif_stat** A data frame containing the results of CATSIB statistics across all evaluated items. From the first column, each column indicates item's ID, CATSIB (*beta*) statistic, standard error of the *beta*, standardized *beta*, p-value of the *beta*, sample size of the reference group, sample size of the focal group, and total sample size, respectively.
- **dif_item** A numeric vector showing potential DIF items flagged by CATSIB statistic.
- **contingency** A contingency table of each item used to compute CATSIB statistic.

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

`with_purify` A list of several sub-objects containing the results of DIF analysis with a purification procedure. The sub-objects are:

- **dif_stat** A data frame containing the results of CATSIB statistics across all evaluated items. From the first column, each column indicates item's ID, CATSIB (*beta*) statistic, standard error of the *beta*, standardized *beta*, p-value of the *beta*, sample size of the reference group, sample size of the focal group, and total sample size, and *n*th iteration where the CATSIB statistic was computed, respectively.
- **dif_item** A numeric vector showing potential DIF items flagged by CATSIB statistic.
- **n.iter** A total number of iterations implemented for the purification.
- **complete** A logical value indicating whether the purification process was completed. If FALSE, it means that the purification process reached the maximum iteration number but it was not complete.
- **contingency** A contingency table of each item used to compute CATSIB statistic.

`alpha` A significance α-level used to compute the p-values of RDIF statistics.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

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

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

See Also

rdif, est_item, info, simdat, shape_df, gen.weight, est_score

Examples

```
# call library
library("dplyr")

## Uniform DIF detection
################################################
# (1) manipulate true uniform DIF data
################################################
# import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# select 36 of 3PLM items which are non-DIF items
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 inject uniform DIF
difpar_ref <-
  shape_df(
    par.drm = list(a = c(0.8, 1.5, 0.8, 1.5), b = c(0.0, 0.0, -0.5, -0.5), g = 0.15),
    item.id = paste0("dif", 1:4), cats = 2, model = "3PLM"
  )

# manipulate uniform DIF on the four new items by adding constants to 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 4 DIF and 36 non-DIF items for both reference and focal groups
# thus, the first four items have uniform DIF
par_ref <- rbind(difpar_ref, par_nstd)
par_foc <- rbind(difpar_foc, par_nstd)

# generate the true thetas
set.seed(123)
theta_ref <- rnorm(500, 0.0, 1.0)
theta_foc <- rnorm(500, 0.0, 1.0)

# generate the 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 the item and ability parameters
#     using the aggregate data
################################################
# estimate the item parameters
est_mod <- est_irt(data = data, D = 1, model = "3PLM")
est_par <- est_mod$par.est

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

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

# (a)-1 compute SIBTEST statistic by providing scores,
#       and without a purification
dif_1 <- catsib(
  x = NULL, data = data, D = 1, score = score, se = se, group = group, focal.name = 1,
  weight.group = "comb", alpha = 0.05, missing = NA, purify = FALSE
)
print(dif_1)

# (a)-2 compute SIBTEST statistic by providing scores,
#       and with a purification
dif_2 <- catsib(
  x = est_par, data = data, D = 1, score = score, se = se, group = group, focal.name = 1,
  weight.group = "comb", alpha = 0.05, missing = NA, purify = TRUE
)
print(dif_2)
```

---

covirt *Asymptotic variance-covariance matrices of item parameter estimates*

Description

This function calculates the analytical asymptotic variance-covariance matrices (e.g., Li & Lissitz, 2004; Thissen & Wainer, 1982) of item parameter estimates for dichotomous and polytomous IRT Models without examinee's responses to test items, given a set of item parameter estimates and sample size. The square roots of variance terms in the matrices can be used as the asymptotic standard errors of maximum likelihood item parameter 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 the item metadata (e.g., item parameters, number of categories, models ...). See irtfit, info, or simdat for more details about the item metadata. This data frame can be easily obtained using the function shape_df.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
nstd	An integer value or a vector of integer values indicating a sample size. When a vector is specified, length of the vector must be the same as the number of test items in the argument x. Default is 1,000. See below for details.
pcm.loc	A vector of integer values indicating the locations of partial credit model (PCM) items. For the PCM items, the variance-covariance matrices are computed only for the item category difficulty parameters. Default is NULL. See below for details.
norm.prior	A numeric vector of two components specifying a mean and standard deviation of the normal prior distribution. These two parameters are used to obtain the gaussian quadrature points and the corresponding weights from the normal distribution. Default is c(0,1).
nquad	An integer value specifying the number of gaussian quadrature points from the normal prior distribution. Default is 41.
weights	A two-column matrix or data frame containing the theta values (in the first column) and the weights (in the second column) for the prior distribution. The weights and theta values can be easily obtained using the function gen.weight. If NULL, default values are used for the prior distribution (see the arguments of norm.prior and nquad). Default is NULL.

Details

The standard errors obtained from the analytical approach are likely to represent lower bounds for the actual standard errors (Thissen & Wainer, 1982). Therefore, they may be useful for assessing the degree of precision of a set of item parameter estimates when the corresponding standard errors of the estimates are not presented in literature or research reports.

Sometimes item parameters need to be estimated using different sample size. If the item parameters in the argument x were calibrated with different number of examinees, a vector of different sample

sizes should be specified in the argument nstd. Suppose that you want to compute the variance-covariance matrices of five IRT 3PLM items and the five items were calibrated with 500, 600, 1,000, 2,000, and 700 examinees, respectively. Then, nstd = c(500, 600, 1000, 2000, 700) must be specified.

Because you can specify only "GPCM" for both the partial credit model (PCM) or the generalized partial credit model (GPCM) in the item metadata, you must indicate which items are the PCM items through the argument pcm.loc. This is because the item category difficulty parameters are estimated from the PCM, meaning that the variance-covariance of item parameter estimates must be computed for the item category difficulty parameters. Suppose that you want to compute the variance-covariance matrices of five polytomous items and the last two items were calibrated with the PCM. Then, pcm.loc = c(4, 5) must be specified.

Value

A list of two internal objects. The first internal object contains a list of the variance-covariance matrices of item parameter estimates. The second internal object contains a list of the standard errors of item parameter estimates.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

Thissen, D. & Wainer, H. (1982). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427-450.

See Also

irtfit, info, simdat, shape_df, gen.weight

Examples

```
## the use of a "-prm.txt" file obtained sfrom a flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

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

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

drm *Dichotomous Response Model (DRM) Probabilities*

Description

This function computes the probability of correct answers for multiple items for a given set of theta values using the IRT 1PL, 2PL, and 3PL models.

Usage

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

Arguments

theta	A vector of ability values.
a	A vector of item discrimination (or slope) parameters.
b	A vector of item difficulty (or threshold) parameters.
g	A vector of item guessing parameters.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

Details

g does not need to be specified when the response probabilities of the 1PL and 2PL models are computed.

Value

This function returns a matrix where a row indicates the ability and a column represents the item.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

prm

Examples

```
## when vectors are used for both theta values and item parameters (3PLM)
drm(c(-0.1, 0.0, 1.5), a = c(1, 2), b = c(0, 1), g = c(0.2, 0.1), D = 1)

## when vectors are only used for item parameters (2PLM)
drm(0.0, a = c(1, 2), b = c(0, 1), D = 1)

## when vectors are only used for theta values (3PLM)
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 (IRT) models to a mixture of dichotomous and polytomous data using the marginal maximum likelihood estimation via the expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981). This function also implements the fixed item parameter calibration (FIPC; Kim, 2006). As Method A (Stocking, 1988), FIPC is one of useful online item calibration methods for computerized adaptive testing (CAT) to put the parameter estimates of pretest items on the same scale of operational item parameter estimates (Ban, Hanson, Wang, Yi, & Harris, 2001). For dichotomous items, IRT one-, two-, and three-parameter logistic models are available. For polytomous items, the graded response model (GRM) and the (generalized) partial credit model (GPCM) are available.

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(iter.max = 200),
  fipc = FALSE,
```

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

Arguments

x	A data frame containing the item metadata. This metadata is necessary to obtain the information of each item (i.e., number of score categories and IRT model) to be calibrated. You can easily create an empty item metadata using the function shape_df. When use.startval = TRUE, the item parameters specified in the item metadata are used as the starting values for the item parameter estimation. If x = NULL, the arguments of model and cats must be specified. Note that when fipc = TRUE to implement the FIPC method, the item metadata of a test form must be provided in the argument x. See below for details. Default is NULL.
data	A matrix containing examinees' response data for the items in the argument x. A row and column indicate the examinees and items, respectively.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
model	A vector of character strings indicating what IRT model is used to calibrate each item. Available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous items, and "GRM" and "GPCM" for polytomous items. "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. Note that "DRM" is considered as "3PLM" in this function. If a single character of the IRT model is specified, that model will be recycled across all items. The provided information in the model argument is used only when x = NULL and fipc = FALSE. Default is NULL.
cats	A numeric vector specifying the number of score categories for each item. For example, a dichotomous item has two score categories. If a single numeric value is specified, that value will be recycled across all items. If cats = NULL and all specified models in the model argument are the dichotomous models (i.e., 1PLM, 2PLM, 3PLM, or DRM), it assumes that all items have two score categories. The provided information in the cats argument is used only when x = NULL and fipc = FALSE. Default is NULL.
item.id	A character vector of item IDs. If NULL, the item IDs are generated automatically. When fipc = TRUE and the Item IDs are given by the item.id argument, the Item IDs in the x argument are overridden. Default is NULL.
fix.a.1pl	A logical value. If TRUE, the slope parameters of the 1PLM items are fixed to a specific value specified in the argument a.val.1pl. Otherwise, the slope parameters of all 1PLM items are constrained to be equal and estimated. Default is FALSE.
fix.a.gpcm	A logical value. If TRUE, the GPCM items are calibrated with the partial credit model and the slope parameters of the GPCM items are fixed to a specific value specified in the argument a.val.gpcm. Otherwise, the slope parameter of each GPCM item is estimated. Default is FALSE.

`fix.g` A logical value. If TRUE, the guessing parameters of the 3PLM items are fixed to a specific value specified in the argument `g.val`. Otherwise, the guessing parameter of each 3PLM item is estimated. Default is FALSE.

`a.val.1pl` A numeric value. This value is used to fixed the slope parameters of the 1PLM items.

`a.val.gpcm` A numeric value. This value is used to fixed the slope parameters of the GPCM items.

`g.val` A numeric value. This value is used to fixed the guessing parameters of the 3PLM items.

`use.aprior` A logical value. If TRUE, a prior distribution for the slope parameters is used for the parameter calibration across all items. Default is FALSE.

`use.bprior` A logical value. If TRUE, a prior distribution for the difficulty (or threshold) parameters is used for the parameter calibration across all items. Default is FALSE.

`use.gprior` A logical value. If TRUE, a prior distribution for the guessing parameters is used for the parameter calibration across all 3PLM items. Default is TRUE.

`aprior` A list containing the information of the prior distribution for item slope parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution is used, "lnorm" should be specified in the first argument. When Normal distribution is used, "norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see `dbeta()`, `dlnorm()`, and `dnorm()` in the **stats** package for more details.

`bprior` A list containing the information of the prior distribution for item difficulty (or threshold) parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution is used, "lnorm" should be specified in the first argument. When Normal distribution is used, "norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see `dbeta()`, `dlnorm()`, and `dnorm()` in the **stats** package for more details.

`gprior` A list containing the information of the prior distribution for item guessing parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution is used, "lnorm" should be specified in the first argument. When Normal distribution is used,

"norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see `dbeta()`, `dlnorm()`, and `dnorm()` in the **stats** package for more details.

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

`Quadrature` A numeric vector of two components specifying the number of quadrature points (in the first component) and the symmetric minimum and maximum values of these points (in the second component). For example, a vector of c(49, 6) indicates 49 rectangular quadrature points over -6 and 6. The quadrature points are used in the E step of the EM algorithm. Default is c(49, 6).

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the latent variable prior distribution. If not NULL, the scale of the latent ability distribution will be will be fixed to the scale of the provided quadrature points and weights. The weights and quadrature points can be easily obtained using the function `gen.weight`. If NULL, a normal prior density is used based on the information provided in the arguments of `Quadrature`, `group.mean`, and `group.var`). Default is NULL.

`group.mean` A numeric value to set the mean of latent variable prior distribution when `weights = NULL`. Default is 0. This value is fixed to remove the indeterminancy of item parameter scale when calibrating items. However, the scale of prior distribution is updated when FIPC is implemented.

`group.var` A positive numeric value to set the variance of latent variable prior distribution when `weights = NULL`. Default is 1. This value is fixed to remove the indeterminancy of item parameter scale when calibrating items. However, the scale of prior distribution is updated when FIPC is implemented.

`EmpHist` A logical value. If TRUE, the empirical histogram of the latent variable prior distribution is simultaneously estimated with the item parameters using Woods's (2007) approach. The items are calibrated against the estimated empirical prior distributions. See below for details.

`use.startval` A logical value. If TRUE, the item parameters provided in the item metadata (i.e., the argument `x`) are used as the starting values for the item parameter estimation. Otherwise, internal starting values of this function are used. Default is FALSE.

`Etol` A positive numeric value. This value sets the convergence criterion for E steps of the EM algorithm. Default is 1e-4.

`MaxE` A positive integer value. This value determines the maximum number of the E steps in the EM algorithm. Default is 500.

`control` A list of control parameters to be passed to the optimization function of `nlminb()` in the **stats** package. The control parameters set the conditions of M steps of the EM algorithm. For example, the maximum number of iterations in each of the iterative M steps can be set by `control = list(iter.max=200)`. Default maximum number of iterations in each M step is 200. See `nlminb()` in the **stats** package for other control parameters.

`fipc` A logical value. If TRUE, FIPC is implemented for item parameter estimation. When `fipc = TRUE`, the information of which items are fixed needs to be provided via either `fix.loc` or `fix.id`. See below for details.

`fipc.method` A character string specifying the FIPC method. Available methods include "OEM" for "No Prior Weights Updating and One EM Cycle (NWU-OEM; Wainer & Mislevy, 1990)" and "MEM" for "Multiple Prior Weights Updating and Multiple EM Cycles (MWU-MEM; Kim, 2006)." When `fipc.method = "OEM"`, the maximum number of the E steps of the EM algorithm is set to 1 no matter what number is specified in the argument `MaxE`.

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

`fix.id` A vector of character strings specifying IDs of the items to be fixed when the FIPC is implemented (i.e., `fipc = TRUE`). For example, suppose that five items in which IDs are CMC1, CMC2, CMC3, CMC4, and CMC5 should be fixed and all item IDs are provided in the `X` argument or `item.id` argument. Then `fix.id = c("CMC1", "CMC2", "CMC3", "CMC4", "CMC5")`. Note that when the `fix.id` argument is not NULL, the information provided into the `fix.loc` argument is ignored. See below for details.

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

`verbose` A logical value. If FALSE, all progress messages including the process information on the EM algorithm are suppressed. Default is TRUE.

Details

A specific form of a data frame should be used for the argument `x`. The first column should have item IDs, the second column should contain unique score category numbers of the items, and the third column should include IRT models being fit to the items. The available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous item data, and "GRM" and "GPCM" for polytomous item data. Note that "DRM" covers all dichotomous IRT models (i.e, "1PLM", "2PLM", and "3PLM") and "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. The next columns should include the item parameters of the fitted IRT models. For dichotomous items, the fourth, fifth, and sixth columns represent the item discrimination (or slope), item difficulty, and item guessing parameters, respectively. When "1PLM" and "2PLM" are specified in the third column, NAs should be inserted in the sixth column for the item guessing parameters. For polytomous items, the item discrimination (or slope) parameters should be included in the fourth column and the item difficulty (or threshold) parameters of category boundaries should be contained from the fifth to the last columns. When the number of unique score categories differs between items, the empty cells of item parameters should be filled with NAs. In the **irtQ** package, the item difficulty (or threshold) parameters of category boundaries for GPCM are expressed as the item location (or overall difficulty) parameter subtracted by the threshold parameter for unique score categories of the item. Note that when an GPCM item has *K* unique score categories, *K-1* item difficulty parameters are necessary because the item difficulty parameter for the first category boundary is always 0. For example, if an GPCM item has five score categories, four item difficulty parameters should be specified. An example of a data frame with a single-format test is as follows:

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

And an example of a data frame for a mixed-format test is as follows:

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 `IRT Models` section in the page of `irtQ-package` for more details about the IRT models used in the **irtQ** package. An easier way to create a data frame for the argument `x` is by using the function `shape_df`.

To fit the IRT models to data, the IRT model and the number of score category information for the estimated items must be provided as well as the item response data. There are two way to provide the IRT model and score category information. The first way is to provide the item metadata to the argument `x`. As explained above, the item metadata can be easily created by the function `shape_df`. The second way is specify the IRT models and the score category information into the arguments of `model` and `cats`. Thus, if `x=NULL`, the specified information in `model` and `cats` are used.

To implement FIPC, however, the item metadata must be provided in the argument `x`. This is because the item parameters of the fixed items in the item metadata are used to estimate the characteristic of the underlying latent variable prior distribution when calibrating the rest of freely estimated items. More specifically, the underlying latent variable prior distribution of the fixed items is estimated during the calibration of the freely estimated items to put the item parameters of the freely estimated items on the scale of the fixed item parameters (Kim, 2006).

In terms of approaches for FIPC, Kim (2006) described five different methods. Among them, two methods are available in the function `est_irt`. The first method is "NWU-OEM" where uses just one E step in the EM algorithm, involving data from only the fixed items, and just one M step, involving data from only non-fixed items. This method is suggested by Wainer and Mislevy (1990) in the context of online calibration. This method can be implemented by setting `fipc.method = "OEM"`. The second method is "MWU-MEM" which iteratively updates the latent variable prior distribution and finds the parameter estimates of the non-fixed items. In this method, the same procedure of NWU-OEM method is applied to the first EM cycle. From the second EM cycle, both the parameters of non-fixed items and the weights of the prior distribution are concurrently updated. This method can be implemented by setting `fipc.method = "MEM"`. See Kim (2006) for more details.

When `fipc = TRUE`, the information of which items are fixed needs to be provided via either `fix.loc` or `fix.id`. For example, suppose that five items in which IDs are CMC1, CMC2, CMC3, CMC4, and CMC5 should be fixed and all item IDs are provided in `X` or `item.id`. Also, the five items are located in the 1st through 5th rows of the item metadata (i.e., `x`). Then the item parameters of

the five items can be fixed by setting `fix.loc = c(1, 2, 3, 4, 5)` or `fix.id = c("CMC1", "CMC2", "CMC3", "CMC4", "CMC5")`. Note that if both arguments are not NULL, the information provided into the `fix.loc` argument is ignored.

When `EmpHist = TRUE`, the empirical histogram (i.e., densities at the quadrature points) of latent variable prior distribution is simultaneously estimated with the item parameters. If `fipc = TRUE` given `EmpHist = TRUE`, the scale parameters (e.g., mean and variance) of the empirical prior distribution are estimated as well. If `fipc = FALSE` given `EmpHist = TRUE`, the scale parameters of the empirical prior distribution are fixed to the values specified in the arguments of `group.mean` and `group.var`. When `EmpHist = FALSE`, the normal prior distribution is used during the item parameter estimation. If `fipc = TRUE` given `EmpHist = FALSE`, the scale parameters of the normal prior distribution are estimated as well as the item parameters. If `fipc = FALSE` given `EmpHist = FALSE`, the scale parameters of the normal prior distribution are fixed to the values specified in the arguments of `group.mean` and `group.var`.

Value

This function returns an object of class `est_irt`. Within this object, several internal objects are contained such as:

`estimates`	A data frame containing both the item parameter estimates and the corresponding standard errors of estimates.
`par.est`	A data frame containing the item parameter estimates.
`se.est`	A data frame containing the standard errors of the item parameter estimates. Note that the standard errors are estimated using the cross-production approximation method (Meilijson, 1989).
`pos.par`	A data frame containing the position number of each item parameter being estimated. The position information is useful when interpreting the variance-covariance matrix of item parameter estimates.
`covariance`	A matrix of variance-covariance matrix of item parameter estimates.
`loglikelihood`	A sum of the log-likelihood values of the observed data set (marginal log-likelihood) across all items in the data set
`aic`	A model fit statistic of Akaike information criterion based on the loglikelihood.
`bic`	A model fit statistic of Bayesian information criterion based on the loglikelihood.
`group.par`	A data frame containing the mean, variance, and standard deviation of latent variable prior distribution.
`weights`	A two-column data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the (updated) latent variable prior distribution.
`posterior.dist`	A matrix of normalized posterior densities for all the response patterns at each of the quadrature points. The row and column indicate each individual's response pattern and the quadrature point, respectively.
`data`	A data.frame of the examinees' response data set.
`scale.D`	A scaling factor in IRT models.
`ncase`	A total number of response patterns.

nitem	A total number of items included in the response data.
Etol	A convergence criteria for E steps of the EM algorithm.
MaxE	The maximum number of E steps in the EM algorithm.
aprior	A list containing the information of the prior distribution for item slope parameters.
gprior	A list containing the information of the prior distribution for item guessing parameters.
npar.est	A total number of the estimated parameters.
niter	The number of EM cycles completed.
maxpar.diff	A maximum item parameter change when the EM cycles were completed.
EMtime	Time (in seconds) spent for the EM cycles.
SEtime	Time (in seconds) spent for computing the standard errors of the item parameter estimates.
TotalTime	Time (in seconds) spent for total compuatation.
test.1	Status of the first-order test to report if the gradients has vanished sufficiently for the solution to be stable.
test.2	Status of the second-order test to report if the information matrix is positive definite, which is a prerequisite for the solution to be a possible maximum.
var.note	A note to report if the variance-covariance matrix of item parameter estimates is obtainable from the information matrix.
fipc	A logical value to indicate if FIPC was used.
fipc.method	A method used for the FIPC.
fix.loc	A vector of integer values specifying the locations of the fixed items when the FIPC was implemented.

The internal objects can be easily extracted using the function `getirt`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

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

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

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

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

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

See Also

est_item, irtfit, info, simdat, shape_df, sx2_fit, traceline.est_irt, getirt

Examples

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

# summary of the estimation
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 the 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))

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

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

# summary of the estimation
summary(mod.2pl)

# assess the fit of the 2PL model to the LSAT5 data using S-X2 fit statistic
(sx2fit.2pl <- sx2_fit(x = mod.2pl))

# compute the item and test information at several theta points
theta <- seq(-4, 4, 0.1)
(info.2pl <- info(x = mod.2pl, theta = theta))
```

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

# draw the item characteristic curve for the 1st item
plot(trace.2pl, item.loc = 1)

# fit the 2PL model to LSAT6 data and
# estimate the empirical histogram of latent variable prior distribution
# also use a less stringent convergence criterion for E-step
(mod.2pl.hist <- est_irt(data = LSAT6, D = 1, model = "2PLM", cats = 2,
                         EmpHist = TRUE, Etol = 0.001))
(emphist <- getirt(mod.2pl.hist, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# fit the 3PL model to LSAT6 data and use the Beta prior distribution for
# 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))
))

# summary of the estimation
summary(mod.3pl)

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

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

# fit the different dichotomous models to each item of LSAT6 data
# fit the constrained 1PL model to the 1st, 2nd, and 3rd items, fit the 2PL model to
# the 4th item, and fit the 3PL model to the 5th item with the Beta prior of
# the guessing parameter
(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))
))
# summary of the estimation
summary(mod.drm.mix)

## ----------------------------------------------------------------------------
# 2. item parameter estimation for the mixed-item format data (simulation data)
## ----------------------------------------------------------------------------
## 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
```

```
# modify the item metadata so that the 39th and 40th items follow 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 the response data
sim.dat1 <- simdat(x = x, theta = score1, D = 1)

# fit the 3PL model to all dichotomous items, fit the GPCM model to 39th and 40th items,
# and fit the GRM model to the 53th, 54th, 55th items.
# use the beta prior distribution for the guessing parameters, use the log-normal
# prior distribution for the slope parameters, and use the normal prior distribution
# for the difficulty (or threshold) parameters.
# also, specify the argument 'x' to provide the IRT model and score category information
# for items
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))
))

# summary of the estimation
summary(mod.mix1)

# estimate examinees' latent scores given the item parameter estimates using the MLE
(score.mle <- est_score(x = mod.mix1, method = "ML", range = c(-4, 4), ncore = 2))

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

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

# residual plots for the last item (polytomous item)
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, fit the GPCM model to 39th and 40th items,
# and fit the GRM model to the 53th, 54th, 55th items.
```

```
# also, specify the arguments of 'model' and 'cats' to provide the IRT model and
# score category information for items
(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))
))

# summary of the estimation
summary(mod.mix2)

# fit the 2PL model to all dichotomous items, fit the GPCM model to 39th and 40th items,
# fit the GRM model to the 53th, 54th, 55th items, and estimate the empirical histogram
# of latent variable prior distribution.
# also, specify the arguments of 'model' and 'cats' to provide the IRT model and
# score category information for items
(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,
# fit the PCM model to 39th and 40th items by fixing the slope parameters to 1,
# and fit the GRM model to the 53th, 54th, 55th items.
# also, specify the arguments of 'model' and 'cats' to provide the IRT model and
# score category information for items
(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
))

# summary of the estimation
summary(mod.mix4)

## ----------------------------------------------------------------------------
# 3. fixed item parameter calibration (FIPC) for the mixed-item format data
#    (simulation data)
## ----------------------------------------------------------------------------
## 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 1,000 examinees' latent abilities from N(0.4, 1.3)
set.seed(20)
score2 <- rnorm(1000, mean = 0.4, sd = 1.3)
```

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

# fit the 3PL model to all dichotomous items, fit the GRM model to all polytomous data,
# fix the five 3PL items (1st - 5th items) and three GRM items (53rd to 55th items)
# also, estimate the empirical histogram of latent variable
# 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
))
(prior.par <- mod.fix1$group.par)
(emphist <- getirt(mod.fix1, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# summary of the estimation
summary(mod.fix1)

# or the same five items can be fixed by providing their item IDs to 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
))

# summary of the estimation
summary(mod.fix1)

# fit the 3PL model to all dichotomous items, fit the GRM model to all polytomous data,
# fix the five 3PL items (1st - 5th items) and three GRM items (53rd to 55th items)
# at this moment, do estimate the empirical histogram of latent variable.
# instead, estimate the scale of normal prior distribution of latent variable
# 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
))
(prior.par <- mod.fix2$group.par)
(emphist <- getirt(mod.fix2, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# fit the 3PL model to all dichotomous items, fit the GRM model to all polytomous data,
# at this moment fix only the five 3PL items (1st - 5th items)
# and estimate the empirical histogram of latent variable.
# use the OEM method. Thus, only 1 EM cycle is used.
fix.loc <- c(1:5)
```

```
(mod.fix3 <- est_irt(
  x = x, data = sim.dat2, D = 1, use.gprior = TRUE,
  gprior = list(dist = "beta", params = c(5, 16)), EmpHist = TRUE,
  Etol = 1e-3, fipc = TRUE, fipc.method = "OEM", fix.loc = fix.loc
))
(prior.par <- mod.fix3$group.par)
(emphist <- getirt(mod.fix3, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# summary of the estimation
summary(mod.fix3)

# fit the 3PL model to all dichotomous items, fit the GRM model to all polytomous data,
# at this moment fix all 55 items and estimate only the latent ability distribution
# using the MEM method.
fix.loc <- c(1:55)
(mod.fix4 <- est_irt(
  x = x, data = sim.dat2, D = 1, EmpHist = TRUE,
  Etol = 1e-3, fipc = TRUE, fipc.method = "MEM", fix.loc = fix.loc
))
(prior.par <- mod.fix4$group.par)
(emphist <- getirt(mod.fix4, what = "weights"))
plot(emphist$weight ~ emphist$theta, type = "h")

# summary of the estimation
summary(mod.fix4)

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

# summary of the estimation
summary(mod.fix4)
```

`est_item` *Fixed ability parameter calibration*

Description

This function performs the fixed ability parameter calibration (FAPC), often called Method A, which is the maximum likelihood estimation of item parameters given the ability estimates (Baker & Kim, 2004; Ban, Hanson, Wang, Yi, & Harris, 2001; Stocking, 1988). Also, this could be considered as a special type of the joint maximum likelihood estimation where only one cycle of

parameter estimation is implemented given the ability estimates (Birnbaum, 1968). FAPC is one of potentially useful online item calibration methods for computerized adaptive testing (CAT) to put the parameter estimates of pretest items on the same scale of operational item parameter estimates and recalibrate the operational items to evaluate the parameter drifts of the operational items (Chen & Wang, 2016; Stocking, 1988).

Usage

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

Arguments

x	A data frame containing the item metadata. This metadata is necessary to obtain the information of each item (i.e., number of score categories and IRT model) to be calibrated. You can easily create an empty item metadata using the function shape_df. When use.startval = TRUE, the item parameters specified in the item metadata are used as the starting values for the item parameter estimation. If x = NULL, the arguments of model and cats must be specified. See irtfit, info or simdat for more details about the item metadata. Default is NULL.
data	A matrix containing examinees' response data for the items in the argument x. A row and column indicate the examinees and items, respectively.
score	A vector of examinees' ability estimates. Length of the vector must be the same as the number of rows in the response data set.

`D` A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

`model` A vector of character strings indicating what IRT model is used to calibrate each item. Available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous items, and "GRM" and "GPCM" for polytomous items. "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. Note that "DRM" is considered as "3PLM" in this function. If a single character of the IRT model is specified, that model will be recycled across all items. The provided information in the `model` argument is used only when `x = NULL`. Default is NULL.

`cats` A numeric vector specifying the number of score categories for each item. For example, a dichotomous item has two score categories. If a single numeric value is specified, that value will be recycled across all items. If `cats = NULL` and all specified models in the `model` argument are the dichotomous models (i.e., 1PLM, 2PLM, 3PLM, or DRM), it assumes that all items have two score categories. The provided information in the `cats` argument is used only when `x = NULL`. Default is NULL.

`item.id` A character vector of item IDs. If NULL, the item IDs are generated automatically. Default is NULL.

`fix.a.1pl` A logical value. If TRUE, the slope parameters of the 1PLM items are fixed to a specific value specified in the argument `a.val.1pl`. Otherwise, the slope parameters of all 1PLM items are constrained to be equal and estimated. Default is FALSE.

`fix.a.gpcm` A logical value. If TRUE, the GPCM items are calibrated with the partial credit model and the slope parameters of the GPCM items are fixed to a specific value specified in the argument `a.val.gpcm`. Otherwise, the slope parameter of each GPCM item is estimated. Default is FALSE.

`fix.g` A logical value. If TRUE, the guessing parameters of the 3PLM items are fixed to a specific value specified in the argument `g.val`. Otherwise, the guessing parameter of each 3PLM item is estimated. Default is FALSE.

`a.val.1pl` A numeric value. This value is used to fixed the slope parameters of the 1PLM items.

`a.val.gpcm` A numeric value. This value is used to fixed the slope parameters of the GPCM items.

`g.val` A numeric value. This value is used to fixed the guessing parameters of the 3PLM items.

`use.aprior` A logical value. If TRUE, a prior distribution for the slope parameters is used for the parameter calibration across all items. Default is FALSE.

`use.bprior` A logical value. If TRUE, a prior distribution for the difficulty (or threshold) parameters is used for the parameter calibration across all items. Default is FALSE.

`use.gprior` A logical value. If TRUE, a prior distribution for the guessing parameters is used for the parameter calibration across all 3PLM items. Default is TRUE.

`aprior` A list containing the information of the prior distribution for item slope parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution is used, "lnorm" should be specified in the first argument. When Normal distribution is used, "norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see `dbeta()`, `dlnorm()`, and `dnorm()` in the **stats** package for more details.

`bprior` A list containing the information of the prior distribution for item difficulty (or threshold) parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution is used, "lnorm" should be specified in the first argument. When Normal distribution is used, "norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see `dbeta()`, `dlnorm()`, and `dnorm()` in the **stats** package for more details.

`gprior` A list containing the information of the prior distribution for item guessing parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution is used, "lnorm" should be specified in the first argument. When Normal distribution is used, "norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see `dbeta()`, `dlnorm()`, and `dnorm()` in the **stats** package for more details.

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

`use.startval` A logical value. If TRUE, the item parameters provided in the item metadata (i.e., the argument `x`) are used as the starting values for the item parameter estimation. Otherwise, internal starting values of this function are used. Default is FALSE.

`control` A list of control parameters to be passed to the optimization function of `nlminb()` in the **stats** package. The control parameters set the conditions of the item parameter estimation process such as the maximum number of iterations. See `nlminb()` in the **stats** package for details.

`verbose` A logical value. If FALSE, all progress messages are suppressed. Default is TRUE.

Details

In most cases, the function `est_item` will return successfully converged item parameter estimates using the default internal starting values. However, if there is a convergence problem in the calibration, one possible solution is using different starting values. When the item parameter values are specified in the item metadata (i.e., the argument `x`), those values can be used as the starting values for the item parameter calibration by setting `use.startval = TRUE`.

Value

This function returns an object of class `est_item`. Within this object, several internal objects are contained such as:

`estimates`	A data frame containing both the item parameter estimates and the corresponding standard errors of estimates.
`par.est`	A data frame containing the item parameter estimates.
`se.est`	A data frame containing the standard errors of the item parameter estimates. Note that the standard errors are estimated using observed information functions.
`pos.par`	A data frame containing the position number of item parameters being estimated. The position information is useful when interpreting the variance-covariance matrix of item parameter estimates.
`covariance`	A matrix of variance-covariance matrix of item parameter estimates.
`loglikelihood`	A sum of the log-likelihood values of the complete data set across all estimated items.
`data`	A data frame of the examinees' response data set.
`score`	A vector of the examinees' ability values used as the fixed effects.
`scale.D`	A scaling factor in IRT models.
`convergence`	A string indicating the convergence status of the item parameter estimation.
`nitem`	A total number of items included in the response data.
`deleted.item`	The items which have no item response data. Those items are excluded from the item parameter estimation.
`npar.est`	A total number of the estimated parameters.
`n.response`	An integer vector indicating the number of item responses for each item used to estimate the item parameters.
`TotalTime`	Time (in seconds) spent for total compuatation.

The internal objects can be easily extracted using the function `getirt`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

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

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

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

See Also

irtfit, info, simdat, shape_df, sx2_fit, traceline.est_item, getirt

Examples

```
## 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

# 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 the response data
data <- simdat(x = x, theta = score, D = 1)


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

# extract the item parameter estimates
```

```
getirt(mod1, what = "par.est")

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

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

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

# extract both item parameter and standard error estimates
getirt(mod2, what = "estimates")
```

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

Description

This function conducts a multiple-group item calibration (Bock & Zimowski, 1997) using the marginal maximum likelihood estimation via the expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981). This function also implements the multiple-group fixed item parameter calibration (MG-FIPC; e.g., Kim & Kolen, 2016). The MG-FIPC is an extension of the single-group FIPC method (Kim, 2006) to multiple-group data. For dichotomous items, IRT one-, two-, and three-parameter logistic models are available. 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 = 200, iter.max = 200),
  fipc = FALSE,
  fipc.method = "MEM",
  fix.loc = NULL,
  fix.id = NULL,
  se = TRUE,
  verbose = TRUE
)
```

Arguments

x	A list containing item metadata across all groups to be analyzed. For example, if five group data are analyzed, the list should include five internal objects of the item metadata for the five groups. The internal objects of the list should be ordered in accordance with the order of the group names specified in the `group.name` argument. An item metadata of each group test from contains the meta information about items (i.e., number of score categories and IRT model) to be calibrated. See `est_irt`, `irtfit`, `info` or `simdat` for more details about the item metadata. When `use.startval = TRUE`, the item parameters specified in the item metadata are used as the starting values for the item parameter estimation. If `x = NULL`, the `model` and `cats` arguments must be specified. Note that when `fipc = TRUE` to implement the MG-FIPC method, a list of item metadata must be provided into `x`. In other words, `x` cannot be NULL in that case. Default is NULL.
data	A list containing item response matrices across all groups to be analyzed. For example, if five group data are analyzed, the list should include five internal objects of the response data matrices for the five groups. The internal objects of the list should be ordered in accordance with the order of the group names specified

in the group.name argument. An item response matrix for each group includes examinees' response data corresponding to the items in the item metadata of that group. In a response data matrix, a row and column indicate the examinees and items, respectively.

group.name A vector of character strings indicating group names. For example, if five group data are analyzed, group.names = c("G1", "G2", "G3", "G4", "G5"). Any group names can be used.

D D A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

model A list containing character vectors of the IRT models used to calibrate items across all groups' data. For example, if five group data are analyzed, the list should include five internal objects of the character vectors for the five groups. The internal objects of the list should be ordered in accordance with the order of the group names specified in the group.name argument. Available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous items, and "GRM" and "GPCM" for polytomous items. "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. Note that "DRM" is considered as "3PLM" in this function. If a single character of the IRT model is provided in the internal objects of the list, that model will be recycled across all items in the corresponding groups. The provided information in the model 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 of items across all groups' data to be analyzed. For example, if five group data are analyzed, the list should include five internal objects of the numeric vectors for the five groups. The internal objects of the list should be ordered in accordance with the order of the group names specified in the group.name argument. If a single numeric value is specified in the internal objects of the list, that value will be recycled across all items in the corresponding groups. If cats = NULL and all specified models in the model argument are the dichotomous models (i.e., 1PLM, 2PLM, 3PLM, or DRM), the function assumes that all items have two score categories across all groups. The provided information in the cats argument is used only when x = NULL and fipc = FALSE. Default is NULL.

item.id A list containing character vectors of item IDs across all groups' data to be analyzed. For example, if five group data are analyzed, the list should include five internal objects of the character vectors of item IDs for the five groups. The internal objects of the list should be ordered in accordance with the order of the group names specified in the group.name argument. When fipc = TRUE and the Item IDs are given by the item.id argument, the Item IDs in the x argument are overridden. Default is NULL.

free.group A numeric or character vector indicating groups in which scales (i.e., a mean and standard deviation) of the latent ability distributions are freely estimated. The scales of other groups not specified in this argument are fixed based on the information provided in the group.mean and group.var arguments, or the weights argument. For example, suppose that five group data are analyzed and the corresponding group names are "G1", "G2", "G3", "G4", and "G5", respectively. If the scales of the 2nd through 5th groups are freely estimated, then free.group

= c(2, 3, 4, 5) or free.group = c("G3", "G4", "G4","G5"). Also, the first group (i.e., "G1") will have the fixed scale (e.g., a mean of 0 and variance of 1 when group.mean = 0, group.var = 1, and weights = NULL).

fix.a.1pl — A logical value. If TRUE, the slope parameters of the 1PLM items are fixed to a specific value specified in the argument a.val.1pl. Otherwise, the slope parameters of all 1PLM items are constrained to be equal and estimated. Default is FALSE.

fix.a.gpcm — A logical value. If TRUE, the GPCM items are calibrated with the partial credit model and the slope parameters of the GPCM items are fixed to a specific value specified in the argument a.val.gpcm. Otherwise, the slope parameter of each GPCM item is estimated. Default is FALSE.

fix.g — A logical value. If TRUE, the guessing parameters of the 3PLM items are fixed to a specific value specified in the argument g.val. Otherwise, the guessing parameter of each 3PLM item is estimated. Default is FALSE.

a.val.1pl — A numeric value. This value is used to fixed the slope parameters of the 1PLM items.

a.val.gpcm — A numeric value. This value is used to fixed the slope parameters of the GPCM items.

g.val — A numeric value. This value is used to fixed the guessing parameters of the 3PLM items.

use.aprior — A logical value. If TRUE, a prior distribution for the slope parameters is used for the parameter calibration across all items. Default is FALSE.

use.bprior — A logical value. If TRUE, a prior distribution for the difficulty (or threshold) parameters is used for the parameter calibration across all items. Default is FALSE.

use.gprior — A logical value. If TRUE, a prior distribution for the guessing parameters is used for the parameter calibration across all 3PLM items. Default is TRUE.

aprior — A list containing the information of the prior distribution for item slope parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution is used, "lnorm" should be specified in the first argument. When Normal distribution is used, "norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see dbeta(), dlnorm(), and dnorm() in the **stats** package for more details.

bprior — A list containing the information of the prior distribution for item difficulty (or threshold) parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution

is used, "lnorm" should be specified in the first argument. When Normal distribution is used, "norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see `dbeta()`, `dlnorm()`, and `dnorm()` in the **stats** package for more details.

`gprior` A list containing the information of the prior distribution for item guessing parameters. Three probability distributions of Beta, Log-normal, and Normal distributions are available. In the list, a character string of the distribution name must be specified in the first internal argument and a vector of two numeric values for the two parameters of the distribution must be specified in the second internal argument. Specifically, when Beta distribution is used, "beta" should be specified in the first argument. When Log-normal distribution is used, "lnorm" should be specified in the first argument. When Normal distribution is used, "norm" should be specified in the first argument. In terms of the two parameters of the three distributions, see `dbeta()`, `dlnorm()`, and `dnorm()` in the **stats** package for more details.

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

`Quadrature` A numeric vector of two components specifying the number of quadrature points (in the first component) and the symmetric minimum and maximum values of these points (in the second component). For example, a vector of c(49, 6) indicates 49 rectangular quadrature points over -6 and 6. The quadrature points are used in the E step of the EM algorithm. Default is c(49, 6).

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the latent variable prior distribution. If not NULL, the scale of the latent ability distributions of the groups that are not freely estimated (i.e., the groups not specified in the `free.group` argument) are fixed to the scale of the provided quadrature points and weights. The weights and quadrature points can be easily obtained using the function `gen.weight`. If NULL, a normal prior density is used based on the information provided in the arguments of `Quadrature`, `group.mean`, and `group.var`). Default is NULL.

`group.mean` A numeric value to set the mean of latent variable prior distribution when `weights = NULL`. Default is 0. The means of the distributions of the groups that are not specified in the `free.group` are fixed to the value provided in this argument to remove the indeterminancy of item parameter scales.

`group.var` A positive numeric value to set the variance of latent variable prior distribution when `weights = NULL`. Default is 1. The variances of the distributions of the groups that are not specified in the `free.group` are fixed to the value provided in this argument to remove the indeterminancy of item parameter scales.

`EmpHist` A logical value. If TRUE, the empirical histograms of the latent variable prior distributions across all groups are simultaneously estimated with the item parameters using Woods's (2007) approach. The items are calibrated against the estimated empirical prior distributions.

`use.startval` A logical value. If TRUE, the item parameters provided in the item metadata (i.e., the argument `x`) are used as the starting values for the item parameter estimation. Otherwise, internal starting values of this function are used. Default is FALSE.

`Etol` A positive numeric value. This value sets the convergence criterion for E steps of the EM algorithm. Default is 1e-3.

`MaxE` A positive integer value. This value determines the maximum number of the E steps in the EM algorithm. Default is 500.

`control` A list of control parameters to be passed to the optimization function of `nlminb()` in the **stats** package. The control parameters set the conditions of M steps of the EM algorithm. For example, the maximum number of iterations in each of the iterative M steps can be set by `control = list(iter.max=200)`. Default maximum number of iterations in each M step is 200. See `nlminb()` in the **stats** package for other control parameters.

`fipc` A logical value. If TRUE, the MG-FIPC is implemented for item parameter estimation. When `fipc = TRUE`, the information of which items are fixed needs to be provided via either of `fix.loc` or `fix.id`. See below for details.

`fipc.method` A character string specifying the FIPC method. Available methods include "OEM" for "No Prior Weights Updating and One EM Cycle (NWU-OEM; Wainer & Mislevy, 1990)" and "MEM" for "Multiple Prior Weights Updating and Multiple EM Cycles (MWU-MEM; Kim, 2006)." When `fipc.method = "OEM"`, the maximum number of the E steps of the EM algorithm is set to 1 no matter what number is specified in the argument `MaxE`.

`fix.loc` A list of positive integer vectors. Each internal integer vector indicates the locations of the items to be fixed in the item metadata (i.e., `x`) for each group when the MG-FIPC is implemented (i.e., `fipc = TRUE`). The internal objects of the list should be ordered in accordance with the order of group names specified in the `group.name` argument. For example, suppose that three group data are analyzed. For the first group data, the 1st, 3rd, and 5th items are fixed, for the second group data, the 2nd, 3rd, 4th, and 7th items are fixed, and for the third group data, the 1st, 2nd, and 6th items are fixed. Then `fix.loc = list(c(1, 3, 5), c(2, 3, 4, 7), c(1, 2, 6))`. Note that when the `fix.id` argument is not NULL, the information provided into the `fix.loc` argument is ignored. See below for details.

`fix.id` A vector of character strings specifying IDs of the items to be fixed when the MG-FIPC is implemented (i.e., `fipc = TRUE`). For example, suppose that three group data are analyzed. For the first group data, three items in which IDs are G1I1, C1I1, are C1I2 are fixed. For the second group data, four items in which IDs are C1I1, C1I2, C2I1, and C2I2 are fixed. For the third group data, three items in which IDs are C2I1, C2I2, and G3I1 are fixed. Then there are six unique items to be fixed across the three groups (i.e., G1I1, C1I1, C1I2, C2I1, C2I2, and G3I1) because C1I1 and C1I2 are common items between the first and the second groups, and C2I1 and C2I2 are common items between the second and the third groups. Thus, `fix.id = c("G1I1", "C1I1", "C1I2", "C2I1", "C2I2", "G3I1")` should be specified. Note that when the `fix.id` argument is not NULL, the information provided into the `fix.loc` argument is ignored. See below for details.

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

`verbose` A logical value. If FALSE, all progress messages including the process information on the EM algorithm are suppressed. Default is TRUE.

Details

The multiple-group (MG) item calibration (Bock & Zimowski, 1996) provides a unified approach to the testing situations or problems involving multiple groups such as nonequivalent groups equating, vertical scaling, and identification of differential item functioning (DIF). In such contexts, typically there exist test takers from different groups responding to the same test form or to the common items shared between different test forms.

The goal of the MG item calibration is to estimate the item parameters and the latent ability distributions of the multiple groups simultaneously (Bock & Zimowski, 1996). In the **irtQ** package, the `est_mg` function supports the MG item calibration using the marginal maximum likelihood estimation via the expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981). In addition, The function also supports the MG fixed item parameter calibration (MG-FIPC; e.g., Kim & Kolen, 2016) if the parameters of certain items need to be fixed across multiple test forms,

In MG IRT analyses, it is commonly seen that the test forms of multiple groups share some common (or anchor) items between the groups. By default the common items that have the same item IDs between different groups are automatically constrained so that they can have the same item parameter estimates across the groups in the `est_mg` function.

Most of the features of the `est_mg` function are similar with those of the `est_irt` function. The main difference is that several arguments in the `est_mg` functions take an object of a list format which contains several internal objects corresponding to the groups to be analyzed. Those arguments include `x`, `data`, `model`, `cats`, `item.id`, and `fix.loc`.

Also, the `est_mg` has two new arguments, the `group.name` and `free.group`. The `group.name` is required to assign a unique group name to each group. The order of internal objects in the lists provided in the `x`, `data`, `model`, `cats`, `item.id`, and `fix.loc` arguments must match that of the group names supplied to the `group.name` argument.

The `free.group` argument is necessary to indicate the freed groups in which scales (i.e., a mean and standard deviation) of the latent ability distributions are estimated. When no item parameters are fixed (i.e., `fipc = FALSE`), at least one group should have a fixed scale (e.g., a mean of 0 and variance of 1) of the latent ability distribution among the multiple groups that shares common items in order to solve the scale indeterminancy in IRT estimation. By providing which are the freed groups into the `free.group` argument, the scales of the groups specified in the `free.group` argument are freely estimated while the scales of all other groups not specified in the argument are fixed based on the information provided in the `group.mean` and `group.var` arguments or the `weights` argument.

The situations where the implementation of MG-FIPC is necessary arise when the new latent ability scales from MG test data are linked to the established scale (e.g., the scale of an item bank). In a single run of MG-FIPC method, all the parameters of freed items across multiple test forms and the density of the latent ability distributions for multiple groups can be estimated on the scale of the fixed items (Kim & Kolen, 2016). 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, and Form 2 and Form 3 share 10 common items, C2I1 to C2I10, while there is no common item between Form 1 and Form 3. Also, suppose that all unique items of Form 1 came from an item bank, which were already calibrated on the scale of the item bank. In this case, the goal of the MG-FIPC is to estimate the parameters of all the items across the three test forms, except the unique items in Form 1, and the latent ability distributions of the three groups on the same scale of the item bank. To accomplish this task, the unique items in Form 1 need to be fixed during the MG-FIPC to link the current MG test data and the item bank.

The est_mg function can implement the MG-FIPC by setting fipc = TRUE. Then, the information of which items are fixed needs to be supplied via either of fix.loc or fix.id. When utilizing the fix.loc argument, a list of the locations of the items that are fixed in each group test form should be prepared. For example, suppose that three group data are analyzed. For the first group test form, the 1st, 3rd, and 5th items are fixed, for the second group test form, the 2nd, 3rd, 4th, and 7th items are fixed, and for the third group test form, the 1st, 2nd, and 6th items are fixed. Then fix.loc = list(c(1, 3, 5), c(2, 3, 4, 7), c(1, 2, 6)). Instead of using the fix.loc, the fix.id argument can be used by providing a vector of the IDs of the items to be fixed. Again suppose that the three group data are analyzed. For the first group test form, the three items in which IDs are G1I1, C1I1, are C1I2 are fixed. For the second group test form, the four items in which IDs are C1I1, C1I2, C2I1, and C2I2 are fixed. For the third group test form, the three items in which IDs are C2I1, C2I2, and G3I1 are fixed. Then there are six unique items to be fixed across the three groups (i.e., G1I1, C1I1, C1I2, C2I1, C2I2, and G3I1) because C1I1 and C1I2 are common items between the first and the second groups, and C2I1 and C2I2 are common items between the second and the third groups. Thus, fix.id = c("G1I1", "C1I1", "C1I2", "C2I1", "C2I2", "G3I1") should be set. Note that when the information of item IDs supplied in the fix.id argument overrides the information provided into the fix.loc argument.

Value

This function returns an object of class est_mg. Within this object, several internal objects are contained such as:

estimates	A list containing two internal objects (i.e., overall and group) of the item parameter estimates and the corresponding standard errors of estimates. The first internal object (overall) is a data frame of the item parameter and standard error estimates for the combined data set across all groups. Accordingly, the data frame includes unique items across all groups. The second internal object (group) is a list of group specific data frames containing item parameter and standard error estimates
par.est	A list containing two internal objects (i.e., overall and group) of the item parameter estimates. The format of the list is the same with the internal object of 'estimates'
se.est	A list containing two internal objects (i.e., overall and group) of the standard errors of item parameter estimates. The format of the list is the same with the internal object of 'estimates'. Note that the standard errors are estimated using the cross-production approximation method (Meilijson, 1989).
pos.par	A data frame containing the position number of each item parameter being estimated. This item position data frame was created based on the combined data sets across all groups (see the first internal object of 'estimates'). The position information is useful when interpreting the variance-covariance matrix of item parameter estimates.
covariance	A matrix of variance-covariance matrix of item parameter estimates. This matrix was created based on the combined data sets across all groups (see the first internal object of 'estimates')
loglikelihood	A list containing two internal objects (i.e., overall and group) of the log-likelihood values of observed data set (marginal log-likelihood). The format of the list is the same with the internal object of 'estimates'. Specifically, the first internal

	object (overall) contains a sum of the log-likelihood values of the observed data set across all unique items of all groups. The second internal object (group) shows the group specific log-likelihood values.
aic	A model fit statistic of Akaike information criterion based on the loglikelihood of all unique items..
bic	A model fit statistic of Bayesian information criterion based on the loglikelihood of all unique items.
group.par	A list containing the summary statistics (i.e., a mean, variance, and standard deviation) of latent variable prior distributions across all groups.
weights	a list of the two-column data frames containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the (updated) latent variable prior distributions for all groups.
posterior.dist	A matrix of normalized posterior densities for all the response patterns at each of the quadrature points. The row and column indicate each individual's response pattern and the quadrature point, respectively.
data	A list containing two internal objects (i.e., overall and group) of the examinees' response data sets. The format of the list is the same with the internal object of 'estimates'.
scale.D	A scaling factor in IRT models.
ncase	A list containing two internal objects (i.e., overall and group) with the total number of response patterns. The format of the list is the same with the internal object of 'estimates'.
nitem	A list containing two internal objects (i.e., overall and group) with the total number of items included in the response data set. The format of the list is the same with the internal object of 'estimates'.
Etol	A convergence criteria for E steps of the EM algorithm.
MaxE	The maximum number of E steps in the EM algorithm.
aprior	A list containing the information of the prior distribution for item slope parameters.
gprior	A list containing the information of the prior distribution for item guessing parameters.
npar.est	A total number of the estimated parameters across all unique items.
niter	The number of EM cycles completed.
maxpar.diff	A maximum item parameter change when the EM cycles were completed.
EMtime	Time (in seconds) spent for the EM cycles.
SEtime	Time (in seconds) spent for computing the standard errors of the item parameter estimates.
TotalTime	Time (in seconds) spent for total compuatation.
test.1	Status of the first-order test to report if the gradients has vanished sufficiently for the solution to be stable.
test.2	Status of the second-order test to report if the information matrix is positive definite, which is a prerequisite for the solution to be a possible maximum.

var.note	A note to report if the variance-covariance matrix of item parameter estimates is obtainable from the information matrix.
fipc	A logical value to indicate if FIPC was used.
fipc.method	A method used for the FIPC.
fix.loc	A list containing two internal objects (i.e., overall and group) with the locations of the fixed items when the FIPC was implemented. The format of the list is the same with the internal object of 'estimates'.

The internal objects can be easily extracted using the function getirt.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

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

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

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

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

See Also

est_irt, shape_df, getirt

Examples

```
## ------------------------------------------------------------------------------
# 1. MG-calibration using simMG data
#  - Details :
#    (a) constrain the common items between the groups to have
#        the same 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 ability
#        distributions except the reference group in which mean
#        and variance are fixed to 0 and 1, respectively.
## ------------------------------------------------------------------------------
# 1-(1). freely estimates the means and variances of groups 2 and 3
# import the true item metadata of 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 of the three groups
data <- simMG$res.dat

# import the group names of the three groups
group.name <- simMG$group.name

# set the groups 2 and 3 as the free groups in which scale
# of the ability distributions will be freely estimated.
# then, the group 1 will be a reference group in which the scale
# of the ability distribution will be fixed to the values specified
# in the 'group.mean' and 'group.var' arguments
free.group <- c(2, 3) # or use 'free.group <- group.name[2:3]'

# estimate the IRT parameters:
# as long as the common items between any groups have the same item IDs,
# their item parameters will be constrained to be the same between
# the groups unless the FIPC is not 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 parameter estimates (i.e., scale parameters)
getirt(fit.1, what = "group.par")

# extract the posterior latent ability distributions of the groups
getirt(fit.1, what = "weights")

# 1-(2). or the same parameter estimation can be implemented by
# inserting a list of item metadata for the groups into the 'x'
# argument. if the item metadata contains the item parameters
# which you want to use as starting values for the estimation,
# you can set 'use.startval = TRUE'.
# also set the groups in which scales of ability distributions
```

```
# will be freely estimated using the 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 the 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 ability
#        distributions of all three groups
## ------------------------------------------------------------------------------
# 2-(1). freely estimates the means and variances of all three groups
# set all three groups as the free groups in which scale
# of the ability distributions will be freely estimated.
free.group <- 1:3 # or use 'free.group <- group.name'

# specify the locations of items to be fixed in each group data
# note that for the first group, the common items C1I1 - C1I12 are located
# in the 1st - 10th and 11th to 12th rows of the first group's item metadata
# for the second group, the common items C1I1 - C1I12 are located
# in the 1st - 12th rows of the second group's item metadata
# also, for the second group, the common items C2I1 - C2I10 are located
# in the 41st - 50th rows of the second group's item metadata
# for the third group, the common items C2I1 - C2I10 are located
# in the 1st - 10th rows of the third group's item metadata
fix.loc <- list(
  c(1:10, 49:50),
  c(1:12, 41:50),
  c(1:10)
)

# estimate the IRT parameters using FIPC:
# when the FIPC is implemented, a list of the item metadata for all
# groups must be provided into the 'x' argument.
# for the fixed items, their item parameters must be specified
# in the item metadata
# for other non-fixed items, any parameter values can be provided
# in the item 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). or the FIPC can be implemented by providing which items
# will be fixed in a different way using the 'fix.id' argument.
# a vector of the fixed item IDs needs to be provided into the
# 'fix.id' argument as such.
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 the FIPC using simMG data
#    (estimate the group parameters only)
#  - Details :
#    (a) fix all item parameters across all three groups
#    (b) freely estimate the means and variances of ability
#        distributions of all three groups
## ----------------------------------------------------------------------------
# 3-(1). freely estimates the means and variances of all three groups
# set all three groups as the free groups in which scale
# of the ability distributions will be freely estimated.
free.group <- 1:3 # or use 'free.group <- group.name'

# specify the locations of all item parameters to be fixed
```

```
# in each item metadata
fix.loc <- list(1:50, 1:50, 1:38)

# estimate the group parameters only using 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 the FIPC using simMG data
#    (fix the unique items of the group 1 only)
#  - Details :
#    (a) fix item parameters of unique items in the group 1 only
#    (b) constrain the common items between the groups to have
#        the same item parameters (i.e., items C1I1 - C1I12 between
#        Groups 1 and 2, and items C2I1 - C2I10 between Groups 2 and 3)
#    (c) freely estimate the means and variances of ability
#        distributions of all three groups
## -----------------------------------------------------------------------------
# 4-(1). freely estimates the means and variances of all three groups
# set all three groups as the free groups in which scale
# 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 the group 1 to be fixed
# in each item metadata using the 'fix.id' argument or
# you can use the 'fix.loc' argument by setting
# 'fix.loc = list(11:48, NULL, NULL)'
fix.id <- paste0("G1I", 1:38)

# estimate the IRT parameters using 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 are maximum likelihood estimation (ML), maximum likelihood estimation with fences (MLF; Han, 2016), weighted likelihood estimation (Warm, 1989), maximum a posteriori estimation (MAP; Hambleton et al., 1991), expected a posteriori estimation (EAP; Bock & Mislevy, 1982), EAP summed scoring (Thissen et al., 1995; Thissen & Orlando, 2001), and inverse test characteristic curve (TCC) scoring (e.g., Kolen & Brennan, 2004; Kolen & Tong, 2010; Stocking, 1996).

Usage

```
est_score(x, ...)

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

## S3 method for class 'est_irt'
```

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

Arguments

x	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...) or an object of class est_irt obtained from the function est_irt. See irtfit, info, or simdat for more details about the item metadata. This data frame can be easily obtained using the function shape_df.
...	additional arguments to pass to parallel::makeCluster.
data	A matrix or vector containing examinees' response data for the items in the argument x. When a matrix is used, a row and column indicate the examinees and items, respectively. When a vector is used, it should contains the item response data for an examinee.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
method	A character string indicating a scoring method. Available methods are "ML" for the maximum likelihood estimation, "MLF" for the maximum likelihood estimation with fences, "WL" for the weighted likelihood estimation, "MAP" for the maximum a posteriori estimation, "EAP" for the expected a posteriori estimation, "EAP.SUM" for the expected a posteriori summed scoring, and "INV.TCC" for the inverse TCC scoring. Default method is "ML".
range	A numeric vector of two components to restrict the range of ability scale for the ML, MLF, WL, and MAP scoring methods. Default is c(-5, 5).
norm.prior	A numeric vector of two components specifying a mean and standard deviation of the normal prior distribution. These two parameters are used to obtain the gaussian quadrature points and the corresponding weights from the normal distribution. Default is c(0,1). Ignored if method is "ML", "MLF", "WL", or "INV.TCC".

nquad — An integer value specifying the number of gaussian quadrature points from the normal prior distribution. Default is 41. Ignored if method is "ML", "MLF", "WL", "MAP", or "INV.TCC".

weights — A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the latent variable prior distribution. The weights and quadrature points can be easily obtained using the function gen.weight. If NULL and method is "EAP" or "EAP.SUM", default values are used (see the arguments of norm.prior and nquad). Ignored if method is "ML", "MLF", "WL", "MAP", or "INV.TCC".

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

fence.b — A numeric vector of two components specifying the lower and upper fences of item difficulty parameters (i.e., *b*-parameters) for the two imaginary items, respectively, in MLF. When fence.b = NULL, the range values were used to set the lower and upper fences of item difficulty parameters. Default is NULL.

tol — A numeric value of the convergent tolerance for the ML, MLF, WL, MAP, and inverse TCC scoring methods. For the ML, MLF, WL, and MAP, Newton Raphson method is implemented for optimization. For the inverse TCC scoring, the bisection method is used. Default is 1e-4.

max.iter — An positive integer value specifying the maximum number of iterations of Newton Raphson method. Default is 100.

se — A logical value. If TRUE, the standard errors of ability estimates are computed. However, if method is "EAP.SUM" or "INV.TCC", the standard errors are always returned. Default is TRUE.

stval.opt — An positive integer value specifying the starting value option for the ML, MLF, WL, and MAP scoring methods. Available options are 1 for the brute-force method, 2 for the observed sum score-based method, and 3 for setting to 0. Default is 1. See below for details.

intpol — A logical value. If TRUE and method = "INV.TCC", linear interpolation method is used to approximate the ability estimates corresponding to the observed sum scores in which ability estimates cannot be obtained using the TCC (e.g., observed sum scores less than the sum of item guessing parameters). Default is TRUE. See below for details.

range.tcc — A numeric vector of two components to be used as the lower and upper bounds of ability estimates when method = "INV.TCC". Default is c(-7, 7).

missing — A value indicating missing values in the response data set. Default is NA. See below for details.

ncore — The number of logical CPU cores to use. Default is 1. See below for details.

Details

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

When there are missing data in the response data set, the missing value must be specified in missing. The missing data are taken into account when either of ML, MLF, WL, MAP, and EAP is

used. When "EAP.SUM" or "INV.TCC" is used, however, any missing responses are replaced with incorrect responses (i.e., 0s).

In the maximum likelihood estimation with fences (MLF; Han, 2016), two 2PLM imaginary items are necessary. The first imaginary item serves as the lower fence and its difficulty parameter (i.e., b-parameters) should be lower than any difficulty parameter values in the test form. Likewise, the second imaginary item serves as the upper fence and its difficulty parameter should be greater than any difficulty parameter values in the test form. Also, the two imaginary items should have a very high item slope parameter (i.e., a-parameter) value. See Han (2016) for more details. When `fence.b = NULL` in MLF, the function automatically sets the lower and upper fences of item difficulty parameters using the values in the `range` argument.

When "INV.TCC" method is used employing the IRT 3-parameter logistic model (3PLM) in a test, ability estimates for the observed sum scores less than the sum of item guessing parameters are not attainable. In this case, linear interpolation can be applied by setting `intpol = TRUE`. Let θ_{min} and θ_{max} be the minimum and maximum ability estimates and θ_X be the ability estimate for the smallest observed sum score, X, but greater than or equal to the sum of item guessing parameters. When linear interpolation method is used, the first value of the `range.tcc` is set to θ_{min}. Then, a linear line is constructed between two points of (x=θ_{min}, y=0) and (x=θ_X, y=X). Also, the first value of the `range.tcc` is set to θ_{max}, which is the ability estimates corresponding to the maximum observed sum score. When it comes to the scoring method of "INV.TCC", the standard errors of ability estimates are computed using an approach suggested by Lim, Davey, and Wells (2020). The code for the inverse TCC scoring was written by modifying the function `irt.eq.tse` of the **SNSequate** R package (González, 2014).

In terms of the starting value to be used for ML, MLF, WL, and MAP scoring methods, the brute-force method is used when `stval.opt = 1`. With this option, the log-likelihood values were evaluated at the discrete theta values with increments of 0.1 given `range`. The theta node that has the largest log-likelihood is used as the starting value. when `stval.opt = 2`, the starting value is obtained based on the observed sum score. For example, if the maximum observed sum score (max.score) is 30 for a test and an examinee has an observed sum score of 20 (obs.score), then the starting value is "log(obs.score / (max.score - obs.score))". For all incorrect response, the starting value is "log(1 / max.score)" and for all correct responses, it is "log(max.score / 1)".

To speed up the ability estimation for ML, MLF, WL, MAP, and EAP methods, this function applies a parallel process using multiple logical CPU cores. You can set the number of logical CPU cores by specifying a positive integer value in the argument `ncore`. Default value is 1.

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

To implement WL method, the `Pi`, `Ji`, and `Ii` functions of **catR** (Magis & Barrada, 2017) were referred.

Value

When `method` is either of "ML", "MLF", "WL", "MAP", or "EAP", a two column data frame including the ability estimates (1st column) and the standard errors of ability estimates (2nd column) is returned. When `method` is "EAP.SUM" or "INV.TCC", a list of two internal objects are returned. The first object is a three column data frame including the observed sum scores (1st column), the ability estimates (2nd column), and the standard errors of ability estimates (3rd column). The second object is a score table including the possible raw sum scores and corresponding ability and standard error estimates.

Methods (by class)

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

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

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

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

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

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

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

Lim, H., Davey, T., & Wells, C. S. (2020). A recursion-based analytical approach to evaluate the performance of MST. *Journal of Educational Measurement, 58*(2), 154-178.

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

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

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

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

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

See Also

`irtfit`, `info`, `simdat`, `shape_df`, `gen.weight`

Examples

```
## the use of a "-prm.txt" file obtained from a flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# read item parameters and transform them to item metadata
x <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# generate examinees abilities
set.seed(12)
theta <- rnorm(10)

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


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

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

# estimate the abilities using MLF with default fences of item difficulty parameters
est_score(x, data, D = 1, method = "MLF", fence.a = 3.0, fence.b = NULL, se = TRUE)

# estimate the abilities using MLF with different fences of item difficulty parameters
est_score(x, data, D = 1, method = "MLF", fence.a = 3.0, fence.b = c(-7, 7), se = TRUE)

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

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

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

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

---

gen.weight *Generate Weights*

Description

This function generates a set of weights based on a set of theta values to be used in the functions est_score and sx2_fit.

Usage

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

Arguments

n	An integer identifying the number of theta (or node) values for which weights are generated. Default is 41.
dist	A character string specifying a probability distribution from which the weights are generated. Available distributions are "norm" for a normal distribution, "unif" for a uniform distribution, and "emp" for an empirical distribution. When dist = "norm", either n or theta can be specified, when dist = "unif", only n can be used, and when dist = "emp", only theta can be used.
mu, sigma	A mean and standard deviation of a normal distribution.
l, u	Lower and upper limits of a uniform distribution.
theta	A vector of empirical theta (or node) values for which weights are generated.

Details

When the argument theta is missing, *n* weights can be generated from either the normal distribution or the uniform distribution. Note that if dist = "norm", gaussian quadrature points and weights from the normal distribution are generated. See gauss.quad.prob() in the **statmod** package for more details.

When the argument theta is not missing, the weights corresponding to the provided theta values are generated. Specifically, if dist = "norm", normalized weights from the normal distribution are returned. If dist = "emp", every specified theta value has the equal values of normalized weights.

Value

This function returns a data frame with two columns, where the first column has theta values (nodes) and the second column provides weights.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

est_score, sx2_fit

Examples

```
## example 1
## generate 41 gaussian quadrature points and weights of normal distribution
gen.weight(n = 41, dist = "norm", mu = 0, sigma = 1)

## example 2
## generate 41 theta values and weights from the uniform normal distribution,
## given the mininum value of -4 and the maximum value of 4
```

```
gen.weight(n = 41, dist = "unif", l = -4, u = 4)

## example 3
## generate the normalized weights from the standardized normal distribution,
## given a set of theta values
theta <- seq(-4, 4, by = 0.1)
gen.weight(dist = "norm", mu = 0, sigma = 1, theta = theta)

## example 4
## generate the same values of normalized weights for the theta values that are
## randomly sampled from the standardized normal distribution
theta <- rnorm(100)
gen.weight(dist = "emp", theta = theta)
```

`getirt` *Extract various elements from 'est_irt', 'est_mg', and 'est_item' objects*

Description

This method extracts various internal objects from an object of class `est_irt`, `est_mg`, or `est_item`.

Usage

```
getirt(x, ...)

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

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

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

Arguments

`x`	An object of class `est_irt`, `est_mg`, or `est_item`.
`...`	Further arguments passed to or from other methods.
`what`	A character string indicating what to extract.

Details

Objects which can be extracted from the object of class `est_irt` include:

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

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

se.est A data frame containing the standard errors of the item parameter estimates. Note that the standard errors are estimated using the cross-production approximation method (Meilijson, 1989).

pos.par A data frame containing the position number of each item parameter being estimated. The position information is useful when interpreting the variance-covariance matrix of item parameter estimates.

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

loglikelihood A sum of the log-likelihood values of the observed data set (marginal log-likelihood) across all items in the data set.

aic A model fit statistic of Akaike information criterion based on the loglikelihood.

bic A model fit statistic of Bayesian information criterion based on the loglikelihood.

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

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

posterior.dist A matrix of normalized posterior densities for all the response patterns at each of the quadrature points. The row and column indicate each individual's response pattern and the quadrature point, respectively.

data A data frame of the examinees' response data set.

scale.D A scaling factor in IRT models.

ncase A total number of response patterns.

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

Etol A convergence criteria for E steps of the EM algorithm.

MaxE The maximum number of E steps in the EM algorithm.

aprior A list containing the information of the prior distribution for item slope parameters.

bprior A list containing the information of the prior distribution for item difficulty (or threshold) parameters.

gprior A list containing the information of the prior distribution for item guessing parameters.

npar.est A total number of the estimated parameters.

niter The number of EM cycles completed.

maxpar.diff A maximum item parameter change when the EM cycles were completed.

EMtime Time (in seconds) spent for the EM cycles.

SEtime Time (in seconds) spent for computing the standard errors of the item parameter estimates.

TotalTime Time (in seconds) spent for total compuatation.

test.1 Status of the first-order test to report if the gradients has vanished sufficiently for the solution to be stable.

test.2 Status of the second-order test to report if the information matrix is positive definite, which is a prerequisite for the solution to be a possible maximum.

var.note A note to report if the variance-covariance matrix of item parameter estimates is obtainable from the information matrix.

fipc A logical value to indicate if FIPC was used.

fipc.method A method used for the FIPC.

fix.loc A vector of integer values specifying the locations of the fixed items when the FIPC was implemented.

Objects which can be extracted from the object of class `est_mg` include:

estimates A list containing two internal objects (i.e., overall and group) of the item parameter estimates and the corresponding standard errors of estimates. The first internal object (overall) is a data frame of the item parameter and standard error estimates for the combined data set across all groups. Accordingly, the data frame includes unique items across all groups. The second internal object (group) is a list of group specific data frames containing item parameter and standard error estimates

par.est A list containing two internal objects (i.e., overall and group) of the item parameter estimates. The format of the list is the same with the internal object of 'estimates'

se.est A list containing two internal objects (i.e., overall and group) of the standard errors of item parameter estimates. The format of the list is the same with the internal object of 'estimates'. Note that the standard errors are estimated using the cross-production approximation method (Meilijson, 1989).

pos.par A data frame containing the position number of each item parameter being estimated. This item position data frame was created based on the combined data sets across all groups (see the first internal object of 'estimates'). The position information is useful when interpreting the variance-covariance matrix of item parameter estimates.

covariance A matrix of variance-covariance matrix of item parameter estimates. This matrix was created based on the combined data sets across all groups (see the first internal object of 'estimates')

loglikelihood A list containing two internal objects (i.e., overall and group) of the log-likelihood values of observed data set (marginal log-likelihood). The format of the list is the same with the internal object of 'estimates'. Specifically, the first internal object (overall) contains a sum of the log-likelihood values of the observed data set across all unique items of all groups. The second internal object (group) shows the group specific log-likelihood values.

aic A model fit statistic of Akaike information criterion based on the loglikelihood of all unique items..

bic A model fit statistic of Bayesian information criterion based on the loglikelihood of all unique items.

group.par A list containing the summary statistics (i.e., a mean, variance, and standard deviation) of latent variable prior distributions across all groups.

weights a list of the two-column data frames containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the (updated) latent variable prior distributions for all groups.

posterior.dist A matrix of normalized posterior densities for all the response patterns at each of the quadrature points. The row and column indicate each individual's response pattern and the quadrature point, respectively.

data A list containing two internal objects (i.e., overall and group) of the examinees' response data sets. The format of the list is the same with the internal object of 'estimates'.

scale.D A scaling factor in IRT models.

ncase A list containing two internal objects (i.e., overall and group) with the total number of response patterns. The format of the list is the same with the internal object of 'estimates'.

nitem A list containing two internal objects (i.e., overall and group) with the total number of items included in the response data set. The format of the list is the same with the internal object of 'estimates'.

Etol A convergence criteria for E steps of the EM algorithm.

MaxE The maximum number of E steps in the EM algorithm.

aprior A list containing the information of the prior distribution for item slope parameters.

gprior A list containing the information of the prior distribution for item guessing parameters.

npar.est A total number of the estimated parameters across all unique items.

niter The number of EM cycles completed.

maxpar.diff A maximum item parameter change when the EM cycles were completed.

EMtime Time (in seconds) spent for the EM cycles.

SEtime Time (in seconds) spent for computing the standard errors of the item parameter estimates.

TotalTime Time (in seconds) spent for total compuatation.

test.1 Status of the first-order test to report if the gradients has vanished sufficiently for the solution to be stable.

test.2 Status of the second-order test to report if the information matrix is positive definite, which is a prerequisite for the solution to be a possible maximum.

var.note A note to report if the variance-covariance matrix of item parameter estimates is obtainable from the information matrix.

fipc A logical value to indicate if FIPC was used.

fipc.method A method used for the FIPC.

fix.loc A list containing two internal objects (i.e., overall and group) with the locations of the fixed items when the FIPC was implemented. The format of the list is the same with the internal object of 'estimates'.

Objects which can be extracted from the object of class `est_item` include:

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

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

se.est A data frame containing the standard errors of the item parameter estimates. Note that the standard errors are estimated using observed information functions.

pos.par A data frame containing the position number of each item parameter being estimated. The position information is useful when interpreting the variance-covariance matrix of item parameter estimates.

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

loglikelihood A sum of the log-likelihood values of the complete data set across all estimated items.

data A data frame of the examinees' response data set.

score A vector of the examinees' ability values used as the fixed effects.

scale.D A scaling factor in IRT models.

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

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

deleted.item The items which have no item response data. Those items are excluded from the item parameter estimation.

npar.est A total number of the estimated parameters.

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

TotalTime Time (in seconds) spent for total compuatation.

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

Value

The internal objects extracted from an object of class `est_irt`, `est_mg`, or `est_item`.

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_item`

Examples

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

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

# extract the 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, Zhu, Choe, & Han, 2023). They are specialized to capture uniform DIF, nonuniform DIF, and mixed DIF, respectively.

Usage

```
grdif(x, ...)

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

## S3 method for class 'est_irt'
grdif(
  x,
  score = NULL,
  group,
  focal.name,
  alpha = 0.05,
  missing = NA,
```

```
  purify = FALSE,
  purify.by = c("grdifrs", "grdifr", "grdifs"),
  max.iter = 10,
  min.resp = NULL,
  post.hoc = TRUE,
  method = "ML",
  range = c(-4, 4),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)

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

Arguments

x	A data frame containing item metadata (e.g., item parameters, number of categories, models, etc.), an object of class est_item obtained from the function est_item, or an object of class est_irt obtained from the function est_irt. The item metadata can be easily created using the function shape_df. See est_irt, irtfit, info or simdat for more details about the item metadata.
...	Additional arguments that will be forwarded to the est_score function.
data	A matrix containing examinees' response data for items in x. Rows and columns represent examinees and items, respectively.
score	A vector of examinees' ability estimates. If abilities are not provided, grdif

estimates abilities before computing GRDIF statistics. See `est_score` for more details about scoring methods. Default is NULL.

`group` A numeric or character vector indicating group membership of examinees. The length of the vector should be the same as the number of rows in the response data matrix.

`focal.name` A character or numeric vector representing levels associated with focal groups. For instance, consider `group = c(0, 0, 1, 2, 2, 3, 3)` where '1', '2', and '3' indicate three distinct focal groups, and '0' represents the reference group. In this case, set `focal.name = c(1, 2, 3)`.

`D` A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

`alpha` A numeric value to specify the significance α-level of the hypothesis test using GRDIF statistics. Default is .05.

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

`purify` A logical value indicating whether a purification process will be implemented or not. Default is FALSE.

`purify.by` A character string specifying a GRDIF statistic with which the purification is implemented. Available statistics are "grdifrs" for $GRDIF_{RS}$, "grdifr" for $GRDIF_{R}$, and "grdifs" for $GRDIF_{S}$.

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

`min.resp` A positive integer value specifying the minimum number of item responses for an examinee required to compute the ability estimate. Default is NULL. See details below for more information.

`post.hoc` A logical value indicating whether to conduct a post-hoc RDIF analysis for all possible combinations of paired groups for statistically flagged items. The default is TRUE. See below for more details.

`method` A character string indicating a scoring method. Available methods are "ML" for maximum likelihood estimation, "WL" for the weighted likelihood estimation, "MAP" for maximum a posteriori estimation, and "EAP" for expected a posteriori estimation. The default method is "ML".

`range` A numeric vector with two components to restrict the ability scale range for ML, WL, EAP, and MAP scoring methods. The default is c(-5, 5).

`norm.prior` A numeric vector with two components specifying the mean and standard deviation of the normal prior distribution. These parameters are used to obtain Gaussian quadrature points and their corresponding weights from the normal distribution. The default is c(0,1). Ignored if `method` is "ML" or "WL".

`nquad` An integer value specifying the number of Gaussian quadrature points from the normal prior distribution. The default is 41. Ignored if `method` is "ML", "WL", or "MAP".

`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 quadrature points can be obtained

using the gen.weight function. If NULL and method is "EAP", default values are used (see the norm.prior and nquad arguments). Ignored if method is "ML", "WL", or "MAP".

ncore — The number of logical CPU cores to use. The default is 1. See est_score for details.

verbose — A logical value. If TRUE, progress messages for the purification procedure are suppressed. The default is TRUE.

Details

The GRDIF framework (Lim et al., 2023) is a generalized version of the RDIF detection framework, designed to assess DIF for multiple groups. The GRDIF framework comprises three statistics: $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$, which focus on detecting uniform, nonuniform, and mixed DIF, respectively. Under the null hypothesis that a test contains no DIF items, $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$ asymptotically follow the χ^2 distributions with G-1, G-1, and 2(G-1) degrees of freedom, respectively, where G represents the total number of groups being compared. For more information on the GRDIF framework, see Lim et al. (2023).

The grdif function calculates all three GRDIF statistics: $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$. The current version of the grdif function supports both dichotomous and polytomous item response data. To compute these statistics, the grdif function requires (1) item parameter estimates obtained from aggregate data, regardless of group membership, (2) examinees' ability estimates (e.g., MLE), and (3) examinees' item response data. Note that the ability estimates must be computed using the aggregate data-based item parameter estimates. 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. When abilities are not given in the score argument (i.e., score = NULL), the grdif function estimates examinees' abilities automatically using the scoring method specified in the method argument (e.g., method = "ML").

The group argument accepts a vector with numeric or character values, indicating the group membership of examinees. The vector may include multiple distinct values, where one value represents the reference group and the others represent the focal groups. The length of the vector should be the same as the number of rows in the response data, with each value indicating the group membership of each examinee. After specifying the group, a numeric or character vector should be provided in the focal.name argument to define which group values in the group argument represent the focal groups. The reference group will be the group not included in the focal.name vector.

Similar to the original RDIF framework for two-groups comparison, the GRDIF framework can implement an iterative purification process. When purify = TRUE, the purification process is executed based on one of the GRDIF statistics specified in the purify.by argument (e.g., purify.by="grdifrs"). During each iterative purification, examinees' latent abilities are calculated using purified items and the scoring method specified in the method argument. The iterative purification process stops when no additional DIF items are identified or when the process reaches a predetermined limit of iterations, which can be set in the max.iter argument. For more information about the purification procedure, refer to Lim et al. (2022).

Scoring with a limited number of items can result in large standard errors, which may impact the effectiveness of DIF detection within the GRDIF framework. The min.resp argument can be employed to avoid using scores with significant standard errors when calculating the GRDIF statistics, particularly during the purification process. For instance, if min.resp is not NULL (e.g., min.resp=5), item responses from examinees whose total item responses fall below the specified

minimum number are treated as missing values (i.e., NA). Consequently, their ability estimates become missing values and are not utilized in computing the GRDIF statistics. If `min.resp=NULL`, an examinee's score will be computed as long as there is at least one item response for the examinee.

The `post.hoc` argument allows you to perform a post-hoc RDIF analysis for all possible combinations of paired groups for items flagged as statistically significant. For example, consider four groups of examinees: A, B, C, and D. If `post.hoc = TRUE`, the `grdif` function will perform a post-hoc RDIF analysis for all possible pairs of groups (A-B, A-C, A-D, B-C, B-D, and C-D) for each flagged item. This helps to identify which specific pairs of groups have DIF for each item, providing a more detailed understanding of the DIF patterns in the data. Note that when purification is implemented (i.e., `purify = TRUE`), the post-hoc RDIF analysis is conducted for each flagged item during each single iteration of the purification process.

Value

This function returns a list of four internal objects. The four objects are:

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

- **dif_stat** A data frame containing the results of three RDIF statistics for all evaluated items. Starting from the first column, each column represents the item's ID, $GRDIF_R$ statistic, $GRDIF_S$ statistic, $GRDIF_{RS}$ statistic, p-value of $GRDIF_R$, p-value of $GRDIF_S$, p-value of $GRDIF_{RS}$, sample size of the reference group, sample sizes of the focal groups, and the total sample size, respectively.
- **moments** A list of three data frames detailing the moments of mean raw residuals (MRRs) and mean squared residuals (MSRs) across all compared groups. The first data frame contains the means of MRR and MSR, the second data frame includes the variances of MRR and MSR, and the last one displays the covariances of MRR and MSR for all groups.
- **dif_item** A list of three numeric vectors indicating potential DIF items flagged by each of the GRDIF statistics. Each numeric vector corresponds to the items identified by $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$, respectively.
- **score** A vector of ability estimates used to compute the GRDIF statistics.
- **post.hoc** A list of three data frames containing the post-hoc RDIF analysis results of all possible combinations of paired groups. The first, second, and third data frames present the post-hoc analysis outcomes for the items identified by the $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$ statistics, respectively.

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

`with_purify` A list of several sub-objects containing the results of DIF analysis with a purification procedure. The sub-objects are:

- **purify.by** A character string indicating which GRDIF statistic is used for the purification. "grdifr", "grdifs", and "grdifrs" refers to $GRDIF_R$, $GRDIF_S$, and $GRDIF_{RS}$, respectively.
- **dif_stat** A data frame containing the results of three GRDIF statistics for all evaluated items. Starting from the first column, each column represents the item's ID, $GRDIF_R$ statistic, $GRDIF_S$ statistic, $GRDIF_{RS}$ statistic, p-value of $GRDIF_R$, p-value of $GRDIF_S$, p-value of $GRDIF_{RS}$, sample

size of the reference group, sample sizes of the focal groups, total sample size, and *n*th iteration where the GRDIF statistics were computed, respectively.

moments A list of three data frames detailing the moments of mean raw residuals (MRRs) and mean squared residuals (MSRs) across all compared groups. The first data frame contains the means of MRR and MSR, the second data frame includes the variances of MRR and MSR, and the last one displays the covariances of MRR and MSR for all groups. In each data frame, the last column indicates the *n*th iteration where the GRDIF statistics were computed.

n.iter The total number of iterations implemented for the purification.

score A vector of final purified ability estimates used to compute the GRDIF statistics.

post.hoc A data frame containing the post-hoc RDIF analysis results for the flagged items across all possible combinations of paired groups. The post-hoc RDIF analysis is conducted for each flagged item at every iteration.

complete A logical value indicating whether the purification process was completed. If FALSE, it means that the purification process reached the maximum iteration number but was not completed.

`alpha` A significance α-level used to compute the p-values of RDIF statistics.

Methods (by class)

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

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Lim, H., & Choe, E. M. (2023). Detecting differential item functioning in CAT using IRT residual DIF approach. *Journal of Educational Measurement*. 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. (2023, April). *Detecting differential item functioning among multiple groups using IRT residual DIF framework*. Paper presented at the Annual Meeting of the National Council on Measurement in Education. Chicago, IL.

See Also

`rdif` `est_item`, `info`, `simdat`, `shape_df`, `gen.weight`, `est_score`

Examples

```
# load library
library("dplyr")

## Uniform DIF detection for four groups (1R/3F)
#########################################################
# (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 of 3PLM items which are non-DIF items
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 where uniform DIF will be manipulated
difpar_ref <-
  shape_df(
    par.drm = list(a = c(0.8, 1.5, 0.8, 1.5), b = c(0.0, 0.0, -0.5, -0.5), g = .15),
    item.id = paste0("dif", 1:4), cats = 2, model = "3PLM"
  )

# Manipulate uniform DIF on the four new items by adjusting the b-parameters
# for the three focal groups
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 item data for both reference and focal groups.
# Thus, the first four items have uniform DIF for thee 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 the true thetas from the different ability 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)
```

```
# Generate the response data
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 the item and ability parameters
#     using the aggregate data
########################################################
# Estimate the item parameters
est_mod <- irtQ::est_irt(data = data, D = 1, model = "3PLM")
est_par <- est_mod$par.est

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

########################################################
# (3) Conduct DIF analysis
########################################################
# Create a vector of group membership indicators,
# where 1, 2 and 3 indicate the three focal groups
group <- c(rep(0, 500), rep(1, 500), rep(2, 500), rep(3, 500))

# (a) Compute GRDIF statistics without purification
#     and implement the post-hoc two-groups comparison analysis for
#     the 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)

# Print the post-hoc analysis results for the fagged items
print(dif_nopuri$no_purify$post.hoc)

# (b) Compute GRDIF statistics with purification
#     based on \eqn{GRDIF_{R}} and implement the post-hoc
#     two-groups comparison analysis for flagged items
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)

# Print the post-hoc analysis results without purification
print(dif_puri_r$no_purify$post.hoc)

# Print the post-hoc analysis results with purification
print(dif_puri_r$with_purify$post.hoc)
```

info *Item and Test Information Function*

Description

This function computes both item and test information functions (Hambleton et al., 1991) given a 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 the item metadata (e.g., item parameters, number of categories, models ...), an object of class est_item obtained from the function est_item, or an object of class est_irt obtained from the function est_irt. The data frame of item metadata can be easily obtained using the function shape_df. See below for details.
...	Further arguments passed to or from other methods.
theta	A vector of theta values where item and test information values are computed.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
tif	A logical value. If TRUE, the test information function is computed. Default is TRUE.

Details

A specific form of a data frame should be used for the argument x. The first column should have item IDs, the second column should contain unique score category numbers of the items, and the third column should include IRT models being fit to the items. The available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous item data, and "GRM" and "GPCM" for polytomous item data. Note that "DRM" covers all dichotomous IRT models (i.e, "1PLM",

"2PLM", and "3PLM") and "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. The next columns should include the item parameters of the fitted IRT models. For dichotomous items, the fourth, fifth, and sixth columns represent the item discrimination (or slope), item difficulty, and item guessing parameters, respectively. When "1PLM" and "2PLM" are specified in the third column, NAs should be inserted in the sixth column for the item guessing parameters. For polytomous items, the item discrimination (or slope) parameters should be included in the fourth column and the item difficulty (or threshold) parameters of category boundaries should be contained from the fifth to the last columns. When the number of unique score categories differs between items, the empty cells of item parameters should be filled with NAs. In the **irtQ** package, the item difficulty (or threshold) parameters of category boundaries for GPCM are expressed as the item location (or overall difficulty) parameter subtracted by the threshold parameter for unique score categories of the item. Note that when an GPCM item has *K* unique score categories, *K-1* item difficulty parameters are necessary because the item difficulty parameter for the first category boundary is always 0. For example, if an GPCM item has five score categories, four item difficulty parameters should be specified. An example of a data frame with a single-format test is as follows:

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

And an example of a data frame for a mixed-format test is as follows:

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 `IRT Models` section in the page of `irtQ-package` for more details about the IRT models used in the **irtQ** package. An easier way to create a data frame for the argument `x` is by using the function `shape_df`.

Value

This function returns an object of class `info`. This object contains item and test information values given the specified theta values.

Methods (by class)

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

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

See Also

plot.info, shape_df, est_item

Examples

```
## example 1.
## using the function "shape_df" to create a data frame of test metadata
# create a list containing the 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 containing the 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 of score categories for the items
cats <- c(2, 4, 2, 2, 5, 2, 2)

# create a character vector of IRT models for the items
model <- c("DRM", "GRM", "DRM", "DRM", "GPCM", "DRM", "DRM")

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

# set theta values
theta <- seq(-2, 2, 0.1)

# compute item and test information values given the theta values
info(x = test, theta = theta, D = 1, tif = TRUE)


## example 2.
```

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

# read item parameters and transform them to item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# set theta values
theta <- seq(-2, 2, 0.1)

# compute item and test information values given the theta values
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 (i.e., χ^2 fit statistic (e.g., Bock, 1960; Yen, 1981), loglikelihood ratio χ^2 fit statistic (G^2; McKinley & Mills, 1985), and infit and outfit statistics (Ames et al., 2015)) and returns contingency tables to compute the χ^2 and G^2 fit statistics. Note that caution is needed in interpreting the infit and outfit statistics for non-Rasch models. The saved object of this function, especially the object of contingency tables, is used in the function of `plot.irtfit` to draw a raw and standardized residual plots (Hambleton et al., 1991).

Usage

```
irtfit(x, ...)

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

```
)

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

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

Arguments

x	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...), an object of class est_item obtained from the function est_item, or an object of class est_irt obtained from the function est_irt. The data frame of item metadata can be easily obtained using the function shape_df. See below for more detail.
...	Further arguments passed to or from other methods.
score	A vector of examinees' ability estimates.
data	A matrix containing examinees' response data for the items in the argument x. A row and column indicate the examinees and items, respectively.
group.method	A character string indicating how to group examinees along the ability scale for computing the χ^2 and G^2 fit statistics. Available methods are "equal.width" for grouping examinees by dividing the ability scale into intervals of equal width

and "equal.freq" for grouping examinees by dividing the ability scale into intervals with equal frequencies of examinees. However, "equal.freq" does not always guarantee exactly the same frequency of examinees for all groups. Default is "equal.width". To divide the ability scale, the range of ability scale and the number of divided groups must be specified in the arguments of `range.score` and `n.width`, respectively. See below for details.

`n.width` An integer value to specify the number of divided groups along the ability scale. Default is 10. See below for more detail.

`loc.theta` A character string to indicate the location of ability point at each group (or interval) where the expected probabilities of score categories are calculated using the IRT models. Available locations are "average" for computing the expected probability at the average point of examinees' ability estimates in each group and "middle" for computing the expected probability at the midpoint of each group. Default is "average".

`range.score` A vector of two numeric values to restrict the range of ability scale. All ability estimates less than the first value are transformed to the first value. All ability estimates greater than the second value are transformed to the second value. If NULL, the minimum and maximum values of ability estimates in the argument `score` is used as the range of ability scale. Note that selection of grouping method in the argument `group.method` has nothing to do with the range of ability scale. Default is NULL.

`D` A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

`alpha` A numeric value to specify significance α-level of the hypothesis test for the χ^2 and G^2 fit statistics. Default is .05.

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

`overSR` A numeric value to specify a criterion to find ability groups (or intervals) which have standardized residuals greater than the specified value. Default is 2.

`min.collapse` An integer value to indicate the minimum frequency of cells to be collapsed when computing the χ^2 and G^2 fit statistics. Neighboring interval groups will be collapsed to avoid expected interval frequencies less than the specified minimum cell frequency. Default is 1.

`pcm.loc` A vector of integer values indicating the locations of partial credit model (PCM) items whose slope parameters are fixed

Details

A specific form of a data frame should be used for the argument `x`. The first column should have item IDs, the second column should contain unique score category numbers of the items, and the third column should include IRT models being fit to the items. The available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous item data, and "GRM" and "GPCM" for polytomous item data. Note that "DRM" covers all dichotomous IRT models (i.e, "1PLM", "2PLM", and "3PLM") and "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. The next columns should include the item parameters of the fitted IRT models. For dichotomous items, the fourth, fifth, and sixth columns represent the item discrimination (or slope), item difficulty, and item guessing parameters, respectively. When

"1PLM" and "2PLM" are specified in the third column, NAs should be inserted in the sixth column for the item guessing parameters. For polytomous items, the item discrimination (or slope) parameters should be included in the fourth column and the item difficulty (or threshold) parameters of category boundaries should be contained from the fifth to the last columns. When the number of unique score categories differs between items, the empty cells of item parameters should be filled with NAs. In the **irtQ** package, the item difficulty (or threshold) parameters of category boundaries for GPCM are expressed as the item location (or overall difficulty) parameter subtracted by the threshold parameter for unique score categories of the item. Note that when an GPCM item has *K* unique score categories, *K-1* item difficulty parameters are necessary because the item difficulty parameter for the first category boundary is always 0. For example, if an GPCM item has five score categories, four item difficulty parameters should be specified. An example of a data frame with a single-format test is as follows:

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

And an example of a data frame for a mixed-format test is as follows:

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 `IRT Models` section in the page of `irtQ-package` for more detail about the IRT models used in the **irtQ** package. An easier way to create a data frame for the argument `x` is by using the function `shape_df`.

To calculate the χ^2 and G^2 fit statistics, two methods are used in the argument `group.method` to divide the ability scale into several groups. If `group.method = "equal.width"`, the examinees are grouped based on equal length of intervals. If `group.method = "equal.freq"`, the examinees are grouped so that all groups have equal frequencies. However, the grouping method of "equal.freq" does guarantee that every group has the exactly same frequency of examinees. This is because the examinees are divided by the same size of quantile.

When dividing the ability scale into intervals to compute the χ^2 and G^2 fit statistics, the intervals should be wide enough not to include too small number of examinees. On the other hand, the interval should be narrow enough to include homogeneous examinees in terms of ability (Hambleton et al, 1991). Thus, if you want to divide the ability scale into other than ten groups, you need to specify the number of groups in the argument `n.width`. Yen (1981) fixed the number of groups to 10, whereas Bock (1960) allowed for any number of groups.

Regarding degrees of freedom (*df*), the χ^2 is assumed to be distributed approximately as a chi-square with *df* equal to the number of groups less the number of the IRT model parameters (Ames

et al., 2015) whereas the G^2 is assumed to be distributed approximately as a chi-square with *df* equal to the number of groups (Ames et al., 2015; Muraki & Bock, 2003)

Note that if "DRM" is specified for an item in the item metadata set, the item is considered as "3PLM" to compute degrees of freedom of the χ^2 fit statistic.

Value

This function returns an object of class `irtfit`. Within this object, several internal objects are contained such as:

`fit_stat` A data frame containing the results of three IRT fit statistics (i.e., χ^2 and G^2, infit, outfit statistics) across all evaluated items. In the data frame, the columns indicate item's ID, χ^2 fit statistic, G^2 fit statistic, degrees of freedom for the χ^2, degrees of freedom for the G^2, critical value for the χ^2, critical value for the G^2, p-value for the χ^2, p-value for the G^2, outfit statistic, infit statistic, the number of examinees used to compute the five fit statistics, and the proportion of ability groups (or intervals), before collapsing the cells, that have standardized residuals greater than the specified criterion in the argument `overSR`, respectively.

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

`contingency.plot`
A list of contingency tables used to draw a raw and standardized residual plots (Hambleton et al., 1991) in the function of `plot.irtfit`. Note that the collapsing cell strategy is *not* implemented to these contingency tables.

`individual.info`
A list of data frames including individual residual and variance values. Those information are used to compute infit and outfit statistics.

`item_df` The item metadata specified in the argument `x`.

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

Methods (by class)

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

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

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

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

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

Wells, C. S., & Bolt, D. M. (2008). Investigation of a nonparametric procedure for assessing goodness-of-fit in item response theory. *Applied Measurement in Education, 21*(1), 22-40.

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

See Also

plot.irtfit, shape_df, est_item

Examples

```
## example 1
## use the simulated CAT data
# find the location of items that have more than 10,000 responses
over10000 <- which(colSums(simCAT_MX$res.dat, na.rm = TRUE) > 10000)

# select the items that have more than 10,000 responses
x <- simCAT_MX$item.prm[over10000, ]

# select the response data for the items
data <- simCAT_MX$res.dat[, over10000]

# select the examinees' abilities
score <- simCAT_MX$score

# compute 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
)

# fit statistics
fit1$fit_stat

# contingency tables
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 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(10)
score <- rnorm(1000, mean = 0, sd = 1)

# simulate the response data
data <- simdat(x = x, theta = score, D = 1)

# compute 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
)

# fit statistics
fit2$fit_stat

# contingency tables
fit2$contingency.fitstat

# residual plots for the first item (dichotomous item)
plot(x = fit2, 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 = fit2, item.loc = 3, type = "both", ci.method = "wald",
     show.table = FALSE, ylim.sr.adjust = TRUE)
```

`llike_score` *Loglikelihood of Ability Parameters*

Description

This function computes the loglikelihood of ability parameters given the 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 the item metadata (e.g., item parameters, number of score categories, models). This can be created easily using the shape_df function. See est_irt, irtfit, info, or simdat for more details about the item metadata.
data	A matrix representing examinees' response data for the items in x. Each row and column corresponds to an examinee and an item, respectively.
theta	A numeric vector of ability parameters for which the loglikelihood values will be computed.
D	A scaling factor in IRT models that adjusts the logistic function to approximate the normal ogive function (set to 1.7). The default is 1.
method	A character string specifying the scoring method. Options include "ML" for maximum likelihood estimation, "MLF" for maximum likelihood estimation with fences, and "MAP" for maximum a posteriori estimation. The default method is "MLE".
norm.prior	A numeric vector of two elements indicating the mean and standard deviation of the normal prior distribution. These parameters are used to obtain the Gaussian quadrature points and corresponding weights from the normal distribution. Default is c(0,1). This parameter is ignored if method is "ML" or "MLF".
fence.a	A numeric value defining the item slope parameter (a-parameter) for the two imaginary items in the MLF method. Default is 3.0.
fence.b	A numeric vector of two elements specifying the lower and upper fences of item difficulty parameters (b-parameters) for the two imaginary items in the MLF method. If fence.b = NULL, the range values are used to set the fences. The default is NULL.
missing	A value used to denote missing values in the response data set. Default is NA.

Details

The function computes the loglikelihood value of the ability parameter given the item parameters and response data for each item. As an example, to assess the loglikelihoods of abilities for two examinees who have taken the same test items specified in x, supply their item response data matrix with two rows in data and a vector of ability values for which loglikelihood needs to be computed in theta.

Value

A data frame of loglikelihood values. Each row indicates the ability parameter for which the loglikelihood was computed, and each column represents a response pattern.

Examples

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

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

# Generate examinees' abilities from N(0, 1)
set.seed(10)
score <- rnorm(5, mean=0, sd=1)

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

# Specify the ability values for which the loglikelihood values will be computed
theta <- seq(-3, 3, 0.5)

# Compute the loglikelihood values (using the MLE method)
llike_score(x=x, data=data, theta=theta, D=1, method="ML")
```

LSAT6 *LSAT6 data*

Description

Well-known LSAT6 dichotomous response data set from Thissen (1982).

Usage

```
LSAT6
```

Format

This data contains 1,000 dichotomous response patterns of five items obtained from the Law School Admissions Test, section 6.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

lwrc *Lord-Wingersky Recursion Formula*

Description

This function computes the conditional distributions of number-correct (or observed) scores given probabilities of category responses to items or given a set of theta values using Lord and Wingersky recursion formula (1984).

Usage

```
lwrc(x = NULL, theta, prob = NULL, cats, D = 1)
```

Arguments

x	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...). See irtfit, info or simdat for more details about the item metadata. This data frame can be easily obtained using the function shape_df. If prob = NULL, this data frame is used in the recursion formula. See below for details.
theta	A vector of theta values where the conditional distribution of observed scores are computed. The theta values are only required when a data frame is specified in the argument x.
prob	A matrix containing the probability of answering each category of an item. Each row indicates an item and each column represents each category of the item. When the number of categories differs between items, the empty cells should be filled with zeros or NA values. If x = NULL, this probability matrix is used in the recursion Formula.
cats	A numeric vector specifying the number of categories for each item. For example, a dichotomous item has two categories. This information is only required when a probability matrix is specified in the argument prob.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

Details

The Lord and Wingersky recursive algorithm is an efficient way of calculating the compound probabilities of any number-correct scores on a test based on IRT models. This algorithm is particularly useful when computing the IRT model-based observed score distribution for a test.

To compute the conditional distributions of observed scores, either the item metadata set specified in x or the probability matrix specified in prob can be used.

Value

When the prob argument is provided, this function returns a vector of the probabilities of obtaining every observed score on a test. When the x argument is specified, the function returns a matrix of conditional probabilities across all possible observed scores and theta values.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

Examples

```
## example 1: when a matrix of probabilities is used as a data set
## this is an example from Kolen and Brennan (2004, p. 183)
# create a matrix of probabilities of getting correct and incorrect answers for three items
probs <- matrix(c(.74, .73, .82, .26, .27, .18), nrow = 3, ncol = 2, byrow = FALSE)

# create a vector of score categories for the three items
cats <- c(2, 2, 2)

# compute the conditional distributions of observed scores
lwrc(prob = probs, cats = cats)

## example 2: when a matrix of probabilities is used as a data set
## with a mixed-format test
# category probabilities for a dichotomous item
p1 <- c(0.2, 0.8, 0, 0, 0)
# category probabilities for a 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)

# rbind the probability vectors
p <- rbind(p1, p2, p3, p4)

# create a vector of score categories for the four items
cats <- c(2, 2, 5, 3)

# compute the conditional distributions of observed scores
lwrc(prob = p, cats = cats)

## example 3: when a data frame for the item metadata of
## a mixed-format test is used.
# import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# read item parameters and transform them to item metadata
x <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df
```

```
# compute the conditional distributions of observed scores
lwrc(x = x, theta = seq(-4, 4, 0.2), D = 1)
```

---

pcd2 *Pseudo-count D2 method*

Description

This function calculates the Pseudo-count D^2 statistic to evaluate item parameter drift, as described by Cappaert et al. (2018) and Stone (2000). The Pseudo-count D^2 statistic is designed to detect item parameter drift efficiently without requiring item recalibration, making it especially valuable in computerized adaptive testing (CAT) environments. This method compares observed and expected response frequencies across quadrature points, which represent latent ability levels. The expected frequencies are computed using the posterior distribution of each examinee's ability (Stone, 2000), providing a robust and sensitive measure of item parameter drift, ensuring the stability and accuracy of the test over time.

Usage

```
pcd2(
  x,
  data,
  D = 1,
  missing = NA,
  Quadrature = c(49, 6),
  weights = NULL,
  group.mean = 0,
  group.var = 1
)
```

Arguments

x	A data frame containing the metadata for the item bank, which includes important information for each item such as the number of score categories and the IRT model applied. See est_irt, irtfit, info or simdat for more detail about the item metadata.
data	A matrix containing examinees' response data of the items in the argument x. A row and column indicate the examinees and items, respectively.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
missing	A value indicating missing values in the response data set. Default is NA.
Quadrature	A numeric vector of two components specifying the number of quadrature points (in the first component) and the symmetric minimum and maximum values of these points (in the second component). For example, a vector of c(49, 6) indicates 49 rectangular quadrature points over -6 and 6.

`weights` A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the latent variable prior distribution. If not NULL, the scale of the latent ability distribution will be will be fixed to the scale of the provided quadrature points and weights. The weights and quadrature points can be easily obtained using the function `gen.weight`. If NULL, a normal prior density is used based on the information provided in the arguments of `Quadrature`, `group.mean`, and `group.var`). Default is NULL.

`group.mean` A numeric value to set the mean of latent variable prior distribution when `weights = NULL`. Default is 0.

`group.var` A positive numeric value to set the variance of latent variable prior distribution when `weights = NULL`. Default is 1.

Details

The Pseudo-count D^2 values are calculated by summing the weighted squared differences between the observed and expected frequencies for each score category across all items. The expected frequencies are determined using the posterior distribution of each examinee's ability (Stone, 2000).

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

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

where r_{0k} and r_{1k} are the pseudo-counts for the incorrect and correct responses at each ability level k, E_{1k} is the expected proportion of correct responses at each ability level k, calculated using item parameters from the item bank, and N is the total count of examinees who received each item

The `pcd2` function is designed to be flexible and allows for detailed control over the computation process through its various arguments:

x The metadata should include key information such as the number of score categories for each item, the IRT model applied (e.g., "1PLM", "2PLM", "3PLM"), and the item parameters (e.g., discrimination, difficulty, guessing). This data frame is crucial because it defines the structure of the items and the parameters used in the calculation of expected frequencies.

data The response matrix should be preprocessed to ensure that missing values are correctly coded (using the 'missing' argument if needed). The matrix is the foundation for calculating both observed and expected frequencies for each item and score category. Ensure that the number of items in this matrix matches the number specified in the 'x' argument.

Quadrature The quadrature points are used to approximate the latent ability distribution, which is essential for accurately estimating the posterior distribution of abilities. Adjust the number and range of quadrature points based on the precision required and the characteristics of the examinee population.

weights This argument allows you to provide a custom two-column matrix or data frame where the first column contains quadrature points and the second column contains the corresponding weights. This provides flexibility in defining the latent ability distribution. If 'weights' is set to 'NULL', the function generates a normal prior distribution based on the values provided in 'Quadrature', 'group.mean', and 'group.var'. Use this argument when you have a specific latent ability distribution you wish to apply, such as one derived from empirical data or an alternative theoretical distribution.

group.mean, group.var These numeric values define the mean and variance of the latent ability distribution when 'weights' is not provided. The default values are 'group.mean = 0' and 'group.var = 1', which assume a standard normal distribution. These values are used to generate quadrature points and weights internally if 'weights' is 'NULL'. Adjust 'group.mean' and 'group.var' to reflect the expected distribution of abilities in your examinee population, particularly if you suspect that the latent trait distribution deviates from normality.

When setting these arguments, consider the specific characteristics of your item bank, the distribution of abilities in your examinee population, and the computational precision required. Properly configured, the function provides robust and accurate Pseudo-count D^2 statistics, enabling effective monitoring of item parameter drift in CAT or other IRT-based testing environments.

Value

A data frame containing the Pseudo-count D^2 statistic for each item in the analysis. The data frame includes the following columns:

`id`	The unique identifier for each item, corresponding to the item IDs provided in the `x` argument.
`pcd2`	The computed Pseudo-count D^2 statistic for each item, which quantifies the degree of item parameter drift by comparing observed and expected response frequencies.
`N`	The number of examinees whose responses were used to calculate the Pseudo-count D^2 statistic for each item, reflecting the sample size involved in the computation.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Cappaert, K. J., Wen, Y., & Chang, Y. F. (2018). Evaluating CAT-adjusted approaches for suspected item parameter drift detection. *Measurement: Interdisciplinary Research and Perspectives, 16*(4), 226-238.

Stone, C. A. (2000). Monte Carlo based null distribution for an alternative goodness-of-fit test statistic in IRT models. *Journal of educational measurement, 37*(1), 58-75.

Examples

```
## Compute the pseudo-count D2 statistics for the dichotomous items
## import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# select the first 30 3PLM item metadata to be examined
x <- bring.flexmirt(file = flex_sam, "par")$Group1$full_df[1:30, 1:6]

# generate examinees' abilities from N(0, 1)
set.seed(25)
score <- rnorm(500, mean = 0, sd = 1)
```

```
# simulate the response data
data <- simdat(x = x, theta = score, D = 1)

# compute the pseudo-count D2 statistics
ps_d2 <- pcd2(x = x, data = data)
print(ps_d2)
```

`plot.info` *Plot Item and Test Information Functions*

Description

This method function plots item or test information function given a specified theta values. In addition, it displays the conditional standard errors at a test level.

Usage

```
## S3 method for class 'info'
plot(
  x,
  item.loc = NULL,
  overlap = FALSE,
  csee = FALSE,
  xlab.text,
  ylab.text,
  main.text,
  lab.size = 15,
  main.size = 15,
  axis.size = 15,
  line.color,
  line.size = 1,
  layout.col = 4,
  strip.size = 12,
  ...
)
```

Arguments

`x`	x An object of class `info`.
`item.loc`	A vector of numeric values indicating that the item information functions of the *n*th items (or the location of items in a test form) are plotted. If NULL, the test information function for the total test form is drawn. Default is NULL.
`overlap`	Logical value indicating whether multiple item information functions are plotted in one panel. If FALSE, multiple item information functions are displayed in multiple panels, one for each.

csee	Logical value indicating whether the function displays the conditional standard error of estimation (CSEE) at a test level. If FALSE, item/test information function is plotted. Note that the CSEE plot is displayed only at a test level.
xlab.text, ylab.text	A title for the x and y axes.
main.text	An overall title for the plot.
lab.size	The size of xlab and ylab. Default is 15.
main.size	The size of main.text. Default is 15.
axis.size	The size of labels along the x and y axes. Default is 15.
line.color	A character string specifying a color for the line. See http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/ for more details about colors used in ggplot2.
line.size	The size of lines. Default is 1.
layout.col	An integer value indicating the number of columns in the panel when displaying the item information functions of the multiple items. Default is 4.
strip.size	The size of facet labels when the item information functions of the multiple items are drawn.
...	Further arguments passed from the function geom_line() in the **ggplot2** package.

Details

All of the plots are drawn using the ggplot2 package. The object of class info can be obtained from the function info.

Value

This method function displays the item or test information function plot. When csee = TRUE, the conditional standard error is returned at the test level.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

info

Examples

```
## the use of a "-prm.txt" file obtained from a flexMIRT
# import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# read item parameters and transform them to item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df
```

```
# set theta values
theta <- seq(-4, 4, 0.1)

# compute item and test information values given the theta values
x <- info(x = test_flex, theta = theta, D = 1, tif = TRUE)

# draw a plot of the test information function
plot(x)

# draw a plot of the item information function for the second item
plot(x, item.loc = 2)

# draw a plot of multiple item information functions across the multiple panels
plot(x, item.loc = 1:8, overlap = FALSE)

# draw a plot of multiple item information functions in one panel
plot(x, item.loc = 1:8, overlap = TRUE)

# draw a plot of conditional standard error at a test level
plot(x, csee = TRUE)
```

`plot.irtfit` *Draw raw and standardized residual plots*

Description

This method function provides graphical displays to look at residuals between the observed data and model-based predictions (Hambleton, Swaminathan, & Rogers, 1991). This function gives two residual plots for each score category of an item: (a) the raw residual plot and (b) the standardized residual plot. Note that for dichotomous items the residual plots are drawn only for the score category of 1.

Usage

```
## S3 method for class 'irtfit'
plot(
  x,
  item.loc = NULL,
  type = "both",
  ci.method = c("wald", "wilson", "wilson.cr"),
  show.table = TRUE,
  layout.col = 2,
  xlab.text,
  ylab.text,
  main.text,
  lab.size = 15,
  main.size = 15,
  axis.size = 15,
```

```
  line.size = 1,
  point.size = 2.5,
  strip.size = 12,
  ylim.icc = c(0, 1),
  ylim.sr.adjust = FALSE,
  ylim.sr = c(-4, 4),
  ...
)
```

Arguments

x	An object of class irtfit.
item.loc	An integer value indicating that the *n*th item (or the location of the item) is plotted. See below for details.
type	A character string indicating what type of residual plot is returned. Available options are "icc" for the raw residual plot, "sr" for the standardized residual plot, and "both" for both of them. Default is "both".
ci.method	A character string indicating what method is used to estimate the confidence interval for the raw residual plot. Available options are "wald" for Wald method, "wilson" for Wilson score interval, and "wilson.cr" for Wilson score interval with continuity correction. Default is "wald". See below for details.
show.table	A logical value. If TRUE, a contingency table containing the information used to draw the residual plots for the studied item is returned. This contingency table is the same as one contained in the internal object of contingency.plot in the object of class irtfit. Default is TRUE.
layout.col	An integer value indicating the number of columns in the panel when a polytomous item is used. Default is 2.
xlab.text	A title for the x axis. If missing, the default string is used.
ylab.text	A title for the y axis. If type = "both", two character strings can be specified for the raw residual and standardized residual plots, respectively. If missing, the default strings are used.
main.text	An overall title for the plot. If type = "both", two character strings can be specified for the raw residual and standardized residual plots, respectively. If missing, the default strings are used.
lab.size	The size of xlab and ylab. Default is 15.
main.size	The size of main.text. Default is 15.
axis.size	The size of labels along the x and y axes. Default is 15.
line.size	The size of lines. Default is 1.
point.size	The size of points. Default is 2.5.
strip.size	The size of facet labels. Default is 12.
ylim.icc	A vector of two numeric values specifying the range of y axis for the raw residual plot. Default is c(0, 1).
ylim.sr.adjust	A logical value. If TRUE, the range of y axis for the standardized residual plot is adjusted for each item. If FALSE, the range of y axis for the standardized residual plot is fixed to the values specified in the argument ylim.sr.

ylim.sr	A vector of two numeric values specifying the range of y axis for the standardized residual plot. Default is c(-4, 4).
...	Further arguments passed from the function ggplot() in the **ggplot2** package.

Details

All of the plots are drawn using the ggplot2 package.

Once the results of the IRT model fit analysis are obtained from the function irtfit, an object of class irtfit can be used to draw the IRT raw residual and standardized residual plots. Especially, the information contained in an internal object of contingency.plot are mainly used to draw the residual plots.

Because the residual plots are drawn for an item at a time, you have to indicate which item will be evaluated. For this, you should specify an integer value, which is the location of the studied item, in the argument item.loc. For example, if you want to draw the residual plots for the third item, then item.loc = 3.

In terms of the raw residual plot, the argument ci.method is used to select a method to estimate the confidence intervals among four methods. Those methods are "wald" for the Wald interval, which is based on the normal approximation (Laplace, 1812), "wilson" for Wilson score interval (Wilson, 1927), and "wilson.cr" for Wilson score interval with continuity correction (Newcombe, 1998). See https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval for more details about the binomial proportion confidence intervals. Note that the width of confidence interval is determined by the α-level specified in the argument alpha of the function irtfit.

Regarding the standardized residual plot, any standardized residuals greater than the specified criterion value in the argument overSR of the function irtfit are displayed with circles. Otherwise, they are displayed with crosses.

Value

This method function displays the IRT raw residual plot, the standard residual plot, or both of the studied item. when show.table = TRUE, a contingency table used to draw the residual plots is also returned. See irtfit for more detail about the contingency table.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

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

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

See Also

irtfit

Examples

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

# select the first two dichotomous items and 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 the 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 ICC and TCC*

Description

This method function plots item or test characteristic curve using the ggplot2 package. The item characteristic (or category) curve (ICC) or item score curve is drawn for an individual item. The test characteristic curve (TCC) is drawn based on a total test form.

Usage

```
## S3 method for class 'traceline'
plot(
  x,
  item.loc = NULL,
  score.curve = FALSE,
  overlap = FALSE,
  layout.col = 2,
  xlab.text,
  ylab.text,
  main.text,
  lab.size = 15,
  main.size = 15,
  axis.size = 15,
  line.color,
  line.size = 1,
  strip.size = 12,
  ...
)
```

Arguments

x	An object of class traceline.
item.loc	A numeric value indicating that the *n*th item (or the location of item) is plotted. If NULL, the TCC based on a total test form is drawn. Default is NULL.
score.curve	Logical value. If TRUE, item score curve (i.e., a weighted sum of item category probabilities over the item scores) is plotted in a panel. Otherwise, ICCs for all score categories are plotted in separate panels. For a dichotomous item, the item score curve is the same as the ICC of score category 1. Ignored when item.loc = NULL. Default is FALSE.
overlap	Logical value indicating whether multiple item score curves are plotted in one panel. If FALSE, the multiple item score curves are displayed with multiple panels, one for each.
layout.col	An integer value indicating the number of columns in the panel when displaying ICCs for an item or when displaying multiple item scores with multiple panels.
xlab.text, ylab.text	A title for the x and y axes.
main.text	An overall title for the plot.
lab.size	The size of xlab and ylab. Default is 15.
main.size	The size of main.text. Default is 15.

axis.size	The size of labels along the x and y axes. Default is 15.
line.color	A character string specifying the color for a line. See http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/ for more details about colors used in ggplot2.
line.size	The size of lines. Default is 1.
strip.size	The size of facet labels when ICCs for an item are plotted.
...	Further arguments passed from the function geom_line() in the **ggplot2** package.

Details

All of the plots are drawn using the ggplot2 package. If item.loc = NULL, the TCC based on the total test form is plotted. In the argument item.loc, a vector of positive integer values should be specified to indicate the *n*th items among the total test form. For example, if there are ten items in the test form and the score curves of the 1st, 2nd, and 3rd items should be plotted, then item.loc = 1:3.

Value

This method function displays ICC or TCC plots of the studied item(s).

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

traceline

Examples

```
## example
## using a "-prm.txt" file obtained from a flexMIRT
# import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# read item parameters and transform them to item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# set theta values
theta <- seq(-3, 3, 0.1)

# compute the item category probabilities and item/test
# characteristic functions given the theta values
x <- traceline(x = test_flex, theta, D = 1)

# plot TCC based on the total test form
plot(x, item.loc = NULL)

# plot ICCs for the first item (dichotomous item)
```

```
plot(x, item.loc = 1, score.curve = FALSE, layout.col = 2)

# plot item score curve for the first item (dichotomous item)
plot(x, item.loc = 1, score.curve = TRUE)

# plot item score curves for the first six dichotomous items
# with multiple panels
plot(x, item.loc = 1:6, score.curve = TRUE, overlap = FALSE)

# plot item score curve for the first six dichotomous items
# in one panel
plot(x, item.loc = 1:6, score.curve = TRUE, overlap = TRUE)

# plot ICCs for the last item (polytomous item)
plot(x, item.loc = 55, score.curve = FALSE, layout.col = 2)

# plot item score curve for the last item (polytomous item)
plot(x, item.loc = 55, score.curve = TRUE)

# plot item score curves for the last three polytomous items
# with multiple panels
plot(x, item.loc = 53:55, score.curve = TRUE, overlap = FALSE)

# plot item score curves for the last three polytomous items
# in one panel
plot(x, item.loc = 53:55, score.curve = TRUE, overlap = TRUE)
```

`prm` *Polytomous Response Model (PRM) Probabilities (GRM and GPCM)*

Description

This function computes the probability of selecting a specific category for an item for a given set of theta values using the graded response model and (generalized) partial credit model.

Usage

```
prm(theta, a, d, D = 1, pr.model = c("GRM", "GPCM"))
```

Arguments

`theta`	A vector of ability values.
`a`	A numeric value of item discrimination (or slope) parameter.
`d`	A vector of item difficulty (or threshold) parameters.
`D`	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
`pr.model`	A character string indicating the polytomous model being used. Available models are "GRM" for the the graded response model and "GPCM" for the (generalized) partial credit model.

Details

When the category probabilities are computed for an item with the partial credit model, provide a = 1 for that item. When model = "GPCM", d should include the item difficulty (or threshold) parameters. In the **irtQ** package, the item difficulty (or threshold) parameters of category boundaries for GPCM are expressed as the item location (or overall difficulty) parameter subtracted by the threshold parameter for unique score categories of the item. Note that when an GPCM item has *K* unique score categories, *K-1* item difficulty parameters are necessary because the item difficulty parameter for the first category boundary is always 0. For example, if an GPCM item has five score categories, four item difficulty parameters should be specified. For more details about the parameterization of the (generalized) partial credit model, See IRT Models section in the page of irtQ-package.

Value

This function returns a matrix where a row indicates the ability and a column represents score categories of the item.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

drm, irtfit

Examples

```
## Category probabilities for an item with four categories
## using a generalized partial credit model
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 a graded response model
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 (Lim & Choe, In press; Lim, Choe, & Han, 2022), which are $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$, for each item. $RDIF_R$ primarily captures the typical contrast in raw residual pattern between two groups caused by uniform DIF whereas $RDIF_S$ primarily captures the typical contrast in squared residual pattern between two groups caused by nonuniform DIF. $RDIF_{RS}$ can reasonably capture both types of DIF.

Usage

```
rdif(x, ...)

## Default S3 method:
rdif(
  x,
  data,
  score = NULL,
  group,
  focal.name,
  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,
  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,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("rdifrs", "rdifr", "rdifs"),
  max.iter = 10,
  min.resp = NULL,
  method = "ML",
  range = c(-5, 5),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  ncore = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...), an object of class est_item obtained from the function est_item, or an object of class est_irt obtained from the function est_irt. The item metadata can be easily created using the function shape_df. See est_irt, irtfit, info or simdat for more details about the item metadata.
...	Additional arguments that will be forwarded to the est_score function.
data	A matrix containing examinees' response data for the items in the argument x. A row and column indicate the examinees and items, respectively.
score	A vector of examinees' ability estimates. If the abilities are not provided, rdif estimates the abilities before computing the RDIF statistics. See est_score for more details about scoring methods. Default is NULL.
group	A numeric or character vector indicating group membership of examinees. The length of the vector should be the same with the number of rows in the response data matrix.
focal.name	A single numeric or character scalar representing the level associated with the focal group. For instance, given group = c(0, 1, 0, 1, 1) and '1' indicating the focal group, set focal.name = 1.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
alpha	A numeric value to specify significance α-level of the hypothesis test using the RDIF statistics. Default is .05.
missing	A value indicating missing values in the response data set. Default is NA.

purify	A logical value indicating whether a purification process will be implemented or not. Default is FALSE.
purify.by	A character string specifying a RDIF statistic with which the purification is implemented. Available statistics are "rdifrs" for $RDIF_{RS}$, "rdifr" for $RDIF_R$, and "rdifs" for $RDIF_S$.
max.iter	A positive integer value specifying the maximum number of iterations for the purification process. Default is 10.
min.resp	A positive integer value specifying the minimum number of item responses for an examinee required to compute the ability estimate. Default is NULL. See details below for more information.
method	A character string indicating a scoring method. Available methods are "ML" for the maximum likelihood estimation, "WL" for the weighted likelihood estimation, "MAP" for the maximum a posteriori estimation, and "EAP" for the expected a posteriori estimation. Default method is "ML".
range	A numeric vector of two components to restrict the range of ability scale for the ML, WL, EAP, and MAP scoring methods. Default is c(-5, 5).
norm.prior	A numeric vector of two components specifying a mean and standard deviation of the normal prior distribution. These two parameters are used to obtain the gaussian quadrature points and the corresponding weights from the normal distribution. Default is c(0,1). Ignored if method is "ML" or "WL".
nquad	An integer value specifying the number of gaussian quadrature points from the normal prior distribution. Default is 41. Ignored if method is "ML", "WL", or "MAP".
weights	A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the latent variable prior distribution. The weights and quadrature points can be easily obtained using the function gen.weight. If NULL and method is "EAP", default values are used (see the arguments of norm.prior and nquad). Ignored if method is "ML", "WL" or "MAP".
ncore	The number of logical CPU cores to use. Default is 1. See est_score for details.
verbose	A logical value. If TRUE, the progress messages of purification procedure are suppressed. Default is TRUE.

Details

The RDIF framework (Lim et al., 2022) consists of three IRT residual-based statistics: $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$. Under the null hypothesis that a test contains no DIF items, $RDIF_R$ and $RDIF_S$ follow normal distributions asymptotically. $RDIF_{RS}$ is a based on a bivariate normal distribution of $RDIF_R$ and $RDIF_S$ statistics. Under the null hypothesis of no DIF items, it follows a χ^2 distribution asymptotically with 2 degrees of freedom. See Lim et al. (2022) for more details about RDIF framework.

The rdif function computes all three RDIF statistics of $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$. The current version of rdif function supports both dichotomous and polytomous item response data. To compute the three 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 need to be computed using the aggregate data-based item parameter estimates. The item parameter estimates should be provided in the `x` argument, the ability estimates should be provided in the `score` argument, and the response data should be provided in the `data` argument. When the abilities are not given in the `score` argument (i.e., `score = NULL`), the `rdif` function estimates examinees' abilities automatically using the scoring method specified in the `method` argument (e.g., `method = "ML"`).

The `group` argument accepts a vector of either two distinct numeric or character variables. Between two distinct variable, one is to represent the reference group and another one is to represent the focal group. The length of the vector should be the same with the number of rows in the response data and each value in the vector should indicate each examinee of the response data. Once the `gruop` is specified, a single numeric or character value needs to be provided in the `focal.name` argument to define which group variable in the `group` argument represents the focal group.

As other DIF detection approaches, an iterative purification process can be implemented for the RDIF framework. When `purify = TRUE`, the purification process is implemented based on one of RDIF statistics specified in the `purify.by` argument (e.g, `purify.by="rdifrs"`). At each iterative purification, examinees' latent abilities are computed using purified items and scoring method specified in the `method` argument. The iterative purification process stops when no further DIF items are found or the process reaches a predetermined limit of iteration, which can be specified in the `max.iter` argument. See Lim et al. (2022) for more details about the purification procedure.

Scoring with a limited number of items can result in large standard errors, which may impact the effectiveness of DIF detection within the RDIF framework. The `min.resp` argument can be employed to avoid using scores with significant standard errors when calculating the RDIF statistics, particularly during the purification process. For instance, if `min.resp` is not NULL (e.g., `min.resp=5`), item responses from examinees whose total item responses fall below the specified minimum number are treated as missing values (i.e., NA). Consequently, their ability estimates become missing values and are not utilized in computing the RDIF statistics. If `min.resp=NULL`, an examinee's score will be computed as long as there is at least one item response for the examinee.

Value

This function returns a list of four internal objects. The four objects are:

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

- **dif_stat** A data frame containing the results of three RDIF statistics across all evaluated items. From the first column, each column indicates item's ID, $RDIF_R$ statistic, standardized $RDIF_R$, $RDIF_S$ statistic, standardized $RDIF_S$, $RDIF_{RS}$ statistic, p-value of the $RDIF_R$, p-value of the $RDIF_S$, p-value of the $RDIF_{RS}$, sample size of the reference group, sample size of the focal group, and total sample size, respectively. Note that $RDIF_{RS}$ does not have its standardized value because it is a χ^2 statistic.
- **moments** A data frame containing the moments of three RDIF statistics. From the first column, each column indicates item's ID, mean of $RDIF_R$, standard deviation of $RDIF_R$, mean of $RDIF_S$, standard deviation of $RDIF_S$, and covariance of $RDIF_R$ and $RDIF_S$, respectively.

dif_item A list of three numeric vectors showing potential DIF items flagged by each of the RDIF statistics. Each of the numeric vector means the items flagged by $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$, respectively.

score A vector of ability estimates used to compute the RDIF statistics.

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

`with_purify` A list of several sub-objects containing the results of DIF analysis with a purification procedure. The sub-objects are:

purify.by A character string indicating which RDIF statistic is used for the purification. "rdifr", "rdifs", and "rdifrs" refers to $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$, respectively.

dif_stat A data frame containing the results of three RDIF statistics across all evaluated items. From the first column, each column indicates item's ID, $RDIF_R$ statistic, standardized $RDIF_R$, $RDIF_S$ statistic, standardized $RDIF_S$, $RDIF_{RS}$ statistic, p-value of the $RDIF_R$, p-value of the $RDIF_S$, p-value of the $RDIF_{RS}$, sample size of the reference group, sample size of the focal group, total sample size, and *n*th iteration where the RDIF statistics were computed, respectively.

moments A data frame containing the moments of three RDIF statistics. From the first column, each column indicates item's ID, mean of $RDIF_R$, standard deviation of $RDIF_R$, mean of $RDIF_S$, standard deviation of $RDIF_S$, covariance of $RDIF_R$ and $RDIF_S$, and *n*th iteration where the RDIF statistics were computed, respectively.

dif_item A list of three numeric vectors showing potential DIF items flagged by each of the RDIF statistics. Each of the numeric vector means the items flagged by $RDIF_R$, $RDIF_S$, and $RDIF_{RS}$, respectively.

n.iter A total number of iterations implemented for the purification.

score A vector of final purified ability estimates used to compute the RDIF statistics.

complete A logical value indicating whether the purification process was completed. If FALSE, it means that the purification process reached the maximum iteration number but it was not complete.

`alpha` A significance α-level used to compute the p-values of RDIF statistics.

Methods (by class)

- `rdif(default)`: Default method to computes three RDIF statistics using a data frame `x` containing the item metadata.
- `rdif(est_irt)`: An object created by the function `est_irt`.
- `rdif(est_item)`: An object created by the function `est_item`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

See Also

est_item, info, simdat, shape_df, gen.weight, est_score

Examples

```
# call library
library("dplyr")

## Uniform DIF detection
################################################
# (1) manipulate true uniform DIF data
################################################
# import the "-prm.txt" output file from flexMIRT
flex_sam <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# select 36 of 3PLM items which are non-DIF items
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 inject uniform DIF
difpar_ref <-
  shape_df(
    par.drm = list(a = c(0.8, 1.5, 0.8, 1.5), b = c(0.0, 0.0, -0.5, -0.5), g = 0.15),
    item.id = paste0("dif", 1:4), cats = 2, model = "3PLM"
  )

# manipulate uniform DIF on the four new items by adding constants to 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 4 DIF and 36 non-DIF items for both reference and focal groups
# thus, the first four items have uniform DIF
par_ref <- rbind(difpar_ref, par_nstd)
par_foc <- rbind(difpar_foc, par_nstd)

# generate the true thetas
set.seed(123)
```

```
theta_ref <- rnorm(500, 0.0, 1.0)
theta_foc <- rnorm(500, 0.0, 1.0)

# generate the 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 the item and ability parameters
#     using the aggregate data
##############################################
# estimate the item parameters
est_mod <- est_irt(data = data, D = 1, model = "3PLM")
est_par <- est_mod$par.est

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

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

# (a)-1 compute RDIF statistics by providing scores,
#       and without a 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 by not providing scores
#       and without a purification
dif_nopuri_2 <- rdif(
  x = est_par, data = data, score = NULL,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  method = "ML"
)
print(dif_nopuri_2)

# (b)-1 compute RDIF statistics with a purification
#       based on RDIF(R)
dif_puri_r <- rdif(
  x = est_par, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "rdifr"
)
print(dif_puri_r)

# (b)-2 compute RDIF statistics with a purification
```

```
#       based on RDIF(S)
dif_puri_s <- rdif(
  x = est_par, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "rdifs"
)
print(dif_puri_s)

# (b)-3 compute RDIF statistics with a purification
#       based on RDIF(RS)
dif_puri_rs <- rdif(
  x = est_par, data = data, score = score,
  group = group, focal.name = 1, D = 1, alpha = 0.05,
  purify = TRUE, purify.by = "rdifrs"
)
print(dif_puri_rs)
```

`reval_mst` *Recursion-based MST evaluation method*

Description

This function evaluates the measurement precision and bias in Multistage-adaptive Test (MST) panels using a recursion-based evaluation method introduced by Lim et al. (2020). This function computes conditional biases and standard errors of measurement (CSEMs) across a range of IRT ability levels, facilitating efficient and accurate MST panel assessments without extensive simulations.

Usage

```
reval_mst(
  x,
  D = 1,
  route_map,
  module,
  cut_score,
  theta = seq(-5, 5, 1),
  intpol = TRUE,
  range.tcc = c(-7, 7),
  tol = 1e-04
)
```

Arguments

`x`	A data frame containing the metadata for the item bank, which includes important information for each item such as the number of score categories and the IRT model applied. This metadata is essential for evaluating the MST panel,

with items selected based on the specifications in the module argument. To construct this item metadata efficiently, the shape_df function is recommended. Further details on utilizing item bank metadata along with module for MST panel evaluation are provided below.

D — A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

route_map — A binary square matrix that defines the MST structure, illustrating transitions between modules and stages. This concept and structure are inspired by the transMatrix argument in the randomMST function from the **mstR** package (Magis et al., 2017), which provides a framework for representing MST pathways. For comprehensive understanding and examples of constructing route_map, refer to the **mstR** package (Magis et al., 2017) documentation. Also see below for details.

module — A binary matrix that maps items from the item bank specified in x to modules within the MST framework. This parameter's structure is analogous to the modules argument in the randomMST function of the **mstR** package, enabling precise item-to-module assignments for MST configurations. For detailed instructions on creating and utilizing the module matrix effectively, consult the documentation of the **mstR** package (Magis et al., 2017). Also see below for details.

cut_score — A list defining cut scores for routing test takers through MST stages. Each list element is a vector of cut scores for advancing participants to subsequent stage modules. In a 1-3-3 MST configuration, for example, cut_score might be defined as cut_score = list(c(-0.5, 0.5), c(-0.6, 0.6)), where c(-0.5, 0.5) are thresholds for routing from the first to the second stage, and c(-0.6, 0.6) for routing from the second to the third stage. This setup facilitates tailored test progression based on performance. Further examples and explanations are available below.

theta — A vector of ability levels (theta) at which the MST panel's performance is assessed. This allows for the evaluation of measurement precision and bias across a continuum of ability levels. The default range is theta = seq(-5, 5, 0.1).

intpol — A logical value to enable linear interpolation in the inverse test characteristic curve (TCC) scoring, facilitating ability estimate approximation for observed sum scores not directly obtainable from the TCC, such as those below the sum of item guessing parameters. Default is TRUE, applying interpolation to bridge gaps in the TCC. Refer to est_score for more details and consult Lim et al. (2020) for insights into the interpolation technique within inverse TCC scoring.

range.tcc — A vector to define the range of ability estimates for inverse TCC scoring, expressed as the two numeric values for lower and upper bounds. Default is to c(-7, 7).

tol — A numeric value of the convergent tolerance for the inverse TCC scoring. For the inverse TCC scoring, the bisection method is used for optimization. Default is 1e-4.

Details

The reval_mst function evaluates an MST panel by implementing a recursion-based method to assess measurement precision across IRT ability levels. This approach, detailed in Lim et al. (2020), enables the computation of conditional biases and CSEMs efficiently, bypassing the need for extensive simulations traditionally required for MST evaluation.

The module argument, used in conjunction with the item bank metadata x, systematically organizes items into modules for MST panel evaluation. Each row of x corresponds to an item, detailing its characteristics like score categories and IRT model. The module matrix, structured with the same number of rows as x and columns representing modules, indicates item assignments with 1s. This precise mapping enables the reval_mst function to evaluate the MST panel's performance by analyzing how items within each module contribute to measurement precision and bias, reflecting the tailored progression logic inherent in MST designs.

The route_map argument is essential for defining the MST's structure by indicating possible transitions between modules. Similar to the transMatrix in the **mstR** package (Magis et al., 2017), route_map is a binary matrix that outlines which module transitions are possible within an MST design. Each "1" in the matrix represents a feasible transition from one module (row) to another (column), effectively mapping the flow of test takers through the MST based on their performance. For instance, a "1" at the intersection of row *i* and column *j* indicates the possibility for test takers to progress from the module corresponding to row *i* directly to the module denoted by column *j*. This structure allows reval_mst to simulate and evaluate the dynamic routing of test takers through various stages and modules of the MST panel.

To further detail the cut_score argument with an illustration: In a 1-3-3 MST configuration, the list cut_score = list(c(-0.5, 0.5), c(-0.6, 0.6)) operates as a decision guide at each stage. Initially, all test takers start in the first module. Upon completion, their scores determine their next stage module: scores below -0.5 route to the first module of the next stage, between -0.5 and 0.5 to the second, and above 0.5 to the third. This pattern allows for dynamic adaptation, tailoring the test path to individual performance levels.

Value

This function returns a list of seven internal objects. The four objects are:

panel.info — A list of several sub-objects containing detailed information about the MST panel configuration, including:

- **config** A nested list indicating the arrangement of modules across stages, showing which modules are included in each stage. For example, the first stage includes module 1, the second stage includes modules 2 to 4, and so forth.
- **pathway** A matrix detailing all possible pathways through the MST panel. Each row represents a unique path a test taker might take, based on their performance and the cut scores defined.
- **n.module** A named vector indicating the number of modules available at each stage.
- **n.stage** A single numeric value representing the total number of stages in the MST panel.

item.by.mod — A list where each entry represents a module in the MST panel, detailing the item metadata within that module. Each module's metadata includes item IDs,

the number of categories, the IRT model used (model), and the item parameters (e.g., par.1, par.2, par.3).

`item.by.path` A list containing item metadata arranged according to the paths through the MST structure. This detailed breakdown allows for an analysis of item characteristics along specific MST paths. Each list entry corresponds to a testing stage and path, providing item metadata. This structure facilitates the examination of how items function within the context of each unique path through the MST.

`eq.theta` Estimated ability levels (θ) corresponding to the observed scores, derived from the inverse TCC scoring method. This provides the estimated θ values for each potential pathway through the MST stages. For each stage, θ values are calculated for each path, indicating the range of ability levels across the test takers. For instance, in a three-stage MST, the `eq.theta` list may contain θ estimates for multiple paths within each stage, reflecting the progression of ability estimates as participants move through the test. The example below illustrates the structure of `eq.theta` output for a 1-3-3 MST panel with varying paths:

- **stage.1** `path.1` shows θ estimates ranging from -7 to +7, demonstrating the initial spread of abilities.
- **stage.2** Multiple paths (`path.1`, `path.2`, ...) each with their own θ estimates, indicating divergence in ability levels based on test performance.
- **stage.3** Further refinement of θ estimates across paths, illustrating the final estimation of abilities after the last stage.

`cdist.by.mod` A list where each entry contains the conditional distributions of the observed scores for each module given the ability levels.

`jdist.by.path` Joint distributions of observed scores for different paths at each stage in a MST panel. The example below outlines the organization of `jdist.by.path` data in a hypothetical 1-3-3 MST panel:

- **stage.1** Represents the distribution at the initial stage, indicating the broad spread of test-taker abilities at the outset.
- **stage.2** Represents the conditional joint distributions of the observed scores as test-takers move through different paths at the stage 2, based on their performance in earlier stages.
- **stage.3** Represents a further refinement of joint distribution of observed scores as test-takers move through different paths at the final stage 3, based on their performance in earlier stages.

`eval.tb` A table summarizing the measurement precision of the MST panel. It contains the true ability levels (`theta`) with the mean ability estimates (`mu`), variance (`sigma2`), bias, and conditional standard error of measurement (CSEM) given the true ability levels. This table highlights the MST panel's accuracy and precision across different ability levels, providing insights into its effectiveness in estimating test-taker abilities.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

Lim, H., Davey, T., & Wells, C. S. (2020). A recursion-based analytical approach to evaluate the performance of MST. *Journal of Educational Measurement, 58*(2), 154-178.

See Also

shape_df, est_score

Examples

```
## ------------------------------------------------------------------------------
# Evaluation of a 1-3-3 MST panel using simMST data.
# This simulation dataset was utilized in Lim et al.'s (2020) simulation study.
# Details:
#    (a) Panel configuration: 1-3-3 MST panel
#    (b) Test length: 24 items (each module contains 8 items across all stages)
#    (c) IRT model: 3-parameter logistic model (3PLM)
## ------------------------------------------------------------------------------
# Load the necessary library
library(dplyr)
library(tidyr)
library(ggplot2)

# Import item bank metadata
x <- simMST$item_bank

# Import module information
module <- simMST$module

# Import routing map
route_map <- simMST$route_map

# Import cut scores for routing to subsequent modules
cut_score <- simMST$cut_score

# Import ability levels (theta) for evaluating measurement precision
theta <- simMST$theta

# Evaluate MST panel using the reval_mst() function
eval <-
  reval_mst(x,
    D = 1.702, route_map = route_map, module = module,
    cut_score = cut_score, theta = theta, range.tcc = c(-5, 5)
  )

# Review evaluation results
# The evaluation result table below includes conditional biases and
# standard errors of measurement (CSEMs) across ability levels
print(eval$eval.tb)
```

```
# Generate plots for biases and CSEMs
p_eval <-
  eval$eval.tb %>%
  dplyr::select(theta, bias, csem) %>%
  tidyr::pivot_longer(
    cols = c(bias, csem),
    names_to = "criterion", values_to = "value"
  ) %>%
  ggplot2::ggplot(mapping = ggplot2::aes(x = theta, y = value)) +
  ggplot2::geom_point(mapping = ggplot2::aes(shape = criterion), size = 3) +
  ggplot2::geom_line(
    mapping = ggplot2::aes(
      color = criterion,
      linetype = criterion
    ),
    linewidth = 1.5
  ) +
  ggplot2::labs(x = expression(theta), y = NULL) +
  ggplot2::theme_classic() +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.key.width = unit(1.5, "cm"))
print(p_eval)
```

`run_flexmirt` *Run flexMIRT through R*

Description

This function implements flexMIRT (Cai, 2017) to run a model specified in the syntax file of flexMIRT (i.e., *.flexmirt) through R. To run this function, flexMIRT software must be installed in advance. This function will be useful especially when conducting a simulation study using flexMIRT.

Usage

```
run_flexmirt(file.syntax, dir.flex = NULL, show.output.on.console = FALSE, ...)
```

Arguments

`file.syntax`	A single string or vector containing the file path(s) of a flexmirt syntax file(s) to be run. An example is "C:/Users/Data/irtmodel.flexmirt".
`dir.flex`	A path of directory where flexMIRT is installed. The path may include a folder name with "flexMIRT" (e.g, flexMIRT3, flexMIRT 3.6). If NULL, a path where flexMIRT is installed will be searched in "C:/Program Files" and it will be used as a default path (e.g., "C:/Program Files/flexMIRT3", "C:/Program Files/flexMIRT 3.6").

`show.output.on.console`
: A logical value to indicate whether to capture the output of the command and show it on the R console. Default is FALSE. See `system`.

`...`
: Further arguments passed from the function `system`.

Details

When a path of directory where flexMIRT (with a version < 3.6) is installed is provided in the argument `dir.flex`, the directory must include following six file of

- WinFlexMIRT.exe
- FlexMIRT_x64.exe
- FlexMIRT_x86.exe
- vpg.dll
- vpg.licensing.client.dll
- vpg.licensing.dll

When a path of directory where flexMIRT (with a version >= 3.6) is installed is provided in the argument `dir.flex`, the directory must include following six files of

- WinFlexMIRT.exe
- vpg.dll
- vpg.licensing.client.dll
- vpg.licensing.dll
- VPGLicenseClientNet.dll

and an additional directory of "Resources" that contains two files which are

- flexMIRT_x64_AVX.exe
- flexMIRT_x86_AVX.exe

Value

output files of flexMIRT

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

Examples

```
# Emxaples below will run when flexMIRT software is installed
# in a default path of "C:/Program Files/flexMIRT3".
# Otherwise provide a path where flexMIRT software is installed
# in the argument 'dir.flex'.

## Not run:
# (1) run a single syntax file
# import an example of flexMIRT syntax file to run the item parameter estimation of IRT 3PL model
file.syntax <- system.file("extdata", "2PLM_example.flexmirt", package = "irtQ")

# run flexMIRT to estimate the item parameters of IRT 3PL model
run_flexmirt(file.syntax = file.syntax, dir.flex = NULL, show.output = TRUE)

# check the output file
out.file <- system.file("extdata", "2PLM_example-prm.txt", package = "irtQ")
bring.flexmirt(out.file, type = "par")

# (2) run multiple syntax files
# import two examples of flexMIRT syntax files
file.syntax1 <- system.file("extdata", "2PLM_example.flexmirt", package = "irtQ")
file.syntax2 <- system.file("extdata", "3PLM_example.flexmirt", package = "irtQ")

# run flexMIRT to estimate the item parameters
run_flexmirt(file.syntax = c(file.syntax1, file.syntax2), dir.flex = NULL, show.output = FALSE)

# check the output file
out.file1 <- system.file("extdata", "2PLM_example-prm.txt", package = "irtQ")
out.file2 <- system.file("extdata", "3PLM_example-prm.txt", package = "irtQ")
bring.flexmirt(out.file1, type = "par")
bring.flexmirt(out.file2, type = "par")

## End(Not run)
```

---

shape_df *Create a data frame of item metadata*

Description

This function creates a data frame which includes item meta (e.g., item parameter, categories, models ...) to be used for the IRT model-data fit analysis as well as other analyses.

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 vectors of dichotomous item parameters. Namely, the item discrimination (a), item difficulty (b), and item guessing parameters.
par.prm	A list containing a vector of polytomous item discrimination (or slope) parameters and a list of polytomous item threshold (or step) parameters. In this list, the argument a should have a vector of slope parameters and the argument d should include a list of threshold (or step) parameters. See below for more details.
item.id	A character vector of item IDs. If NULL, an ID is automatically given to each item.
cats	A vector containing the number of score categories for items.
model	A character vector of IRT models corresponding to items. The available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous items, and "GRM" and "GPCM" for polytomous items. Note that "DRM" covers all dichotomous IRT models (i.e, "1PLM", "2PLM", and "3PLM") and "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively.
default.par	A logical value to create an item meta with default item parameters. If TRUE, the number of score categories and corresponding IRT models should be specified in the arguments of cats and model, respectively. In the default item meta, the item slope parameter has a fixed value of 1, the item difficulty (or threshold) parameter(s) has(have) a fixed value of 0, and the item guessing parameter has a fixed value of .2. Default is FALSE.

Details

For any item where "1PLM" or "2PLM" is specified in model, the item guessing parameter will be NA. If model is a vector of $length = 1$, the specified model is replicated across all items. As in the function simdat, it is important to clearly specify cats according to the order of items in the test form when a data frame for a mixed-format test needs to be created. See simdat for more details about how to specify cats.

When specifying item parameters in par.drm and/or par.prm, keep the order of item parameter types. For example, in the par.drm argument, the first argument a should contain the slope parameter vector, the second argument b should contain the difficulty vector, and the third argument g should contain the guessing parameter vector. In the par.drm argument, the first argument a should contain the slope parameter vector and the second argument d should contain a list including vectors of item threshold (or step) parameters for polytomous response IRT models. Note that when an item follows the (generalized) partial credit model, the item step parameters are the overall item difficulty (or location) parameter subtracted by the difficulty (or threshold) parameter for each category. Thus, the number of step parameters for item with m categories is m-1 because a step parameter for the first category does not affect the category probabilities.

Value

This function returns a data frame.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

info

Examples

```
## a mixed-item format test form
## with five dichotomous and two polytomous items
# create a list containing the 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 containing the polytomous item parameters
par.prm <- list(
  a = c(1.4, 0.6),
  d = list(
    c(0.0, -1.9, 1.2),
    c(0.4, -1.1, 1.5, 0.2)
  )
)

# create a numeric vector of score categories for the items
cats <- c(2, 4, 2, 2, 5, 2, 2)

# create a character vector of IRT models for the items
model <- c("DRM", "GRM", "DRM", "DRM", "GPCM", "DRM", "DRM")

# create an item meta set
shape_df(par.drm = par.drm, par.prm = par.prm, cats = cats, model = model)

## an empty item meta with five dichotomous and two polytomous items
# create a numeric vector of score categories for the items
cats <- c(2, 4, 3, 2, 5, 2, 2)

# create a character vector of IRT models for the items
model <- c("1PLM", "GRM", "GRM", "2PLM", "GPCM", "DRM", "3PLM")

# create an empty item meta set
shape_df(cats = cats, model = model, default.par = TRUE)

## an item meta for a single-item format test form with five dichotomous
shape_df(par.drm = par.drm, cats = rep(2, 5), model = "DRM")
```

simCAT_DC *Simulated single-item format CAT Data*

Description

This data set contains an item pool information, response data, and examinee's ability estimates.

Usage

```
simCAT_DC
```

Format

This data includes a list of length three. The first internal object is a data.frame of the item pool consisting of 100 dichotomous items. The item parameters of the first 90 items were generated with the IRT 2PL model and calibrated with the same model. However, the item parameters of the last 10 items were generated with the IRT 3PL model but calibrated with the IRT 2PL model. The second internal object is the response data set including a sparse response data set of 10,000 examinees for the items in the item pool. The third internal object is the examinee's ability estimates for 10,000 examinees.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

simCAT_MX *Simulated mixed-item format CAT Data*

Description

This data set contains an item pool information, response data, and examinee's ability estimates.

Usage

```
simCAT_MX
```

Format

This data includes a list of length three. The first internal object is a data.frame of the item pool consisting of 200 dichotomous items and 30 polytomous items. The dichotomous items were calibrated with the IRT 3PL model and the polytomous items were calibrated with the generalized partial credit model. All polytomous items have three score categories (i.e., 0, 1, 2). The second internal object is the response data set including a sparse response data set of 30,000 examinees for the items in the item pool. The third internal object is the examinee's ability estimates for 30,000 examinees.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

`simdat` *Simulated Response Data*

Description

This function generates a simulated response data for a single- or a mixed-format test forms. For dichotomous item response data, the IRT 1PL, 2PL, and 3PL models are available. For polytomous item response data, the graded response model, the partial credit model, and the generalized partial credit model are available.

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 the item metadata (e.g., item parameters, number of categories, models ...). This data frame can be easily obtained using the function `shape_df`. See below for details.
`theta`	A vector of theta values.
`a.drm`	A vector of item discrimination (or slope) parameters for dichotomous response IRT models.
`b.drm`	A vector of item difficulty (or threshold) parameters for dichotomous response IRT models.
`g.drm`	A vector of item guessing parameters for dichotomous IRT models.
`a.prm`	A vector of item discrimination (or slope) parameters for polytomous response IRT models.
`d.prm`	A list containing vectors of item threshold (or step) parameters for polytomous response IRT models.
`cats`	A vector containing the number of score categories for items.

`pr.model` A vector of character strings specifying the polytomous model with which response data are simulated. For each polytomous model, "GRM" for the graded response model or "GPCM" for the (generalized) partial credit model can be specified.

`D` A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

Details

There are two ways of generating the simulated response data. The first way is by using the argument `x` to read in a data frame of item metadata. In the data frame, the first column should have item IDs, the second column should contain unique score category numbers of the items, and the third column should include IRT models being fit to the items. The available IRT models are "1PLM", "2PLM", "3PLM", and "DRM" for dichotomous item data, and "GRM" and "GPCM" for polytomous item data. Note that "DRM" covers all dichotomous IRT models (i.e, "1PLM", "2PLM", and "3PLM") and "GRM" and "GPCM" represent the graded response model and (generalized) partial credit model, respectively. The next columns should include the item parameters of the fitted IRT models. For dichotomous items, the fourth, fifth, and sixth columns represent the item discrimination (or slope), item difficulty, and item guessing parameters, respectively. When "1PLM" and "2PLM" are specified in the third column, NAs should be inserted in the sixth column for the item guessing parameters. For polytomous items, the item discrimination (or slope) parameters should be included in the fourth column and the item difficulty (or threshold) parameters of category boundaries should be contained from the fifth to the last columns. When the number of unique score categories differs between items, the empty cells of item parameters should be filled with NAs. In the **irtQ** package, the item difficulty (or threshold) parameters of category boundaries for GPCM are expressed as the item location (or overall difficulty) parameter subtracted by the threshold parameter for unique score categories of the item. Note that when an GPCM item has *K* unique score categories, *K-1* item difficulty parameters are necessary because the item difficulty parameter for the first category boundary is always 0. For example, if an GPCM item has five score categories, four item difficulty parameters should be specified. An example of a data frame with a single-format test is as follows:

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

And an example of a data frame for a mixed-format test is as follows:

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 IRT Models section in the page of irtQ-package for more details about the IRT models used in the **irtQ** package. An easier way to create a data frame for the argument x is by using the function shape_df.

The second way is by directly specifying item parameters for each item for which response data should be simulated (i.e., without using a data frame, as shown in the examples that follow). In addition to item parameters, theta, cats, pr.model, and D should be specified as well. g.drm does not need to be specified when only the 1PL and 2PL models are used for dichotomous item response data. For dichotomous items, 2s should be specified in cats. For polytomous items, the number of unique score categories should be specified in cats. When a response data set is generated with a mixed-format test, it is important to clearly specify cats according to the order of items in the test form. Suppose that the response data of ten examinees are simulated with five items, including three dichotomous items and two polytomous items with three categories. Also, suppose that the second and the forth items are the polytomous items. Then, cats = c(2, 3, 2, 3, 2) should be used. Additionally, among those two polytomous items, if the first and second item response data are simulated from the graded response model and generalized partial credit model, respectively, then pr.model = c('GRM', 'GPCM').

Value

This function returns a vector or a matrix. When a matrix is returned, rows indicate theta values and columns represent items.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

drm, prm

Examples

```
## example 1.
## simulates response data with a mixed-format test.
## for the first two polytomous items, the generalized partial credit model is used
## for the last polytomous item, the graded response model is used
# 100 examinees are sampled
theta <- rnorm(100)

# set item parameters for three dichotomous items with 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 item parameters
# note that 4, 4, and 5 categories are used for polytomous items
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))

# create a numeric vector of score categories for both dichotomous and polytomous item data
# this score category vector is used to specify the location of the polytomous items
```

```
cats <- c(2, 2, 4, 4, 5, 2)

# create a character vector of the IRT model for the polytomous items
pr.model <- c("GPCM", "GPCM", "GRM")

# simulate the response data
simdat(
  theta = theta, a.drm = a.drm, b.drm = b.drm, g.drm = NULL,
  a.prm = a.prm, d.prm = d.prm, cats = cats, pr.model = pr.model, D = 1
)


## example 2.
## simulates response data with a single-format test with the 2PL model.
# create a numeric vector of score categories for the three 2PL model items
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.
## the use of a "-prm.txt" file obtained from a flexMIRT
# import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# read item parameters and transform them to item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# simulate the response data
simdat(x = test_flex, theta = theta, D = 1) # use a data.frame of item meta information
```

---

simMG *Simulated multiple-group data*

Description

This data set has a list consisting of item metadata, item response data, and group names of three simulated groups.

Usage

```
simMG
```

Format

This data set includes a list of three internal objects: (1) a list of item metadata (item.prm) for three groups, (2) a list of item response data (res.dat) for the three groups, and (3) a vector of group names (group.name) for the three groups.

The first internal object (item.prm) contains a list of item metadata of three test forms for the three groups. In terms of test forms, the test forms for the first and second groups have fifty items consisting of forty seven 3PLM items and three GRM items. The test form for the third group has thirty eight items consisting of thirty seven 3PLM items and one GRM item. Among the three forms, the first and second test forms share twelve common items (C1I1 through C1I12) and the second and third test forms share ten common items (C2I1 through c2I10). There is no common item between the first and third forms. The item parameters in the item metadata were used to simulate the item response data sets for the three groups (see the second object of the list).

Regrading the second internal object, all three response data sets were simulated with 2,000 latent abilities randomly sampled from $N(0, 1)$ (Group 1), $N(0.5, 0.8^2)$ (Group 2), and $N(-0.3, 1.3^2)$ (Group 3), respectively, using the true item parameters provided in the item metadata.

The third internal object is a vector of three group names which are "Group1", "Group2", and "Group3".

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

`simMST` *Simulated 1-3-3 MST panel data*

Description

This simulated 1-3-3 MST panel data set was used in Lim et al.' (2020) simulation study.

Usage

```
simMST
```

Format

This data set includes a list of five internal objects:

`item_bank` The item bank metadata, containing item parameters and other information.

`module` A binary matrix that maps items from the item bank to modules within the MST panel. This parameter enables precise item-to-module assignments for MST configurations, analogous to the `modules` argument in the `randomMST` function of the **mstR** package (Magis et al., 2017).

`route_map` A binary square matrix that defines the MST structure, illustrating transitions between modules and stages. This concept is inspired by the `transMatrix` argument in the `randomMST` function from the **mstR** package (Magis et al., 2017).

`cut_score` A list defining cut scores for routing test takers through MST stages. Each list element is a vector of cut scores for advancing test takers to subsequent stage modules.

`theta` A vector of ability levels (theta) at which the MST panel's performance is assessed, allowing for the evaluation of measurement precision across a continuum of ability levels.

This 1-3-3 MST panel consists of 7 modules in total with 3 stages. Each module contains eight items calibrated with the IRT 3 parameter logistic model.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

Lim, H., Davey, T., & Wells, C. S. (2020). A recursion-based analytical approach to evaluate the performance of MST. *Journal of Educational Measurement, 58*(2), 154-178.

`summary` *Summary of item calibration*

Description

This method function summarizes the IRT calibration results of `est_irt` or `est_item` object.

Usage

```
summary(object, ...)

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

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

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

Arguments

`object`	An object of class `est_irt` or `est_item`.
`...`	Further arguments passed to or from other methods.

Value

The method function returns a list of internal objects extracted from `est_irt` or `est_item` object and displays a summary of the IRT calibration results on the console panel.

Methods (by class)

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

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

est_irt, est_item

Examples

```
# fit the 1PL model to LSAT6 data and constrain the slope parameters to be equal
fit.1pl <- est_irt(data = LSAT6, D = 1, model = "1PLM", cats = 2, fix.a.1pl = FALSE)

# summary of the estimation
summary(fit.1pl)
```

---

sx2_fit *S-X2 fit statistic*

Description

This function computes $S - X^2$ (Orlando & Thissen, 2000, 2003) item fit statistic.

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 the item metadata (e.g., item parameters, number of categories, models ...), an object of class est_item obtained from the function est_item, or an object of class est_irt obtained from the function est_irt. See irtfit, info, or simdat for more details about the item metadata. The data frame of item metadata can be easily obtained using the function shape_df.
...	Further arguments passed to or from other methods.
data	A matrix containing examinees' response data for the items in the argument x. A row and column indicate the examinees and items, respectively.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.
alpha	A numeric value to specify significance α-level of the hypothesis test for $S-X^2$ fit statistic. Default is .05.
min.collapse	An integer value to indicate the minimum frequency of cells to be collapsed. Default is 1. See below for details.
norm.prior	A numeric vector of two components specifying a mean and standard deviation of the normal prior distribution. These two parameters are used to obtain the gaussian quadrature points and the corresponding weights from the normal distribution. Default is c(0,1).
nquad	An integer value specifying the number of gaussian quadrature points from the normal prior distribution. Default is 30.
weights	A two-column matrix or data frame containing the quadrature points (in the first column) and the corresponding weights (in the second column) of the latent variable prior distribution. The weights and quadrature points can be easily obtained using the function gen.weight. If missing, default values are used (see the arguments of norm.prior and nquad).

`pcm.loc` A vector of integer values indicating the locations of partial credit model (PCM) items whose slope parameters are fixed to certain values. Default is NULL.

Details

Often, very small expected frequencies in the contingency tables used to compute χ^2 fit statistics could compromise the accuracy of the χ^2 approximation for their distribution (Orlando & Thissen, 2000). To avoid this problem, Orlando and Thissen (2000) used an algorithm of collapsing adjacent test score groups to maintain a minimum expected category frequency of 1. However, if Orlando and Thissen's cell collapsing approach is applied to polytomous data, too much information would be lost (Kang & Chen, 2008). Thus, Kang and Chen (2008) collapsed adjacent cells of item score categories for a specific score group to ensure a minimum expected category frequency of 1. The same collapsing strategies were applied in the function `sx2_fit`. If a minimum expected category frequency needs to be set to different number, you can specify the minimum value in the argument `min.collapse`.

Note that if "DRM" is specified for an item in the item metadata set, the item is considered as "3PLM" to compute degree of freedom of the $S - X^2$ fit statistic.

Also, any missing responses in `data` are replaced with incorrect responses (i.e., 0s).

Value

This function returns a list. Within a list, several internal objects are contained such as:

`fit_stat` A data frame containing the results of $S - X^2$ fit statistics for all items.

`item_df` The item metadata specified in the argument `x`.

`exp_freq` A list containing the collapsed expected frequency tables for all items.

`obs_freq` A list containing the collapsed observed frequency tables for all items.

`exp_prob` A list containing the collapsed expected probability tables for all items.

`obs_prop` A list containing the collapsed observed proportion tables for all items.

Methods (by class)

- `sx2_fit(default)`: Default method to compute $S - X^2$ fit statistics for a data frame `x` containing the item metadata.
- `sx2_fit(est_item)`: An object created by the function `est_item`.
- `sx2_fit(est_irt)`: An object created by the function `est_irt`.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Kang, T., & Chen, T. T. (2008). Performance of the generalized S-X2 item fit index for polytomous IRT models. *Journal of Educational Measurement, 45*(4), 391-406.

Orlando, M., & Thissen, D. (2000). Likelihood-based item-fit indices for dichotomous item response theory models. *Applied Psychological Measurement, 24*(1), 50-64.

Orlando, M., & Thissen, D. (2003). Further investigation of the performance of S-X2: An item fit index for use with dichotomous item response theory models. *Applied Psychological Measurement, 27*(4), 289-298.

See Also

irtfit, info, simdat, shape_df, est_item

Examples

```
## example 1: all five polytomous IRT models are 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 the response data
data <- simdat(x = x, theta = score, D = 1)


# compute fit statistics
fit1 <- sx2_fit(x = x, data = data, nquad = 30)

# fit statistics
fit1$fit_stat


## example 2: first 39th and 40th items follows GRM and 53rd, 54th, and 55th items
##            follow PCM (thus, the slope parameters are fixed to 1)
# replace the model names with GPCM and
# assign 1 to the slope parameters for the 53rd, 54th, and 55th items
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 the 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)

# fit statistics
fit2$fit_stat
```

traceline *Compute Item/Test Characteristic Functions*

Description

This function computes the item category probabilities, item characteristic function, and test characteristic function given a set of theta values. The returned object of this function can be used to draw the item or test characteristic curve using the function 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 the item metadata (e.g., item parameters, number of categories, models ...), an object of class est_item obtained from the function est_item, or an object of class est_irt obtained from the function est_irt. See irtfit, info, or simdat for more details about the item metadata. The data frame of item metadata can be easily obtained using the function shape_df.
...	Further arguments passed to or from other methods.
theta	A vector of theta values.
D	A scaling factor in IRT models to make the logistic function as close as possible to the normal ogive function (if set to 1.7). Default is 1.

Value

This function returns an object of class traceline. This object contains a list containing the item category probabilities, item characteristic function, and test characteristic function.

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_item`

Examples

```
## example
## using a "-prm.txt" file obtained from a flexMIRT
# import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtQ")

# read item parameters and transform them to item metadata
test_flex <- bring.flexmirt(file = flex_prm, "par")$Group1$full_df

# set theta values
theta <- seq(-3, 3, 0.5)

# compute the item category probabilities and item/test
# characteristic functions given the theta values
traceline(x = test_flex, theta, D = 1)
```

`write.flexmirt` *Write a "-prm.txt" file for flexMIRT*

Description

This function writes an output file of "-prm.txt" for flexMIRT (Cai, 2017). The current version of this function can be used only for the unidimensional IRT models. This function was written by modifying the function `read.flexmirt` (Pritikin & Falk, 2020).

Usage

```
write.flexmirt(
  x,
  file = NULL,
  norm.pop = c(0, 1),
  rePar = TRUE,
  mgroup = FALSE,
  group.name = NULL
)
```

Arguments

x	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...) for a single group or a list of the item metadata for multiple groups. See est_irt, irtfit, info, or simdat for more details about the item metadata. The item metadata can be easily created using the function shape_df.
file	The destination file name.
norm.pop	A numeric vector of two components specifying a mean and standard deviation of the normal population ability distribution for a single group or a list of the numeric vectors of length two for multiple groups. When a list is provided, each internal numeric vector should contain a mean and standard deviation of the ability distribution of each group (e.g., norm.pop = list(c(0, 1), c(0, 0.8), c(0.5, 1.2)) for three groups). When mgroup = TRUE and a single vector of length two is provided (e.g., norm.pop = c(0, 1)), the same vector will be recycled across all groups. Default is c(0,1).
rePar	A logical value indicating whether the item parameters in the item metadata are the reparameterized item parameters. If TRUE, the item intercepts and logits of item guessing parameters should be included in the item metadata. If FALSE, the item difficulty and item guessing parameters should be included in the item metadata.
mgroup	A logical value indicating whether a "-prm.txt" file is created for a single group or multiple groups. Default is FALSE.
group.name	A character vector of group names. If NULL, the group names are automatically generated (e.g., Group1).

Value

A "-prm.txt" file.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

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

Examples

```
## 1. Create "-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 a name of "-prm.txt" file
temp_prm <- file.path(tempdir(), "single_group_temp-prm.txt")

# 1-(3) write out the "-prm.txt" file
write.flexmirt(x, file = temp_prm, norm.pop = c(0, 1), rePar = FALSE)

## 2. Create "-prm.txt" file for multiple groups
##    using the simulated three multiple group data
# 2-(1) extract the item metadata
x <- simMG$item.prm

# set a name of "-prm.txt" file
temp_prm <- file.path(tempdir(), "mg_group_temp-prm1.txt")

# write out the "-prm.txt" file
write.flexmirt(x,
  file = temp_prm, norm.pop = list(c(0, 1), c(0.5, 0.8), c(-0.3, 1.3)),
  rePar = FALSE, mgroup = TRUE, group.name = c("GR1", "GR2", "GR3")
)

# or write out the "-prm.txt" file so that
# all groups have the same ability distributions
# and the group names are generate autoumatically
temp_prm <- file.path(tempdir(), "mg_group_temp-prm2.txt")
write.flexmirt(x,
  file = temp_prm, norm.pop = c(0, 1),
  rePar = FALSE, mgroup = TRUE, group.name = NULL
)
```

Index