Package 'irtplay'

November 4, 2021

Type Package

Title Unidimensional Item Response Theory Modeling

Version 1.6.3

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 unidimensional IRT models. For the item parameter estimation, marginal maximum likelihood estimation via expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin (1981) <doi:10.1007/BF02294168>) is used. For the online calibration, the fixed item parameter calibration method (Kim (2006) <doi:10.1111/j.1745-3984.2006.00021.x>) and the fixed ability parameter calibration method (Ban, Hanson, Wang, Yi, & Harris (2011) <doi:10.1111/j.1745-3984.2001.tb01123.x>) are provided. For the ability estimation, several popular scoring methods (e.g., MLE, EAP, and MAP) are implemented. In terms of assessing the IRT model-data fit, one of distinguished features of this package is that it gives not only well-known item fit statistics (e.g., chi-square (X2), likelihood ratio chi-square (G2), infit and oufit statistics, and S-X2 statistic (Ames & Penfield (2015) <doi:10.1111/emip.12067>)) but also graphical displays to look at residuals between the observed data and model-based predictions (Hambleton, Swaminathan, & Rogers (1991, ISBN:9780803936478)). In addition, there are many useful functions such as analyzing differential item functioning, computing asymptotic variance-covariance matrices of item parameter estimates (Li & Lissitz (2004) <doi:10.1111/j.1745-3984.2004.tb01109.x>), 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 (Lord & Wingersky (1984) <doi:10.1177/014662168400800409>), computing the loglikelihood of individual items, computing the loglikelihood of abilities, 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.

Depends R (>= 3.6)

License GPL (>= 2)

Encoding UTF-8

LazyData true

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

RoxygenNote 7.1.1

Suggests mirt

URL https://github.com/hwangQ/irtplay

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

NeedsCompilation no

Author Hwanggyu Lim [aut, cre], Craig S. Wells [ctb]

Maintainer Hwanggyu Lim <hglim83@gmail.com>

Repository CRAN

Date/Publication 2021-11-04 16:20:02 UTC

R topics documented:

`irtplay-package` *irtplay: 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 examine the IRT model-data fit on item-level in different ways as well as provide useful functions related to unidimensional IRT.

For the item parameter estimation, the marginal maximum likelihood estimation via expectation-maximization (MMLE-EM) algorithm (Bock & Aitkin, 1981) is used. For the online calibration, 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., MLE, EAP, and MAP) are implemented. In terms of assessing the IRT model-data fit, one of distinguished features of this package is that it gives not only item fit statistics (e.g., χ^2 fit statistic (e.g., Bock, 1960; Yen, 1981), likelihood ratio χ^2 fit statistic (G^2; McKinley & Mills, 1985), infit and outfit statistics (Ames et al., 2015), and $S - X^2$ (Orlando & Thissen, 2000, 2003)) but also graphical displays to look at residuals between the observed data and model-based predictions (Hambleton, Swaminathan, & Rogers, 1991).

In addition, there are many useful functions such as analyzing differential item functioning (DIF), computing asymptotic variance-covariance matrices of item parameter estimates,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 the loglikelihood of individual items, computing the loglikelihood of abilities, 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:	irtplay
Version:	1.6.3
Date:	2021-11-04
Depends:	R (>= 3.6)
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) the process of evaluating the IRT model-data fit, c) two examples for the online calibration and evaluating the IRT model-data fit, and d) IRT Models used in **irtplay** 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 **irtplay** 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 **irtplay** 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.

The process of evaluating the IRT model-data fit

One way to assess goodness of IRT model-data fit is through an item fit analysis by examining the traditional item fit statistics and looking at the discrepancy between the observed data and model-based predictions. Using **irtplay** package, the traditional approach of evaluating the IRT model-data fit on item-level can be implemented with three main steps:

1. Prepare a data set for the IRT item fit analysis (i.e., item metadata, ability estimates, and response data).
2. Obtain the IRT fit statistics such as χ^2, G^2, infit, and outfit statistics using the function `irtfit`.
3. Based on the results of IRT model fit analysis (i.e., an object of class `irtfit`) obtained in step 2, draw the IRT residual plots (i.e., raw residual and standardized residual plots) using the function `plot.irtfit`.

1. Preparing a data set Before conducting the IRT model fit analysis, it is necessary to prepare a data set. To run the function `irtfit`, it requires three data sets:

1. Item metadata including the item ID, number of score categories, IRT models, and item parameters. The item metadata should be in the format of data frame. You can prepare the data either by using the function `shape_df` or by creating a data frame of the item metadata by yourself. If you have output files of item parameter estimates obtained from one of the IRT software such as BILOG-MG 3, PARSCALE 4, flexMIRT, and mirt (R package), the item metadata can be easily obtained using the functions of `bring.bilog`, `bring.parscale`, `bring.flexmirt`, and `bring.mirt`. See `irtfit`, `test.info`, or `simdat` for more details about the item metadata format.

2. Examinees' ability (or proficiency) estimates. It should be in the format of a numeric vector.
3. Examinees' 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. The order of the items in the response data set must be exactly the same as that of the items in the item metadata.

2. Computing the IRT model-data fit statistics The function `irtfit` computes the traditional IRT item fit statistics such as χ^2, G^2, infit, and outfit statistics. To calculate the χ^2 and G^2 statistics, two methods are available to divide the ability scale into several groups. The two methods are "equal.width" for dividing the scale by an equal length of the interval and "equal.freq" for dividing the scale by an equal frequency of examinees. Also, you need to specify the location of ability point at each group (or interval) where the expected probabilities of score categories are calculated from 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.

To use the function `irtfit`, you need to insert the item metadata in the argument `x`, the ability estimates in the argument `score`, and the response data in the argument `data`. 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`. In addition, if the response data include missing values, you must indicate the missing value in argument `missing`.

Once the function `irtfit` has been implemented, you'll get the fit statistic results and the contingency tables for every item used to calculate the χ^2 and G^2 fit statistics.

3. Drawing the IRT residual plots Using the saved object of class `irtfit`, you can use the `plot` method to evaluate the IRT raw residual and standardized residual plots.

Because the `plot` method can draw the residual plots for an item at a time, you have to indicate which item will be examined. For this, you can specify an integer value, which is the location of the studied item, in the argument `item.loc`.

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), "cp" for Clopper-Pearson interval (Clopper & Pearson, 1934), "wilson" for Wilson score interval (Wilson, 1927), and "wilson.cr" for Wilson score interval with continuity correction (Newcombe, 1998).

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(irtplay)

##-----------------------------------------------------------------------------
# 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 = "irtplay")

# 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 Method A:
##-----------------------------------------------------------------------------

## 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 = "irtplay")

# 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)


##----------------------------------------------------------------------------
# 3. The example code below shows how to prepare the data sets and how to conduct
#    the IRT model-data fit analysis:
##----------------------------------------------------------------------------

## Step 1: prepare a data set for IRT
## In this example, we use the simulated mixed-item format of CAT Data
## But, only items that have examinees' responses more than 1,000 are assessed.

# find the location of items that have more than 1,000 item responses
over1000 <- which(colSums(simCAT_MX$res.dat, na.rm=TRUE) > 1000)

# (1) item metadata
x <- simCAT_MX$item.prm[over1000, ]
```

```
# (2) examinee's ability estimates
score <- simCAT_MX$score

# (3) response data
data <- simCAT_MX$res.dat[, over1000]

## Step 2: Compute the IRT mode-data fit statistics
# (1) the use of "equal.width"
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)

# what kinds of internal objects does the results have?
names(fit1)

# show the results of the fit statistics
fit1$fit_stat[1:10, ]

# show the contingency tables for the first item (dichotomous item)
fit1$contingency.fitstat[[1]]

# (2) the use of "equal.freq"
fit2 <- irtfit(x=x, score=score, data=data, group.method="equal.freq",
               n.width=10, loc.theta="average", range.score=NULL, D=1,
               alpha=0.05, missing=NA)

# show the results of the fit statistics
fit2$fit_stat[1:10, ]

# show the contingency table for the fourth item (polytomous item)
fit2$contingency.fitstat[[4]]

## Step 3: Draw the IRT residual plots
# 1. for the dichotomous item
# (1) both raw and standardized residual plots using the object "fit1"
plot(x=fit1, item.loc=1, type = "both", ci.method = "wald",
     ylim.sr.adjust=TRUE)

# (2) the raw residual plots using the object "fit1"
plot(x=fit1, item.loc=1, type = "icc", ci.method = "wald",
     ylim.sr.adjust=TRUE)

# (3) the standardized residual plots using the object "fit1"
plot(x=fit1, item.loc=113, type = "sr", ci.method = "wald",
     ylim.sr.adjust=TRUE)

# 2. for the polytomous item
```

```
# (1) both raw and standardized residual plots using the object "fit1"
plot(x=fit1, item.loc=113, type = "both", ci.method = "wald",
     ylim.sr.adjust=TRUE)

# (2) the raw residual plots using the object "fit1"
plot(x=fit1, item.loc=113, type = "icc", ci.method = "wald",
     layout.col=2, ylim.sr.adjust=TRUE)

# (3) the standardized residual plots using the object "fit1"
plot(x=fit1, item.loc=113, type = "sr", ci.method = "wald",
     layout.col=4, ylim.sr.adjust=TRUE)
```

IRT Models

In the **irtplay** 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 ($k = 0, 1, ..., K$) 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 ($k = 0, 1, ..., K$) 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 **irtplay** 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.

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.

Clopper, C. J., & Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika, 26*(4), 404-413.

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.

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., & Han, K. T. (Under review). A residual-based differential item functioning detection framework in item response theory. *Journal of Educational Measurement*.

Lim, H., Choe, E. M., Han, K. T., Lee, S., & Hong, M. (2021, June). *IRT residual approach to detecting DIF.* Paper presented at the Annual Meeting of the National Council on Measurement in Education. Online.

Lim, H., Davey, T., & Wells, C. S. (2020). A recursion-based analytical approach to evaluate the performance of MST. *Journal of Educational Measurement*. DOI: 10.1111/jedm.12276.

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

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.

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.

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.

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"))
```

Arguments

`List`	A list containing different length of numeric vectors
`type`	A character string specifying whether rbind is used or cbind is used.

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")
```

---

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"),
  rePrm = TRUE,
  rePrm.gpc = TRUE,
  n.factor = 1
)

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

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

bring.mirt(x)

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.
rePrm	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.
rePrm.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, 2018). The functions bring.bilog and bring.parscale were written by modifying the functions read.bilog and read.parscale (Weeks, 2017), 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 `test.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

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 = "irtplay")

# 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 = "irtplay")
```

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

`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`, `test.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, test.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 = "irtplay")

# 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=40)
```

---

drm *Dichotomous Response Model Probabilities*

Description

This function computes the probability of correct answers for one or more 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 vector or matrix. When a matrix is returned, rows indicate theta values and columns represent items.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

plm

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 marginal maximum likelihood estimation with 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,
  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.
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. This information is only required when x = NULL and fipc = FALSE.
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 NULL and all items are binary items (i.e., dichotomous items), it assumes that all items have two score categories. This information is only required when x = NULL and fipc = FALSE.

`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. 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. 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. 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 item parameters are calibrated against the estimated empirical histogram prior distribution. 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. 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 location of the items to be fixed in the item metadata (i.e., `x`) when the FIPC is implemented. 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)`.

`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 **irtplay** 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 `irtplay-package` for more details about the IRT models used in the **irtplay** 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 `EmpHist = TRUE`, the empirical histogram 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 observed information functions. The standard errors are estimated using the cross-production approximation method (Meilijson, 1989).
`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 observed data set (marginal log-likelihood) across all estimated items.
`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 matrix or 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 the 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 location 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, test.info, simdat, shape_df, sx2_fit, traceline.est_item, 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 <- test.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 differnt 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 = "irtplay")

# 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, empty.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 = "MLE", 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 = "irtplay")

# 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)

# 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)
```

`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, test.info or simdat for more details about the item metadata.
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. This information is only required when x = 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 NULL and all items are binary items (i.e., dichotomous items), it assumes that all items have two score categories. This information is only required when `x = 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`, `test.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 = "irtplay")

# 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_score` *Estimate examinees' ability (proficiency) parameters*

Description

This function estimates examinees' latent ability parameters. Available scoring methods are maximum likelihood estimation (MLE), maximum likelihood estimation with fences (MLEF; Han, 2016), 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 = "MLE",
  range = c(-4, 4),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  fence.a = 3,
  fence.b = NULL,
  se = TRUE,
  obs.info = TRUE,
  constant = 0.1,
  constraint = FALSE,
  range.tcc = c(-7, 7),
  missing = NA,
  ncore = 1,
  ...
)

## S3 method for class 'est_irt'
est_score(
  x,
  method = "MLE",
  range = c(-4, 4),
  norm.prior = c(0, 1),
  nquad = 41,
  weights = NULL,
  fence.a = 3,
```

```
  fence.b = NULL,
  se = TRUE,
  obs.info = TRUE,
  constant = 0.1,
  constraint = FALSE,
  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, test.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 "MLE" for the maximum likelihood estimation, "MLEF" for the maximum likelihood estimation with fences, "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 "MLE".
range	A numeric vector of two components to restrict the range of ability scale for the MLE. Default is c(-4, 4).
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 "MLE", "MLEF", 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 "MLE", "MLEF", "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 "MLE", "MLEF", "MAP", or "INV.TCC".

`fence.a` A numeric value specifying the item slope parameter (i.e., *a*-parameter) for the two imaginary items in MLEF. 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 MLEF. When `fence.b = NULL`, the lower and upper fences of item difficulty parameters were automatically set. See below for details. Default is NULL.

`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.

`obs.info` A logical value. If TRUE, the observed item information functions are used to compute the standard errors of ability estimates when "MLE", "MLEF", or "MAP" is specified in `method`. If FALSE, the expected item information (a.k.a. Fisher information) functions are used to compute the standard errors. Note that under the 1PL and 2PL models, the observed item information function is exactly equal to the expected item information function. Default is TRUE.

`constant` A numeric value used to adjust zero and perfect raw sum scores, or the raw sum score equal to the sum of item guessing parameters, if necessary, to find estimable solutions for those raw sum scores when `method = "INV.TCC"`. The zero raw score is forced to become the score of "zero raw score + constant" and the perfect raw score is forced to become the score of "perfect raw score - constant". If the 3PLM items are included in the item metadata, the raw sum score equal to the sum of item guessing parameters is forced to become the score of "the raw sum score + constant". Default is .1.

`constraint` A logical value indicating whether the ability estimates will be restricted within a specific ability range specified in the argument `range.tcc` when `method = "INV.TCC"`. If `constraint = TRUE`, all ability estimates less than the first value in the vector specified in the argument `range.tcc` are transformed to the first value and all ability estimates greater than the second value in the vector specified in the argument `range.tcc` are transformed to the second value. Also, when `constraint = TRUE` and the 3PLM items are contained in the item metadata, linear interpolation method is used to find the ability estimates for the raw sum scores less than the sum of item guessing parameters. When `constraint = FALSE` and the 3PLM items are contained in the item metadata, linear extrapolation method is used to find the ability estimates for the raw sum scores less than the sum of item guessing parameters. See below for details. Default is FALSE.

`range.tcc` A numeric vector of two components to be used as the lower and upper bounds of ability estimates when `method = "INV.TCC"` and `constraint = TRUE`. 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 MLE, MLEF, MAP, and EAP is used. However, there must be no missing data in the response data set when "EAP.SUM" or "INV.TCC" is used. One of possible ways to use "EAP.SUM" or "INV.TCC" method when missing values exist is to remove rows with any missing values.

In the maximum likelihood estimation with fences (MLEF; 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 MLEF, the function automatically sets the lower and upper fences of item difficulty parameters using two steps. More specifically, in the first step, the lower fence of the item difficulty parameter is set to the greatest integer value less than the minimum of item difficulty parameters in the item metadata and the upper fence of the item difficulty parameter is set to the smallest integer value greater than the maximum of item difficulty parameters in the item metadata. Then, in the second step, if the lower fence set in the first step is greater than -3.5, the lower fence is constrained to -3.5 and if the upper fence set in the first step is less than 3.5, the upper fence is constrained to 3.5. Otherwise, the fence values of item difficulty parameters set in the first step are used.

When "INV.TCC" method is used employing the IRT 3-parameter logistic model (3PLM) in a test, ability estimates for the raw sum scores less than the sum of item guessing parameters are not attainable. In this case, either of linear interpolation and linear extrapolation can be applied. Note that if `constraint = TRUE`, linear interpolation method is used. Otherwise, linear extrapolation method is used. Let θ_{min} and θ_{max} be the minimum and maximum ability estimates and θ_X be the ability estimate for the smallest raw score, X, greater than or equal to the sum of item guessing parameters. When linear interpolation method is used, a linear line is constructed between two points of (x=θ_{min}, y=0) and (x=θ_X, y=X). Because `constraint = TRUE`, θ_{min} is the first value in the argument `range.tcc`. When linear extrapolation method is used, a linear line is constructed using two points of (x=θ_X, y=X) and (x=θ_{max}, y=maximum raw score). Then, ability estimates for the raw sum scores between zero and the smallest raw score greater than or equal to the sum of item guessing parameters are found using the constructed linear line. 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).

To speed up the ability estimation for MLE, MLEF, 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 observed information functions for MLE, MLEF, and MAP methods.

Value

A list including a vector of the ability estimates and a vector of the standard errors of ability estimates. When `method` is "EAP.SUM" or "INV.TCC", raw sum scores of examinees and a table with the possible raw sum scores and corresponding ability estimates are returned as well.

Methods (by class)

- `default`: Default method to estimate examinees' latent ability parameters using a data frame `x` containing the item metadata.
- `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.

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.

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*. DOI: 10.1111/jedm.12276.

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.

See Also

`irtfit`, `test.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 = "irtplay")

# 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 MLE
est_score(x, data, D=1, method="MLE", range=c(-4, 4), se=TRUE, ncore=2)

# estimate the abilities using MLEF with default fences of item difficulty parameters
est_score(x, data, D=1, method="MLEF", fence.a=3.0, fence.b=NULL, se=TRUE, ncore=2)

# estimate the abilities using MLEF with different fences of item difficulty parameters
est_score(x, data, D=1, method="MLEF", fence.a=3.0, fence.b=c(-5, 5), se=TRUE, ncore=2)

# estimate the abilities using MAP
est_score(x, data, D=1, method="MAP", norm.prior=c(0, 1), nquad=30, se=TRUE, ncore=2)

# estimate the abilities using EAP
est_score(x, data, D=1, method="EAP", norm.prior=c(0, 1), nquad=30, se=TRUE, ncore=2)

# 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", constant=0.1, constraint=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' or 'est_item' objects*

Description

This function extracts various internal objects from an object of class est_irt or est_item .

Usage

```
getirt(x, ...)

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

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

Arguments

x	An object of class est_irt 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 observed information functions. The standard errors are estimated using the cross-production approximation method (Meilijson, 1989).

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 observed data set (marginal log-likelihood) across all estimated items.

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 matrix or 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 the 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 location of the fixed items when the FIPC was implemented.

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 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.

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

Methods (by class)

- `est_irt`: An object created by the function `est_irt`.
- `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"))
```

`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,
  ...
)

## 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,
  ...
)
```

```
## 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,
  ...
)
```

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.
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 details.
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.

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 **irtplay** 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 `irtplay-package` for more details about the IRT models used in the **irtplay** 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)

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

# 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_item` *Loglikelihood of Items*

Description

This function computes the loglikelihoods of individual items given the item parameters, ability values, and response data.

Usage

```
llike_item(
  x,
  data,
  score,
  D = 1,
  use.aprior = FALSE,
  use.bprior = FALSE,
  use.gprior = FALSE,
  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
)
```

Arguments

`x`	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...). See `irtfit`, `test.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.
`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.

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

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

`use.gprior` A logical value. If TRUE, a prior distribution for the guessing parameters is used when computing the loglikelihood values 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.

Value

A vector of loglikelihood values. Each element represents a sum of loglikeihoods across all ability values for each item.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

Examples

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

# 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(10, mean=0, sd=1)

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

# compute the loglikelihood values (no priors are used)
llike_item(x, data, score, D=1, use.aprior=FALSE, use.gprior=FALSE)
```

---

llike_score *Loglikelihood of ability*

Description

This function computes the loglikelihood of abilities for examinees given the item parameters and response data.

Usage

```
llike_score(
  x,
  data,
  theta,
  D = 1,
  method = "MLE",
  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 categories, models ...). See irtfit, test.info, or simdat for more details about the item metadata. This data frame can be easily obtained using the function shape_df.
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.
theta	A numeric vector of abilities of which loglikelihood 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.
method	A character string indicating a scoring method. Available methods are "MLE" for the maximum likelihood estimation, "MLEF" for the maximum likelihood estimation with fences, "MAP" for the maximum a posteriori estimation. Default method is "MLE".
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 "MLE" or "MLEF".
fence.a	A numeric value specifying the item slope parameter (i.e., *a*-parameter) for the two imaginary items in MLEF. 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 MLEF. When fence.b = NULL, the lower and upper fences of item difficulty parameters were automatically set. See below for details. Default is NULL.
missing	A value indicating missing values in the response data set. Default is NA.

Details

The loglikelihood function of ability for an examinee can be computed given the item parameters and the examinee's response data for the items. For example, if you want to examine the loglikelihood functions of abilities for two examinees given the same test items specified in the argument x, then you should provide the item response data matrix with two rows in the argument data and a vector of ability points where the loglikelihood values need to be computed in the argument theta. Or if you want to examine the loglikelihood function of ability for an examinee given the test items specified in the argument x, then you should provide the item response data matrix with one row (or a vector of item response data) in the argument data and a vector of ability points where the loglikelihood values need to be computed in the argument theta.

Value

A data frame of loglikelihood values. A row indicates the ability value where the loglikelihood is computed and a column represents a response pattern, respectively.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

Examples

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

# read item parameters and transform them to 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)

# set the ability values where the loglikelihood values are computed
theta <- seq(-3, 3, 0.5)

# compute the loglikelihood values (When MLE method is used)
llike_score(x=x, data=data, theta=theta, D=1, method="MLE")
```

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, test.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 is used instead of a probabiliy matrix
## with a mixed-format test
# import the "-prm.txt" output file from flexMIRT
flex_prm <- system.file("extdata", "flexmirt_sample-prm.txt", package = "irtplay")

# 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(-1, 1, 0.1), D=1)
```

---

plm *Polytomous Response Model 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, partial credit model, and generalized partial credit model.

Usage

```
plm(theta, a, d, D = 1, pmodel = 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.
pmodel	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, `a = 1` for that item. When `pmodel = "GPCM"`, `d` should include the item difficulty (or threshold) parameters. In the **irtplay** 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 `irtplay-package`.

Value

This function returns a vector or matrix. When a matrix is returned, rows indicate theta values and columns represent categories of an 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
plm(theta=c(-0.2, 0, 0.5), a=1.4, d=c(-0.2, 0, 0.5), D=1, pmodel='GPCM')

## Category probabilities for an item with five categories
## using a graded response model
plm(theta=c(-0.2, 0, 0.5), a=1.2, d=c(-0.4, -0.2, 0.4, 1.5), D=1, pmodel='GRM')
```

---

plot.irtfit *Draw raw and standardized residual plots*

Description

This 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", "cp", "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, "cp" for Clopper-Pearson interval, "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), "cp" for Clopper-Pearson interval (Clopper & Pearson, 1934), "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 as triangles. Otherwise, they are displayed as circles.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

Clopper, C. J., & Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika, 26*(4), 404-413.

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 = "irtplay")
```

```
# 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.test.info` *Plot Item and Test Information Functions*

Description

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

Usage

```
## S3 method for class 'test.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 test.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 test.info can be obtained from the function test.info.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

test.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 = "irtplay")

# 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 <- test.info(x=test_flex, theta=theta, D=1)

# 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 across 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.traceline *Plot ICC and TCC*

Description

This 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.

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 = "irtplay")

# 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 poltyomous items
# in one panel
plot(x, item.loc=53:55, score.curve=TRUE, overlap=TRUE)
```

---

post_den *Updated prior (a.k.a. posterior) latent ability distribution*

Description

This function computes updated prior (a.k.a. posterior) densities of the latent ability distribution given a prior ability distribution, item parameters, and item response data.

Usage

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

Arguments

x	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...). See irtfit, test.info, or simdat for more details about the item metadata. This data frame can be easily obtained using the function shape_df.
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.

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. 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. 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. Default is 0.
group.var	A positive numeric value to set the variance of latent variable prior distribution. Default is 1.
missing	A value indicating missing values in the response data set. Default is NA.

Value

This function returns a list containing two internal objects. The first internal object is a data frame with two columns, where the first column has theta values (nodes) and the second column provides the weights of the posterior latent ability distribution. The second internal object is a data frame containing the mean, variance, and standard deviation of the distribution.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

See Also

shape_df, irtfit, test.info, simdat

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
(x <- getirt(x=mod.2pl, what="par.est"))

# update the standard normal prior deisnty of the ability distribution
# using the estimated item parameters
(upd_prior <- post_den(x=x, data=LSAT6, D=1, group.mean=0, group.var=1))
```

rdif *IRT residual-based DIF (RDIF) detection framework*

Description

This function computes three RDIF statistics (Lim, Choe, & Han, Under review; Lim, Choe, Han, Lee, & Hong, 2021), 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,
  data,
  score = NULL,
  group,
  focal.name,
  D = 1,
  alpha = 0.05,
  missing = NA,
  purify = FALSE,
  purify.by = c("rdif_rs", "rdif_r", "rdif_s"),
  max.iter = 10,
  min.resp = NULL,
  method = "MLE",
  range = c(-4, 4),
  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 data frame of item metadata can be easily obtained using the function shape_df. See est_irt, irtfit, test.info or simdat for more details about the item metadata.
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 function estimates the abilities before computing 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 vector should the same with the number of rows in the response data matrix.
focal.name	A single numeric or character indicating the level of group which corresponds to the focal group. For example, if group = c(0,1,0,1,1) and '1' indicates the focal group, then 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 fit 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 "rdif_rs" for $RDIF_{RS}$, "rdif_r" for $RDIF_R$, and "rdif_s" for $RDIF_S$.
max.iter	An positive integer value specifying the maximum number of iterations for the purification process. Default is 10.
min.resp	An positive integer value specifying the minimum number of item responses for an examinee when scoring is conducted. Default is NULL. See details below for more information.
method	A character string indicating a scoring method. Available methods are "MLE" for the maximum likelihood estimation, "MAP" for the maximum a posteriori estimation, and "EAP" for the expected a posteriori estimation. Default method is "MLE".
range	A numeric vector of two components to restrict the range of ability scale for the MLE. Default is c(-4, 4).
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 "MLE".
nquad	An integer value specifying the number of gaussian quadrature points from the normal prior distribution. Default is 41. Ignored if method is "MLE" 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 "MLE" 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., Under review; Lim et al., 2021) 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. (2021) 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 only supports dichotomous 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., MLE), 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 = "MLE"`).

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="rdif_rs"`). 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. (2021) for more details about the purification procedure.

Scoring with a few items entails large standard errors which in turn could compromise DIF detection with RDIF framework. The `min.resp` argument can be used to avoid using scores with large standard errors when computing the RDIF statistics, espeically during the purification process. For example, if `min.resp` is not NULL (e.g., `min.resp=5`), item responses of examinees whose tally of item responses are less than the specified minimum number are treated as missing values (i.e., NA). Accordingly, their ability estimates become missing values and are not used for computing the RDIF statistics. If `min.resp=NULL`, an examinee's score will be computed as long as there exists, at least, 1 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 focal group, sample size of the reference 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. "rdif_r", "rdif_s", and "rdif_rs" 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 focal group, sample size of the reference 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 repleated 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.

Author(s)

Hwanggyu Lim <hglim83@gmail.com>

References

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

Lim, H., Choe, E. M., Han, K. T., Lee, S., & Hong, M. (2021, June). *IRT residual approach to detecting DIF.* Paper presented at the Annual Meeting of the National Council on Measurement in Education. Online.

See Also

est_item, test.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 = "irtplay")

# 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.dc=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 MLE
score <- est_score(x=est_par, data=data, method="MLE")$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="MLE")
print(dif_nopuri_2)

# (b)-1 compute RDIF statistics with a purification
#       based on \eqn{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="rdif_r")
print(dif_puri_r)

# (b)-2 compute RDIF statistics with a purification
#       based on \eqn{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="rdif_s")
print(dif_puri_s)

# (b)-3 compute RDIF statistics with a purification
```

```
#       based on \eqn{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="rdif_rs")
print(dif_puri_rs)
```

`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 = "irtplay")

# 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 = "irtplay")
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 = "irtplay")
file.syntax2 <- system.file("extdata", "3PLM_example.flexmirt", package = "irtplay")

# 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 = "irtplay")
out.file2 <- system.file("extdata", "3PLM_example-prm.txt", package = "irtplay")
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.dc = list(a = NULL, b = NULL, g = NULL),
  par.py = list(a = NULL, d = NULL),
  item.id = NULL,
  cats,
  model,
  empty.par = FALSE
)
```

Arguments

par.dc	A list containing three vectors of dichotomous item parameters. Namely, the item discrimination (a), item difficulty (b), and item guessing parameters.
par.py	A list containing a vector of polytomous item discrimination (or slope) parameters and a list of polytomous item threshold (or step) parameters. In the 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.

`empty.par` A logical value to create an empty item meta. If TRUE, the number of score categories and corresponding IRT models should be specified in the arguments of `cats` and `model`, respectively. In the empty item meta, the item slope parameter has a fixed value of 1, the item difficulty (or threshold) parameter has 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.dc` and `par.dc`, keep the order of item parameter types. For example, in the list of `par.dc`, the order of items parameters should be the slope, the difficulty, and the guessing parameters.

When specifying item parameters in `par.dc`, note that in the list of the threshold (or step) parameters, each vector should contain the threshold (or step) parameters for each item. 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

`test.info`

Examples

```
## a mixed-item format test form
## with five dichotomous and two polytomous items
# create a list containing the dichotomous item parameters
par.dc <- 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.py <- 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.dc=par.dc, par.py=par.py, 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, empty.par=TRUE)

## an item meta for a single-item format test form with five dichotomous
shape_df(par.dc=par.dc, 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.dc,
  b.dc,
  g.dc = NULL,
  a.py,
  d.py,
  cats,
  pmodel,
  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.dc	A vector of item discrimination (or slope) parameters for dichotomous IRT models.
b.dc	A vector of item difficulty (or threshold) parameters for dichotomous IRT models.
g.dc	A vector of item guessing parameters for dichotomous IRT models.
a.py	A vector of item discrimination (or slope) parameters for polytomous IRT models.
d.py	A list containing vectors of item threshold (or step) parameters for polytomous IRT models.
cats	A vector containing the number of score categories for items.
pmodel	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 **irtplay** 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 `irtplay-package` for more details about the IRT models used in the **irtplay** 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`, `pmodel`, and `D` should be specified as well. `g.dc` 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 `pmodel = 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, plm

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.dc <- c(1, 1.2, 1.3); b.dc <- c(-1, 0, 1); g.dc <- 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.py <- c(1.3, 1.2, 1.7)
d.py <- 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 categoires 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
pmodel <- c('GPCM', 'GPCM', 'GRM')

# simulate the response data
simdat(theta=theta, a.dc=a.dc, b.dc=b.dc, g.dc=NULL,
       a.py=a.py, d.py=d.py, cats=cats, pmodel=pmodel, D=1)


## example 2.
## simulates response data with a sigle-format test with the 2PL model.
# create a numeric vector of score categoires for the three 2PL model items
cats <- rep(2, 3)

# simulate the response data
simdat(theta=theta, a.dc=a.dc, b.dc=b.dc, 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 = "irtplay")

# 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.farame of item meta information
```

---

summary *Summary of item calibration*

Description

This 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_item'
summary(object, ...)
```

Arguments

`object`	An object of class `est_irt` or `est_item`.
`...`	Further arguments passed to or from other methods.

Methods (by class)

- `est_irt`: An object created by the function `est_irt`.
- `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,
  ...
)

## S3 method for class 'est_item'
sx2_fit(
  x,
  alpha = 0.05,
  min.collapse = 1,
  norm.prior = c(0, 1),
  nquad = 30,
  weights,
  ...
)

## S3 method for class 'est_irt'
sx2_fit(
```

```
  x,
  alpha = 0.05,
  min.collapse = 1,
  norm.prior = c(0, 1),
  nquad = 30,
  weights,
  ...
)
```

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, test.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).

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.

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)

- default: Default method to compute $S - X^2$ fit statistics for a data frame x containing the item metadata.
- est_item: An object created by the function est_item.
- 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, test.info, simdat, shape_df, est_item

Examples

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

# select the first twenty dichotomous items and last polytomous item
# assuming that the test consists of twenty-one items
x <- bring.flexmirt(file=flex_sam, "par")$Group1$full_df[c(1:20, 55), ]

# 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
fit <- sx2_fit(x=x, data=data, nquad=30)

# fit statistics
fit$fit_stat
```

`test.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

```
test.info(x, ...)

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

## S3 method for class 'est_item'
test.info(x, theta, ...)

## S3 method for class 'est_irt'
test.info(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. 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.

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 **irtplay** 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 `irtplay-package` for more details about the IRT models used in the **irtplay** 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 `test.info`. This object contains item and test information values given the specified theta values.

Methods (by class)

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

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.

See Also

`plot.test.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.dc <- 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.py <- 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 metadata set
test <- shape_df(par.dc=par.dc, par.py=par.py, 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
test.info(x=test, theta=theta, D=1)


## 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 = "irtplay")

# 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
test.info(x=test_flex, theta=theta, D=1)
```

`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, test.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)

- 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.
- est_item: An object created by the function est_item.
- 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 = "irtplay")

# 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.

Usage

```
write.flexmirt(x, file = NULL, norm.pop = c(0, 1), rePrm = TRUE)
```

Arguments

`x`	A data frame containing the item metadata (e.g., item parameters, number of categories, models ...). See `irtfit`, `test.info`, or `simdat` for more details about the item metadata. This data frame can be easily obtained 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 distribution. Default is c(0,1).
`rePrm`	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.

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.

Examples

```
## use the simulated CAT data
# extract the item metadata
x <- simCAT_MX$item.prm

# set a name of "-prm.txt" file
```

```
temp_prm <- file.path(tempdir(), "temp-prm.txt")

# write out the "-prm.txt" file
write.flexmirt(x, file=temp_prm, norm.pop=c(0, 1), rePrm=FALSE)
```

Index