Package 'TestDesign'

September 22, 2021

Type Package

Title Optimal Test Design Approach to Fixed and Adaptive Test Construction

Version 1.2.6

Date 2021-09-10

Maintainer Seung W. Choi <schoi@austin.utexas.edu>

Description Use the optimal test design approach by Birnbaum (1968, ISBN:9781593119348) and van der Linden (2018) <doi:10.1201/9781315117430> in constructing fixed and adaptive tests. Supports the following mixed-integer programming (MIP) solver packages: 'lpsymphony', 'Rsymphony', 'gurobi', 'lpSolve', and 'Rglpk'. The 'gurobi' package is not available from CRAN; see <https://www.gurobi.com/downloads/>.

URL https://choi-phd.github.io/TestDesign/ (documentation)

BugReports https://github.com/choi-phd/TestDesign/issues/

License GPL (>= 2)

Depends R (>= 2.10)

biocViews

Imports Rcpp (>= 1.0.0), methods, lpSolve, foreach, logitnorm, crayon

SystemRequirements C++11

Suggests lpsymphony, Rsymphony, gurobi, Rglpk, mirt, progress, shiny, shinythemes, shinyWidgets, shinyjs, DT, knitr, rmarkdown, kableExtra, testthat (>= 2.1.0), pkgdown, pkgload

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.2

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Collate 'RcppExports.R' 'import.R' 'extensions.R' 'item_class.R' 'item_functions.R' 'loading_functions.R' 'static_class.R' 'shadow_class.R' 'item_pool_operators.R' 'item_attrib_operators.R' 'st_attrib_operators.R' 'constraints_operators.R' 'static_functions.R' 'shadow_functions.R' 'bayes_functions.R' 'constraint_functions.R' 'cpp_documents.R' 'datasets.R' 'eligibility_functions.R' 'exposure_control_functions.R' 'solver_functions.R' 'helper_functions.R' 'item_pool_cluster_operators.R' 'other_functions.R' 'plot_functions.R' 'summary_class.R' 'print_functions.R' 'runshiny.R' 'shadowtest_functions.R' 'summary_functions.R' 'show_functions.R' 'test_operators.R' 'theta_functions.R' 'xdata_functions.R'

NeedsCompilation yes

Author Seung W. Choi [aut, cre] (<https://orcid.org/0000-0003-4777-5420>), Sangdon Lim [aut] (<https://orcid.org/0000-0002-2988-014X>)

Repository CRAN

Date/Publication 2021-09-22 08:40:11 UTC

R topics documented:

---

app *Open TestDesign app*

Description

app and OAT are aliases of TestDesign.

Usage

```
app()

OAT()
```

Details

TestDesign is a caller function to open the Shiny interface of TestDesign package.

Examples

```
## Not run:
if (interactive()) {
  TestDesign()
}

## End(Not run)
```

---

buildConstraints *Build constraints (shortcut to other loading functions)*

Description

buildConstraints is a data loading function to create a constraints object. buildConstraints is a shortcut that calls other data loading functions. The constraints must be in the expected format; see the vignette in vignette("constraints").

Usage

```
buildConstraints(
  object,
  item_pool,
  item_attrib,
  st_attrib = NULL,
  pool = NULL,
  constraints = NULL
)
```

Arguments

object	constraint specifications. Can be a data.frame or the file path of a .csv file. See the vignette for the expected format.
item_pool	item parameters. Can be a item_pool object, a data.frame or the file path of a .csv file.
item_attrib	item attributes. Can be an item_attrib object, a data.frame or the file path of a .csv file.
st_attrib	(optional) stimulus attributes. Can be an st_attrib object, a data.frame or the file path of a .csv file.
pool	(deprecated) use item_pool argument instead.
constraints	(deprecated) use object argument instead.

Value

buildConstraints returns a constraints object. This object is used in Static and Shadow.

Examples

```
## Read from objects:
constraints_science <- buildConstraints(constraints_science_data,
  itempool_science, itemattrib_science)
constraints_reading <- buildConstraints(constraints_reading_data,
  itempool_reading, itemattrib_reading, stimattrib_reading)

## Read from data.frame:
constraints_science <- buildConstraints(constraints_science_data,
  itempool_science_data, itemattrib_science_data)
constraints_reading <- buildConstraints(constraints_reading_data,
  itempool_reading_data, itemattrib_reading_data, stimattrib_reading_data)

## Read from file: write to tempdir() for illustration and clean afterwards
f1 <- file.path(tempdir(), "constraints_science.csv")
f2 <- file.path(tempdir(), "itempool_science.csv")
f3 <- file.path(tempdir(), "itemattrib_science.csv")
write.csv(constraints_science_data, f1, row.names = FALSE)
write.csv(itempool_science_data   , f2, row.names = FALSE)
write.csv(itemattrib_science_data , f3, row.names = FALSE)
constraints_science <- buildConstraints(f1, f2, f3)
```

```
file.remove(f1)
file.remove(f2)
file.remove(f3)
```

---

calcEscore *Calculate expected scores*

Description

calcEscore is a function to calculate expected scores.

Usage

```
calcEscore(object, theta)

## S4 method for signature 'item_1PL,numeric'
calcEscore(object, theta)

## S4 method for signature 'item_2PL,numeric'
calcEscore(object, theta)

## S4 method for signature 'item_3PL,numeric'
calcEscore(object, theta)

## S4 method for signature 'item_PC,numeric'
calcEscore(object, theta)

## S4 method for signature 'item_GPC,numeric'
calcEscore(object, theta)

## S4 method for signature 'item_GR,numeric'
calcEscore(object, theta)

## S4 method for signature 'item_pool,numeric'
calcEscore(object, theta)

## S4 method for signature 'item_1PL,matrix'
calcEscore(object, theta)

## S4 method for signature 'item_2PL,matrix'
calcEscore(object, theta)

## S4 method for signature 'item_3PL,matrix'
calcEscore(object, theta)

## S4 method for signature 'item_PC,matrix'
calcEscore(object, theta)
```

```
## S4 method for signature 'item_GPC,matrix'
calcEscore(object, theta)

## S4 method for signature 'item_GR,matrix'
calcEscore(object, theta)

## S4 method for signature 'item_pool,matrix'
calcEscore(object, theta)

## S4 method for signature 'item_pool_cluster,numeric'
calcEscore(object, theta)
```

Arguments

object	an item or an item_pool object.
theta	theta values to use.

Value

item **object:** calcEscore a vector containing expected score of the item at the theta values.

item_pool **object:** calcEscore returns a vector containing the pool-level expected score at the theta values.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
item_1     <- new("item_1PL", difficulty = 0.5)
item_2     <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3     <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4     <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
item_5     <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
item_6     <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)

ICC_item_1 <- calcEscore(item_1, seq(-3, 3, 1))
ICC_item_2 <- calcEscore(item_2, seq(-3, 3, 1))
ICC_item_3 <- calcEscore(item_3, seq(-3, 3, 1))
ICC_item_4 <- calcEscore(item_4, seq(-3, 3, 1))
ICC_item_5 <- calcEscore(item_5, seq(-3, 3, 1))
ICC_item_6 <- calcEscore(item_6, seq(-3, 3, 1))
TCC_pool   <- calcEscore(itempool_science, seq(-3, 3, 1))
```

`calcFisher` *Calculate Fisher information*

Description

`calcFisher` is a function to calculate Fisher information.

Usage

```
calcFisher(object, theta)

## S4 method for signature 'item_1PL,numeric'
calcFisher(object, theta)

## S4 method for signature 'item_2PL,numeric'
calcFisher(object, theta)

## S4 method for signature 'item_3PL,numeric'
calcFisher(object, theta)

## S4 method for signature 'item_PC,numeric'
calcFisher(object, theta)

## S4 method for signature 'item_GPC,numeric'
calcFisher(object, theta)

## S4 method for signature 'item_GR,numeric'
calcFisher(object, theta)

## S4 method for signature 'item_pool,numeric'
calcFisher(object, theta)
```

```
## S4 method for signature 'item_1PL,matrix'
calcFisher(object, theta)

## S4 method for signature 'item_2PL,matrix'
calcFisher(object, theta)

## S4 method for signature 'item_3PL,matrix'
calcFisher(object, theta)

## S4 method for signature 'item_PC,matrix'
calcFisher(object, theta)

## S4 method for signature 'item_GPC,matrix'
calcFisher(object, theta)

## S4 method for signature 'item_GR,matrix'
calcFisher(object, theta)

## S4 method for signature 'item_pool,matrix'
calcFisher(object, theta)

## S4 method for signature 'item_pool_cluster,numeric'
calcFisher(object, theta)
```

Arguments

object	an item or an item_pool object.
theta	theta values to use.

Value

item **object:** calcFisher returns a (*nq*, *1*) matrix of information values.

item_pool **object:** calcProb returns a (*nq*, *ni*) matrix of information values.

notations
- *nq* denotes the number of theta values.
- *ni* denotes the number of items in the item_pool object.

A vector of Fisher information values over theta (nq values) for a single item or a matrix of dimension (nq, ni) for an "item_pool".

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
item_1      <- new("item_1PL", difficulty = 0.5)
item_2      <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3      <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4      <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
item_5      <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
item_6      <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)

info_item_1 <- calcFisher(item_1, seq(-3, 3, 1))
info_item_2 <- calcFisher(item_2, seq(-3, 3, 1))
info_item_3 <- calcFisher(item_3, seq(-3, 3, 1))
info_item_4 <- calcFisher(item_4, seq(-3, 3, 1))
info_item_5 <- calcFisher(item_5, seq(-3, 3, 1))
info_item_6 <- calcFisher(item_6, seq(-3, 3, 1))
info_pool   <- calcFisher(itempool_science, seq(-3, 3, 1))
```

`calcHessian` *Calculate second derivative of log-likelihood*

Description

`calcHessian` is a function to calculate the second derivative of the log-likelihood function.

Usage

```
calcHessian(object, theta, resp)

## S4 method for signature 'item_1PL,numeric,numeric'
calcHessian(object, theta, resp)

## S4 method for signature 'item_2PL,numeric,numeric'
calcHessian(object, theta, resp)

## S4 method for signature 'item_3PL,numeric,numeric'
calcHessian(object, theta, resp)

## S4 method for signature 'item_PC,numeric,numeric'
calcHessian(object, theta, resp)

## S4 method for signature 'item_GPC,numeric,numeric'
calcHessian(object, theta, resp)

## S4 method for signature 'item_GR,numeric,numeric'
calcHessian(object, theta, resp)

## S4 method for signature 'item_pool,numeric,numeric'
calcHessian(object, theta, resp)

## S4 method for signature 'item_pool_cluster,numeric,list'
calcHessian(object, theta, resp)
```

Arguments

`object`	an `item` or an `item_pool` object.
`theta`	theta values to use.
`resp`	the response data to use. This must be a single value for an `item`, or a length *ni* vector for an `item_pool`.

Details

notations

- *nq* denotes the number of theta values.
- *ni* denotes the number of items in the `item_pool` object.

Value

`item` **object:** `calcHessian` returns a length *nq* vector containing the second derivative of the log-likelihood function, of observing the response at each theta.

`item_pool` **object:** `calcHessian` returns a (*nq*, *ni*) matrix containing the second derivative of the log-likelihood function, of observing the response at each theta.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17.*

Examples

```
item_1    <- new("item_1PL", difficulty = 0.5)
item_2    <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3    <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4    <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
item_5    <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
item_6    <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)

h_item_1 <- calcHessian(item_1, seq(-3, 3, 1), 0)
h_item_2 <- calcHessian(item_2, seq(-3, 3, 1), 0)
h_item_3 <- calcHessian(item_3, seq(-3, 3, 1), 0)
h_item_4 <- calcHessian(item_4, seq(-3, 3, 1), 0)
h_item_5 <- calcHessian(item_5, seq(-3, 3, 1), 0)
h_item_6 <- calcHessian(item_6, seq(-3, 3, 1), 0)
h_pool   <- calcHessian(
  itempool_science, seq(-3, 3, 1),
  rep(0, itempool_science@ni)
)
```

calcJacobian *Calculate first derivative of log-likelihood*

Description

calcJacobian is a function to calculate the first derivative of the log-likelihood function.

Usage

```
calcJacobian(object, theta, resp)

## S4 method for signature 'item_1PL,numeric,numeric'
calcJacobian(object, theta, resp)

## S4 method for signature 'item_2PL,numeric,numeric'
calcJacobian(object, theta, resp)

## S4 method for signature 'item_3PL,numeric,numeric'
calcJacobian(object, theta, resp)

## S4 method for signature 'item_PC,numeric,numeric'
calcJacobian(object, theta, resp)

## S4 method for signature 'item_GPC,numeric,numeric'
calcJacobian(object, theta, resp)

## S4 method for signature 'item_GR,numeric,numeric'
calcJacobian(object, theta, resp)

## S4 method for signature 'item_pool,numeric,numeric'
calcJacobian(object, theta, resp)

## S4 method for signature 'item_pool_cluster,numeric,list'
calcJacobian(object, theta, resp)
```

Arguments

object	an item or an item_pool object.
theta	theta values to use.
resp	the response data to use.

Value

item **object:** calcJacobian returns a length *nq* vector containing the first derivative of the log-likelihood function, of observing the response at each theta.

item_pool **object:** calcJacobian returns a (*nq*, *ni*) matrix containing the first derivative of the log-likelihood function, of observing the response at each theta.

notations • *nq* denotes the number of theta values.
• *ni* denotes the number of items in the `item_pool` object.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
item_1    <- new("item_1PL", difficulty = 0.5)
item_2    <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3    <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4    <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
item_5    <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
item_6    <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)

j_item_1 <- calcJacobian(item_1, seq(-3, 3, 1), 0)
j_item_2 <- calcJacobian(item_2, seq(-3, 3, 1), 0)
j_item_3 <- calcJacobian(item_3, seq(-3, 3, 1), 0)
j_item_4 <- calcJacobian(item_4, seq(-3, 3, 1), 0)
j_item_5 <- calcJacobian(item_5, seq(-3, 3, 1), 0)
j_item_6 <- calcJacobian(item_6, seq(-3, 3, 1), 0)
j_pool   <- calcJacobian(
  itempool_science, seq(-3, 3, 1),
  rep(0, itempool_science@ni)
)
```

`calcLocation-methods` *Calculate central location (overall difficulty)*

Description

`calcLocation` is a function to calculate the central location (overall difficulty) of items.

Usage

```
calcLocation(object)

## S4 method for signature 'item_1PL'
calcLocation(object)

## S4 method for signature 'item_2PL'
calcLocation(object)

## S4 method for signature 'item_3PL'
calcLocation(object)

## S4 method for signature 'item_PC'
calcLocation(object)

## S4 method for signature 'item_GPC'
calcLocation(object)

## S4 method for signature 'item_GR'
calcLocation(object)

## S4 method for signature 'item_pool'
calcLocation(object)
```

Arguments

`object`	an `item` or an `item_pool` object.

Value

`item` **object:** `calcLocation` returns a theta value representing the central location.

`item_pool` **object:** `calcProb` returns a length *ni* list, each containing the central location of the item.

notations • *ni* denotes the number of items in the `item_pool` object.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
item_1      <- new("item_1PL", difficulty = 0.5)
item_2      <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3      <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4      <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
item_5      <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
item_6      <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)

loc_item_1 <- calcLocation(item_1)
loc_item_2 <- calcLocation(item_2)
loc_item_3 <- calcLocation(item_3)
loc_item_4 <- calcLocation(item_4)
loc_item_5 <- calcLocation(item_5)
loc_item_6 <- calcLocation(item_6)
loc_pool   <- calcLocation(itempool_science)
```

calcLogLikelihood *Calculate log-likelihood*

Description

calcLogLikelihood is a function to calculate log-likelihood values.

Usage

```
calcLogLikelihood(object, theta, resp)

## S4 method for signature 'item_pool,numeric,numeric'
calcLogLikelihood(object, theta, resp)

## S4 method for signature 'item_pool,numeric,matrix'
calcLogLikelihood(object, theta, resp)

## S4 method for signature 'item_pool,matrix,numeric'
calcLogLikelihood(object, theta, resp)

## S4 method for signature 'item_pool,matrix,matrix'
calcLogLikelihood(object, theta, resp)
```

Arguments

object	an item_pool object.
theta	theta values to use.
resp	the response data to use.

Value

calcLogLikelihood returns values of log-likelihoods.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
j_pool   <- calcLogLikelihood(itempool_science, seq(-3, 3, 1), 0)
```

`calcProb-methods` *Calculate item response probabilities*

Description

`calcProb` is a function to calculate item response probabilities.

Usage

```
calcProb(object, theta)

## S4 method for signature 'item_1PL,numeric'
calcProb(object, theta)

## S4 method for signature 'item_2PL,numeric'
calcProb(object, theta)

## S4 method for signature 'item_3PL,numeric'
calcProb(object, theta)

## S4 method for signature 'item_PC,numeric'
calcProb(object, theta)

## S4 method for signature 'item_GPC,numeric'
calcProb(object, theta)

## S4 method for signature 'item_GR,numeric'
calcProb(object, theta)

## S4 method for signature 'item_pool,numeric'
```

```
calcProb(object, theta)

## S4 method for signature 'item_1PL,matrix'
calcProb(object, theta)

## S4 method for signature 'item_2PL,matrix'
calcProb(object, theta)

## S4 method for signature 'item_3PL,matrix'
calcProb(object, theta)

## S4 method for signature 'item_PC,matrix'
calcProb(object, theta)

## S4 method for signature 'item_GPC,matrix'
calcProb(object, theta)

## S4 method for signature 'item_GR,matrix'
calcProb(object, theta)

## S4 method for signature 'item_pool,matrix'
calcProb(object, theta)

## S4 method for signature 'item_pool_cluster,numeric'
calcProb(object, theta)
```

Arguments

object	an item or an item_pool object.
theta	theta values to use.

Value

item **object:** calcProb returns a (*nq*, *ncat*) matrix of probability values.

item_pool **object:** calcProb returns a length *ni* list, each containing a matrix of probability values.

notations

- *nq* denotes the number of theta values.
- *ncat* denotes the number of response categories.
- *ni* denotes the number of items in the item_pool object.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
item_1      <- new("item_1PL", difficulty = 0.5)
item_2      <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3      <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4      <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
item_5      <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
item_6      <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)

prob_item_1 <- calcProb(item_1, seq(-3, 3, 1))
prob_item_2 <- calcProb(item_2, seq(-3, 3, 1))
prob_item_3 <- calcProb(item_3, seq(-3, 3, 1))
prob_item_4 <- calcProb(item_4, seq(-3, 3, 1))
prob_item_5 <- calcProb(item_5, seq(-3, 3, 1))
prob_item_6 <- calcProb(item_6, seq(-3, 3, 1))
prob_pool   <- calcProb(itempool_science, seq(-3, 3, 1))
```

---

calc_info *Calculate Fisher information (multiple items)*

Description

`calc_info` and `calc_info_matrix` are functions to calculate Fisher information. These functions are designed for multiple items.

Usage

```
calc_info(x, item_parm, ncat, model)

calc_info_matrix(x, item_parm, ncat, model)
```

Arguments

x	the theta value. This must be a column vector in matrix form for `array_info_*` functions.
item_parm	a matrix containing item parameters. Each row represents each item.
ncat	a vector containing the number of response categories of each item.
model	a vector indicating item models of each item, using • 1: 1PL model • 2: 2PL model • 3: 3PL model • 4: PC model • 5: GPC model • 6: GR model

Details

`calc_info` accepts a single theta value, and `calc_info_matrix` accepts multiple theta values.

Currently supports unidimensional models.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
# item parameters
item_parm <- matrix(c(
  1, NA,   NA,
  1,  2,   NA,
  1,  2, 0.25,
  0,  1,   NA,
  2,  0,    1,
  2,  0,    2),
  nrow = 6,
  byrow = TRUE
)

ncat  <- c(2, 2, 2, 3, 3, 3)
model <- c(1, 2, 3, 4, 5, 6)

# single theta example
x <- 0.5
calc_info(x, item_parm, ncat, model)
# same as
info_1pl(x, 1)
info_2pl(x, 1, 2)
info_3pl(x, 1, 2, 0.25)
info_pc(x, c(0, 1))
info_gpc(x, 2, c(0, 1))
info_gr(x, 2, c(0, 2))

# multiple thetas example
x <- matrix(seq(0.1, 0.5, 0.1)) # column vector in matrix form
calc_info_matrix(x, item_parm, ncat, model)
# same as
array_info_1pl(x, 1)
array_info_2pl(x, 1, 2)
array_info_3pl(x, 1, 2, 0.25)
array_info_pc(x, c(0, 1))
array_info_gpc(x, 2, c(0, 1))
array_info_gr(x, 2, c(0, 2))
```

`calc_info_EB` *Calculate the Fisher information using empirical Bayes*

Description

Calculate the Fisher information using empirical Bayes.

Usage

```
calc_info_EB(x, item_parm, ncat, model)
```

Arguments

x	A numeric vector of MCMC sampled theta values.
item_parm	A numeric matrix of item parameters.
ncat	a numeric vector specifying the number of response categories in each item.
model	a numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).

---

calc_info_FB *Calculate the Fisher information using full Bayesian*

Description

Calculate the Fisher information using full Bayesian.

Usage

```
calc_info_FB(x, items_list, ncat, model, useEAP = FALSE)
```

Arguments

x	A numeric vector of MCMC sampled theta values.
items_list	A list of item parameter matrices.
ncat	a numeric vector specifying the number of response categories in each item.
model	a numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
useEAP	TRUE to use the mean of MCMC theta draws.

---

calc_likelihood *Calculate likelihoods*

Description

calc_likelihood and calc_likelihood_function are functions to calculate likelihoods.

Usage

```
calc_likelihood(x, item_parm, resp, ncat, model)

calc_likelihood_function(theta_grid, item_parm, resp, ncat, model)

calc_log_likelihood(x, item_parm, resp, ncat, model, prior, prior_parm)

calc_log_likelihood_function(
  theta_grid,
  item_parm,
  resp,
  ncat,
  model,
  prior,
  prior_parm
)
```

Arguments

`x, theta_grid`	the theta value. This must be a column vector in matrix form for `calc_*_function` functions.
`item_parm`	a matrix containing item parameters. Each row represents each item.
`resp`	a vector containing responses on each item.
`ncat`	a vector containing the number of response categories of each item.
`model`	a vector indicating item models of each item, using • `1`: 1PL model • `2`: 2PL model • `3`: 3PL model • `4`: PC model • `5`: GPC model • `6`: GR model
`prior`	an integer indicating the type of prior distribution, using • `1`: normal distribution • `2`: uniform distribution
`prior_parm`	a vector containing parameters for the prior distribution.

Details

`calc_log_likelihood` and `calc_log_likelihood_function` are functions to calculate log likelihoods.

These functions are designed for multiple items.

`calc_*` functions accept a single theta value, and `calc_*_function` functions accept multiple theta values.

Currently supports unidimensional models.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
# item parameters
item_parm <- matrix(c(
  1, NA,   NA,
  1,  2,   NA,
  1,  2, 0.25,
  0,  1,   NA,
  2,  0,    1,
  2,  0,    2),
  nrow = 6,
  byrow = TRUE
)

ncat  <- c(2, 2, 2, 3, 3, 3)
model <- c(1, 2, 3, 4, 5, 6)
resp  <- c(0, 1, 0, 1, 0, 1)

x <- 3
l  <- calc_likelihood(x, item_parm, resp, ncat, model)
ll <- calc_log_likelihood(x, item_parm, resp, ncat, model, 2, NA)
log(l) == ll

theta_grid <- matrix(seq(-3, 3, .1))
```

```
l  <- calc_likelihood_function(theta_grid, item_parm, resp, ncat, model)
ll <- calc_log_likelihood_function(theta_grid, item_parm, resp, ncat, model, 2, NA)
all(log(l) == ll)
```

`calc_MI_FB` *Calculate the mutual information using full Bayesian*

Description

Calculate the mutual information using full Bayesian.

Usage

```
calc_MI_FB(x, items_list, ncat, model)
```

Arguments

`x`	A numeric vector of MCMC sampled theta values.
`items_list`	A list of item parameter matrices.
`ncat`	a numeric vector specifying the number of response categories in each item.
`model`	a numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).

`calc_posterior` *Calculate a posterior value of theta*

Description

Calculate a posterior value of theta.

Usage

```
calc_posterior(x, item_parm, resp, ncat, model, prior, prior_parm)
```

Arguments

`x`	A length-one numeric vector for a theta value.
`item_parm`	A numeric matrix of item parameters.
`resp`	a numeric vector containing item responses.
`ncat`	A numeric vector of the number of response categories by item.
`model`	A numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
`prior`	The type of prior distribution (1: normal, 2: uniform).
`prior_parm`	A numeric vector of hyperparameters for the prior distribution, c(mu, sigma) or c(ll, ul).

calc_posterior_function *Calculate a posterior distribution of theta*

Description

Calculate a posterior distribution of theta.

Usage

```
calc_posterior_function(
  theta_grid,
  item_parm,
  resp,
  ncat,
  model,
  prior,
  prior_parm
)
```

Arguments

theta_grid	An equi-spaced grid of theta values.
item_parm	A numeric matrix of item parameters.
resp	a numeric vector containing item responses.
ncat	A numeric vector of the number of response categories by item.
model	A numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
prior	The type of prior distribution (1: normal, 2: uniform).
prior_parm	A numeric vector of hyperparameters for the prior distribution, c(mu, sigma) or c(ll, ul).

calc_posterior_single *Calculate a posterior value of theta for a single item*

Description

Calculate a posterior value of theta for a single item.

Usage

```
calc_posterior_single(x, item_parm, resp, ncat, model, prior, prior_parm)
```

Arguments

x	A length-one numeric vector for a theta value.
item_parm	A numeric vector of item parameters (for one item).
resp	A length-one numeric vector of item responses.
ncat	A length-one numeric vector of the number of response categories by item.
model	A length-one numeric vector of the IRT model by item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
prior	The type of prior distribution (1: normal, 2: uniform).
prior_parm	A numeric vector of hyperparameters for the prior distribution, c(mu, sigma) or c(ll, ul).

---

checkConstraints *Check the consistency of constraints and item usage*

Description

Check the consistency of constraints and item usage.

Usage

```
checkConstraints(constraints, usage_matrix, true_theta = NULL)
```

Arguments

constraints	A constraints object generated by loadConstraints.
usage_matrix	A matrix of item usage data from Shadow.
true_theta	A vector of true theta values.

---

config_Shadow-class *Create a config_Shadow object*

Description

createShadowTestConfig is a config function to create a config_Shadow object for Shadow test assembly. Default values are used for any unspecified parameters/slots.

Usage

```
createShadowTestConfig(
  item_selection = NULL,
  content_balancing = NULL,
  MIP = NULL,
  MCMC = NULL,
  refresh_policy = NULL,
  exposure_control = NULL,
  stopping_criterion = NULL,
  interim_theta = NULL,
  final_theta = NULL,
  theta_grid = seq(-4, 4, 0.1)
)
```

Arguments

`item_selection` a named list containing item selection criteria.

- `method` the type of selection criteria. Accepts `MFI`,`MPWI`,`FB`,`EB`,`GFI`. (default = `MFI`)
- `info_type` the type of information. Accepts `FISHER`. (default = `FISHER`)
- `initial_theta` (optional) initial theta values to use.
- `fixed_theta` (optional) fixed theta values to use throughout all item positions.
- `target_value` (optional) the target value to use for `method = 'GFI'`.

`content_balancing`
a named list containing content balancing options.

- `method` the type of balancing method. Accepts `NONE`,`STA`. (default = `STA`)

`MIP` a named list containing solver options.

- `solver` the type of solver. Accepts `lpsymphony`,`Rsymphony`,`gurobi`,`lpSolve`,`Rglpk`. (default = `LPSOLVE`)
- `verbosity` verbosity level of the solver. (default = `-2`)
- `time_limit` time limit in seconds. Used in solvers `lpsymphony`,`Rsymphony`,`gurobi`,`Rglpk`. (default = `60`)
- `gap_limit` search termination criterion. Gap limit in relative scale passed onto the solver. Used in solver `gurobi`. (default = `.05`)
- `gap_limit_abs` search termination criterion. Gap limit in absolute scale passed onto the solver. Used in solvers `lpsymphony`,`Rsymphony`. (default = `0.05`)
- `obj_tol` search termination criterion. The lower bound to use on the minimax deviation variable. Used when `item_selection$method` is `GFI`, and ignored otherwise. (default = `0.05`)
- `retry` number of times to retry running the solver if the solver returns no solution. Some solvers incorrectly return no solution even when a solution exists. This is the number of attempts to verify that the problem is indeed infeasible in such cases. Set to `0` to not retry. (default = `5`)

MCMC a named list containing Markov-chain Monte Carlo configurations for obtaining posterior samples.

- burn_in the number of chains from the start to discard. (default = 100)
- post_burn_in the number of chains to use after discarding the first burn_in chains. (default = 500)
- thin thinning interval to apply. 1 represents no thinning. (default = 1)
- jump_factor the jump factor to use. 1 represents no jumping. (default = 1)

refresh_policy a named list containing the refresh policy for when to obtain a new shadow test.

- method the type of policy. Accepts ALWAYS,POSITION,INTERVAL,THRESHOLD,INTERVAL-THRESHOL (default = ALWAYS)
- interval used in methods INTERVAL,INTERVAL-THRESHOLD. Set to 1 to refresh at each position, 2 to refresh at every two positions, and so on. (default = 1)
- threshold used in methods THRESHOLD,INTERVAL-THRESHOLD. The absolute change in between interim theta estimates to trigger the refresh. (default = 0.1)
- position used in methods POSITION. Item positions to trigger the refresh. (default = 1)

exposure_control

a named list containing exposure control settings.

- method the type of exposure control method. Accepts NONE,ELIGIBILITY,BIGM,BIGM-BAYESIAN. (default = ELIGIBILITY)
- M used in methods BIGM,BIGM-BAYESIAN. the Big M penalty to use on item information.
- max_exposure_rate target exposure rates for each segment. (default = rep(0.25,7))
- acceleration_factor the acceleration factor to apply. (default = 1)
- n_segment the number of theta segments to use. (default = 7)
- first_segment (optional) the theta segment assumed at the beginning of test for all participants.
- segment_cut theta segment cuts. (default = c(-Inf,seq(-2.5,2.5,1),Inf))
- initial_eligibility_stats (optional) initial eligibility statistics to use.
- fading_factor the fading factor to apply. (default = .999)
- diagnostic_stats set to TRUE to generate segment-wise diagnostic statistics. (default = FALSE)

stopping_criterion

a named list containing stopping criterion.

- method the type of stopping criterion. Accepts FIXED. (default = FIXED)
- test_length test length.
- min_ni the maximum number of items to administer.
- max_ni the minimum number of items to administer.
- se_threshold standard error threshold. Item administration is stopped when theta estimate standard error becomes lower than this value.

`interim_theta` a named list containing interim theta estimation options.

- `method` the type of estimation. Accepts `EAP,MLE,MLEF,EB,FB`. (default = `EAP`)
- `shrinkage_correction` set `TRUE` to apply shrinkage correction. Used when `method` is `EAP`. (default = `FALSE`)
- `prior_dist` the type of prior distribution. Accepts `NORMAL,UNIFORM`. (default = `NORMAL`)
- `prior_par` distribution parameters for `prior_dist`. (default = `c(0,1)`)
- `bound_ML` theta bound in `c(lower_bound,upper_bound)` format. Used when `method` is `MLE`. (default = `-4,4`)
- `truncate_ML` set `TRUE` to truncate ML estimate within `bound_ML`. (default = `FALSE`)
- `max_iter` maximum number of Newton-Raphson iterations. Used when `method` is `MLE`. (default = `50`)
- `crit` convergence criterion. Used when `method` is `MLE`. (default = `1e-03`)
- `max_change` maximum change in ML estimates between iterations. Changes exceeding this value is clipped to this value. Used when `method` is `MLE`. (default = `1.0`)
- `use_step_size` set `TRUE` to use `step_size`. Used when `method` is `MLE` or `MLEF`. (default = `FALSE`)
- `step_size` upper bound to impose on the absolute change in initial theta and estimated theta. Absolute changes exceeding this value will be capped to `step_size`. Used when `method` is `MLE` or `MLEF`. (default = `0.5`)
- `do_Fisher` set `TRUE` to use Fisher's method of scoring. Used when `method` is `MLE`. (default = `TRUE`)
- `fence_slope` slope parameter to use for `method = 'MLEF'`. This must have two values in total, for the lower and upper bound item respectively. Use one value to use the same value for both bounds. (default = `5`)
- `fence_difficulty` difficulty parameters to use for `method = 'MLEF'`. This must have two values in total, for the lower and upper bound item respectively. (default = `c(-5,5)`)

`final_theta` a named list containing final theta estimation options.

- `method` the type of estimation. Accepts `EAP,MLE,MLEF,EB,FB`. (default = `EAP`)
- `shrinkage_correction` set `TRUE` to apply shrinkage correction. Used when `method` is `EAP`. (default = `FALSE`)
- `prior_dist` the type of prior distribution. Accepts `NORMAL,UNIFORM`. (default = `NORMAL`)
- `prior_par` distribution parameters for `prior_dist`. (default = `c(0,1)`)
- `bound_ML` theta bound in `c(lower_bound,upper_bound)` format. Used when `method` is `MLE`. (default = `-4,4`)
- `truncate_ML` set `TRUE` to truncate ML estimate within `bound_ML`. (default = `FALSE`)
- `max_iter` maximum number of Newton-Raphson iterations. Used when `method` is `MLE`. (default = `50`)

- crit convergence criterion. Used when method is MLE. (default = 1e-03)
- max_change maximum change in ML estimates between iterations. Changes exceeding this value is clipped to this value. Used when method is MLE. (default = 1.0)
- use_step_size set TRUE to use step_size. Used when method is MLE or MLEF. (default = FALSE)
- step_size upper bound to impose on the absolute change in initial theta and estimated theta. Absolute changes exceeding this value will be capped to step_size. Used when method is MLE or MLEF. (default = 0.5)
- do_Fisher set TRUE to use Fisher's method of scoring. Used when method is MLE. (default = TRUE)
- fence_slope slope parameter to use for method = 'MLEF'. This must have two values in total, for the lower and upper bound item respectively. Use one value to use the same value for both bounds. (default = 5)
- fence_difficulty difficulty parameters to use for method = 'MLEF'. This must have two values in total, for the lower and upper bound item respectively. (default = c(-5,5))

theta_grid the theta grid to use as quadrature points.

Examples

```
cfg1 <- createShadowTestConfig(refresh_policy = list(
  method = "STIMULUS"
))
cfg2 <- createShadowTestConfig(refresh_policy = list(
  method = "POSITION",
  position = c(1, 5, 9)
))
```

---

config_Static-class *Create a config_Static object*

Description

createStaticTestConfig is a config function to create a config_Static object for Static (fixed-form) test assembly. Default values are used for any unspecified parameters/slots.

Usage

```
createStaticTestConfig(item_selection = NULL, MIP = NULL)
```

Arguments

item_selection a named list containing item selection criteria.

- method the type of selection criteria. Accepts MAXINFO,TIF,TCC. (default = MAXINFO)

- info_type the type of information. Accepts FISHER. (default = FISHER)
- target_location a numeric vector containing the locations of target theta points. (e.g. c(-1,0,1)) (default = c(-1.2,0,1.2))
- target_value a numeric vector containing the target values at each theta location. This should have the same length with target_location. Ignored if method is MAXINFO. (default = NULL)
- target_weight a numeric vector containing the weights for each theta location. This should have the same length with target_location. (default = rep(1,length(target_location))

MIP a named list containing solver options.

- solver the type of solver. Accepts lpsymphony,Rsymphony,gurobi,lpSolve,Rglpk. (default = LPSOLVE)
- verbosity verbosity level of the solver. (default = -2)
- time_limit time limit in seconds. Used in solvers lpsymphony,Rsymphony,gurobi,Rglpk. (default = 60)
- gap_limit search termination criterion. Gap limit in relative scale passed onto the solver. Used in solver gurobi. (default = .05)
- gap_limit_abs search termination criterion. Gap limit in absolute scale passed onto the solver. Used in solvers lpsymphony,Rsymphony. (default = 0.05)
- obj_tol search termination criterion. The lower bound to use on the minimax deviation variable. Used when item_selection$method is TIF or TCC. (default = 0.05)
- retry number of times to retry running the solver if the solver returns no solution. Some solvers incorrectly return no solution even when a solution exists. This is the number of attempts to verify that the problem is indeed infeasible in such cases. Set to 0 to not retry. (default = 5)

Value

createStaticTestConfig returns a config_Static object. This object is used in Static.

Examples

```
cfg1 <- createStaticTestConfig(
  list(
    method = "MAXINFO",
    info_type = "FISHER",
    target_location = c(-1, 0, 1),
    target_weight = c(1, 1, 1)
  )
)

cfg2 <- createStaticTestConfig(
  list(
    method = "TIF",
    info_type = "FISHER",
    target_location = c(-1, 0, 1),
```

```
    target_weight = c(1, 1, 1),
    target_value = c(8, 10, 12)
  )
)

cfg3 <- createStaticTestConfig(
  list(
    method = "TCC",
    info_type = "FISHER",
    target_location = c(-1, 0, 1),
    target_weight = c(1, 1, 1),
    target_value = c(10, 15, 20)
  )
)
```

`constraint-class` *Class 'constraint': a single constraint*

Description

`constraint` is an S4 class to represent a single constraint.

Slots

`constraint` the numeric index of the constraint.

`constraint_id` the character ID of the constraint.

`nc` the number of MIP-format constraints translated from this constraint.

`mat,dir,rhs` these represent MIP-format constraints. A single MIP-format constraint is associated with a row in `mat`, a value in `rhs`, and a value in `dir`.

- the *i*-th row of `mat` represents LHS coefficients to use on decision variables in the *i*-th MIP-format constraint.
- the *i*-th value of `rhs` represents RHS values to use in the *i*-th MIP-format constraint.
- the *i*-th value of `dir` represents the imposed constraint between LHS and RHS.

`suspend` `TRUE` if the constraint is not to be imposed.

`constraints-class` *Class 'constraints': a set of constraints*

Description

`constraints` is an S4 class to represent a set of constraints and its associated objects.

Details

See `constraints-operators` for object manipulation functions.

Slots

`constraints` a `data.frame` containing the constraint specifications.

`list_constraints` a list containing the `constraint` object representation of each constraint.

`pool` the `item_pool` object associated with the constraints.

`item_attrib` the `item_attrib` object associated with the constraints.

`st_attrib` the `st_attrib` object associated with the constraints.

`test_length` the test length specified in the constraints.

`nv` the number of decision variables. Equals `ni + ns`.

`ni` the number of items to search from.

`ns` the number of stimulus to search from.

`id` the item/stimulus ID string of each item/stimulus.

`index,mat,dir,rhs` these represent MIP-format constraints. A single MIP-format constraint is associated with a value in `index`, a row in `mat`, a value in `rhs`, and a value in `dir`.

- the *i*-th value of `index` represents which constraint specification in the `constraints` argument it was translated from.
- the *i*-th row of `mat` represents LHS coefficients to use on decision variables in the *i*-th MIP-format constraint.
- the *i*-th value of `rhs` represents RHS values to use in the *i*-th MIP-format constraint.
- the *i*-th value of `dir` represents the imposed constraint between LHS and RHS.

`set_based` `TRUE` if the constraint is set-based. `FALSE` otherwise.

`item_order` the item attribute of each item to use in imposing an item order constraint, if any.

`item_order_by` the name of the item attribute to use in imposing an item order constraint, if any.

`stim_order` the stimulus attribute of each stimulus to use in imposing a stimulus order constraint, if any.

`stim_order_by` the name of the stimulus attribute to use in imposing a stimulus order constraint, if any.

`item_index_by_stimulus` a list containing item indices of each stimulus.

`stimulus_index_by_item` the stimulus indices of each item.

`constraints-operators` *Basic operators for constraints objects*

Description

Create a subset of a `constraints` object:

- `constraints[i]`
- `subsetConstraints(constraints,1:10)`

Combine two `constraints` objects:

- `c(constraints1,constraints2)`
- `combineConstraints(constraints1,constraints2)`

Usage

```
subsetConstraints(x, i = NULL)

combineConstraints(x1, x2)

## S4 method for signature 'constraints,numeric'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'constraints'
c(x, ...)
```

Arguments

x, x1, x2	a constraints object.
i, j	indices to use in subsetting.
...	not used, exists for compatibility.
drop	not used, exists for compatibility.

Examples

```
c1 <- constraints_science
c2 <- c1[1:10]
c3 <- c1[c(1, 11:36)] # keep constraint 1 for test length
c4 <- c(c2, c3)
```

---

dataset_bayes *Bayes dataset*

Description

Item-based example item pool with standard errors (320 items).

Details

This pool is associated with the following objects:

- itempool_bayes an item_pool object containing 320 items.
- itemattrib_bayes a item_attrib object containing 5 item-level attributes.
- constraints_bayes a constraints object containing 14 constraints.

Also, the following objects are intended for illustrating expected data structures.

- itempool_bayes_data a data.frame containing item parameters.
- itempool_se_bayes_data a data.frame containing item parameter standard errors.
- itemattrib_bayes_data a data.frame containing item attributes.
- constraints_bayes_data a data.frame containing constraint specifications.

Examples

```
itempool_bayes    <- loadItemPool(itempool_bayes_data, itempool_se_bayes_data)
itemattrib_bayes  <- loadItemAttrib(itemattrib_bayes_data, itempool_bayes)
constraints_bayes <- loadConstraints(constraints_bayes_data,
  itempool_bayes, itemattrib_bayes)
```

---

dataset_fatigue *Fatigue dataset*

Description

Item-based example pool with item contents (95 items).

Details

This pool is associated with the following objects:

- `itempool_fatigue` an `item_pool` object containing 95 items.
- `itemattrib_fatigue` an `item_attrib` object containing 7 item-level attributes.
- `constraints_fatigue` a `constraints` object containing 111 constraints.

Also, the following objects are intended for illustrating expected data structures.

- `itempool_fatigue_data` a `data.frame` containing item parameters.
- `itemattrib_fatigue_data` a `data.frame` containing item attributes.
- `itemcontent_fatigue_data` a `data.frame` containing item contents.
- `constraints_fatigue_data` a `data.frame` containing constraint specifications.
- `resp_fatigue_data` a `data.frame` containing raw response data.

Examples

```
itempool_fatigue   <- loadItemPool(itempool_fatigue_data)
itemattrib_fatigue <- loadItemAttrib(itemattrib_fatigue_data, itempool_fatigue)
constraints_fatigue <- loadConstraints(constraints_fatigue_data,
  itempool_fatigue, itemattrib_fatigue)
```

dataset_reading *Reading dataset*

Description

Stimulus-based example item pool (303 items, 35 stimuli).

Details

This pool is associated with the following objects:

- `itempool_reading` an `item_pool` object containing 303 items.
- `itemattrib_reading` an `item_attrib` object containing 12 item-level attributes.
- `stimattrib_reading` a `st_attrib` object containing 4 stimulus-level attributes.
- `constraints_reading` a `constraints` object containing 18 constraints.

Also, the following objects are intended for illustrating expected data structures.

- `itempool_reading_data` a `data.frame` containing item parameters.
- `itemattrib_reading_data` a `data.frame` containing item attributes.
- `stimattrib_reading_data` a `data.frame` containing stimulus attributes.
- `constraints_reading_data` a `data.frame` containing constraint specifications.

Examples

```
itempool_reading    <- loadItemPool(itempool_reading_data)
itemattrib_reading  <- loadItemAttrib(itemattrib_reading_data, itempool_reading)
stimattrib_reading  <- loadStAttrib(stimattrib_reading_data, itemattrib_reading)
constraints_reading <- loadConstraints(constraints_reading_data,
  itempool_reading, itemattrib_reading, stimattrib_reading)
```

dataset_science *Science dataset*

Description

Item-based example item pool (1000 items).

Details

This pool is associated with the following objects:

- `itempool_science` an `item_pool` object containing 1000 items.
- `itemattrib_science` an `item_attrib` object containing 9 item-level attributes.
- `constraints_science` a `constraints` object containing 36 constraints.

Also, the following objects are intended for illustrating expected data structures.

- `itempool_science_data` a `data.frame` containing item parameters.
- `itemattrib_science_data` a `data.frame` containing item attributes.
- `constraints_science_data` a `data.frame` containing constraint specifications.

Examples

```
itempool_science    <- loadItemPool(itempool_science_data)
itemattrib_science  <- loadItemAttrib(itemattrib_science_data, itempool_science)
constraints_science <- loadConstraints(constraints_science_data,
  itempool_science, itemattrib_science)
```

---

eap *Compute expected a posteriori estimates of theta*

Description

`eap` is a function to compute expected a posteriori estimates of theta.

Usage

```
eap(
  object,
  select = NULL,
  resp,
  theta_grid = seq(-4, 4, 0.1),
  prior = rep(1/81, 81)
)

## S4 method for signature 'item_pool'
eap(
  object,
  select = NULL,
  resp,
  theta_grid = seq(-4, 4, 0.1),
  prior = rep(1/81, 81)
)
```

```
EAP(object, select = NULL, prior, reset_prior = FALSE)

## S4 method for signature 'test'
EAP(object, select = NULL, prior, reset_prior = FALSE)

## S4 method for signature 'test_cluster'
EAP(object, select = NULL, prior, reset_prior = FALSE)
```

Arguments

object	an item_pool object.
select	(optional) if item indices are supplied, only the specified items are used.
resp	item response on all (or selected) items in the object argument. Can be a vector, a matrix, or a data frame. length(resp) or ncol(resp) must be equal to the number of all (or selected) items.
theta_grid	the theta grid to use as quadrature points. (default = seq(-4,4,.1))
prior	a prior distribution, a numeric vector for a common prior or a matrix for individualized priors. (default = rep(1 / 81,81))
reset_prior	used for test_cluster objects. If TRUE, to reset the prior distribution before each test.

Value

eap returns a list containing estimated values.

- th theta value.
- se standard error.

Examples

```
eap(itempool_fatigue, resp = resp_fatigue_data[10, ])
eap(itempool_fatigue, select = 1:20, resp = resp_fatigue_data[10, 1:20])
```

---

find_segment *Classify theta into segments*

Description

find_segment is a function to classify theta values into segments based on supplied cutpoints.

Usage

```
find_segment(x, segment)
```

Arguments

x	the theta value. This can be a vector.
segment	segment cutpoints.

Examples

```
cuts <- c(-Inf, -2, 0, 2, Inf)

find_segment(-3, cuts)
find_segment(-1, cuts)
find_segment(1, cuts)
find_segment(3, cuts)
find_segment(seq(-3, 3, 2), cuts)
```

---

getSolution *Print solution items*

Description

Print solution items

Usage

```
getSolution(object, examinee = NA, position = NA, index_only = TRUE)

## S4 method for signature 'list'
getSolution(object, examinee = NA, position = NA, index_only = TRUE)

## S4 method for signature 'output_Static'
getSolution(object, examinee = NA, position = NA, index_only = TRUE)
```

Arguments

object	an output_Static object or an output_Shadow object.
examinee	(optional) the examinee index to display the solution. Used when the 'object' argument is an output_Shadow object.
position	(optional) if supplied, display the item attributes of the assembled test at that item position. If not supplied, display the item attributes of the administered items. Used when the 'object' argument is an output_Shadow object.
index_only	if TRUE, only print item indices. if FALSE, print all item attributes. (default = TRUE)

Value

Item attributes of solution items.

`getSolutionAttributes` *Retrieve constraints-related attributes from solution*

Description

`getSolutionAttributes` is a helper function to retrieve constraints-related attributes from a solution.

Usage

```
getSolutionAttributes(constraints, item_idx, all_values = FALSE)
```

Arguments

`constraints`	a `constraints` object.
`item_idx`	item indices from a solution.
`all_values`	if `TRUE`, return all values as-is without taking the mean when there are multiple values. If `FALSE`, return the mean when there are multiple values. This has an effect when there is a constraint on items per stimulus, where there are multiple values of number of items per stimulus. In this case, if `TRUE`, the number of items for every stimuli are returned as-is. If `FALSE`, the average number of items across stimuli is returned. (default = `FALSE`)

Value

- If `all_values == FALSE`, `getSolutionAttributes` returns a `data.frame` containing constraints data and their associated attributes.
- If `all_values == TRUE`, `getSolutionAttributes` returns a `list` containing attributes associated to each constraint.

Examples

```
item_idx <-
  c( 29,  33,  26,  36,  34,
    295, 289, 296, 291, 126,
    133, 124, 134, 129,  38,
     47,  39,  41,  46,  45,
    167, 166, 170, 168, 113,
    116, 119, 117, 118, 114)

getSolutionAttributes(constraints_reading, item_idx, FALSE)
getSolutionAttributes(constraints_reading, item_idx, TRUE)
```

info_1pl *Calculate Fisher information (single item)*

Description

info_* and array_info_* are functions to calculate Fisher information.

Usage

```
info_1pl(x, b)

info_2pl(x, a, b)

info_3pl(x, a, b, c)

info_pc(x, b)

info_gpc(x, a, b)

info_gr(x, a, b)

array_info_1pl(x, b)

array_info_2pl(x, a, b)

array_info_3pl(x, a, b, c)

array_info_pc(x, b)

array_info_gpc(x, a, b)

array_info_gr(x, a, b)
```

Arguments

x	the theta value. This must be a column vector in matrix form for array_info_* functions.
b	the *b*-parameter.
a	the *a*-parameter.
c	the *c*-parameter.

Details

info_* functions accept a single theta value, and array_info_* functions accept multiple theta values.

Currently supports unidimensional models.

- `info_1pl`, `array_info_1pl`: 1PL models
- `info_2pl`, `array_info_2pl`: 2PL models
- `info_3pl`, `array_info_3pl`: 3PL models
- `info_pc`, `array_info_pc`: PC (partial credit) models
- `info_gpc`, `array_info_gpc`: GPC (generalized partial credit) models
- `info_gr`, `array_info_gr`: GR (graded response) models

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
x <- 0.5

info_1pl(x, 1)
info_2pl(x, 1, 2)
info_3pl(x, 1, 2, 0.25)
info_pc(x, c(0, 1))
info_gpc(x, 2, c(0, 1))
info_gr(x, 2, c(0, 2))

x <- matrix(seq(0.1, 0.5, 0.1)) # column vector in matrix form

array_info_1pl(x, 1)
array_info_2pl(x, 1, 2)
```

```
array_info_3pl(x, 1, 2, 0.25)
array_info_pc(x, c(0, 1))
array_info_gpc(x, 2, c(0, 1))
array_info_gr(x, 2, c(0, 2))
```

---

iparPosteriorSample *Sample item parameter estimates from their posterior distributions*

Description

Sample item parameter estimates from their posterior distributions.

Usage

```
iparPosteriorSample(pool, n_sample = 500)
```

Arguments

pool	An item_pool object.
n_sample	An integer as the number of sampled parameters.

Examples

```
ipar <- iparPosteriorSample(itempool_science, 5)
```

---

item-classes *Item classes*

Description

- item_1PL class represents a 1PL item.
- item_2PL class represents a 2PL item.
- item_3PL class represents a 3PL item.
- item_PC class represents a partial credit item.
- item_GPC class represents a generalized partial credit item.
- item_GR class represents a graded response item.

Slots

slope a slope parameter value

difficulty a difficulty parameter value

guessing a guessing parameter value

threshold a vector of threshold parameter values

category a vector of category boundary values

ncat the number of response categories

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
item_1 <- new("item_1PL", difficulty = 0.5)
item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4 <- new("item_PC", threshold = c(-0.5, 0.5), ncat = 3)
item_5 <- new("item_GPC", slope = 1.0, threshold = c(-0.5, 0.0, 0.5), ncat = 4)
item_6 <- new("item_GR", slope = 1.0, category = c(-2.0, -1.0, 0, 1.0, 2.0), ncat = 6)
```

---

item_attrib-class *Load item attributes*

Description

`loadItemAttrib` is a data loading function to create an `item_attrib` object. `loadItemAttrib` can read item attributes a `data.frame` or a .csv file.

Usage

```
loadItemAttrib(object, pool, file = NULL)
```

Arguments

object	item attributes. Can be a data.frame or the file path of a .csv file. The content should at least include column 'ID' that matches with the item_pool object.
pool	an item_pool object. Use loadItemPool for this.
file	(deprecated) use object argument instead.

Value

loadItemAttrib returns an item_attrib object.

- data a data.frame containing item attributes.

See Also

dataset_science, dataset_reading, dataset_fatigue, dataset_bayes for examples.

Examples

```
## Read from data.frame:
itempool_science   <- loadItemPool(itempool_science_data)
itemattrib_science <- loadItemAttrib(itemattrib_science_data, itempool_science)

## Read from file: write to tempdir() for illustration and clean afterwards
f <- file.path(tempdir(), "itemattrib_science.csv")
write.csv(itemattrib_science_data, f, row.names = FALSE)
itemattrib_science <- loadItemAttrib(f, itempool_science)
file.remove(f)

## TestDesign 1.1.0 - Deprecated arguments
## Not run:
loadItemAttrib(object = "iatt.csv", pool) # is equivalent to
loadItemAttrib(file   = "iatt.csv", pool) # pre 1.1.0

## End(Not run)
```

---

item_attrib-operators *Basic functions for item attribute objects*

Description

Basic functions for item attribute objects

Usage

```
## S4 method for signature 'item_attrib,numeric'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'item_attrib'
dim(x)

## S4 method for signature 'item_attrib'
colnames(x)

## S4 method for signature 'item_attrib'
rownames(x)

## S4 method for signature 'item_attrib'
names(x)

## S4 method for signature 'item_attrib'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	an item_attrib object.
i, j	indices to use in subsetting.
...	not used, exists for compatibility.
drop	not used, exists for compatibility.
row.names	not used, exists for compatibility.
optional	not used, exists for compatibility.

Examples

```
x <- itemattrib_science
x[1:10]
dim(x)
ncol(x)
nrow(x)
colnames(x)
rownames(x)
names(x)
as.data.frame(x)
```

---

item_pool-class *Class 'item_pool': an item pool*

Description

item_pool is an S4 class to represent an item pool.

Details

See item_pool-operators for object manipulation functions.

Slots

ni the number of items in the pool.

max_cat the maximum number of response categories across the pool.

index the numeric index of each item.

id the ID string of each item.

model the item class name of each item. See item-classes.

NCAT the number of response categories of each item.

parms a list containing item class objects. See item-classes.

ipar a matrix containing item parameters.

se a matrix containing item parameter standard errors.

raw the raw input data.frame used in loadItemPool to create this object.

raw_se the raw input data.frame used in loadItemPool to create this object.

unique whether item IDs must be unique for this object to be a valid object.

---

item_pool-operators *Basic operators for item pool objects*

Description

Create a subset of an item_pool object:

- pool[i]
- subsetItemPool(pool,i)

Combine two item_pool objects:

- c(pool1,pool2)
- combineItemPool(pool1,pool2)
- pool1 + pool2

pool1 -pool2 excludes items in pool2 from pool1.

pool1 == pool2 tests whether two item_pool objects are identical.

Usage

```
subsetItemPool(x, i = NULL)

combineItemPool(x1, x2, unique = TRUE, verbose = TRUE)

## S4 method for signature 'item_pool,numeric'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'item_pool'
c(x, ...)

## S3 method for class 'item_pool'
x1 + x2

## S3 method for class 'item_pool'
x1 - x2

## S3 method for class 'item_pool'
x1 == x2
```

Arguments

x, x1, x2	an item_pool object.
i	item indices to use in subsetting.
unique	if TRUE, remove items with duplicate IDs after combining. (default = TRUE)
verbose	if TRUE, raise a warning if duplicate IDs are found after combining. (default = TRUE)
j, drop, ...	not used, exists for compatibility.

Examples

```
p1 <- itempool_science[1:100]
p2 <- c(itempool_science, itempool_reading)
p3 <- p2 - p1

p1 <- itempool_science[1:500]
p2 <- itempool_science - p1
p3 <- itempool_science[501:1000]
identical(p2, p3)  ## TRUE

p <- p1 + p3
p == itempool_science ## TRUE
```

item_pool_cluster-class *Class 'item_pool_cluster': an item pool*

Description

`item_pool_cluster` is an S4 class to represent a group of item pools.

Slots

`np` the number of item pools.

`pools` a list of `item_pool` objects.

`names` a vector containing item pool names.

lnHyperPars *Calculate hyperparameters for log-normal distribution*

Description

Calculate hyperparameters for log-normal distribution.

Usage

```
lnHyperPars(mean, sd)
```

Arguments

`mean`	Mean of the distribution.
`sd`	Standard deviation of the distribution.

Examples

```
lnHyperPars(.5, 1)
```

loadConstraints *Load constraints*

Description

loadConstraints is a data loading function to create a constraints object. loadConstraints can read constraints from a data.frame or a .csv file. The contents must be in the expected format; see the vignette in vignette("constraints").

Usage

```
loadConstraints(object, pool, item_attrib, st_attrib = NULL, file = NULL)
```

Arguments

object	constraint specifications. Can be a data.frame or the file path of a .csv file. See the vignette for the expected format.
pool	an item_pool object. Use loadItemPool for this.
item_attrib	an item_attrib object. Use loadItemAttrib for this.
st_attrib	(optional) an st_attrib object. Use loadStAttrib for this.
file	(deprecated) use object argument instead.

Value

loadConstraints returns a constraints object. This object is used in Static and Shadow.

See Also

dataset_science, dataset_reading, dataset_fatigue, dataset_bayes for examples.

Examples

```
## Read from data.frame:
itempool_science    <- loadItemPool(itempool_science_data)
itemattrib_science  <- loadItemAttrib(itemattrib_science_data, itempool_science)
constraints_science <- loadConstraints(constraints_science_data,
  itempool_science, itemattrib_science)

## Read from file: write to tempdir() for illustration and clean afterwards
f <- file.path(tempdir(), "constraints_science.csv")
write.csv(constraints_science_data, f, row.names = FALSE)
constraints_science <- loadConstraints(f,
  itempool_science, itemattrib_science)
file.remove(f)

## TestDesign 1.1.0 - Deprecated arguments
## Not run:
loadConstraints(object = "consts.csv", pool, item_attrib) # is equivalent to
```

```
loadConstraints(file   = "consts.csv", pool, item_attrib) # pre 1.1.0

## End(Not run)
```

`loadItemPool` *Load item pool*

Description

`loadItemPool` is a data loading function to create an `item_pool` object. `loadItemPool` can read item parameters and standard errors from a `data.frame` or a .csv file.

Usage

```
loadItemPool(ipar, ipar_se = NULL, file = NULL, se_file = NULL, unique = FALSE)
```

Arguments

`ipar`	item parameters. Can be a `data.frame` or the file path of a .csv file. The content should at least include columns 'ID' and 'MODEL'.
`ipar_se`	(optional) standard errors. Can be a `data.frame` or the file path of a .csv file.
`file`	(deprecated) use `ipar` argument instead.
`se_file`	(deprecated) use `ipar_se` argument instead.
`unique`	if `TRUE`, item IDs must be unique to create a valid `item_pool` object. (default = `FALSE`)

Value

`loadItemPool` returns an `item_pool` object.

- `ni` the number of items in the pool.
- `max_cat` the maximum number of response categories across all items in the pool.
- `index` the numeric item index of each item.
- `id` the item ID string of each item.
- `model` the object class names of each item representing an item model type. Can be `item_1PL`, `item_2PL`, `item_3PL`, `item_PC`, `item_GPC`, or `item_GR`.
- `NCAT` the number of response categories of each item.
- `parms` a list containing the item object of each item.
- `ipar` a matrix containing all item parameters.
- `se` a matrix containing all item parameter standard errors. The values will be 0 if the argument `ipar_se` was not supplied.
- `raw` the original input `data.frame` used to create this object.

See Also

dataset_science, dataset_reading, dataset_fatigue, dataset_bayes for examples.

Examples

```
## Read from data.frame:
itempool_science <- loadItemPool(itempool_science_data)

## Read from file: write to tempdir() for illustration and clean afterwards
f <- file.path(tempdir(), "itempool_science.csv")
write.csv(itempool_science_data, f, row.names = FALSE)
itempool_science <- loadItemPool(f)
file.remove(f)

## TestDesign 1.1.0 - Deprecated arguments
## Not run:
loadItemPool(ipar = "ipar.csv", ipar_se = "se.csv") # is equivalent to
loadItemPool(file = "ipar.csv", se_file = "se.csv") # pre 1.1.0

## End(Not run)
```

---

logitHyperPars *Calculate hyperparameters for logit-normal distribution*

Description

Calculate hyperparameters for logit-normal distribution.

Usage

```
logitHyperPars(mean, sd)
```

Arguments

mean	Mean of the distribution.
sd	Standard deviation of the distribution.

Examples

```
logitHyperPars(.5, 1)
```

makeItemPoolCluster *Create an item pool cluster object*

Description

Create a item_pool_cluster object.

item_pool_cluster1 == item_pool_cluster2 tests equality of two item_pool_cluster objects.

Usage

```
makeItemPoolCluster(x, ..., names = NULL)

## S4 method for signature 'item_pool'
makeItemPoolCluster(x, ..., names = NULL)

## S3 method for class 'item_pool_cluster'
item_pool_cluster1 == item_pool_cluster2
```

Arguments

x, ...	item_pool objects.
names	(optional) names to use for item_pool.
item_pool_cluster1	an item_pool_cluster object.
item_pool_cluster2	an item_pool_cluster object.

Examples

```
cluster <- makeItemPoolCluster(itempool_science, itempool_reading)
cluster1 <- makeItemPoolCluster(itempool_science, itempool_reading)
cluster2 <- makeItemPoolCluster(cluster1@pools[[1]], cluster1@pools[[2]])
cluster1 == cluster2  ## TRUE
```

makeTest *Generate a test object*

Description

makeTest is a function for creating a test object. This is used in Shadow to determine all necessary data prior to the main simulation, so that they are not affected by random number generation.

Usage

```
makeTest(
  object,
  theta = seq(-4, 4, 0.1),
  info_type = "FISHER",
  true_theta = NULL
)

## S4 method for signature 'item_pool'
makeTest(
  object,
  theta = seq(-4, 4, 0.1),
  info_type = "FISHER",
  true_theta = NULL
)
```

Arguments

object	an item_pool object.
theta	a grid of theta values.
info_type	the type of information.
true_theta	(optional) true theta values to simulate response data.

Examples

```
test <- makeTest(itempool_science, seq(-3, 3, 1))
```

---

makeTestCluster *Generate a test cluster object*

Description

Generate a test_cluster object

Usage

```
makeTestCluster(object, theta, true_theta)

## S4 method for signature 'item_pool_cluster,numeric,numeric'
makeTestCluster(object, theta, true_theta)

## S4 method for signature 'item_pool_cluster,numeric,list'
makeTestCluster(object, theta, true_theta)
```

Arguments

object	An item_pool_cluster object
theta	A grid of theta values
true_theta	An optional vector of true theta values to simulate response data

---

mle *Compute maximum likelihood estimates of theta*

Description

mle is a function to compute maximum likelihood estimates of theta.

Usage

```
mle(
  object,
  select = NULL,
  resp,
  start_theta = NULL,
  max_iter = 100,
  crit = 0.001,
  truncate = FALSE,
  theta_range = c(-4, 4),
  max_change = 1,
  use_step_size = FALSE,
  step_size = 0.5,
  do_Fisher = TRUE
)

## S4 method for signature 'item_pool'
mle(
  object,
  select = NULL,
  resp,
  start_theta = NULL,
  max_iter = 50,
  crit = 0.005,
  truncate = FALSE,
  theta_range = c(-4, 4),
  max_change = 1,
  use_step_size = FALSE,
  step_size = 0.5,
  do_Fisher = TRUE
)

MLE(
```

```
  object,
  select = NULL,
  start_theta = NULL,
  max_iter = 100,
  crit = 0.001,
  theta_range = c(-4, 4),
  truncate = FALSE,
  max_change = 1,
  do_Fisher = TRUE
)

## S4 method for signature 'test'
MLE(
  object,
  select = NULL,
  start_theta = NULL,
  max_iter = 100,
  crit = 0.001,
  theta_range = c(-4, 4),
  truncate = FALSE,
  max_change = 1,
  do_Fisher = TRUE
)

## S4 method for signature 'test_cluster'
MLE(object, select = NULL, start_theta = NULL, max_iter = 100, crit = 0.001)
```

Arguments

object	an item_pool object.
select	(optional) if item indices are supplied, only the specified items are used.
resp	item response on all (or selected) items in the object argument. Can be a vector, a matrix, or a data frame. length(resp) or ncol(resp) must be equal to the number of all (or selected) items.
start_theta	(optional) initial theta values. If not supplied, EAP estimates using uniform priors are used as initial values. Uniform priors are computed using the theta_range argument below, with increments of .1.
max_iter	maximum number of iterations. (default = 100)
crit	convergence criterion to use. (default = 0.001)
truncate	set TRUE to impose a bound using theta_range on the estimate. (default = FALSE)
theta_range	a range of theta values to bound the estimate. Only effective when truncate is TRUE. (default = c(-4,4))
max_change	upper bound to impose on the absolute change in theta between iterations. Absolute changes exceeding this value will be capped to max_change. (default = 1.0)

use_step_size	set TRUE to use step_size. (default = FALSE)
step_size	upper bound to impose on the absolute change in initial theta and estimated theta. Absolute changes exceeding this value will be capped to step_size. (default = 0.5)
do_Fisher	set TRUE to use Fisher scoring instead of Newton-Raphson method. (default = TRUE)

Value

mle returns a list containing estimated values.

- th theta value.
- se standard error.
- conv TRUE if estimation converged.
- trunc TRUE if truncation was applied on th.

Examples

```
mle(itempool_fatigue, resp = resp_fatigue_data[10, ])
mle(itempool_fatigue, select = 1:20, resp = resp_fatigue_data[10, 1:20])
```

---

mlef *Compute maximum likelihood estimates of theta using fence items*

Description

mlef is a function to compute maximum likelihood estimates of theta using fence items.

Usage

```
mlef(
  object,
  select = NULL,
  resp,
  fence_slope = 5,
  fence_difficulty = c(-5, 5),
  start_theta = NULL,
  max_iter = 100,
  crit = 0.001,
  truncate = FALSE,
  theta_range = c(-4, 4),
  max_change = 1,
  use_step_size = FALSE,
  step_size = 0.5,
  do_Fisher = TRUE
)
```

```
## S4 method for signature 'item_pool'
mlef(
  object,
  select = NULL,
  resp,
  fence_slope = 5,
  fence_difficulty = c(-5, 5),
  start_theta = NULL,
  max_iter = 50,
  crit = 0.005,
  truncate = FALSE,
  theta_range = c(-4, 4),
  max_change = 1,
  use_step_size = FALSE,
  step_size = 0.5,
  do_Fisher = TRUE
)
```

Arguments

object	an item_pool object.
select	(optional) if item indices are supplied, only the specified items are used.
resp	item response on all (or selected) items in the object argument. Can be a vector, a matrix, or a data frame. length(resp) or ncol(resp) must be equal to the number of all (or selected) items.
fence_slope	the slope parameter to use on fence items. Can be one value, or two values for the lower and the upper fence respectively. (default = 5)
fence_difficulty	the difficulty parameter to use on fence items. Must have two values for the lower and the upper fence respectively. (default = c(-5,5))
start_theta	(optional) initial theta values. If not supplied, EAP estimates using uniform priors are used as initial values. Uniform priors are computed using the theta_range argument below, with increments of .1.
max_iter	maximum number of iterations. (default = 100)
crit	convergence criterion to use. (default = 0.001)
truncate	set TRUE to impose a bound using theta_range on the estimate. (default = FALSE)
theta_range	a range of theta values to bound the estimate. Only effective when truncate is TRUE. (default = c(-4,4))
max_change	upper bound to impose on the absolute change in theta between iterations. Absolute changes exceeding this value will be capped to max_change. (default = 1.0)
use_step_size	set TRUE to use step_size. (default = FALSE)

step_size	upper bound to impose on the absolute change in initial theta and estimated theta. Absolute changes exceeding this value will be capped to step_size. (default = 0.5)
do_Fisher	set TRUE to use Fisher scoring instead of Newton-Raphson method. (default = TRUE)

Value

mlef returns a list containing estimated values.

- th theta value.
- se standard error.
- conv TRUE if estimation converged.
- trunc TRUE if truncation was applied on th.

References

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.

Examples

```
mlef(itempool_fatigue, resp = resp_fatigue_data[10, ])
mlef(itempool_fatigue, select = 1:20, resp = resp_fatigue_data[10, 1:20])
```

---

output_Shadow-class *Class 'output_Shadow': adaptive assembly solution for one simulee*

Description

output_Shadow is an S4 class to represent the adaptive assembly solution for one simulee.

Slots

simulee_id the numeric ID of the simulee.

true_theta the true theta of the simulee, if was specified.

true_theta_segment the segment number of the true theta.

final_theta_est final theta estimate.

final_se_est the standard error of final_theta_est.

administered_item_index item IDs administered at each position.

administered_item_resp item responses from the simulee at each position.

administered_item_ncat the number of categories of each administered item.

administered_stimulus_index stimulus IDs administered at each position.

shadow_test_refreshed TRUE indicates the shadow test was refreshed for the position.

`shadow_test_feasible` `TRUE` indicates the MIP was feasible with all constraints.

`solve_time` elapsed time in running the solver at each position.

`initial_theta_est` initial theta estimate.

`interim_theta_est` interim theta estimates at each position.

`interim_se_est` the standard error of the interim estimate at each position.

`theta_segment_index` segment numbers of interim theta estimates.

`prior` prior distribution, if was specified.

`prior_par` prior parameters, if were specified.

`posterior` the posterior distribution after completing test.

`posterior_sample` posterior samples of interim theta before the estimation of final theta. `mean(posterior_sample) == interim_theta_est[test_length]` holds.

`likelihood` the likelihood distribution after completing test.

`shadow_test` the list containing the item IDs within the shadow test used in each position.

`max_cat_pool` the maximum number of response categories the item pool had.

`ni_pool` the total number of items the item pool had.

`ns_pool` the total number of stimuli the item pool had.

`test_length_constraints` the test length constraint used in assembly.

`set_based` whether the item pool was set-based.

`item_index_by_stimulus` the list of items by each stimulus the item pool had.

`output_Shadow_all-class` *Class 'output_Shadow_all': a set of adaptive assembly solutions*

Description

`output_Shadow_all` is an S4 class to represent a set of adaptive assembly solutions.

Details

notations

- *ni* denotes the number of items in the `item_pool` object.
- *ns* denotes the number of stimuli.
- *nj* denotes the number of participants.

Slots

`output` a length-*nj* list of `output_Shadow` objects, containing the assembly results for each participant.

`final_theta_est` a length-*nj* vector containing final theta estimates for each participant.

`final_se_est` a length-*nj* vector standard errors of the final theta estimates for each participant.

`exposure_rate` a matrix containing item-level exposure rates of all items in the pool. Also contains stimulus-level exposure rates if the assembly was set-based.

`usage_matrix` a *nj* by (*ni* + *ns*) matrix representing whether the item/stimulus was administered to each participant. Stimuli representations are appended to the right side of the matrix.

`true_segment_count` a length-*nj* vector containing the how many examinees are now in their segment based on the true theta. This will tend to increase. This can be reproduced with true theta values alone.

`est_segment_count` a length-*nj* vector containing the how many examinees are now in their segment based on the estimated theta. This will tend to increase. This can be reproduced with estimated theta values alone.

`eligibility_stats` exposure record for diagnostics.

`check_eligibility_stats` detailed segment-wise exposure record for diagnostics. available when `config_Shadow@exposure_control$diagnostic_stats` is `TRUE`.

`no_fading_eligibility_stats` detailed segment-wise exposure record without fading for diagnostics. available when `config_Shadow@exposure_control$diagnostic_stats` is `TRUE`.

`freq_infeasible` a table representing the number of times the assembly was initially infeasible.

`pool` the `item_pool` used in the assembly.

`config` the `config_Shadow` used in the assembly.

`constraints` the `constraints` used in the assembly.

`true_theta` the `true_theta` argument used in the assembly.

`data` the `data` argument used in the assembly.

`prior` the `prior` argument used in the assembly.

`prior_par` the `prior_par` argument used in the assembly.

`output_Static-class` *Class 'output_Static': fixed-form assembly solution*

Description

`output_Static` is an S4 class to represent a fixed-form assembly solution.

Slots

`MIP` a list containing the result from MIP solver.

`selected` a `data.frame` containing the selected items and their attributes.

`obj_value` the objective value of the solution.

`solve_time` the elapsed time in running the solver.

`achieved` a `data.frame` containing attributes of the assembled test, by each constraint.

`pool` the `item_pool` used in the assembly.

`config` the `config_Static` used in the assembly.

`constraints` the `constraints` used in the assembly.

plot *Extension of plot() for objects in TestDesign package*

Description

Extension of plot() for objects in TestDesign package

Usage

```
## S4 method for signature 'item_pool'
plot(
  x,
  y,
  type = "info",
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  examinee_id = 1,
  position = NULL,
  theta_range = c(-5, 5),
  ylim = NULL,
  color = "blue",
  z_ci = 1.96,
  simple = TRUE,
  theta_segment = "Estimated",
  color_final = "blue",
  segment = NULL,
  rmse = FALSE,
  use_segment_label = TRUE,
  ...
)

## S4 method for signature 'output_Static'
plot(
  x,
  y,
  type = NULL,
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  examinee_id = 1,
  position = NULL,
  theta_range = c(-5, 5),
  ylim = NULL,
  color = "blue",
```

```
  z_ci = 1.96,
  simple = TRUE,
  theta_segment = "Estimated",
  color_final = "blue",
  segment = NULL,
  rmse = FALSE,
  use_segment_label = TRUE,
  ...
)

## S4 method for signature 'constraints'
plot(
  x,
  y,
  type = "info",
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  examinee_id = 1,
  position = NULL,
  theta_range = c(-5, 5),
  ylim = NULL,
  color = "blue",
  z_ci = 1.96,
  simple = TRUE,
  theta_segment = "Estimated",
  color_final = "blue",
  segment = NULL,
  ...
)

## S4 method for signature 'output_Shadow'
plot(
  x,
  y,
  type = "audit",
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  examinee_id = 1,
  theta_range = c(-5, 5),
  ylim = NULL,
  color = "blue",
  z_ci = 1.96,
  simple = FALSE,
  theta_segment = "Estimated",
```

```
  color_final = "blue",
  segment = NULL,
  rmse = FALSE,
  use_segment_label = TRUE,
  ...
)

## S4 method for signature 'output_Shadow_all'
plot(
  x,
  y,
  type = "audit",
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  examinee_id = 1,
  position = NULL,
  theta_range = c(-5, 5),
  ylim = NULL,
  color = "blue",
  z_ci = 1.96,
  simple = FALSE,
  theta_segment = "Estimated",
  color_final = "blue",
  segment = NULL,
  rmse = FALSE,
  use_segment_label = TRUE,
  ...
)
```

Arguments

x	accepts the following signatures: • item_pool: plot information and expected scores. • constraints: plot information range based on the test length constraint. • output_Static: plot information and expected scores based on the fixed assembly solution. • output_Shadow_all: plot audit trail, shadow test chart, and exposure rates from the adaptive assembly solution. • output_Shadow: plot audit trail and shadow test chart from the adaptive assembly solution.
y	not used, exists for compatibility with plot in the base R package.
type	the type of plot. • info plots information from item_pool, output_Static, and output_Shadow_all. • score plots expected scores from item_pool and output_Static. • audit plots audit trail from output_Shadow_all and output_Shadow.

- shadow plots shadow test chart from output_Shadow_all and output_Shadow.
- exposure plots exposure rates from output_Shadow_all.

theta — the theta grid to use in plotting. (default = seq(-3,3,.1))

info_type — the type of information. Currently accepts FISHER. (default = FISHER)

plot_sum — used in item_pool objects.

- if TRUE then plot pool-level values.
- if FALSE then plot item-level values, and repeat for all items in the pool.
- (default = TRUE)

select — used in item_pool objects. Item indices to subset.

examinee_id — used in output_Shadow and output_Shadow_all with type = 'audit' and type = 'shadow'. The examinee numeric ID to draw the plot.

position — used in output_Shadow_all with type = 'info'. The item position to draw the plot.

theta_range — used in output_Shadow and output_Shadow_all with type = 'audit'. The theta range to plot. (default = c(-5,5))

ylim — (optional) the y-axis plot range. Used in most plot types.

color — the color of the curve.

z_ci — used in output_Shadow and output_Shadow_all with type = 'audit'. The range to use for confidence intervals. (default = 1.96)

simple — used in output_Shadow and output_Shadow_all with type = 'shadow'. If TRUE, simplify the chart by hiding unused items.

theta_segment — used in output_Shadow_all with type = 'exposure'. The type of theta to determine exposure segments. Accepts Estimated or True. (default = Estimated)

color_final — used in output_Shadow_all with type = 'exposure'. The color of item-wise exposure rates, only counting the items administered in the final theta segment as exposed.

segment — used in output_Shadow_all with type = 'exposure'. (optional) The segment index to draw the plot. Leave empty to use all segments.

rmse — used in output_Shadow_all with type = 'exposure'. If TRUE, display the RMSE value for each segment. (default = FALSE)

use_segment_label — used in output_Shadow_all with type = 'exposure'. If TRUE, display the segment label for each segment. (default = TRUE)

... — arguments to pass onto plot.

Details

The base plot() does not allow directly storing the plot as a object. TestDesign::plot() calls recordPlot() internally to allow this. This adds convenience, but also introduces a caveat when using with the 'knitr' package. The caveat is that using plot() alone will not render the plot. This issue can be resolved by using p <-plot() and print(p) in two separate blocks in the markdown document.

Examples

```
subitempool <- itempool_science[1:8]

## Plot item information of a pool
plot(subitempool)
plot(itempool_science, select = 1:8)

## Plot expected score of a pool
plot(subitempool, type = "score")
plot(itempool_science, type = "score", select = 1:8)

## Plot assembly results from Static()
cfg <- createStaticTestConfig()
solution <- Static(cfg, constraints_science)
plot(solution)                # defaults to the objective type
plot(solution, type = "score") # plot expected scores

## Plot attainable information range from constraints
plot(constraints_science)

## Plot assembly results from Shadow()
cfg <- createShadowTestConfig()
set.seed(1)
solution <- Shadow(cfg, constraints_science, true_theta = rnorm(1))
plot(solution, type = 'audit' , examinee_id = 1)
plot(solution, type = 'shadow', examinee_id = 1, simple = TRUE)

## plot(solution, type = 'exposure')
```

`print` *Extension of print() for objects in TestDesign package*

Description

Extension of print() for objects in TestDesign package

Usage

```
## S4 method for signature 'item_1PL'
print(x)

## S4 method for signature 'item_2PL'
print(x)

## S4 method for signature 'item_3PL'
print(x)
```

```
## S4 method for signature 'item_PC'
print(x)

## S4 method for signature 'item_GPC'
print(x)

## S4 method for signature 'item_GR'
print(x)

## S4 method for signature 'item_pool'
print(x)

## S4 method for signature 'item_attrib'
print(x)

## S4 method for signature 'st_attrib'
print(x)

## S4 method for signature 'summary_item_attrib'
print(x)

## S4 method for signature 'summary_st_attrib'
print(x)

## S4 method for signature 'constraints'
print(x)

## S4 method for signature 'config_Static'
print(x)

## S4 method for signature 'config_Shadow'
print(x)

## S4 method for signature 'output_Static'
print(x, index_only = TRUE)

## S4 method for signature 'output_Shadow'
print(x)

## S4 method for signature 'output_Shadow_all'
print(x)

## S4 method for signature 'exposure_rate_plot'
print(x)

## S4 method for signature 'summary_item_pool'
print(x)
```

```
## S4 method for signature 'summary_constraints'
print(x)

## S4 method for signature 'summary_output_Static'
print(x, digits = 3)

## S4 method for signature 'summary_output_Shadow_all'
print(x, digits = 3)
```

Arguments

x	an object to print.
index_only	if TRUE then only print item indices. If FALSE then print all item attributes. (default = TRUE)
digits	minimal number of *significant* digits. See print.default.

---

p_1pl *Calculate item response probability*

Description

p_* and array_p_* are functions to calculate item response probability.

Usage

```
p_1pl(x, b)

p_2pl(x, a, b)

p_3pl(x, a, b, c)

p_pc(x, b)

p_gpc(x, a, b)

p_gr(x, a, b)

array_p_1pl(x, b)

array_p_2pl(x, a, b)

array_p_3pl(x, a, b, c)

array_p_pc(x, b)

array_p_gpc(x, a, b)

array_p_gr(x, a, b)
```

Arguments

x	the theta value. This must be a column vector in matrix form for `array_p_*` functions.
b	the *b*-parameter.
a	the *a*-parameter.
c	the *c*-parameter.

Details

`p_*` functions accept a single theta value, and `array_p_*` functions accept multiple theta values.

Currently supports unidimensional models.

- `p_1pl`, `array_p_1pl`: 1PL models
- `p_2pl`, `array_p_2pl`: 2PL models
- `p_3pl`, `array_p_3pl`: 3PL models
- `p_pc`, `array_p_pc`: PC (partial credit) models
- `p_gpc`, `array_p_gpc`: GPC (generalized partial credit) models
- `p_gr`, `array_p_gr`: GR (graded response) models

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
x <- 0.5

p_1pl(x, 1)
p_2pl(x, 1, 2)
p_3pl(x, 1, 2, 0.25)
p_pc(x, c(0, 1))
p_gpc(x, 2, c(0, 1))
p_gr(x, 2, c(0, 2))

x <- matrix(seq(0.1, 0.5, 0.1)) # column vector in matrix form

array_p_1pl(x, 1)
array_p_2pl(x, 1, 2)
array_p_3pl(x, 1, 2, 0.25)
array_p_pc(x, c(0, 1))
array_p_gpc(x, 2, c(0, 1))
array_p_gr(x, 2, c(0, 2))
```

RE *Calculate Relative Errors*

Description

Calculate Relative Errors.

Usage

```
RE(RMSE_foc, RMSE_ref)
```

Arguments

RMSE_foc	A vector of RMSE values for the focal group.
RMSE_ref	A vector of RMSE values for the reference group.

RMSE *Calculate Root Mean Squared Error*

Description

Calculate Root Mean Squared Error.

Usage

```
RMSE(x, y, conditional = TRUE)
```

Arguments

x	A vector of values.
y	A vector of values.
conditional	If TRUE, calculate RMSE conditional on x.

`runAssembly` *Run Test Assembly*

Description

`runAssembly` is a function to perform test assembly. This function is used internally in `Static` and `Shadow`.

Usage

```
runAssembly(config, constraints, xdata = NULL, objective = NULL)
```

Arguments

config	a `config_Static` or a `config_Shadow` object containing configuration options. Use `createStaticTestConfig` and `createShadowTestConfig` for this.
constraints	a `constraints` object. Use `loadConstraints` for this.
xdata	a list containing extra constraints in MIP form, to force-include previously administered items.
objective	the information value for each item in the pool.

Value

a list containing the following entries:

- `MIP` a list containing the result from MIP solver.
- `status` the MIP status value, indicating whether an optimal solution was found.
- `shadow_test` the attributes of the selected items.
- `obj_value` the objective value of the solution.
- `solve_time` the elapsed time in running the solver.

References

van der Linden, W. J. (2005). *Linear models for optimal test design.* Springer Science & Business Media.

Shadow *Run adaptive test assembly*

Description

Shadow is a test assembly function to perform adaptive test assembly based on the generalized shadow-test framework.

Usage

```
Shadow(
  config,
  constraints = NULL,
  true_theta = NULL,
  data = NULL,
  prior = NULL,
  prior_par = NULL,
  exclude = NULL,
  include_items_for_estimation = NULL,
  force_solver = FALSE,
  session = NULL,
  seed = NULL
)

## S4 method for signature 'config_Shadow'
Shadow(
  config,
  constraints = NULL,
  true_theta = NULL,
  data = NULL,
  prior = NULL,
  prior_par = NULL,
  exclude = NULL,
  include_items_for_estimation = NULL,
  force_solver = FALSE,
  session = NULL,
  seed = NULL
)
```

Arguments

config	a config_Shadow object. Use createShadowTestConfig for this.
constraints	a constraints object representing test specifications. Use loadConstraints for this.
true_theta	(optional) true theta values to use in simulation. Either true_theta or data must be supplied.

`data` (optional) a matrix containing item response data to use in simulation. Either `true_theta` or `data` must be supplied.

`prior` (optional) prior density at each `config@theta_grid`. This overrides `prior_par`. Can be a vector to use the same prior for all *nj* participants, or a *nj*-row matrix to use a different prior for each participant.

`prior_par` (optional) normal distribution parameters `c(mean,sd)` to use as prior. Can be a vector to use the same prior for all *nj* participants, or a *nj*-row matrix to use a different prior for each participant.

`exclude` (optional) a list containing item names in `$i` and set names in `$s` to exclude from selection for each participant. The length of the list must be equal to the number of participants.

`include_items_for_estimation`
(optional) an examinee-wise list containing:

- `administered_item_pool` items to include in theta estimation as `item_pool` object.
- `administered_item_resp` item responses to include in theta estimation.

`force_solver` if `TRUE`, do not check whether the solver is one of recommended solvers for doing set-based assembly. Has no effect on discrete assembly. (default = `FALSE`)

`session` (optional) used to communicate with Shiny app `TestDesign`.

`seed` (optional) used to perform data generation internally.

Value

`Shadow` returns an `output_Shadow_all` object containing assembly results.

References

van der Linden, W. J., Reese, L. M. (1998). A model for optimal constrained adaptive testing. *Applied Psychological Measurement, 22*, 259-270.

van der Linden, W. J. (1998). Optimal assembly of psychological and educational tests. *Applied Psychological Measurement, 22*, 195-211.

van der Linden, W. J. (2000). Optimal assembly of tests with item sets. *Applied Psychological Measurement, 24*, 225-240.

van der Linden, W. J. (2005). *Linear models for optimal test design.* Springer Science & Business Media.

Examples

```
config <- createShadowTestConfig()
true_theta <- rnorm(1)
solution <- Shadow(config, constraints_science, true_theta)
solution@output
```

show *Extension of show() for objects in TestDesign package*

Description

Extension of show() for objects in TestDesign package

Usage

```
## S4 method for signature 'item_1PL'
show(object)

## S4 method for signature 'item_2PL'
show(object)

## S4 method for signature 'item_3PL'
show(object)

## S4 method for signature 'item_PC'
show(object)

## S4 method for signature 'item_GPC'
show(object)

## S4 method for signature 'item_GR'
show(object)

## S4 method for signature 'item_pool'
show(object)

## S4 method for signature 'item_pool_cluster'
show(object)

## S4 method for signature 'item_attrib'
show(object)

## S4 method for signature 'st_attrib'
show(object)

## S4 method for signature 'constraints'
show(object)

## S4 method for signature 'summary_item_pool'
show(object)

## S4 method for signature 'summary_item_attrib'
show(object)
```

```
## S4 method for signature 'summary_st_attrib'
show(object)

## S4 method for signature 'summary_constraints'
show(object)

## S4 method for signature 'config_Static'
show(object)

## S4 method for signature 'config_Shadow'
show(object)

## S4 method for signature 'output_Static'
show(object)

## S4 method for signature 'output_Shadow'
show(object)

## S4 method for signature 'output_Shadow_all'
show(object)

## S4 method for signature 'summary_output_Static'
show(object)

## S4 method for signature 'summary_output_Shadow_all'
show(object)

## S4 method for signature 'exposure_rate_plot'
show(object)
```

Arguments

object	an object to display.

`simResp` *Simulate item response data*

Description

`simResp` is a function to simulate item response data.

Usage

```
simResp(object, theta)

## S4 method for signature 'item_1PL,numeric'
```

```
simResp(object, theta)

## S4 method for signature 'item_2PL,numeric'
simResp(object, theta)

## S4 method for signature 'item_3PL,numeric'
simResp(object, theta)

## S4 method for signature 'item_PC,numeric'
simResp(object, theta)

## S4 method for signature 'item_GPC,numeric'
simResp(object, theta)

## S4 method for signature 'item_GR,numeric'
simResp(object, theta)

## S4 method for signature 'item_pool,numeric'
simResp(object, theta)

## S4 method for signature 'item_pool_cluster,numeric'
simResp(object, theta)

## S4 method for signature 'item_pool_cluster,list'
simResp(object, theta)

## S4 method for signature 'item_pool_cluster,list'
simResp(object, theta)
```

Arguments

`object`	an `item` or an `item_pool` object.
`theta`	theta values to use.

Details

notations • *nq* denotes the number of theta values.
• *ni* denotes the number of items in the `item_pool` object.

Value

`item` **object:** `simResp` returns a length *nq* vector containing simulated item response data.

`item_pool` **object:** `simResp` returns a (*nq*, *ni*) matrix containing simulated item response data.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
item_1    <- new("item_1PL", difficulty = 0.5)
item_2    <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3    <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4    <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
item_5    <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
item_6    <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)

sim_item_1 <- simResp(item_1, seq(-3, 3, 1))
sim_item_2 <- simResp(item_2, seq(-3, 3, 1))
sim_item_3 <- simResp(item_3, seq(-3, 3, 1))
sim_item_4 <- simResp(item_4, seq(-3, 3, 1))
sim_item_5 <- simResp(item_5, seq(-3, 3, 1))
sim_item_6 <- simResp(item_6, seq(-3, 3, 1))
sim_pool   <- simResp(itempool_science, seq(-3, 3, 1))
```

`Static` *Run fixed-form test assembly*

Description

`Static` is a test assembly function to perform fixed-form test assembly based on the generalized shadow-test framework.

Usage

```
Static(config, constraints, force_solver = FALSE)

## S4 method for signature 'config_Static'
Static(config, constraints, force_solver = FALSE)
```

Arguments

config	a config_Static object. Use createStaticTestConfig for this.
constraints	a constraints object representing test specifications. Use loadConstraints for this.
force_solver	if TRUE, do not check whether the solver is one of recommended solvers for doing set-based assembly. Has no effect on discrete assembly. (default = FALSE)

Value

Static returns a output_Static object containing the selected items.

References

van der Linden, W. J. (2005). *Linear models for optimal test design.* Springer Science & Business Media.

Examples

```
config_science <- createStaticTestConfig(
  list(
    method = "MAXINFO",
    target_location = c(-1, 1)
  )
)
solution <- Static(config_science, constraints_science)
```

---

st_attrib-class *Load set/stimulus/passage attributes*

Description

loadStAttrib is a data loading function to create an st_attrib object. loadStAttrib can read stimulus attributes a data.frame or a .csv file.

Usage

```
loadStAttrib(object, item_attrib, file = NULL)
```

Arguments

object	set attributes. Can be a data.frame or the file path of a .csv file. The content should at least include the column 'STID' referring to the column 'STID' in the data slot of the item_attrib object.
item_attrib	an item_attrib object. Use loadItemAttrib for this.
file	(deprecated) use object argument instead.

Value

loadStAttrib returns a st_attrib object.

- data a data.frame containing stimulus attributes.

See Also

dataset_reading for examples.

Examples

```
## Read from data.frame:
itempool_reading   <- loadItemPool(itempool_reading_data)
itemattrib_reading <- loadItemAttrib(itemattrib_reading_data, itempool_reading)
stimattrib_reading <- loadStAttrib(stimattrib_reading_data, itemattrib_reading)

## Read from file: write to tempdir() for illustration and clean afterwards
f <- file.path(tempdir(), "stimattrib_reading.csv")
write.csv(stimattrib_reading_data, f, row.names = FALSE)
stimattrib_reading <- loadStAttrib(f, itemattrib_reading)
file.remove(f)

## TestDesign 1.1.0 - Deprecated arguments
## Not run:
loadStAttrib(object = "satt.csv", item_attrib) # is equivalent to
loadStAttrib(file   = "satt.csv", item_attrib) # pre 1.1.0

## End(Not run)
```

---

st_attrib-operators *Basic functions for stimulus attribute objects*

Description

Basic functions for stimulus attribute objects

Usage

```
## S4 method for signature 'st_attrib,numeric'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'st_attrib'
dim(x)

## S4 method for signature 'st_attrib'
colnames(x)

## S4 method for signature 'st_attrib'
rownames(x)

## S4 method for signature 'st_attrib'
names(x)

## S4 method for signature 'st_attrib'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	a st_attrib object.
i, j	indices to use in subsetting.
...	not used, exists for compatibility.
drop	not used, exists for compatibility.
row.names	not used, exists for compatibility.
optional	not used, exists for compatibility.

Examples

```
x <- stimattrib_reading
x[1:10]
dim(x)
ncol(x)
nrow(x)
colnames(x)
rownames(x)
names(x)
as.data.frame(x)
```

---

summary *Extension of summary() for objects in TestDesign package*

Description

Extension of summary() for objects in TestDesign package

Usage

```
## S4 method for signature 'item_pool'
summary(object)

## S4 method for signature 'item_attrib'
summary(object)

## S4 method for signature 'st_attrib'
summary(object)

## S4 method for signature 'constraints'
summary(object)

## S4 method for signature 'output_Static'
summary(object, simple = FALSE)

## S4 method for signature 'output_Shadow_all'
summary(object, simple = FALSE)
```

Arguments

object	an object to summarize.
simple	if TRUE, do not print constraints. (default = FALSE)

Examples

```
summary(itempool_science)
summary(itemattrib_science)

cfg <- createStaticTestConfig()
solution <- Static(cfg, constraints_science)
summary(solution)
summary(solution, simple = TRUE)

cfg <- createShadowTestConfig()
solution <- Shadow(cfg, constraints_science, true_theta = seq(-1, 1, 1))
summary(solution)
summary(solution, simple = TRUE)
```

---

summary-classes *Summary classes*

Description

Summary classes

test-class *Class 'test': data for test assembly*

Description

`test` is an S4 class to represent data for test assembly.

Slots

`pool` the `item_pool` object.

`theta` the theta grid to use as quadrature points.

`prob` the list containing item response probabilities.

`info` the matrix containing item information values.

`true_theta` (optional) the true theta values.

`data` (optional) the matrix containing item responses.

TestDesign *Open TestDesign app*

Description

`TestDesign` is a caller function to open the Shiny interface of TestDesign package.

Usage

```
TestDesign()
```

Examples

```
## Not run:
if (interactive()) {
  TestDesign()
}

## End(Not run)
```

testSolver *Test solver*

Description

Test solver

Usage

```
testSolver(solver)
```

Arguments

solver	a solver package name. Accepts `lpSolve`, `Rsymphony`, `lpsymphony`, `gurobi`, `Rglpk`.

Value

empty string `""` if solver works. A string containing error messages otherwise.

test_cluster-class *Class 'test_cluster': data for test assembly*

Description

`test_cluster` is an S4 class to represent data for test assembly.

Slots

`nt` the number of `test` objects in this cluster.

`tests` the list containing `test` objects.

`names` test ID strings for each `test` object.

test_operators *Basic operators for test objects*

Description

Create a subset of a test object.

Usage

```
subsetTest(x, i = NULL)

## S4 method for signature 'test,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

x	a test object.
i	item indices to use in subsetting.
j, drop, ...	not used, exists for compatibility.

theta_EAP *Calculate an EAP estimate of theta for one examinee*

Description

Calculate an expected a posterior estimate of theta for one examinee.

Usage

```
theta_EAP(theta_grid, item_parm, resp, ncat, model, prior, prior_parm)
```

Arguments

theta_grid	An equi-spaced theta grid.
item_parm	A numeric matrix of item parameters.
resp	a numeric vector containing item responses.
ncat	A numeric vector of the number of response categories by item.
model	A numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
prior	The type of prior distribution (1: normal, 2: uniform).
prior_parm	A numeric vector of hyperparameters for the prior distribution, c(mu, sigma) or c(ll, ul).

theta_EAP_matrix *Calculate EAP estimates of theta for a group of examinees*

Description

Calculate expected a posteriori estimates of theta for a group of examinees.

Usage

```
theta_EAP_matrix(theta_grid, item_parm, resp, ncat, model, prior, prior_parm)
```

Arguments

theta_grid	An equi-spaced theta grid.
item_parm	A numeric matrix of item parameters.
resp	A numeric matrix of item responses.
ncat	A numeric vector of the number of response categories by item.
model	A numeric vector of the IRT model by item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
prior	The type of prior distribution (1: normal, 2: uniform).
prior_parm	A numeric vector of hyperparameters for the prior distribution, c(mu, sigma) or c(ll, ul).

theta_EB *Calculate theta estimates using EB (Empirical Bayes) method*

Description

theta_EB_single and theta_EB are functions to calculate theta estimates using EB (Empirical Bayes) method.

Usage

```
theta_EB(
  nx,
  theta_init,
  theta_prop,
  item_parm,
  resp,
  ncat,
  model,
  prior,
  prior_parm
```

```
)

theta_EB_single(
  nx,
  theta_init,
  theta_prop,
  item_parm,
  resp,
  ncat,
  model,
  prior,
  prior_parm
)
```

Arguments

`nx`	the number of MCMC draws.
`theta_init`	initial estimate of theta.
`theta_prop`	SD of the proposal distribution.
`item_parm`	a matrix containing item parameters. Each row represents each item.
`resp`	a vector (or a value if for one item) containing responses on each item.
`ncat`	a vector (or a value if for one item) containing the number of response categories of each item.
`model`	a vector (or a value if for one item) indicating item models of each item, using • `1`: 1PL model • `2`: 2PL model • `3`: 3PL model • `4`: PC model • `5`: GPC model • `6`: GR model
`prior`	an integer indicating the type of prior distribution, using • `1`: normal distribution • `2`: uniform distribution
`prior_parm`	a vector containing parameters for the prior distribution.

Details

`theta_EB_single` is designed for one item, and `theta_EB` is designed for multiple items.

Currently supports unidimensional models.

References

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danish Institute for Educational Research.

Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.

Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397-479. Reading, MA: Addison-Wesley.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph, 17*.

Examples

```
# item parameters
item_parm <- matrix(c(
  1, NA,   NA,
  1,  2,   NA,
  1,  2, 0.25,
  0,  1,   NA,
  2,  0,    1,
  2,  0,    2),
  nrow = 6,
  byrow = TRUE
)

ncat  <- c(2, 2, 2, 3, 3, 3)
model <- c(1, 2, 3, 4, 5, 6)
resp  <- c(0, 1, 0, 1, 0, 1)

nx <- 100
theta_init <- 0
theta_prop <- 1.0
set.seed(1)
theta_EB_single(nx, theta_init, theta_prop, item_parm[1, ], resp[1], ncat[1], model[1], 1, c(0, 1))
theta_EB(nx, theta_init, theta_prop, item_parm, resp, ncat, model, 1, c(0, 1))
```

`theta_FB` *Calculate a fully Bayesian estimate of theta for an examinee*

Description

Calculate a fully Bayesian estimate of theta for an examinee.

Usage

```
theta_FB(
  nx,
  theta_init,
  theta_prop,
  items_list,
  item_init,
  resp,
  ncat,
  model,
  prior,
  prior_parm
)
```

Arguments

`nx`	The number of MCMC draws.
`theta_init`	A value for initial estimate of theta.
`theta_prop`	SD of the proposal distribution.
`items_list`	A list of item_parm matrices.
`item_init`	A matrix of item parameter estimates (one row per item).
`resp`	a numeric vector containing item responses.
`ncat`	A numeric vector of the number of response categories by item.
`model`	A numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
`prior`	The type of prior distribution (1: normal, 2: uniform).
`prior_parm`	A numeric vector of hyperparameters for the prior distribution, c(mu, sigma) or c(ll, ul).

`theta_FB_single` *Calculate a fully Bayesian estimate of theta for a single item*

Description

Calculate a fully Bayesian estimate of theta for a single item.

Usage

```
theta_FB_single(
  nx,
  theta_init,
  theta_prop,
  item_mcmc,
  item_init,
  resp,
  ncat,
  model,
  prior,
  prior_parm
)
```

Arguments

`nx`	The number of MCMC draws.
`theta_init`	A value for initial estimate of theta.
`theta_prop`	SD of the proposal distribution.
`item_mcmc`	A matrix of sampled item parameters for a single item.
`item_init`	A matrix of item parameter estimates (one row per item).
`resp`	a numeric vector containing item responses.
`ncat`	A numeric vector of the number of response categories by item.
`model`	A numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
`prior`	The type of prior distribution (1: normal, 2: uniform).
`prior_parm`	A numeric vector of hyperparameters for the prior distribution, c(mu, sigma) or c(ll, ul).

`toggleConstraints` *Toggle constraints*

Description

`toggleConstraints` is a function to toggle individual constraints in a `constraints` object.

Usage

```
toggleConstraints(object, on = NULL, off = NULL)
```

Arguments

`object`	a `constraints` object from `loadConstraints`.
`on`	constraint indices to mark as active. Also accepts character IDs.
`off`	constraint indices to mark as inactive. Also accepts character IDs.

Value

`toggleConstraints` returns the updated `constraints` object.

Examples

```
constraints_science2 <- toggleConstraints(constraints_science, off = 32:36)
constraints_science3 <- toggleConstraints(constraints_science2, on = 32:36)
constraints_science4 <- toggleConstraints(constraints_science, off = "C32")
```

Index