Package ‘TestDesign’

February 6, 2020

Type Package
Title Optimal Test Design Approach to Fixed and Adaptive Test Construction
Version 1.0.2
Date 2020-01-25
Maintainer Seung W. Choi <schoi@austin.utexas.edu>

URL https://github.com/choi-phd/TestDesign

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

License GPL (>= 2)
Depends R (>= 2.10)
biocViews
Imports Rcpp (>= 1.0.0), methods, Matrix, lpSolve, foreach, logitnorm, Rdpack, crayon
Suggests lpsymphony, Rsymphony, gurobi, Rglpk, shiny, shinythemes, shinyWidgets, shinyjs, DT, knitr, rmarkdown, kableExtra, testthat (>= 2.1.0)

LinkingTo Rcpp
RoxygenNote 7.0.2
Encoding UTF-8
LazyData true
RdMacros Rdpack
VignetteBuilder knitr
Collate 'RcppExports.R' 'import.R' 'item_class.R' 'item_functions.R'
   'loading_functions.R' 'static_class.R' 'shadow_class.R'
   'static_functions.R' 'shadow_functions.R' 'datasets.R'
   'solver_functions.R' 'helper_functions.R' 'runshiny.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 2020-02-06 00:50:02 UTC

R topics documented:

addTrans ................................................. 4
array_info_1pl ........................................ 5
array_info_2pl ........................................ 5
array_info_3pl ........................................ 6
array_info_gpc ........................................ 7
array_info_gr .......................................... 7
array_info_pc .......................................... 8
array_p_1pl ........................................... 8
array_p_2pl ........................................... 9
array_p_3pl ........................................... 9
array_p_gpc .......................................... 10
array_p_gr ........................................... 10
array_p_pc ........................................... 11
buildConstraints ...................................... 11
calcDerivative ......................................... 12
calcDerivative2 ....................................... 14
calcEscore ............................................ 16
calcFisher ............................................ 17
calcHessian ........................................... 19
calcJacobian ........................................... 21
calcLocation .......................................... 23
calcProb ............................................... 25
calcRP ................................................. 26
calc_info ............................................. 27
calc_info_EB ......................................... 28
calc_info_FB ......................................... 28
calc_info_matrix .................................... 29
calc_likelihood ........................................ 29
calc_likelihood_function ............................. 30
calc_log_likelihood ................................... 30
calc_log_likelihood_function ......................... 31
calc_MI_FB .......................................... 31
calc_posterior ........................................ 32
calc_posterior ....................................... 32
topics documented:
calc_posterior_single ........................................... 33
checkConstraints ................................................. 34
config_Shadow-class ............................................. 34
config_Static-class ............................................... 37
constraint-class ................................................... 38
constraints-class ................................................. 39
dataset_fatigue ..................................................... 39
dataset_reading ..................................................... 40
dataset_science ..................................................... 41
EAP ................................................................. 42
eap ................................................................. 43
extract-methods ..................................................... 43
find_segment ....................................................... 44
getSolution ......................................................... 44
info_1pl ............................................................ 45
info_2pl ............................................................ 45
info_3pl ............................................................ 46
info_gpc ........................................................... 47
info_gr ............................................................. 47
info_pc ............................................................. 48
iparPosteriorSample .............................................. 48
item_1PL-class ..................................................... 49
item_2PL-class ..................................................... 49
item_3PL-class ..................................................... 50
item_attrib-class .................................................. 50
item_GPC-class .................................................... 51
item_GR-class ..................................................... 51
item_PC-class ..................................................... 52
item_pool-class ................................................... 52
item_pool_operators ............................................... 53
lnHyperPars ......................................................... 54
loadConstraints ..................................................... 54
loadItemAttrib ...................................................... 55
loadItemPool ....................................................... 56
loadStAttrib ....................................................... 57
logitHyperPars ..................................................... 58
makeItemPoolCluster .............................................. 58
makeTest .......................................................... 59
makeTestCluster ................................................... 60
MLE ............................................................... 60
mle ............................................................... 61
OAT ............................................................... 62
output_Shadow-class ............................................. 62
plotCAT ............................................................ 64
plotEligibilityStats .............................................. 65
plotExposure ....................................................... 66
plotExposureRateBySegment ................................... 67
plotExposureRateFinal ........................................... 68
addTrans

Add transparency to color.

**Usage**

`addTrans(color, alpha)`
array_info_1pl

Arguments

- **color**: A vector of color names or RGB color codes.
- **alpha**: A vector of integers between 0 and 255 (0 = fully transparent, 255 = fully visible).

Description

Calculate Fisher information at theta values according to the 1PL model.

Usage

array_info_1pl(x, b)

Arguments

- **x**: Numeric. A vector of theta values.
- **b**: Numeric. A difficulty parameter value.

References


array_info_2pl

Calculate Fisher information at multiple thetas (2PL)

Description

Calculate the Fisher information at theta values according to the 2PL model.

Usage

array_info_2pl(x, a, b)

Arguments

- **x**: Numeric. A vector of theta values.
- **a**: Numeric. A slope parameter value.
- **b**: Numeric. A difficulty parameter value.
References


array_info_3pl

*Calculate Fisher information at multiple thetas (3PL)*

Description

Calculate the Fisher information at theta values according to the 3PL model.

Usage

array_info_3pl(x, a, b, c)

Arguments

- **x**: Numeric. A vector of theta values.
- **a**: Numeric. A slope parameter value.
- **b**: Numeric. A difficulty parameter value.
- **c**: Numeric. A guessing parameter value.

References

array_info_gpc

Calculate Fisher information at multiple thetas (GPC)

Description
Calculate the Fisher information at theta values according to the generalized partial credit model.

Usage
array_info_gpc(x, a, b)

Arguments
- x: Numeric. A vector of theta values.
- a: Numeric. A slope parameter value.

References

array_info_gr

Calculate Fisher information at multiple thetas (GR)

Description
Calculate the Fisher information at theta values according to the graded response model.

Usage
array_info_gr(x, a, b)

Arguments
- x: Numeric. A vector of theta values.
- a: Numeric. A slope parameter value.

References
array_info_pc

*Calculate Fisher information at multiple thetas (PC)*

**Description**

Calculate the Fisher information at theta values according to the partial credit model.

**Usage**

```r
array_info_pc(x, b)
```

**Arguments**

- `x` Numeric. A vector of theta values.
- `b` Numeric. A vector of threshold parameter values.

**References**


---

array_p_1pl

*Calculate probability at multiple thetas (1PL)*

**Description**

Calculate the probability of correct response at theta values, under the 1PL model.

**Usage**

```r
array_p_1pl(x, b)
```

**Arguments**

- `x` Numeric. A vector of theta values.
- `b` Numeric. A difficulty parameter value.

**References**

array_p_2pl  

Calculate probability at multiple thetas (2PL)

Description

Calculate the probability of correct response at theta values, under the 2PL model.

Usage

array_p_2pl(x, a, b)

Arguments

x  Numeric. A vector of theta values.

a  Numeric. A slope parameter value.

b  Numeric. A difficulty parameter value.

References


array_p_3pl  

Calculate probability at multiple thetas (3PL)

Description

Calculate the probability of correct response at theta values, under the 3PL model.

Usage

array_p_3pl(x, a, b, c)

Arguments

x  Numeric. A vector of theta values.

a  Numeric. A slope parameter value.

b  Numeric. A difficulty parameter value.

c  Numeric. A guessing parameter value.
References


array_p_gpc

*Calculate probability at multiple thetas (GPC)*

Description

Calculate the probability of correct response at theta values, under the generalized partial credit model.

Usage

array_p_gpc(x, a, b)

Arguments

x Numeric. A vector of theta values.
a Numeric. A slope parameter value.
b Numeric. A vector of threshold parameter values.

References


array_p_gr

*Calculate probability at multiple thetas (GR)*

Description

Calculate the probability of correct response at theta values, under the graded response model.

Usage

array_p_gr(x, a, b)

Arguments

x Numeric. A vector of theta values.
a Numeric. A slope parameter value.
b Numeric. A vector of category boundary parameter values.
array_p_pc

References


array_p_pc Calculate probability at multiple thetas (PC)

Description

Calculate the probability of correct response at theta values, under the partial credit model.

Usage

array_p_pc(x, b)

Arguments

x Numeric. A vector of theta values.
b Numeric. A vector of threshold parameter values.

References


buildConstraints Build constraints

Description

Read constraints from specified files.

Usage

buildConstraints(
    pool,
    file_constraints,
    file_item_attrib,
    file_st_attrib = NULL
)
Arguments

- pool: An item_pool object. Use `loadItemPool` for this.
- file_constraints: Character. The name of the file containing constraint specifications.
- file_item_attrib: Character. The name of the file containing item attributes.
- file_st_attrib: (Optional) Character. The name of the file containing set attributes.

Value

A list containing the parsed constraints, to be used in `Static` and `Shadow`.

Examples

```r
## Write to tempdir() and clean afterwards
f1 <- file.path(tempdir(), "constraints_science.csv")
write.csv(constraints_science_raw, f1, row.names = FALSE)
f2 <- file.path(tempdir(), "itemattrib_science.csv")
write.csv(itemattrib_science_raw, f2, row.names = FALSE)

constraints <- buildConstraints(itempool_science, f1, f2)
file.remove(f1)
file.remove(f2)
```

calcDerivative Calculation first derivative

Description

An S4 generic and its methods to calculate the first derivative of the probability function.

Usage

```r
calcDerivative(object, theta)
```

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

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

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

## S4 method for signature 'item_PC,numeric'
```
calcDerivative

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

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

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

## S4 method for signature 'pool_cluster,numeric'
calcDerivative(object, theta)

### Arguments

- **object**: An instance of an item class.
- **theta**: A vector of theta values.

### Value

First derivative values.

### References


Examples

```r
item_1 <- new("item_1PL", difficulty = 0.5)
d.item_1 <- calcDerivative(item_1, seq(-3, 3, 1))
item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
d.item_2 <- calcDerivative(item_2, seq(-3, 3, 1))
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
d.item_3 <- calcDerivative(item_3, seq(-3, 3, 1))
item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
d.item_4 <- calcDerivative(item_4, seq(-3, 3, 1))
item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
d.item_5 <- calcDerivative(item_5, seq(-3, 3, 1))
item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)
d.item_6 <- calcDerivative(item_6, seq(-3, 3, 1))
d_itempool <- calcDerivative(itempool_science, seq(-3, 3, 1))
```

calcDerivative2

### Description

An S4 generic and its methods to calculate the second derivative of the probability function.

### Usage

```r
calcDerivative2(object, theta)
```

```r
## S4 method for signature 'item_1PL', numeric'
calcDerivative2(object, theta)
```

```r
## S4 method for signature 'item_2PL', numeric'
calcDerivative2(object, theta)
```

```r
## S4 method for signature 'item_3PL', numeric'
calcDerivative2(object, theta)
```

```r
## S4 method for signature 'item_PC', numeric'
calcDerivative2(object, theta)
```

```r
## S4 method for signature 'item_GPC', numeric'
calcDerivative2(object, theta)
```

```r
## S4 method for signature 'item_GR', numeric'
calcDerivative2(object, theta)
```

```r
## S4 method for signature 'item_pool', numeric'
calcDerivative2(object, theta)
```

```r
## S4 method for signature 'pool_cluster', numeric'
calcDerivative2(object, theta)
```
Arguments

object An instance of an item class.
theta A vector of theta values.

Value

Second derivative values.

References


Examples

```r
item_1 <- new("item_1PL", difficulty = 0.5)
dd_item_1 <- calcDerivative2(item_1, seq(-3, 3, 1))
item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
dd_item_2 <- calcDerivative2(item_2, seq(-3, 3, 1))
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
dd_item_3 <- calcDerivative2(item_3, seq(-3, 3, 1))
item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
dd_item_4 <- calcDerivative2(item_4, seq(-3, 3, 1))
item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
dd_item_5 <- calcDerivative2(item_5, seq(-3, 3, 1))
item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)
dd_item_6 <- calcDerivative2(item_6, seq(-3, 3, 1))
itempool <- new("itempool", seq(-3, 3, 1))
```

Description

An S4 generic and its methods to calculate expected scores given a vector of thetas for different item classes.

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 'pool_cluster,numeric'
calcEscore(object, theta)

Arguments

object An instance of an item class.
theta A vector of theta values.

Value

A vector of expected scores of length nq (the number of values on theta grid).
References


Examples

```r
item_1 <- new("item_1PL", difficulty = 0.5)
ICC_item_1 <- calcEscore(item_1, seq(-3, 3, 1))

item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
ICC_item_2 <- calcEscore(item_2, seq(-3, 3, 1))

item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
ICC_item_3 <- calcEscore(item_3, seq(-3, 3, 1))

item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
ICC_item_4 <- calcEscore(item_4, seq(-3, 3, 1))

item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
ICC_item_5 <- calcEscore(item_5, seq(-3, 3, 1))

item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)
ICC_item_6 <- calcEscore(item_6, seq(-3, 3, 1))

TCC_itempool <- calcEscore(itempool_science, seq(-3, 3, 1))
```

calcFisher

*Calculate Fisher information*

Description

An S4 generic and its methods to calculate Fisher information given a vector of thetas for different item classes.
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 'pool_cluster,numeric'
calcFisher(object, theta)

Arguments

object An instance of an item class.
theta A vector of theta values.

Value

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


calcHessian

Calculate second derivative of log-likelihood

Description

An S4 generic and its methods 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 'itempool_science,numeric,numeric'
calcHessian(object, theta, resp)

Examples

```r
item_1 <- new("item_1PL", difficulty = 0.5)
info_item_1 <- calcFisher(item_1, seq(-3, 3, 1))

item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
info_item_2 <- calcFisher(item_2, seq(-3, 3, 1))

item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
info_item_3 <- calcFisher(item_3, seq(-3, 3, 1))

item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
info_item_4 <- calcFisher(item_4, seq(-3, 3, 1))

item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
info_item_5 <- calcFisher(item_5, seq(-3, 3, 1))

item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)
info_item_6 <- calcFisher(item_6, seq(-3, 3, 1))

info_itempool <- calcFisher(itempool_science, seq(-3, 3, 1))
```
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 'pool_cluster, numeric, list'
calcHessian(object, theta, resp)

### Arguments

- **object**: An instance of an item class.
- **theta**: A vector of theta values.
- **resp**: Response data.

### Value

Second derivative values of log-likelihoods.

### References


**Examples**

```r
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)
```

**Description**

An S4 generic and its methods to calculate the first derivative of the log-likelihood function.

**Usage**

```r
calcJacobian(object, theta, resp)
```

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

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

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

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

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

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

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

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

**Arguments**

- **object**: An instance of an item class.
- **theta**: A vector of theta values.
- **resp**: Response data.

**Value**

First derivative values of log-likelihoods.

**References**


**Examples**

```r
item_1 <- new("item_1PL", difficulty = 0.5)
j_item_1 <- calcJacobian(item_1, seq(-3, 3, 1), 0)
item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
```
calcLocation <- calcJacobian(item_2, seq(-3, 3, 1), 0)
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
j_item_3 <- calcJacobian(item_3, seq(-3, 3, 1), 0)
item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
j_item_4 <- calcJacobian(item_4, seq(-3, 3, 1), 0)
item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
j_item_5 <- calcJacobian(item_5, seq(-3, 3, 1), 0)
item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)
j_item_6 <- calcJacobian(item_6, seq(-3, 3, 1), 0)
j_itempool <- calcJacobian(itempool_science, seq(-3, 3, 1), 0)

calcLocation |
---|
Calculate item location

**Description**

An S4 generic and its methods to calculate item location.

**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)

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

**Arguments**

object

An instance of an item class.
Value

Item location values.

References


Examples

```r
item_1 <- new("item_1PL", difficulty = 0.5)
theta_item_1 <- calcLocation(item_1)
item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
theta_item_2 <- calcLocation(item_2)
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
theta_item_3 <- calcLocation(item_3)
item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
theta_item_4 <- calcLocation(item_4)
item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
theta_item_5 <- calcLocation(item_5)
item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)
theta_item_6 <- calcLocation(item_6)
theta_itempool <- calcLocation(itempool_science)
```
Description

An S4 generic and its methods to calculate item response probabilities for different item classes

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 'pool_cluster,numeric'
calcProb(object, theta)

Arguments

object An instance of an item class.
theta A vector of theta values.

Value

A matrix of probability values with a dimension (nq, ncat) for a single item or a list of matrices for an instance of "item_pool".
References


Examples

```r
item_1 <- new("item_1PL", difficulty = 0.5)
prob_item_1 <- calcProb(item_1, seq(-3, 3, 1))
item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
prob_item_2 <- calcProb(item_2, seq(-3, 3, 1))
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
prob_item_3 <- calcProb(item_3, seq(-3, 3, 1))
item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)
prob_item_4 <- calcProb(item_4, seq(-3, 3, 1))
item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
prob_item_5 <- calcProb(item_5, seq(-3, 3, 1))
item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)
prob_item_6 <- calcProb(item_6, seq(-3, 3, 1))
prob_itempool <- calcProb(itempool_science, seq(-3, 3, 1))
```

calcRP

*Find matching theta to supplied probability*

Description

Find theta corresponding to a response probability value for each item.
**Usage**

calcRP(object, rp = 0.5, max_iter = 100, conv = 1e-04, start_theta = 0)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An item_pool object.</td>
</tr>
<tr>
<td>rp</td>
<td>A response probability value.</td>
</tr>
<tr>
<td>max_iter</td>
<td>A maximum number of iterations.</td>
</tr>
<tr>
<td>conv</td>
<td>A convergence criterion.</td>
</tr>
<tr>
<td>start_theta</td>
<td>A starting theta value.</td>
</tr>
</tbody>
</table>

**Description**

Calculate the Fisher information matrix for a single theta value and a set of items, potentially with a mixture of different models

**Usage**

calc_info(x, item_parm, ncat, model)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Numeric. A single theta value.</td>
</tr>
<tr>
<td>item_parm</td>
<td>A matrix of item parameters.</td>
</tr>
<tr>
<td>ncat</td>
<td>A numeric vector of the number of response categories by item.</td>
</tr>
<tr>
<td>model</td>
<td>A numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).</td>
</tr>
</tbody>
</table>
calc_info_EB  
*Calculate the Fisher information using empirical Bayes*

**Description**

Calculate the Fisher information using empirical Bayes.

**Usage**

```r
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 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).

---

calc_info_FB  
*Calculate the Fisher information using full Bayesian*

**Description**

Calculate the Fisher information using full Bayesian.

**Usage**

```r
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 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).
- `useEAP` TRUE to use the mean of MCMC theta draws.
calc_info_matrix

Calculate the Fisher information matrix for a vector of theta values and a set of items, potentially with a mixture of different models.

### Description

Calculate the Fisher information matrix for a vector of theta values and a set of items, potentially with a mixture of different models.

### Usage

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

### Arguments

- **x**: Numeric. A vector of theta values.
- **item_parm**: A matrix of item parameters.
- **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).

---

calc_likelihood

Calculate a likelihood value of theta.

### Description

Calculate a likelihood value of theta.

### Usage

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

### Arguments

- **x**: Numeric. A single theta value.
- **item_parm**: A numeric matrix of item parameters.
- **resp**: A numeric vector of 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).
**calc_likelihood_function**

*Calculate a likelihood function of theta*

**Description**

Calculate a likelihood function of theta.

**Usage**

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

**Arguments**

- `theta_grid` An equi-spaced grid of theta values.
- `item_parm` A numeric matrix of item parameters.
- `resp` A numeric vector of 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).

**calc_log_likelihood**

*Calculate a log-likelihood value of theta*

**Description**

Calculate a log-likelihood value of theta.

**Usage**

```r
calc_log_likelihood(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 of 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_log_likelihood_function

*Calculate a log-likelihood function of theta*

### Description

Calculate a log-likelihood function of theta.

### Usage

```r
calc_log_likelihood_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 of 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_MI_FB

*Calculate the mutual information using full Bayesian*

### Description

Calculate the mutual information using full Bayesian.

### Usage

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

Arguments

- **x**: A numeric vector of MCMC sampled theta values.
- **items_list**: A list of item parameter matrices.
- **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).

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 of 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 of 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

**Description**

Check the consistency of constraints and item usage.

**Usage**

```r
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

**Description**

Create a `config_Shadow` object for Shadow Test Assembly (STA).

**Usage**

```r
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),
  audit_trail = F
)
```
Arguments

- **item_selection** A list containing item selection criteria.
  - **method** The type of criteria. Accepts one of MFI, MPWI, FB, EB.
  - **info_type** The type of information. Accepts FISHER.
  - **initial_theta** Initial theta value(s) for the first item selection.
  - **fixed_theta** Fixed theta value(s) to optimize for all items to select.

- **content_balancing** A list containing content balancing options.
  - **method** The type of balancing method. Accepts one of NONE, STA.

- **MIP** A list containing solver options.
  - **solver** The type of solver. Accepts one of lpSymphony, Rsymphony, gurobi, lpSolve, Rglpk.
  - **verbosity** Verbosity level.
  - **time_limit** Time limit to be passed onto solver. Used in solvers lpSymphony, Rsymphony, gurobi, Rglpk.
  - **gap_limit** Gap limit (relative) to be passed onto solver. Used in solver gurobi. Uses the solver default when NULL.
  - **gap_limit_abs** Gap limit (absolute) to be passed onto solver. Used in solver lpSymphony, Rsymphony. Uses the solver default when NULL.

- **MCMC** A list containing Markov-chain Monte Carlo configurations.
  - **burn_in** Numeric. The number of chains from the start to discard.
  - **post_burn_in** Numeric. The number of chains to use after discarding the first burn_in chains.
  - **thin** Numeric. Thinning interval.
  - **jumpfactor** Numeric. Jump factor.

- **refresh_policy** A list containing refresh policy for obtaining a new shadow test.
  - **method** The type of policy. Accepts one of ALWAYS, POSITION, INTERVAL, THRESHOLD, INTERVAL-THRESHOLD.
  - **interval** Integer. Set to 1 to refresh at each position, 2 to refresh at every two positions, and so on.
  - **threshold** Numeric. The shadow test is refreshed when the absolute change in theta estimate is greater than this value.
  - **position** Numeric. Position(s) at which refresh to occur.

- **exposure_control** A list containing exposure control settings.
  - **method** Accepts one of "NONE", "ELIGIBILITY", "BIGM", "BIGM-BAYESIAN".
  - **M** Big M constant.
  - **max_exposure_rate** Maximum target exposure rate.
  - **acceleration_factor** Acceleration factor.
  - **n_segment** Number of theta segments.
  - **first_segment** Theta segment assumed at the beginning of test.
  - **segment_cut** A numeric vector of segment cuts.
  - **initial_eligibility_stats** A list of eligibility statistics from a previous run.
config_Shadow-class

- fading_factor Fading factor.
- diagnostic_stats TRUE to generate diagnostic statistics.

stopping_criterion
A list containing stopping criterion.
- method Accepts one of "FIXED".
- test_length Test length.
- min_ni Maximum number of items to administer.
- max_ni Minimum number of items to administer.
- se_threshold Standard error threshold for stopping.

interim_theta
A list containing interim theta estimation options.
- method The type of estimation. Accepts one of EAP, EB, FB.
- shrinkage_correction Set TRUE to correct for shrinkage in EAP.
- prior_dist The type of prior distribution. Accepts one of NORMAL, UNIF.
- prior_par Distributional parameters for the prior.
- bound_ML Theta bound for MLE.
- truncate_ML Set TRUE to truncate MLE within bound_ML.
- max_iter Maximum number of Newton-Raphson iterations.
- crit Convergence criterion.
- max_change Maximum change in ML estimates between iterations.
- do_fisher Set TRUE to use Fisher’s method of scoring.

final_theta
A list containing final theta estimation options.
- method The type of estimation. Accepts one of EAP, EB, FB.
- shrinkage_correction Set TRUE to correct for shrinkage in EAP.
- prior_dist The type of prior distribution. Accepts one of NORMAL, UNIF.
- prior_par Distributional parameters for the prior.
- bound_ML Theta bound for MLE.
- truncate_ML Set TRUE to truncate MLE within bound_ML.
- max_iter Maximum number of Newton-Raphson iterations.
- crit Convergence criterion.
- max_change Maximum change in ML estimates between iterations.
- do_fisher Set TRUE to use Fisher’s method of scoring.

theta_grid
A numeric vector. Theta values to represent the continuum.

audit_trail
Set TRUE to generate audit trails.

Examples

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

Create a config_Static object for Static (fixed-form) test assembly.

Usage

createStaticTestConfig(item_selection = NULL, MIP = NULL)

Arguments

item_selection A list containing item selection criteria. This should have the following entries:

- method The type of criteria. Accepts MAXINFO, TIF, TCC.
- info_type The type of information. Accepts FISHER.
- target_location A numeric vector containing the locations of target theta points. (e.g. c(-1, 0, 1))
- 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.
- target_weight A numeric vector containing the weights for each theta location. This should have the same length with target_location. Defaults to a vector of 1s.

MIP A list containing solver options. This should have the following entries:

- solver The type of solver. Accepts lpsymphony, Rsymphony, gurobi, lpSolve, Rglpk.
- verbosity Verbosity level of the solver. Defaults to -2.
- time_limit Time limit in seconds passed onto the solver. Defaults to 60. Used in solvers lpsymphony, Rsymphony, gurobi, Rglpk.
- gap_limit Termination criterion. Gap limit in relative scale passed onto the solver. Defaults to .05. Used in solver gurobi.
- gap_limit_abs Termination criterion. Gap limit in absolute scale passed onto the solver. Defaults to .05. Used in solver lpsymphony, Rsymphony.
- obj_tol Termination criterion. Tolerance on target objective value in absolute difference scale. Defaults to .05. Ignored if method is MAXINFO.

Examples

cfg1 <- createStaticTestConfig(
  list(
    method = "MAXINFO",
    info_type = "FISHER",
    target_location = c(-1, 0, 1),
    target_weight = c(1, 1, 1)
  )
)
An S4 class to represent a single constraint.

Description

An S4 class to represent a single constraint.

Slots

constraint Character. The index of the constraint.

mat A matrix representing the left-hand side weights. Has nc rows.

dir A vector of length nc. Each entry represents a logical operator relating the left-hand side to the right-hand side.

rhs A vector of length nc. Each entry represents the right-hand side of the constraint.

nc Numeric. The number of constraints represented in the constraint set.

suspend TRUE if the constraint is to be turned off.
An S4 class to represent a set of constraints

**Description**

An S4 class to represent a set of constraints.

**Slots**

- **slope** Numeric. A slope parameter value.
- **difficulty** Numeric. A difficulty parameter value.

**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.
- itemattrib_fatigue A data frame containing item attributes.
- constraints_fatigue A list containing 111 constraints.

Also, the following datasets are intended for illustrating expected data structures. See examples below.

- itempool_fatigue_raw Item parameters.
- itemattrib_fatigue_raw Item attributes.
- itemcontent_fatigue_raw Item contents.
- constraints_fatigue_raw Constraints.
- resp_fatigue_raw Raw response data.
Examples

```r
## Write to tempdir() and clean afterwards
f <- file.path(tempdir(), "itempool_fatigue.csv")
write.csv(itempool_fatigue_raw, f, row.names = FALSE)
itempool_fatigue <- loadItemPool(f)
file.remove(f)

f <- file.path(tempdir(), "itemattrib_fatigue.csv")
write.csv(itemattrib_fatigue_raw, f, row.names = FALSE)
itemattrib_fatigue <- loadItemAttrib(f, itempool_fatigue)
file.remove(f)

f <- file.path(tempdir(), "constraints_fatigue.csv")
write.csv(constraints_fatigue_raw, f, row.names = FALSE)
constraints_fatigue <- loadConstraints(f,
  itempool_fatigue, itemattrib_fatigue)
file.remove(f)

## Item contents for use in shiny app
f <- file.path(tempdir(), "itemcontent_fatigue.csv")
write.csv(itemcontent_fatigue_raw, f, row.names = FALSE)
file.remove(f)

## Raw item responses for reference
f <- file.path(tempdir(), "resp_fatigue.csv")
write.table(resp_fatigue_raw, f, row.names = FALSE, col.names = FALSE, sep = ",")
file.remove(f)
```

Description

Stimulus-based example item pool (303 items).

Details

This pool is associated with the following objects:

- `itempool_reading` An `item_pool` object.
- `itemattrib_reading` A data frame containing item attributes.
- `stimattrib_reading` A data frame containing stimulus attributes.
- `constraints_reading` A list containing 18 constraints.

Also, the following datasets are intended for illustrating expected data structures. See examples below.

- `itempool_reading_raw` Item parameters.
### Examples

```r
## Write to tempdir() and clean afterwards
f <- file.path(tempdir(), "itempool_reading.csv")
write.csv(itempool_reading_raw, f, row.names = FALSE)
itempool_reading <- loadItemPool(f)
file.remove(f)

f <- file.path(tempdir(), "itemattrib_reading.csv")
write.csv(itemattrib_reading_raw, f, row.names = FALSE)
itemattrib_reading <- loadItemAttrib(f, itempool_reading)
file.remove(f)

f <- file.path(tempdir(), "stimattrib_reading.csv")
write.csv(stimattrib_reading_raw, f, row.names = FALSE)
stimattrib_reading <- loadStAttrib(f, itemattrib_reading)
file.remove(f)

f <- file.path(tempdir(), "constraints_reading.csv")
write.csv(constraints_reading_raw, f, row.names = FALSE)
constraints_reading <- loadConstraints(f, itempool_reading, itemattrib_reading, stimattrib_reading)
file.remove(f)
```

---

**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.
- **itemattrib_science** A data frame containing item attributes.
- **constraints_science** A list containing 36 constraints.

Also, the following datasets are intended for illustrating expected data structures. See examples below.

- **itempool_science_raw** Item parameters.
- **itemattrib_science_raw** Item attributes.
- **constraints_science_raw** Constraints.
Examples

```r
## Write to tempdir() and clean afterwards
f <- file.path(tempdir(), "itempool_science.csv")
write.csv(itempool_science_raw, f, row.names = FALSE)
itempool_science <- loadItemPool(f)
file.remove(f)

f <- file.path(tempdir(), "itemattrib_science.csv")
write.csv(itemattrib_science_raw, f, row.names = FALSE)
itemattrib_science <- loadItemAttrib(f, itempool_science)
file.remove(f)

f <- file.path(tempdir(), "constraints_science.csv")
write.csv(constraints_science_raw, f, row.names = FALSE)
constraints_science <- loadConstraints(f,
    itempool_science, itemattrib_science)
file.remove(f)
```

---

**EAP**

*Generate expected a posteriori estimates of theta*

**Description**

Generate expected a posteriori estimates of theta.

**Usage**

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

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

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

**Arguments**

- `object` A `test` or a `test_cluster` object.
- `prior` A prior distribution, a numeric vector for a common prior or a matrix for individualized priors.
- `select` A vector of indices identifying the items to subset.
- `reset_prior` Set TRUE to reset the prior distribution for each test when object is of class `test_cluster`.
**eap**

Generate expected a posteriori estimates of theta.

### Usage

eap(object, theta, prior, resp, select = NULL)

```
## S4 method for signature 'item_pool'
eap(object, theta, prior, resp, select = NULL)
```

### Arguments

- **object**
  An `item_pool` object.
- **theta**
  A theta grid.
- **prior**
  A prior distribution, a numeric vector for a common prior or a matrix for individualized priors.
- **resp**
  A numeric matrix of item responses, one row per examinee.
- **select**
  A vector of indices identifying the items to subset.

---

**extract-methods**

Extract

### Description

Extract

### Usage

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

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

### Arguments

- **x**
- **i**
- **j**
- **...**
- **drop**
find_segment  

Find the segment to which each theta value belongs

Description

Find the segment to which each theta value belongs.

Usage

find_segment(segment, x)

Arguments

segment  
A numeric vector of segment cuts.

x  
A numeric vector of theta values.

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)

Arguments

object  
Output from Static or Shadow.

examinee  
Examinee id to display the solution. Used when 'object' is from Shadow.

position  
If supplied, display the item attributes of the shadow test at that item position. If not supplied, display the item attributes of the administered items. Applied when 'object' is from Shadow.

index_only  
If TRUE (default), print the item indexes only. Otherwise, print all item attributes.

Value

Item attributes of solution items.
**info_1pl**

Calculate Fisher information at a single theta (1PL)

**Description**

Calculate the Fisher information at a theta value according to the 1PL model.

**Usage**

```r
info_1pl(x, b)
```

**Arguments**

- `x` Numeric. A single theta value.
- `b` Numeric. A difficulty parameter value.

**References**


**info_2pl**

Calculate Fisher information at a single theta (2PL)

**Description**

Calculate the Fisher information at a theta value according to the 2PL model.

**Usage**

```r
info_2pl(x, a, b)
```

**Arguments**

- `x` Numeric. A single theta value.
- `a` Numeric. A slope parameter value.
- `b` Numeric. A difficulty parameter value.
References


### info_3pl

*Calculate Fisher information at a single theta (3PL)*

#### Description

Calculate the Fisher information at a theta value according to the 3PL model.

#### Usage

```r
test(x, a, b, c)
```

#### Arguments

- **x**: Numeric. A single theta value.
- **a**: Numeric. A slope parameter value.
- **b**: Numeric. A difficulty parameter value.
- **c**: Numeric. A guessing parameter value.

#### References

Calculate Fisher information at a single theta (GPC).

**Description**

Calculate the Fisher information at a theta value according to the generalized partial credit model.

**Usage**

```
info_gpc(x, a, b)
```

**Arguments**

- `x`: Numeric. A single theta value.
- `a`: Numeric. A slope parameter value.

**References**


Calculate Fisher information at a single theta (GR).

**Description**

Calculate the Fisher information at a theta value according to the graded response model.

**Usage**

```
info_gr(x, a, b)
```

**Arguments**

- `x`: Numeric. A single theta value.
- `a`: Numeric. A slope parameter value.

**References**

info_pc  Calculate Fisher information at a single theta (PC)

**Description**

Calculate the Fisher information at a theta value according to the partial credit model.

**Usage**

```r
info_pc(x, b)
```

**Arguments**

- `x`  Numeric. A single theta value.
- `b`  Numeric. A vector of threshold parameter values.

**References**


iparPosteriorSample  Sample item parameter estimates from their posterior distributions

**Description**

Sample item parameter estimates from their posterior distributions.

**Usage**

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

**Arguments**

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

**Examples**

```r
ipar <- iparPosteriorSample(itempool_science, 5)
```
item_1PL-class

An S4 class to represent a 1PL item

Description

An S4 class to represent a 1PL item.

Slots

difficulty Numeric. A difficulty parameter value.

References


Examples

item_1 <- new("item_1PL", difficulty = 0.5)

item_2PL-class

An S4 class to represent a 2PL item

Description

An S4 class to represent a 2PL item.

Slots

slope Numeric. A slope parameter value.
difficulty Numeric. A difficulty parameter value.

References


Examples

item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3PL-class  An S4 class to represent a 3PL item

Description

An S4 class to represent a 3PL item.

Slots

- slope  Numeric. A slope parameter value.
- difficulty  Numeric. A difficulty parameter value.
- guessing  Numeric. A guessing parameter value.

References


Examples

```r
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
```

item_attrib-class  An S4 class to represent a set of constraints.

Description

An S4 class to represent a set of constraints.

Slots

- slope  Numeric. A slope parameter value.
- difficulty  Numeric. A difficulty parameter value.
item_GPC-class

An S4 class to represent a generalized partial credit item

Description

An S4 class to represent a generalized partial credit item.

Slots

slope Numeric. A slope parameter value.
threshold Numeric. A vector of threshold parameter values.
ncat Numeric. The number of response categories.

References


Examples

item_5 <- new("item_GPC", slope = 1.0, threshold = c(-0.5, 0.0, 0.5), ncat = 4)

item_GR-class

An S4 class to represent a graded response item

Description

An S4 class to represent a graded response item.

Slots

slope Numeric. A slope parameter value.
category Numeric. A vector of category boundary values.
ncat Numeric. The number of response categories.

References


Examples

item_6 <- new("item_GR", slope = 1.0, category = c(-2.0, -1.0, 0, 1.0, 2.0), ncat = 6)
item_PC-class

An S4 class to represent a partial credit item

Description
An S4 class to represent a partial credit item.

Slots
- threshold Numeric. A vector of threshold parameter values.
- ncat Numeric. The number of response categories.

References

Examples
```r
item_4 <- new("item_PC", threshold = c(-0.5, 0.5), ncat = 3)
```

item_pool-class

An S4 class to represent an item pool

Description
An S4 class to represent an item pool.

Slots
- ni Numeric. The number of items in the item pool.
- max_cat Numeric. The maximum number of response categories across all items.
- index Numeric. A vector of item indices.
- id Character. A vector of item ids.
- model Numeric. A vector of item model codes (1: item.1pl, 2: item_2PL, 3: item_3PL, 4: item_PC, 5: item_GPC, 6: item_GR).
- NCAT Numeric. A vector of the number of response categories for each item.
- parms A list of item parameters in the pool.
- ipar A matrix of item parameters in the pool.
- se A matrix representing standard errors of the item parameters.
- raw A data.frame containing raw input data.
Description

pool1 + pool2 combines two item_pool objects.

pool1 - pool2 excludes the items in the second item pool from the first. The two item_pool objects must overlap for this to be performed.

pool1 == pool2 tests equality of the two item_pool objects.

pool_cluster1 == pool_cluster2 tests equality of the two pool_cluster objects.

Usage

```r
# S3 method for class 'item_pool'
pool1 + pool2

# S3 method for class 'item_pool'
pool1 - pool2

# S3 method for class 'item_pool'
pool1 == pool2

# S3 method for class 'pool_cluster'
pool_cluster1 == pool_cluster2
```

Arguments

- **pool1**: An item_pool object.
- **pool2**: An item_pool object.
- **pool_cluster1**: A pool_cluster object.
- **pool_cluster2**: A pool_cluster object.

Examples

```r
itempool <- itempool_science + itempool_reading
subitempool <- subsetItemPool(itempool_science, 1:500)
itempool <- itempool_science - subitempool

itempool <- subsetItemPool(itempool_science, 1:500)
subitempool1 <- itempool_science - itempool
subitempool2 <- subsetItemPool(itempool_science, 501:1000)
subitempool1 == subitempool2  # TRUE

cluster1 <- makeItemPoolCluster(c(itempool_science, itempool_reading))
cluster2 <- makeItemPoolCluster(c(cluster1@pools[[1]], cluster1@pools[[2]]))
```
loadConstraints

cluster1 == cluster2  ## TRUE

<table>
<thead>
<tr>
<th>lnHyperPars</th>
<th>Calculate hyperparameters for log-normal distribution</th>
</tr>
</thead>
</table>

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)

<table>
<thead>
<tr>
<th>loadConstraints</th>
<th>Load constraints</th>
</tr>
</thead>
</table>

Description

Read constraints from specified file.

Usage

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

Arguments

file  Character. The name of the file containing specifications for constraints.
pool  An item_pool object.
item_attrib  An item_attrib object containing item attributes. Use loadItemAttrib for this.
st_attrib  (Optional) An st_attrib object containing stimulus attributes. Use loadStAttrib for this.
Details

Use vignette("constraints") for instructions on how to create the constraint file.

Value

A constraints object containing the parsed constraints, to be used in Static and Shadow.

See Also

dataset_science for example usage.

Examples

```r
## Write to tempdir() and clean afterwards
f <- file.path(tempdir(), "itempool_science.csv")
write.csv(itempool_science_raw, f, row.names = FALSE)
itempool_science <- loadItemPool(f)
file.remove(f)

f <- file.path(tempdir(), "itemattrib_science.csv")
write.csv(itemattrib_science_raw, f, row.names = FALSE)
itemattrib_science <- loadItemAttrib(f, itempool_science)
file.remove(f)

f <- file.path(tempdir(), "constraints_science.csv")
write.csv(constraints_science_raw, f, row.names = FALSE)
constraints_science <- loadConstraints(f, itempool_science, itemattrib_science)
file.remove(f)
```

loadItemAttrib

Load item attributes

Description

Read item attributes from specified file.

Usage

loadItemAttrib(file, pool)

Arguments

file Character. The name of the file containing item attributes.

pool An item_pool object. Use loadItemPool for this.
loadItemPool

Load item parameters

Description

Read item parameters from a .csv file or a data.frame and create an item_pool class.

Usage

loadItemPool(file, ipar = NULL, se_file = NULL)

Arguments

file
  File path of a .csv file containing item parameters. The file content should at least include columns ‘ID’ and ‘MODEL’.

ipar
  A data.frame containing the item parameters. If supplied, this argument is used over ‘file’.

se_file
  File path of a .csv file containing standard errors.

Value

An item_pool object.

See Also

dataset_science for example usage.

Examples

## Write to tempdir() and clean afterwards
f <- file.path(tempdir(), "itempool_science.csv")
write.csv(itempool_science_raw, f, row.names = FALSE)
itempool_science <- loadItemPool(f)
file.remove(f)

f <- file.path(tempdir(), "itemattrib_science.csv")
write.csv(itemattrib_science_raw, f, row.names = FALSE)
itemattrib_science <- loadItemAttrib(f, itempool_science)
file.remove(f)
Examples

```r
## Write to tempdir() and clean afterwards
f <- file.path(tempdir(), "itempool_science.csv")
write.csv(itempool_science_raw, f, row.names = FALSE)
itempool_science <- loadItemPool(f)
file.remove(f)
```

## Write to tempdir() and clean afterwards
```r
f <- file.path(tempdir(), "itemattrib_reading.csv")
write.csv(itemattrib_reading_raw, f, row.names = FALSE)
itemattrib_reading <- loadItemAttrib(f, itempool_reading)
file.remove(f)
```

## Write to tempdir() and clean afterwards
```r
f <- file.path(tempdir(), "stimattrib_reading.csv")
write.csv(stimattrib_reading_raw, f, row.names = FALSE)
stimattrib_reading <- loadStAttrib(f, itemattrib_reading)
```

### loadStAttrib

**Load set/stimulus/passage attributes**

**Description**

Read set, stimulus, or passage attributes from specified file.

**Usage**

```r
loadStAttrib(file, item_attrib)
```

**Arguments**

- `file` Character. The name of the file containing stimulus attributes.
- `item_attrib` An `item_attrib` object containing item attributes. Use `loadItemAttrib` for this.

**Value**

A `st_attrib` object containing stimulus attributes.

**See Also**

- `dataset_reading` for example usage.

**Examples**

```r
## Write to tempdir() and clean afterwards
f <- file.path(tempdir(), "itempool_reading.csv")
write.csv(itempool_reading_raw, f, row.names = FALSE)
itempool_reading <- loadItemPool(f)
file.remove(f)
```

```r
f <- file.path(tempdir(), "itemattrib_reading.csv")
write.csv(itemattrib_reading_raw, f, row.names = FALSE)
itemattrib_reading <- loadItemAttrib(f, itempool_reading)
file.remove(f)
```

```r
f <- file.path(tempdir(), "stimattrib_reading.csv")
write.csv(stimattrib_reading_raw, f, row.names = FALSE)
stimattrib_reading <- loadStAttrib(f, itemattrib_reading)
```
makeItemPoolCluster

Create an item pool cluster object

Description

Create a pool_cluster object.

Usage

makeItemPoolCluster(pools, names = NULL)

Arguments

pools A list of item_pool objects.
names An optional vector of item_pool names.

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)
makeTest

Examples

cluster <- makeItemPoolCluster(c(itempool_science, itempool_reading))

declare makeTest Generate a test object

Description

Generate a test object

Usage

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 An information type.
true_theta An optional vector of 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 'pool_cluster,numeric,numeric'
makeTestCluster(object, theta, true_theta)

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An pool_cluster object</td>
</tr>
<tr>
<td>theta</td>
<td>A grid of theta values</td>
</tr>
<tr>
<td>true_theta</td>
<td>An optional vector of true theta values to simulate response data</td>
</tr>
</tbody>
</table>

MLE  Generate maximum likelihood estimates of theta

Description
Generate maximum likelihood estimates of theta.

Usage

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

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

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

**Arguments**

- **object** A `test` object.
- **start_theta** An optional vector of start theta values.
- **max_iter** Maximum number of iterations.
- **crit** Convergence criterion.
- **select** A vector of indices identifying the items to subset.
- **theta_range** A range of theta values: c(minTheta, maxTheta).
- **truncate** Set TRUE to bound MLE to theta_range.
- **max_change** Maximum change between iterations.
- **do_Fisher** Set TRUE to use Fisher’s method of scoring.

**mle**

*Generate maximum likelihood estimates of theta*

**Description**

Generate maximum likelihood estimates of theta.

**Usage**

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

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

### Arguments

- **object**
  - A *item_pool* object.

- **resp**
  - A vector (or matrix) of item responses.

- **start_theta**
  - An optional vector of start theta values.

- **max_iter**
  - Maximum number of iterations.

- **crit**
  - Convergence criterion.

- **select**
  - A vector of indices identifying the items to subset.

- **theta_range**
  - A range of theta values.

- **truncate**
  - Set TRUE to bound MLE to theta_range: c(minTheta, maxTheta).

- **max_change**
  - Maximum change between iterations.

- **do_Fisher**
  - TRUE to use Fisher’s method of scoring.

### Examples

```r
mle(itempool_fatigue, resp_fatigue_raw[10,])
```

---

## Description

Launch Shiny app locally.
Usage

OAT()

app()

Examples

if (interactive()) {
  OAT()
  ## or
  app()
}

Description

output_Shadow

Slots

  simulee_id Numeric. The index of the simulee.
  true_theta Numeric or NULL. True theta value of the simulee if supplied in advance.
  true_theta_segment Numeric or NULL. Which segment the true theta value is in.
  final_theta_est Numeric. The estimated theta after the last administered item.
  final_se_est Numeric. The standard error of estimation after the last administered item.
  administered_item_index Numeric. A vector of item indices administered at each position.
  administered_item_resp Numeric. A vector of item responses at each position.
  administered_item_ncat Numeric. A vector containing the number of categories for each administered item.
  administered_stimulus_index Numeric. A vector of stimulus indices administered at each position.
  shadow_test_refreshed Logical. A vector of logical values indicating whether the shadow test was refreshed before administering an item at each position.
  shadow_test_feasible Logical. A vector of logical values indicating whether a feasible solution to the shadow test was available in each position.
  solve_time Numeric. A vector of values indicating the time taken in obtaining a shadow test.
  interim_theta_est Numeric. A vector containing estimated thetas at each position.
  interim_se_est Numeric. A vector containing standard errors at each position.
  theta_segment_index Numeric. A vector containing which segments the estimated thetas were in at each position.
prior Numeric. A prior distribution.
prior_par Numeric. The hyper parameters for the prior distribution.
posterior Numeric. A posterior distribution.
posterior_sample Numeric. A vector containing MCMC samples.
likelihood Numeric. A likelihood distribution.
shadow_test A list of vectors containing item indices of the shadow test at each position.

---

**plotCAT**

*Draw an audit trail plot*

**Description**

Draw an audit trail plot.

**Usage**

```r
plotCAT(
  object,
  examinee_id = 1,
  min_theta = -5,
  max_theta = 5,
  min_score = 0,
  max_score = 1,
  z_ci = 1.96,
  file_pdf = NULL,
  ...
)
```

---

## S4 method for signature 'list'

```r
plotCAT(
  object,
  examinee_id = 1,
  min_theta = -5,
  max_theta = 5,
  min_score = 0,
  max_score = 1,
  z_ci = 1.96,
  file_pdf = NULL,
  ...
)
```

## S4 method for signature 'output_Shadow'

```r
plotCAT(
  object,
  examinee_id = 1,
```
### plotEligibilityStats

**Draw item eligibility statistics plots**

#### Description

Draw item eligibility statistics plots.

#### Usage

```r
plotEligibilityStats(
  config,
  object = NULL,
  object_no_fading = NULL,
  file = NULL,
  file_no_fading = NULL,
  segment = 1,
  ...
)
```

#### Arguments

- **object**
  
  An output object generated by `Shadow`.

- **examinee_id**
  
  Numeric ID of the examinee to draw the plot.

- **min_theta**
  
  A lower bound of theta.

- **max_theta**
  
  An upper bound of theta.

- **min_score**
  
  A minimum item score.

- **max_score**
  
  A maximum item score.

- **z_ci**
  
  A quantile of the normal distribution for confidence intervals.

- **file_pdf**
  
  If supplied a filename, save as a PDF file.

- **...**
  
  Additional options to be passed on to `pdf()`.

#### Examples

```r
config <- createShadowTestConfig()
true_theta <- rnorm(1)
solution <- Shadow(config, constraints_science, true_theta)
plotCAT(solution, 1)
```
plotExposure

items = c(1), file_pdf = NULL, max_rate = 0.25, discard_first = NULL

Arguments

config A config_Shadow object.
object An object containing eligibility statistics generated by Shadow.
object_no_fading An object containing eligibility statistics generated without fading.
file The filename of an object containing eligibility statistics generated by Shadow.
file_no_fading The filename of an object containing eligibility statistics generated without fading.
segment A theta segment index.
items A vector of item indices to generate the plots.
file_pdf If supplied a filename, save as a PDF file.
max_rate A target item exposure rate.
discard_first A integer identifying the first x simulees to discard as burn-in.

plotExposure  

Draw an item exposure plot

Description

Draw a plot of item exposure rates

Usage

plotExposure(
  object, max_rate = 0.25, theta_segment = "Estimated", color = "blue", color_final = "blue", file_pdf = NULL,
  ...
)

## S4 method for signature 'list'
plotExposure(
  object, max_rate = 0.25,
plotExposureRateBySegment

theta_segment = "estimated",
color = "blue",
color_final = "blue",
file_pdf = NULL,
...
)

Arguments

object An output object generated by Shadow.
max_rate A target exposure rate.
theta_segment True or Estimated theta used to create segments ("Estimated" or "True").
color Color of item-wise exposure rates.
color_final Color of item-wise exposure rates, only counting the items while in the final theta segment as exposed.
file_pdf If supplied a filename, save as a PDF file.
... Additional options to be passed on to pdf().

Examples

ture_theta <- runif(10, min = -3.5, max = 3.5)
resp_science <- makeTest(itempool_science, info_type = "FISHER", true_theta = true_theta)@data
constraints_science2 <- updateConstraints(constraints_science, off = c(14:20, 32:36))
config_science <- createShadowTestConfig(
  MIP = list(solver = "lpSolve"),
  exposure_control = list(method = "ELIGIBILITY")
)
solution <- Shadow(config_science, constraints_science2, true_theta, data = resp_science)
p <- plotExposure(solution)

plotExposureRateBySegment

Draw exposure rate plots by theta segment

Description

Draw exposure rate plots by theta segment.

Usage

plotExposureRateBySegment(
  object,
  config,
  max_rate = 0.25,
plotExposureRateFinal

Arguments

object An output object generated by Shadow.
config A config_Shadow object.
max_rate A target item exposure rate.
file_pdf If supplied a filename, save as a PDF file.
width Width of the graphics device.
height Height of the graphics device.
mfrow Number of multiple figures defined as c(nrow, ncol).

Description

Draw exposure rate plots by final theta segment.

Usage

plotExposureRateFinal(
  object,
  config = NULL,
  max_rate = 0.25,
  theta = "Estimated",
  segment_cut = NULL,
  color = "red",
  file_pdf = NULL,
  width = 7,
  height = 6,
  mfrow = c(2, 4),
  burn = 0,
  retain = NULL
)

Arguments

object An output object generated by Shadow.
config A config_Shadow object.
max_rate A target item exposure rate.
theta  By which theta to base the segments, either "Estimated" or "True".
segment_cut  A vector of cut values defining theta segments.
color  A vector of colors.
file_pdf  If supplied a filename, save as a PDF file.
width  Width of the graphics object.
height  Height of the graphics object.
mfrow  Number of multiple figures defined as c(nrow, ncol).
burn  An integer identifying the first x simulees to discard as burn-in.
retain  An optional vector of indices identifying the simulees to retain.

Examples

true_theta <- runif(10, min = -3.5, max = 3.5)
resp_science <- makeTest(itempool_science, info_type = "FISHER", true_theta = true_theta)@data
constraints_science2 <- updateConstraints(constraints_science, off = c(14:20, 32:36))
config_science <- createShadowTestConfig(
  MIP = list(solver = "LPSOLVE"),
  exposure_control = list(method = "ELIGIBILITY")
)
solution <- Shadow(config_science, constraints_science2, true_theta, data = resp_science)
p <- plotExposureRateFinal(solution, config_science, 0.25)

plotExposureRateFinalFlag

Draw item information plots for flagged items by segment

Description

Draw item information plots for flagged items by segment.

Usage

plotExposureRateFinalFlag(
  object,
  pool,
  theta = seq(-3, 3, 0.1),
  flag_from = 0.4,
  file_pdf = NULL,
  width = 7,
  height = 6,
  color = "red",
  mfrow = c(2, 4)
)
Arguments

- **object**: A list object generated by `plotExposureRateFinal`.
- **pool**: An `item_pool` object.
- **theta**: A theta grid.
- **flag_from**: A flagging criterion.
- **file_pdf**: If supplied a filename, save as a PDF file.
- **width**: Width of the graphics device.
- **height**: Height of the graphics device.
- **color**: Plotting color.
- **mfrow**: Number of multiple figures defined as c(nrow, ncol).

plotInfo

*Draw item information plots*

Description

Draw item information plots.

Usage

```r
plotInfo(
  object,
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  color = "blue",
  file_pdf = NULL,
  width = 7,
  height = 6,
  mfrow = c(2, 4)
)
```

```r
# S4 method for signature 'list'
plotInfo(
  object,
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  color = "blue",
  file_pdf = NULL,
  width = 7,
  height = 6,
```
plotInfo

mfrow = c(2, 4)

## S4 method for signature 'item_pool'
plotInfo(
  object,
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  color = "blue",
  file_pdf = NULL,
  width = 7,
  height = 6,
  mfrow = c(1, 1)
)

## S4 method for signature 'constraints'
plotInfo(
  object,
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
  plot_sum = TRUE,
  select = NULL,
  color = "black",
  file_pdf = NULL,
  width = 7,
  height = 6,
  mfrow = c(1, 1)
)

Arguments

object An item_pool object to draw pool-level or item-level information, or a list from Static to draw test-level information.
theta Theta values for drawing the curve. Default is seq(-3, 3,.1).
info_type Type of information. Currently only accepts FISHER (default).
plot_sum When 'object' is an item_pool object, if TRUE then draw pool-level information, and if FALSE draw item-level information for every item in the pool.
select A vector of indices identifying the items to subset, for when 'object' is an item_pool object.
color The color of the curve.
file_pdf If supplied a filename, save as a PDF file.
width Width of graphics device.
height Width of graphics device.
mfrow Multipanel configurations as c(nrow, ncol).
Examples

```r
subitempool <- subsetItemPool(itempool_science, 1:8)
plotInfo(subitempool)

config <- createStaticTestConfig()
solution <- Static(config, constraints_science)
plotInfo(solution)
```

---

**plotInfoOverlay**

*Overlay item information plots*

**Description**

Overlay item information plots.

**Usage**

```r
plotInfoOverlay(
  object,  # An item_pool object.
  theta,   # A theta grid.
  info_type = "FISHER",  # Type of information.
  select = NULL,  # A vector of indices identifying the items to subset.
  file_pdf = NULL,  # If supplied a filename, save as a PDF file.
  color = "red",  # Plotting color.
  width = 7,  # Width of the graphics device.
  height = 6  # Height of the graphics device.
)
```

**Arguments**

- **object**: An *item_pool* object.
- **theta**: A theta grid.
- **info_type**: Type of information.
- **select**: A vector of indices identifying the items to subset.
- **file_pdf**: If supplied a filename, save as a PDF file.
- **color**: Plotting color.
- **width**: Width of the graphics device.
- **height**: Height of the graphics device.
### plotRMSE

**Draw RMSE plots**

#### Description

Draw RMSE plots.

#### Usage

```r
plotRMSE(
  ...,            # A series of RMSE values.
  title = NULL,   # A plot title.
  legend_title = NULL, # A legend title.
  legend_labels = NULL, # A vector of labels for the series.
  lty_set = NULL,  # A vector of line types for the series.
  col_set = NULL,  # A vector of colors for the series.
  theta = seq(-2, 2, 1)  # A theta grid.
)
```

#### Arguments

- `...`: A series of RMSE values.
- `title`: A plot title.
- `legend_title`: A legend title.
- `legend_labels`: A vector of labels for the series.
- `lty_set`: A vector of line types for the series.
- `col_set`: A vector of colors for the series.
- `theta`: A theta grid.

### plotShadow

**Draw a shadow test chart**

#### Description

Draw a chart of shadow tests constructed for each simulee. The index of a column represents the position of item administration process, and each column represents the item pool.
Usage

plotShadow(
  object,
  examinee_id = 1,
  sort_by_difficulty = FALSE,
  file_pdf = NULL,
  simple = FALSE,
  ...
)

## S4 method for signature 'list'
plotShadow(
  object,
  examinee_id = 1,
  sort_by_difficulty = FALSE,
  file_pdf = NULL,
  simple = FALSE,
  ...
)

Arguments

object An output from `Shadow` function.
examinee_id Numeric ID of the examinee to draw the plot.
sort_by_difficulty Sort the items by difficulty. (not implemented)
file_pdf If supplied a filename, save as a PDF file.
simple If TRUE, simplify the chart by hiding unused items.
... Additional options to be passed on to `pdf()`.

Examples

cfg <- createShadowTestConfig()
true_theta <- rnorm(1)
solution <- Shadow(cfg, constraints_science, true_theta)
plotShadow(solution, 1)
plotShadow(solution, 1, simple = TRUE)

---

**pool_cluster-class**

An S4 class to represent a cluster of item pools

Description

An S4 class to represent a cluster of item pools.
**Slots**

- `np` A scalar to indicate the number of item pools in the cluster.
- `pools` A list of `item_pool` objects.
- `names` A character vector of item pool names of length `np`.

---

**p_1pl**  
**Calculate probability at a single theta (1PL)**

**Description**

Calculate the probability of correct response at a theta value, under the 1PL model.

**Usage**

`p_1pl(x, b)`

**Arguments**

- `x` Numeric. A single theta value.
- `b` Numeric. A difficulty parameter value.

**References**


---

**p_2pl**  
**Calculate probability at a single theta (2PL)**

**Description**

Calculate the probability of correct response at a theta value, under the 2PL model.

**Usage**

`p_2pl(x, a, b)`

**Arguments**

- `x` Numeric. A single theta value.
- `a` Numeric. A slope parameter value.
- `b` Numeric. A difficulty parameter value.
References


p_3pl

*Calculate probability at a single theta (3PL)*

Description

Calculate the probability of correct response at a theta value, under the 3PL model.

Usage

\[
p_\text{3pl}(x, a, b, c)
\]

Arguments

- **x**: Numeric. A single theta value.
- **a**: Numeric. A slope parameter value.
- **b**: Numeric. A difficulty parameter value.
- **c**: Numeric. A guessing parameter value.

References

Calculate probability at a single theta (GPC)

Description
Calculate the probability of correct response at a theta value, under the generalized partial credit model.

Usage
\[ p_{\text{gpc}}(x, a, b) \]

Arguments
- \( x \) Numeric. A single theta value.
- \( a \) Numeric. A slope parameter value.
- \( b \) Numeric. A vector of threshold parameter values.

References

Calculate probability at a single theta (GR)

Description
Calculate the probability of correct response at a theta value, under the graded response model.

Usage
\[ p_{\text{gr}}(x, a, b) \]

Arguments
- \( x \) Numeric. A single theta value.
- \( a \) Numeric. A slope parameter value.
- \( b \) Numeric. A vector of category boundary parameter values.

References
p_pc

*Calculate probability at a single theta (PC)*

**Description**

Calculate the probability of correct response at a theta value, under the partial credit model.

**Usage**

\[
p_{pc}(x, b)
\]

**Arguments**

- **x**
  - Numeric. A single theta value.
- **b**
  - Numeric. A vector of threshold parameter values.

**References**


---

**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**

\[ \text{RMSE}(x, y, \text{conditional} = \text{TRUE}) \]

**Arguments**

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

**runAssembly**

*Run Test Assembly*

**Description**

Perform test assembly with specified configurations. This function is used internally in Static and Shadow.

**Usage**

\[ \text{runAssembly}(\text{config}, \text{constraints}, \text{xdata} = \text{NULL}, \text{objective} = \text{NULL}) \]

**Arguments**

- **config**: A config_Static or a config_Shadow object containing configuration options. Use createStaticTestConfig and createShadowTestConfig for this.
- **constraints**: A list representing optimization constraints. Use loadConstraints for this.
- **xdata**: A list containing extra data to be used in Shadow, representing the constraints for force-including previously administered items.
- **objective**: Information 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


**saveOutput**

*Save or print audit trails*

**Description**

Save or print audit trails for all simulees.

**Usage**

```
saveOutput(object_list, file = NULL)
```

**Arguments**

- `object_list`: A list of output objects generated from STA.
- `file`: An optional file name as a character string to save the output.

**Value**

None

**Shadow**

*Run adaptive test assembly.*

**Description**

Perform adaptive test assembly based on generalized shadow-test approach, with specified configurations.

**Usage**

```
Shadow(
  config,
  constraints = NULL,
  true_theta = NULL,
  data = NULL,
  prior = NULL,
  prior_par = NULL,
  session = NULL
)
```

## S4 method for signature 'config_Shadow'
Shadow(
  config,
  constraints = NULL,
  true_theta = NULL,
  data = NULL,
  prior = NULL,
  prior_par = NULL,
  session = NULL
)

Arguments

config A config_Shadow object.
constraints A list representing optimization constraints. Use loadConstraints for this.
true_theta Numeric. A vector of true theta values to be used in simulation.
data Numeric. A matrix containing item response data.
prior Numeric. A matrix or a vector containing priors.
prior_par Numeric. A vector of parameters for prior distribution.
session Used to communicate with a Shiny session.

References


Examples

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

showConstraints Show constraints

Description

Show constraints. This function is a shortcut to access ‘’ slot.
Usage

showConstraints(constraints)

Arguments

constraints Output from loadConstraints.

Description

An S4 generic and its methods to simulate responses.

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 'pool_cluster,numeric'
simResp(object, theta)

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

Arguments

object An instance of an item class.
theta A vector of theta values.

Value

Simulated responses.

References


Examples

```r
item_1 <- new("item_1PL", difficulty = 0.5) sim_item_1 <- simResp(item_1, seq(-3, 3, 1))
item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5) sim_item_2 <- simResp(item_2, seq(-3, 3, 1))
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2) sim_item_3 <- simResp(item_3, seq(-3, 3, 1))
item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4) sim_item_4 <- simResp(item_4, seq(-3, 3, 1))
item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4) sim_item_5 <- simResp(item_5, seq(-3, 3, 1))
item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4) sim_item_6 <- simResp(item_6, seq(-3, 3, 1)) sim_itempool <- simResp(itempool_science, seq(-3, 3, 1))
```
Static

**Run Static Test Assembly**

**Description**
Perform static (fixed-form) test assembly with specified configurations.

**Usage**

```
Static(config, constraints)
```

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

**Arguments**

- **config**
  A `config_Static` object containing configuration options. Use `createStaticTestConfig` for this.

- **constraints**
  A list representing optimization constraints. Use `loadConstraints` for this.

**Value**
A list containing the following entries:

- **MIP**
  A list containing the result from MIP solver.
  - **solution**
    Solution vector. Each value represents an item. A value of 1 indicates the item was selected.
  - **objval**
    Objective value of the solution.
  - **status**
    Status value indicating whether an optimal solution was found.

- **selected**
  The attributes of the selected items.

- **solver**
  The name of the MIP solver used in the assembly.

- **obj_value**
  Objective value of the solution. Identical to the one above.

- **solve_time**
  The elapsed time in running the solver.

**References**

**Examples**

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

---

**st_attrib-class**

An S4 class to represent a set of constraints.

**Description**

An S4 class to represent a set of constraints.

**Slots**

- **slope** Numeric. A slope parameter value.
- **difficulty** Numeric. A difficulty parameter value.

---

**subsetItemPool**

Create a subset of an item pool object

**Description**

Create a subset of an item_pool object.

**Usage**

subsetItemPool(pool, select = NULL)

**Arguments**

- **pool** An item_pool object.
- **select** A vector of indices identifying the items to subset.

**Examples**

subitempool <- subsetItemPool(itempool_science, 1:100)
subsetTest  

Create a subset of a test object

Description

Create a subset of a test object.

Usage

subsetTest(test, select = NULL)

Arguments

test  
An test object.

select  
A vector of item indices to subset.

Examples

test <- makeTest(itempool_science, seq(-3, 3, 1))
subtest <- subsetTest(test, 1:100)

test-class  
An S4 class to represent a test

Description

An S4 class to represent a test.

Slots

pool  An item_pool object.

theta  A theta grid.

prob  A list of item response probabilities.

info  A matrix of item information values.

true_theta  An optional vector of true theta values.

data  An optional matrix of item responses.
test_cluster-class

An S4 class to represent a test cluster

Description

An S4 class to represent a test cluster from a list of test objects.

Slots

nt Numeric. A scalar to indicate the number of test objects to be clustered.
tests A list test objects.
names Character. A vector of names corresponding to the test objects.

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 of item responses.
cmp 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**

```r
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 an empirical Bayes estimate of theta for one examinee*

**Description**

Calculate an empirical Bayes estimate of theta for one examinee.

**Usage**

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

theta_EB_single

Arguments

- **nx**: The number of MCMC draws.
- **theta_init**: A value for initial estimate of theta.
- **theta_prop**: SD of the proposal distribution.
- **item_parm**: A numeric matrix of item parameters.
- **resp**: A numeric vector of 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).

### Description

Calculate an empirical Bayes estimate of theta for a single item.

### Usage

```r
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**: A value for initial estimate of theta.
- **theta_prop**: SD of the proposal distribution.
- **item_parm**: A numeric matrix of item parameters.
- **resp**: A numeric vector of item responses.
- **ncat**: A numeric vector of the number of response categories by item.
theta_FB

theta_FB

<table>
<thead>
<tr>
<th>theta_FB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate a fully Bayesian estimate of theta for an examinee</td>
</tr>
</tbody>
</table>

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
theta_init
theta_prop
items_list
item_init
resp
ncat
model
prior
prior_parm

The number of MCMC draws.
A value for initial estimate of theta.
SD of the proposal distribution.
A list of item_parm matrices.
A matrix of item parameter estimates (one row per item).
A numeric vector of item responses.
A numeric vector of the number of response categories by item.
A numeric vector indicating the IRT models of each item (1: 1PL, 2: 2PL, 3: 3PL, 4: PC, 5: GPC, 6: GR).
The type of prior distribution (1: normal, 2: uniform).
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 of 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).
Description
Update the constraints list

Usage
updateConstraints(object, on = NULL, off = NULL)

Arguments
object  A constraints object from loadConstraints.
on  a vector of constraints index to turn on.
off  a vector of constraints index to turn off.

Value
An updated constraints object, to be used in Static and Shadow.

Examples
constraints_science2 <- updateConstraints(constraints_science, off = 32:36)
constraints_science3 <- updateConstraints(constraints_science, on = 32:36)
Index

*Topic datasets
  dataset_fatigue, 39
dataset_reading, 40
dataset_science, 41
+.item_pool (item_pool.operators), 53
-.item_pool (item_pool.operators), 53
==.item_pool (item_pool.operators), 53
==.pool_cluster (item_pool.operators), 53
[,item_pool,ANY,ANY,ANY-method
  (extract-methods), 43
[,test,ANY,ANY,ANY-method
  (extract-methods), 43
addTrans, 4
app (OAT), 62
array_info_1pl, 5
array_info_2pl, 5
array_info_3pl, 6
array_info_gpc, 7
array_info_gr, 7
array_info_pc, 8
array_p_1pl, 8
array_p_2pl, 9
array_p_3pl, 9
array_p_gpc, 10
array_p_gr, 10
array_p_pc, 11
buildConstraints, 11

calc_info, 27
calc_info_EB, 28
calc_info_FB, 28
calc_info_matrix, 29
calc_likelihood, 29
calc_likelihood_function, 30
calc_log_likelihood, 30
calc_log_likelihood_function, 31
calc_MI_FB, 31
calc_posterior, 32
calc_posterior_function, 32
calc_posterior_single, 33
calcDerivative, 12
calcDerivative,item_1PL,numeric-method
  (calcDerivative), 12
calcDerivative,item_2PL,numeric-method
  (calcDerivative), 12
calcDerivative,item_3PL,numeric-method
  (calcDerivative), 12
calcDerivative,item_GPC,numeric-method
  (calcDerivative), 12
calcDerivative,item_GR,numeric-method
  (calcDerivative), 12
calcDerivative,item_PC,numeric-method
  (calcDerivative), 12
calcDerivative2, 14
calcDerivative2,item_1PL,numeric-method
  (calcDerivative2), 14
calcDerivative2,item_2PL,numeric-method
  (calcDerivative2), 14
calcDerivative2,item_3PL,numeric-method
  (calcDerivative2), 14
calcDerivative2,item_GPC,numeric-method
  (calcDerivative2), 14
calcDerivative2,item_GR,numeric-method
  (calcDerivative2), 14
calcDerivative2,item_PC,numeric-method
  (calcDerivative2), 14
calcEscore, 16
calcEscore,item_1PL,numeric-method
(calcEscore), 16
(calcEscore, item_2PL, numeric-method
(calcEscore), 16
(calcEscore, item_3PL, numeric-method
(calcEscore), 16
(calcEscore, item_GPC, numeric-method
(calcEscore), 16
(calcEscore, item_GR, numeric-method
(calcEscore), 16
(calcEscore, item_PC, numeric-method
(calcEscore), 16
(calcEscore, item_pool, numeric-method
(calcEscore), 16
(calcEscore, pool_cluster, numeric-method
(calcEscore), 16
(calcFisher, 17
(calcFisher, item_1PL, numeric-method
(calcFisher), 17
(calcFisher, item_2PL, numeric-method
(calcFisher), 17
(calcFisher, item_3PL, numeric-method
(calcFisher), 17
(calcFisher, item_GPC, numeric-method
(calcFisher), 17
(calcFisher, item_GR, numeric-method
(calcFisher), 17
(calcFisher, item_PC, numeric-method
(calcFisher), 17
(calcFisher, item_pool, numeric-method
(calcFisher), 17
(calcFisher, pool_cluster, numeric-method
(calcFisher), 17
(calcHessian, 19
(calcHessian, item_1PL, numeric-method
(calcHessian), 19
(calcHessian, item_2PL, numeric-method
(calcHessian), 19
(calcHessian, item_GPC, numeric-method
(calcHessian), 19
(calcHessian, item_GR, numeric-method
(calcHessian), 19
(calcHessian, item_PC, numeric-method
(calcHessian), 19
(calcHessian, item_pool, numeric-method
(calcHessian), 19
(calcHessian, pool_cluster, numeric-method
(calcHessian), 19
(calcJacobian, 21
(calcJacobian, item_1PL, numeric-method
(calcJacobian), 21
(calcJacobian, item_2PL, numeric-method
(calcJacobian), 21
(calcJacobian, item_3PL, numeric-method
(calcJacobian), 21
(calcJacobian, item_GPC, numeric-method
(calcJacobian), 21
(calcJacobian, item_GR, numeric-method
(calcJacobian), 21
(calcJacobian, item_PC, numeric-method
(calcJacobian), 21
(calcJacobian, item_pool, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-list-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-list-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-list-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-list-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-list-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-list-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-list-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-method
(calcJacobian), 21
(calcJacobian, pool_cluster, numeric-list-method
(calcJacobian), 21
calcJacobian, pool_cluster, numeric-method (calcJacobian), 21
calcLocation, 23
calcLocation, item_1PL, numeric-method (calcLocation), 23
calcLocation, item_1PL-method (calcLocation), 23
calcLocation, item_2PL, numeric-method (calcLocation), 23
calcLocation, item_2PL-method (calcLocation), 23
calcLocation, item_3PL, numeric-method (calcLocation), 23
calcLocation, item_3PL-method (calcLocation), 23
calcLocation, item_GPC, numeric-method (calcLocation), 23
calcLocation, item_GPC-method (calcLocation), 23
calcLocation, item_GR, numeric-method (calcLocation), 23
calcLocation, item_GR-method (calcLocation), 23
calcLocation, item_PC, numeric-method (calcLocation), 23
calcLocation, item_PC-method (calcLocation), 23
calcLocation, pool_cluster, numeric-method (calcLocation), 23
calcLocation, pool_cluster-method (calcLocation), 23
calcProb, 25
calcProb, item_1PL, numeric-method (calcProb), 25
calcProb, item_2PL, numeric-method (calcProb), 25
calcProb, item_3PL, numeric-method (calcProb), 25
calcProb, item_GPC, numeric-method (calcProb), 25
calcProb, item_GR, numeric-method (calcProb), 25
calcProb, item_PC, numeric-method (calcProb), 25
calcProb, pool_cluster, numeric-method (calcProb), 25
calcRP, 26
checkConstraints, 34
config_Shadow, 34, 66, 68, 79, 81
config_Shadow-class, 34
config_Static, 37, 79, 84
config_Static-class, 37
constraint-class, 38
constraints, 34, 92
constraints-class, 39
constraints_fatigue (dataset_fatigue), 39
constraints_fatigue_raw (dataset_fatigue), 39
constraints_reading (dataset_reading), 40
constraints_reading_raw (dataset_reading), 40
constraints_science (dataset_science), 41
constraints_science_raw (dataset_science), 41
createShadowTestConfig, 79
createShadowTestConfig (config_Shadow-class), 34
createStaticTestConfig, 79, 84
createStaticTestConfig (config_Static-class), 37
dataset_fatigue, 39
dataset_reading, 40, 57
dataset_science, 41, 55, 56
EAP, 42
eap, 43
eap, item_pool-method (eap), 43
EAP, test-method (EAP), 42
EAP, test_cluster-method (EAP), 42
extract-methods, 43
find_segment, 44
getSolution, 44
getSolution, list-method (getSolution), 44
info_1pl, 45
INDEX

info_2pl, 45
info_3pl, 46
info_gpc, 47
info_gr, 47
info_pc, 48
iparPosteriorSample, 48
item_1PL-class, 49
item_2PL-class, 49
item_3PL-class, 50
item_attrib, 56, 57
item_attrib-class, 50
item_GPC-class, 51
item_GR-class, 51
item_PC-class, 52
item_pool, 27, 39–41, 43, 48, 53, 55, 56, 58, 59, 62, 70–72, 85, 86
item_pool-class, 52
item_pool.operators, 53
itemattrib_fatigue (dataset_fatigue), 39
itemattrib_fatigue_raw (dataset_fatigue), 39
itemattrib_reading (dataset_reading), 40
itemattrib_reading_raw (dataset_reading), 40
itemattrib_science (dataset_science), 41
itemattrib_science_raw (dataset_science), 41
itemcontent_fatigue_raw (dataset_fatigue), 39
itempool_fatigue (dataset_fatigue), 39
itempool_fatigue_raw (dataset_fatigue), 39
itempool_reading (dataset_reading), 40
itempool_reading_raw (dataset_reading), 40
itempool_science (dataset_science), 41
itempool_science_raw (dataset_science), 41
lnHyperPars, 54
loadConstraints, 34, 54, 79, 81, 82, 84, 92
loadItemAttrib, 54, 55, 57
loadItemPool, 12, 55, 56
loadStAttrib, 54, 57
logitHyperPars, 58
makeItemPoolCluster, 58
makeTest, 59
makeTest, item_pool-method (makeTest), 59
makeTestCluster, 60
makeTestCluster, pool_cluster, numeric, list-method (makeTestCluster), 60
makeTestCluster, pool_cluster, numeric, numeric-method (makeTestCluster), 60
MLE, 60
mle, 61
mle, item_pool-method (mle), 61
MLE, test-method (MLE), 60
MLE, test_cluster-method (MLE), 60
OAT, 62
output_Shadow-class, 63
p_1pl, 75
p_2pl, 75
p_3pl, 76
p_gpc, 77
p_gr, 77
p_pc, 78
plotCAT, 64
plotCAT, list-method (plotCAT), 64
plotCAT, output_Shadow-method (plotCAT), 64
plotEligibilityStats, 65
plotExposure, 66
plotExposure, list-method (plotExposure), 66
plotExposureRateBySegment, 67
plotExposureRateFinal, 68, 70
plotExposureRateFinalFlag, 69
plotInfo, 70
plotInfo, constraints-method (plotInfo), 70
plotInfo, item_pool-method (plotInfo), 70
plotInfo, list-method (plotInfo), 70
plotInfoOverlay, 72
plotRMSE, 73
plotShadow, 73
plotShadow, list-method (plotShadow), 73
pool_cluster, 53, 58, 60
pool_cluster-class, 74
RE, 78
resp_fatigue_raw (dataset_fatigue), 39
RMSE, 79
runAssembly, 79
saveOutput, 80
INDEX

Shadow, 12, 34, 44, 55, 65–68, 74, 79, 80, 92
Shadow, config_Shadow-method (Shadow), 80
showConstraints, 81
simResp, 82
simResp, item_1PL, numeric-method
    (simResp), 82
simResp, item_2PL, numeric-method
    (simResp), 82
simResp, item_3PL, numeric-method
    (simResp), 82
simResp, item_GPC, numeric-method
    (simResp), 82
simResp, item_GR, numeric-method
    (simResp), 82
simResp, item_PC, numeric-method
    (simResp), 82
simResp, item_pool, numeric-method
    (simResp), 82
simResp, pool_cluster, list-method
    (simResp), 82
simResp, pool_cluster, numeric-method
    (simResp), 82
st_attrib, 57
st_attrib-class, 85
Static, 12, 44, 55, 71, 79, 84, 92
Static, config_Static-method (Static), 84
stimattrib_reading (dataset_reading), 40
stimattrib_reading_raw
    (dataset_reading), 40
subsetItemPool, 85
subsetTest, 86
test, 42, 59, 61, 86, 87
test-class, 86
test_cluster, 42, 60
test_cluster-class, 87
theta_EAP, 87
theta_EAP_matrix, 88
theta_EB, 88
theta_EB_single, 89
theta_FB, 90
theta_FB_single, 91
updateConstraints, 92