

# Package ‘TestDesign’

October 2, 2020

**Type** Package

**Title** Optimal Test Design Approach to Fixed and Adaptive Test Construction

**Version** 1.1.3

**Date** 2020-10-01

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

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

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

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

**License** GPL (>= 2)

**Depends** R (>= 2.10)

**biocViews**

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

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

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 7.1.1

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr

**Collate** 'RcppExports.R' 'import.R' 'extensions.R' 'item\_class.R'  
 'item\_functions.R' 'loading\_functions.R' 'static\_class.R'  
 'shadow\_class.R' 'item\_pool\_operators.R'  
 'item\_attr\_operators.R' 'st\_attr\_operators.R'  
 'constraints\_operators.R' 'static\_functions.R'  
 'shadow\_functions.R' 'bayes\_functions.R'  
 'constraint\_functions.R' 'cpp\_documents.R' 'datasets.R'  
 'eligibility\_functions.R' 'exposure\_control\_functions.R'  
 'solver\_functions.R' 'helper\_functions.R'  
 'item\_pool\_cluster\_operators.R' 'other\_functions.R'  
 'plot\_functions.R' 'summary\_class.R' 'print\_functions.R'  
 'runshiny.R' 'summary\_functions.R' 'show\_functions.R'  
 'test\_operators.R' 'theta\_functions.R' 'xdata\_functions.R'

**NeedsCompilation** yes

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

**Repository** CRAN

**Date/Publication** 2020-10-01 23:20:12 UTC

## R topics documented:

app . . . . .	4
buildConstraints . . . . .	5
calcEScore . . . . .	6
calcFisher . . . . .	8
calcHessian . . . . .	11
calcJacobian . . . . .	13
calcLocation-methods . . . . .	15
calcLogLikelihood . . . . .	17
calcProb-methods . . . . .	18
calc_info . . . . .	21
calc_info_EB . . . . .	23
calc_info_FB . . . . .	23
calc_likelihood . . . . .	24
calc_MI_FB . . . . .	26
calc_posterior . . . . .	26
calc_posterior_function . . . . .	27
calc_posterior_single . . . . .	28
checkConstraints . . . . .	28
config_Shadow-class . . . . .	29
config_Static-class . . . . .	32
constraint-class . . . . .	34
constraints-class . . . . .	34
constraints-operators . . . . .	35
dataset_bayes . . . . .	36
dataset_fatigue . . . . .	37

dataset_reading	38
dataset_science	38
eap	39
find_segment	40
getSolution	41
getSolutionAttributes	42
info_1pl	43
iparPosteriorSample	45
item-classes	45
item_attrib-class	46
item_attrib-operators	47
item_pool-class	48
item_pool-operators	49
item_pool_cluster-class	50
lnHyperPars	51
loadConstraints	51
loadItemPool	52
logitHyperPars	54
makeItemPoolCluster	54
makeTest	55
makeTestCluster	56
mle	56
mlef	58
output_Shadow-class	60
output_Shadow_all-class	61
output_Static-class	62
plot	63
plotCAT	66
plotExposure	68
plotInfo	70
plotShadow	72
print	73
p_1pl	75
RE	77
RMSE	77
runAssembly	78
saveOutput	79
Shadow	79
show	81
showConstraints	82
simResp	83
Static	85
st_attrib-class	86
st_attrib-operators	87
summary	88
summary-classes	89
test-class	89
TestDesign	89

test_cluster-class . . . . .	90
test_operators . . . . .	90
theta_EAP . . . . .	91
theta_EAP_matrix . . . . .	91
theta_EB . . . . .	92
theta_FB . . . . .	94
theta_FB_single . . . . .	95
toggleConstraints . . . . .	96
updateConstraints . . . . .	96

**Index****98**


---

app *Open TestDesign app*

---

**Description**

`app` and `OAT` are aliases of `TestDesign`.

**Usage**

```
app()
```

```
OAT()
```

**Details**

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

**Examples**

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

## End(Not run)
```

---

buildConstraints	<i>Build constraints (shortcut to other loading functions)</i>
------------------	--

---

## Description

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

## Usage

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

## Arguments

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

## Value

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

## Examples

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

```

constraints_science <- buildConstraints(constraints_science_data,
  itempool_science_data, itemattrib_science_data)
constraints_reading <- buildConstraints(constraints_reading_data,
  itempool_reading_data, itemattrib_reading_data, stimattrib_reading_data)

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

```

---

calcEscore

*Calculate expected scores*


---

## Description

`calcEscore` is a function to calculate expected scores.

## Usage

```

calcEscore(object, theta)

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

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

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

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

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

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

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

```

```
## S4 method for signature 'item_1PL,matrix'  
calcEscore(object, theta)  
  
## S4 method for signature 'item_2PL,matrix'  
calcEscore(object, theta)  
  
## S4 method for signature 'item_3PL,matrix'  
calcEscore(object, theta)  
  
## S4 method for signature 'item_PC,matrix'  
calcEscore(object, theta)  
  
## S4 method for signature 'item_GPC,matrix'  
calcEscore(object, theta)  
  
## S4 method for signature 'item_GR,matrix'  
calcEscore(object, theta)  
  
## S4 method for signature 'item_pool,matrix'  
calcEscore(object, theta)  
  
## S4 method for signature 'item_pool_cluster,numeric'  
calcEscore(object, theta)
```

### Arguments

**object**            an [item](#) or an [item\\_pool](#) object.  
**theta**            theta values to use.

### Value

**item object:** [calcEscore](#) a vector containing expected score of the item at the theta values.

**item\_pool object:** [calcEscore](#) returns a vector containing the pool-level expected score at the theta values.

### References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

### Examples

```

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

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

```

---

calcFisher

*Calculate Fisher information*

---

### Description

`calcFisher` is a function to calculate Fisher information.

### Usage

```

calcFisher(object, theta)

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

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

```



```
## S4 method for signature 'item_3PL,numeric'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_PC,numeric'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_GPC,numeric'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_GR,numeric'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_pool,numeric'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_1PL,matrix'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_2PL,matrix'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_3PL,matrix'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_PC,matrix'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_GPC,matrix'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_GR,matrix'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_pool,matrix'  
calcFisher(object, theta)  
  
## S4 method for signature 'item_pool_cluster,numeric'  
calcFisher(object, theta)
```

### Arguments

object            an [item](#) or an [item\\_pool](#) object.  
theta            theta values to use.

### Value

**item object:** [calcFisher](#) returns a  $(nq, I)$  matrix of information values.

**item\_pool object:** `calcProb` returns a  $(nq, ni)$  matrix of information values.

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

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

## References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

```

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

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

```

```
info_pool <- calcFisher(itempool_science, seq(-3, 3, 1))
```

---

calcHessian	<i>Calculate second derivative of log-likelihood</i>
-------------	--

---

## Description

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

## Usage

```
calcHessian(object, theta, resp)

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

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

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

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

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

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

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

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

## Arguments

object	an <code>item</code> or an <code>item_pool</code> object.
theta	theta values to use.
resp	the response data to use. This must be a single value for an <code>item</code> , or a length <i>ni</i> vector for an <code>item_pool</code> .

## Details

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

## Value

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

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

## References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

```

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

h_item_1 <- calcHessian(item_1, seq(-3, 3, 1), 0)

```

```

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

```

---

calcJacobian

*Calculate first derivative of log-likelihood*


---

## Description

[calcJacobian](#) is a function to calculate the first derivative of the log-likelihood function.

## Usage

```

calcJacobian(object, theta, resp)

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

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

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

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

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

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

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

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

```

**Arguments**

object	an <code>item</code> or an <code>item_pool</code> object.
theta	theta values to use.
resp	the response data to use.

**Value**

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

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

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

**References**

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

**Examples**

```

item_1 <- new("item_1PL", difficulty = 0.5)
item_2 <- new("item_2PL", slope = 1.0, difficulty = 0.5)
item_3 <- new("item_3PL", slope = 1.0, difficulty = 0.5, guessing = 0.2)
item_4 <- new("item_PC", threshold = c(-1, 0, 1), ncat = 4)

```

```
item_5 <- new("item_GPC", slope = 1.2, threshold = c(-0.8, -1.0, 0.5), ncat = 4)
item_6 <- new("item_GR", slope = 0.9, category = c(-1, 0, 1), ncat = 4)

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

---

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

---

## Description

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

## Usage

```
calcLocation(object)

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

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

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

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

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

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

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

**Arguments**

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

**Value**

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

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

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

**References**

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

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

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

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

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

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

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

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

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

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

**Examples**

```

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

loc_item_1 <- calcLocation(item_1)
loc_item_2 <- calcLocation(item_2)

```



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

---

calcLogLikelihood      *Calculate log-likelihood*

---

### Description

`calcLogLikelihood` is a function to calculate log-likelihood values.

### Usage

```
calcLogLikelihood(object, theta, resp)

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

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

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

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

### Arguments

object	an <code>item_pool</code> object.
theta	theta values to use.
resp	the response data to use.

### Value

`calcLogLikelihood` returns values of log-likelihoods.

### References

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

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

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

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

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

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

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

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

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

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

## Examples

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

---

calcProb-methods	<i>Calculate item response probabilities</i>
------------------	--

---

## Description

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

## Usage

```
calcProb(object, theta)

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

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

```

### Arguments

object            an [item](#) or an [item\\_pool](#) object.  
theta             theta values to use.

### Value

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

**item\_pool object:** [calcProb](#) returns a length  $ni$  list, each containing a matrix of probability values.

- notations**
- $nq$  denotes the number of theta values.
  - $ncat$  denotes the number of response categories.
  - $ni$  denotes the number of items in the [item\\_pool](#) object.

## References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

```

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

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

```

---

calc_info	<i>Calculate Fisher information (multiple items)</i>
-----------	--

---

### Description

calc\_info and calc\_info\_matrix are functions to calculate Fisher information. These functions are designed for multiple items.

### Usage

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

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

### Arguments

x	the theta value. This must be a column vector in matrix form for array_info_* functions.
item_parm	a matrix containing item parameters. Each row represents each item.
ncat	a vector containing the number of response categories of each item.
model	a vector indicating item models of each item, using <ul style="list-style-type: none"> <li>• 1: 1PL model</li> <li>• 2: 2PL model</li> <li>• 3: 3PL model</li> <li>• 4: PC model</li> <li>• 5: GPC model</li> <li>• 6: GR model</li> </ul>

### Details

calc\_info accepts a single theta value, and calc\_info\_matrix accepts multiple theta values. Currently supports unidimensional models.

### References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

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

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

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

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

```
array_info_gr(x, 2, c(0, 2))
```

---

calc_info_EB	<i>Calculate the Fisher information using empirical Bayes</i>
--------------	---

---

### Description

Calculate the Fisher information using empirical Bayes.

### Usage

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

### Arguments

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

---

calc_info_FB	<i>Calculate the Fisher information using full Bayesian</i>
--------------	---

---

### Description

Calculate the Fisher information using full Bayesian.

### Usage

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

### Arguments

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

---

calc_likelihood	<i>Calculate likelihoods</i>
-----------------	------------------------------

---

### Description

calc\_likelihood and calc\_likelihood\_function are functions to calculate likelihoods.

### Usage

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

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

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

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

### Arguments

x, theta_grid	the theta value. This must be a column vector in matrix form for calc_*_function functions.
item_parm	a matrix containing item parameters. Each row represents each item.
resp	a vector containing responses on each item.
ncat	a vector containing the number of response categories of each item.
model	a vector indicating item models of each item, using <ul style="list-style-type: none"> <li>• 1: 1PL model</li> <li>• 2: 2PL model</li> <li>• 3: 3PL model</li> <li>• 4: PC model</li> <li>• 5: GPC model</li> <li>• 6: GR model</li> </ul>
prior	an integer indicating the type of prior distribution, using <ul style="list-style-type: none"> <li>• 1: normal distribution</li> <li>• 2: uniform distribution</li> </ul>
prior_parm	a vector containing parameters for the prior distribution.



## Details

calc\_log\_likelihood and calc\_log\_likelihood\_function are functions to calculate log likelihoods.

These functions are designed for multiple items.

calc\_\* functions accept a single theta value, and calc\_\*\_function functions accept multiple theta values.

Currently supports unidimensional models.

## References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

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

```

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

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

theta_grid <- matrix(seq(-3, 3, .1))
l <- calc_likelihood_function(theta_grid, item_parm, resp, ncat, model)
ll <- calc_log_likelihood_function(theta_grid, item_parm, resp, ncat, model, 2, NA)
all(log(l) == ll)

```

---

calc\_MI\_FB

*Calculate the mutual information using full Bayesian*


---

### Description

Calculate the mutual information using full Bayesian.

### Usage

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

### Arguments

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

---

calc\_posterior

*Calculate a posterior value of theta*


---

### Description

Calculate a posterior value of theta.

### Usage

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

**Arguments**

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

---

 calc\_posterior\_function

*Calculate a posterior distribution of theta*


---

**Description**

Calculate a posterior distribution of theta.

**Usage**

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

**Arguments**

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

---

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

---

### Description

Calculate a posterior value of theta for a single item.

### Usage

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

### Arguments

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

---

`checkConstraints` *Check the consistency of constraints and item usage*

---

### Description

Check the consistency of constraints and item usage.

### Usage

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

### Arguments

<code>constraints</code>	A <a href="#">constraints</a> object generated by <a href="#">loadConstraints</a> .
<code>usage_matrix</code>	A matrix of item usage data from <a href="#">Shadow</a> .
<code>true_theta</code>	A vector of true theta values.

---

config\_Shadow-class    *Create a config\_Shadow object*

---

## Description

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

## Usage

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

## Arguments

`item_selection` a named list containing item selection criteria.

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

`content_balancing`

a named list containing content balancing options.

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

`MIP`

a named list containing solver options.

- `solver` the type of solver. Accepts `lpsymphony`, `Rsymphony`, `gurobi`, `lpSolve`, `Rglpk`. (default = LPSOLVE)
- `verbosity` verbosity level of the solver. (default = -2)
- `time_limit` time limit in seconds. Used in solvers `lpsymphony`, `Rsymphony`, `gurobi`, `Rglpk`. (default = 60)
- `gap_limit` search termination criterion. Gap limit in relative scale passed onto the solver. Used in solver `gurobi`. (default = .05)

- `gap_limit_abs` search termination criterion. Gap limit in absolute scale passed onto the solver. Used in solvers `lpsymphony`, `Rsymphony`. (default =  $0.05$ )
  - `retry` number of times to retry running the solver if the solver returns no solution. Some solvers incorrectly return no solution even when a solution exists. This is the number of attempts to verify that the problem is indeed infeasible in such cases. Set to  $0$  to not retry. (default = 5)
- `MCMC` a named list containing Markov-chain Monte Carlo configurations for obtaining posterior samples.
- `burn_in` the number of chains from the start to discard. (default = 100)
  - `post_burn_in` the number of chains to use after discarding the first `burn_in` chains. (default = 500)
  - `thin` thinning interval to apply. 1 represents no thinning. (default = 1)
  - `jump_factor` the jump factor to use. 1 represents no jumping. (default = 1)
- `refresh_policy` a named list containing the refresh policy for when to obtain a new shadow test.
- `method` the type of policy. Accepts `ALWAYS`, `POSITION`, `INTERVAL`, `THRESHOLD`, `INTERVAL-THRESHOLD`. (default = `ALWAYS`)
  - `interval` used in methods `INTERVAL`, `INTERVAL-THRESHOLD`. Set to 1 to refresh at each position, 2 to refresh at every two positions, and so on. (default = 1)
  - `threshold` used in methods `THRESHOLD`, `INTERVAL-THRESHOLD`. The absolute change in between interim theta estimates to trigger the refresh. (default =  $0.1$ )
  - `position` used in methods `POSITION`. Item positions to trigger the refresh. (default = 1)
- `exposure_control` a named list containing exposure control settings.
- `method` the type of exposure control method. Accepts `NONE`, `ELIGIBILITY`, `BIGM`, `BIGM-BAYESIAN`. (default = `ELIGIBILITY`)
  - `M` used in methods `BIGM`, `BIGM-BAYESIAN`. the Big M penalty to use on item information.
  - `max_exposure_rate` target exposure rates for each segment. (default = `rep(0.25, 7)`)
  - `acceleration_factor` the acceleration factor to apply. (default = 1)
  - `n_segment` the number of theta segments to use. (default = 7)
  - `first_segment` (optional) the theta segment assumed at the beginning of test for all participants.
  - `segment_cut` theta segment cuts. (default = `c(-Inf, seq(-2.5, 2.5, 1), Inf)`)
  - `initial_eligibility_stats` (optional) initial eligibility statistics to use.
  - `fading_factor` the fading factor to apply. (default = `.999`)
  - `diagnostic_stats` set to `TRUE` to generate segment-wise diagnostic statistics. (default = `FALSE`)
- `stopping_criterion` a named list containing stopping criterion.

- method the type of stopping criterion. Accepts FIXED. (default = FIXED)
  - test\_length test length.
  - min\_ni the maximum number of items to administer.
  - max\_ni the minimum number of items to administer.
  - se\_threshold standard error threshold. Item administration is stopped when theta estimate standard error becomes lower than this value.
- interim\_theta a named list containing interim theta estimation options.
- method the type of estimation. Accepts EAP,MLE,MLEF,EB,FB. (default = EAP)
  - shrinkage\_correction set TRUE to apply shrinkage correction. Used when method is EAP. (default = FALSE)
  - prior\_dist the type of prior distribution. Accepts NORMAL,UNIFORM. (default = NORMAL)
  - prior\_par distribution parameters for prior\_dist. (default =  $c(0,1)$ )
  - bound\_ML theta bound in  $c(\text{lower\_bound},\text{upper\_bound})$  format. Used when method is MLE. (default = -4, 4)
  - truncate\_ML set TRUE to truncate ML estimate within bound\_ML. (default = FALSE)
  - max\_iter maximum number of Newton-Raphson iterations. Used when method is MLE. (default = 50)
  - crit convergence criterion. Used when method is MLE. (default =  $1e-03$ )
  - max\_change maximum change in ML estimates between iterations. Changes exceeding this value is clipped to this value. Used when method is MLE. (default = 1.0)
  - do\_Fisher set TRUE to use Fisher's method of scoring. Used when method is MLE. (default = TRUE)
  - fence\_slope slope parameter to use for method = 'MLEF'. This must have two values in total, for the lower and upper bound item respectively. Use one value to use the same value for both bounds. (default = 5)
  - fence\_difficulty difficulty parameters to use for method = 'MLEF'. This must have two values in total, for the lower and upper bound item respectively. (default =  $c(-5,5)$ )
- final\_theta a named list containing final theta estimation options.
- method the type of estimation. Accepts EAP,MLE,MLEF,EB,FB. (default = EAP)
  - shrinkage\_correction set TRUE to apply shrinkage correction. Used when method is EAP. (default = FALSE)
  - prior\_dist the type of prior distribution. Accepts NORMAL,UNIFORM. (default = NORMAL)
  - prior\_par distribution parameters for prior\_dist. (default =  $c(0,1)$ )
  - bound\_ML theta bound in  $c(\text{lower\_bound},\text{upper\_bound})$  format. Used when method is MLE. (default = -4, 4)
  - truncate\_ML set TRUE to truncate ML estimate within bound\_ML. (default = FALSE)

- `max_iter` maximum number of Newton-Raphson iterations. Used when method is MLE. (default = 50)
  - `crit` convergence criterion. Used when method is MLE. (default =  $1e-03$ )
  - `max_change` maximum change in ML estimates between iterations. Changes exceeding this value is clipped to this value. Used when method is MLE. (default = 1.0)
  - `do_Fisher` set TRUE to use Fisher's method of scoring. Used when method is MLE. (default = TRUE)
  - `fence_slope` slope parameter to use for method = 'MLEF'. This must have two values in total, for the lower and upper bound item respectively. Use one value to use the same value for both bounds. (default = 5)
  - `fence_difficulty` difficulty parameters to use for method = 'MLEF'. This must have two values in total, for the lower and upper bound item respectively. (default = `c(-5, 5)`)
- `theta_grid`      the theta grid to use as quadrature points.
- `audit_trail`     set TRUE to plot audit trails.

### Examples

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

---

`config_Static-class`      *Create a config\_Static object*

---

### Description

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

### Usage

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

### Arguments

`item_selection` a named list containing item selection criteria.

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



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

MIP

a named list containing solver options.

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

## Value

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

## Examples

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

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

```

    target_value = c(8, 10, 12)
  )
)

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

```

---

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

---

### Description

[constraint](#) is an S4 class to represent a single constraint.

### Slots

`constraint` the constraint ID string of the constraint.

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

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

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

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

---

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

---

### Description

[constraints](#) is an S4 class to represent a set of constraints and its associated objects.

### Details

See [constraints-operators](#) for object manipulation functions.

**Slots**

`constraints` a `data.frame` containing the constraint specifications.  
`list_constraints` a list containing the `constraint` object representation of each constraint.  
`pool` the `item_pool` object associated with the constraints.  
`item_attrib` the `item_attrib` object associated with the constraints.  
`st_attrib` the `st_attrib` object associated with the constraints.  
`test_length` the test length specified in the constraints.  
`nv` the number of decision variables. Equals  $n_i + n_s$ .  
`ni` the number of items to search from.  
`ns` the number of stimulus to search from.  
`id` the item/stimulus ID string of each item/stimulus.  
`index,mat,dir,rhs` these represent MIP-format constraints. A single MIP-format constraint is associated with a value in `index`, a row in `mat`, a value in `rhs`, and a value in `dir`.

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

`set_based` TRUE if the constraint is set-based. FALSE otherwise.  
`item_order` the item attribute of each item to use in imposing an item order constraint, if any.  
`item_order_by` the name of the item attribute to use in imposing an item order constraint, if any.  
`stim_order` the stimulus attribute of each stimulus to use in imposing a stimulus order constraint, if any.  
`stim_order_by` the name of the stimulus attribute to use in imposing a stimulus order constraint, if any.  
`item_index_by_stimulus` a list containing item indices of each stimulus.  
`stimulus_index_by_item` the stimulus indices of each item.

---

constraints-operators *Basic operators for constraints objects*

---

**Description**

Create a subset of a `constraints` object:

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

Combine two `constraints` objects:

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

**Usage**

```
subsetConstraints(x, i = NULL)

combineConstraints(x1, x2)

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

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

**Arguments**

x, x1, x2	a <a href="#">constraints</a> object.
i, j	indices to use in subsetting.
...	not used, exists for compatibility.
drop	not used, exists for compatibility.

**Examples**

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

---

dataset\_bayes

*Bayes dataset*


---

**Description**

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

**Details**

This pool is associated with the following objects:

- `itempool_bayes` an [item\\_pool](#) object containing 320 items.
- `itemattrib_bayes` a [item\\_attrib](#) object containing 5 item-level attributes.
- `constraints_bayes` a [constraints](#) object containing 14 constraints.

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

- `itempool_bayes_data` a [data.frame](#) containing item parameters.
- `itempool_se_bayes_data` a [data.frame](#) containing item parameter standard errors.
- `itemattrib_bayes_data` a [data.frame](#) containing item attributes.
- `constraints_bayes_data` a [data.frame](#) containing constraint specifications.

## Examples

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

---

dataset_fatigue	<i>Fatigue dataset</i>
-----------------	------------------------

---

## Description

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

## Details

This pool is associated with the following objects:

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

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

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

## Examples

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

---

dataset_reading	<i>Reading dataset</i>
-----------------	------------------------

---

### Description

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

### Details

This pool is associated with the following objects:

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

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

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

### Examples

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

---

dataset_science	<i>Science dataset</i>
-----------------	------------------------

---

### Description

Item-based example item pool (1000 items).

## Details

This pool is associated with the following objects:

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

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

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

## Examples

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

---

eap

*Compute expected a posteriori estimates of theta*

---

## Description

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

## Usage

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

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

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

```
## S4 method for signature 'test'
```

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

```
## S4 method for signature 'test_cluster'
```

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

### Arguments

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

### Value

`eap` returns a list containing estimated values.

- th theta value.
- se standard error.

### Examples

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

---

find\_segment

*Classify theta into segments*

---

### Description

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

### Usage

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



**Arguments**

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

**Examples**

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

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

---

<code>getSolution</code>	<i>Print solution items</i>
--------------------------	-----------------------------

---

**Description**

Print solution items

**Usage**

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

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

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

**Arguments**

object            an [output\\_Static](#) object or an [output\\_Shadow](#) object.  
 examinee        (optional) the examinee index to display the solution. Used when the 'object' argument is an [output\\_Shadow](#) object.  
 position        (optional) if supplied, display the item attributes of the assembled test at that item position. If not supplied, display the item attributes of the administered items. Used when the 'object' argument is an [output\\_Shadow](#) object.  
 index\_only      if TRUE, only print item indices. if FALSE, print all item attributes. (default = TRUE)

**Value**

Item attributes of solution items.

---

getSolutionAttributes *Retrieve constraints-related attributes from solution*

---

## Description

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

## Usage

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

## Arguments

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

## Value

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

## Examples

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

---

info_1pl	<i>Calculate Fisher information (single item)</i>
----------	---

---

**Description**

info\_\* and array\_info\_\* are functions to calculate Fisher information.

**Usage**

```
info_1pl(x, b)
info_2pl(x, a, b)
info_3pl(x, a, b, c)
info_pc(x, b)
info_gpc(x, a, b)
info_gr(x, a, b)
array_info_1pl(x, b)
array_info_2pl(x, a, b)
array_info_3pl(x, a, b, c)
array_info_pc(x, b)
array_info_gpc(x, a, b)
array_info_gr(x, a, b)
```

**Arguments**

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

**Details**

info\_\* functions accept a single theta value, and array\_info\_\* functions accept multiple theta values.

Currently supports unidimensional models.

- info\_1pl, array\_info\_1pl: 1PL models
- info\_2pl, array\_info\_2pl: 2PL models
- info\_3pl, array\_info\_3pl: 3PL models
- info\_pc, array\_info\_pc: PC (partial credit) models
- info\_gpc, array\_info\_gpc: GPC (generalized partial credit) models
- info\_gr, array\_info\_gr: GR (graded response) models

## References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

```
x <- 0.5

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

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

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

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

---

iparPosteriorSample     *Sample item parameter estimates from their posterior distributions*

---

### Description

Sample item parameter estimates from their posterior distributions.

### Usage

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

### Arguments

pool                    An [item\\_pool](#) object.  
n\_sample                An integer as the number of sampled parameters.

### Examples

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

---

item-classes            *Item classes*

---

### Description

- [item\\_1PL](#) class represents a 1PL item.
- [item\\_2PL](#) class represents a 2PL item.
- [item\\_3PL](#) class represents a 3PL item.
- [item\\_PC](#) class represents a partial credit item.
- [item\\_GPC](#) class represents a generalized partial credit item.
- [item\\_GR](#) class represents a graded response item.

### Slots

slope a slope parameter value  
difficulty a difficulty parameter value  
guessing a guessing parameter value  
threshold a vector of threshold parameter values  
category a vector of category boundary values  
ncat the number of response categories

## References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

```

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

```

---

item\_attr-class      *Load item attributes*

---

## Description

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

## Usage

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

**Arguments**

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

**Value**

[loadItemAttrib](#) returns an [item\\_attrib](#) object.

- data a [data.frame](#) containing item attributes.

**See Also**

[dataset\\_science](#), [dataset\\_reading](#), [dataset\\_fatigue](#), [dataset\\_bayes](#) for examples.

**Examples**

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

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

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

## End(Not run)
```

---

item\_attrib-operators *Basic functions for item attribute objects*

---

**Description**

Basic functions for item attribute objects

**Usage**

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

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

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

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

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

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

**Arguments**

x	an <code>item_attr</code> object.
i, j	indices to use in subsetting.
...	not used, exists for compatibility.
drop	not used, exists for compatibility.
row.names	not used, exists for compatibility.
optional	not used, exists for compatibility.

**Examples**

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

---

item\_pool-class

*Class 'item\_pool': an item pool*


---

**Description**

`item_pool` is an S4 class to represent an item pool.



## Details

See [item\\_pool-operators](#) for object manipulation functions.

## Slots

ni the number of items in the pool.

max\_cat the maximum number of response categories across the pool.

index the numeric index of each item.

id the ID string of each item.

model the item class name of each item. See [item-classes](#).

NCAT the number of response categories of each item.

parms a list containing item class objects. See [item-classes](#).

ipar a matrix containing item parameters.

se a matrix containing item parameter standard errors.

raw the raw input [data.frame](#) used in `loadItemPool` to create this object.

raw\_se the raw input [data.frame](#) used in `loadItemPool` to create this object.

---

item\_pool-operators    *Basic operators for item pool objects*

---

## Description

Create a subset of an [item\\_pool](#) object:

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

Combine two [item\\_pool](#) objects:

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

`pool1 - pool2` excludes items in `pool2` from `pool1`.

`pool1 == pool2` tests whether two [item\\_pool](#) objects are identical.

**Usage**

```

subsetItemPool(x, i = NULL)

combineItemPool(x1, x2)

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

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

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

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

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

```

**Arguments**

x, x1, x2        an `item_pool` object.

i                item indices to use in subsetting.

j, drop, ...     not used, exists for compatibility.

**Examples**

```

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

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

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

```

---

```
item_pool_cluster-class
```

*Class 'item\_pool\_cluster': an item pool*

---

**Description**

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

**Slots**

np the number of item pools.  
 pools a list of [item\\_pool](#) objects.  
 names a vector containing item pool names.

---

InHyperPars                      *Calculate hyperparameters for log-normal distribution*

---

**Description**

Calculate hyperparameters for log-normal distribution.

**Usage**

```
InHyperPars(mean, sd)
```

**Arguments**

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

**Examples**

```
InHyperPars(.5, 1)
```

---

loadConstraints                      *Load constraints*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**See Also**

`dataset_science`, `dataset_reading`, `dataset_fatigue`, `dataset_bayes` for examples.

**Examples**

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

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

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

## End(Not run)
```

---

loadItemPool

*Load item pool*

---

**Description**

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

**Usage**

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

**Arguments**

<code>ipar</code>	item parameters. Can be a <code>data.frame</code> or the file path of a <code>.csv</code> file. The content should at least include columns 'ID' and 'MODEL'.
<code>ipar_se</code>	(optional) standard errors. Can be a <code>data.frame</code> or the file path of a <code>.csv</code> file.
<code>file</code>	(deprecated) use <code>ipar</code> argument instead.
<code>se_file</code>	(deprecated) use <code>ipar_se</code> argument instead.

**Value**

`loadItemPool` returns an `item_pool` object.

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

**See Also**

`dataset_science`, `dataset_reading`, `dataset_fatigue`, `dataset_bayes` for examples.

**Examples**

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

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

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

## End(Not run)
```

---

logitHyperPars	<i>Calculate hyperparameters for logit-normal distribution</i>
----------------	--

---

**Description**

Calculate hyperparameters for logit-normal distribution.

**Usage**

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

**Arguments**

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

**Examples**

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

---

makeItemPoolCluster	<i>Create an item pool cluster object</i>
---------------------	---

---

**Description**

Create a [item\\_pool\\_cluster](#) object.

`item_pool_cluster1 == item_pool_cluster2` tests equality of two `item_pool_cluster` objects.

**Usage**

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

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

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

**Arguments**

x, ...	<a href="#">item_pool</a> objects.
names	(optional) names to use for <a href="#">item_pool</a> .
item_pool_cluster1	an <a href="#">item_pool_cluster</a> object.
item_pool_cluster2	an <a href="#">item_pool_cluster</a> object.

## Examples

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

---

makeTest	<i>Generate a test object</i>
----------	-------------------------------

---

## Description

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

## Usage

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

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

## Arguments

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

## Examples

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

---

makeTestCluster	<i>Generate a test cluster object</i>
-----------------	---------------------------------------

---

### Description

Generate a `test_cluster` object

### Usage

```
makeTestCluster(object, theta, true_theta)

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

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

### Arguments

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

---

mle	<i>Compute maximum likelihood estimates of theta</i>
-----	--

---

### Description

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

### Usage

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



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

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

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

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

### Arguments

object	an <a href="#">item_pool</a> object.
select	(optional) if item indices are supplied, only the specified items are used.
resp	item response on all (or selected) items in the object argument. Can be a vector, a matrix, or a data frame. <code>length(resp)</code> or <code>ncol(resp)</code> must be equal to the

	number of all (or selected) items.
start_theta	(optional) initial theta values. If not supplied, EAP estimates using uniform priors are used as initial values. Uniform priors are computed using the theta_range argument below, with increments of .1.
max_iter	maximum number of iterations. (default = 100)
crit	convergence criterion to use. (default = 0.001)
truncate	set TRUE to impose a bound on the estimate. (default = FALSE)
theta_range	a range of theta values to bound the estimate. Only effective when truncate is TRUE. (default = c(-4, 4))
max_change	upper bound to impose on the absolute change in theta between iterations. Absolute changes exceeding this value will be capped to max_change. (default = 1.0)
do_Fisher	set TRUE to use Fisher scoring instead of Newton-Raphson method. (default = TRUE)

### Value

`mle` returns a list containing estimated values.

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

### Examples

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

---

mlef

*Compute maximum likelihood estimates of theta using fence items*

---

### Description

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

### Usage

```
mlef(
  object,
  select = NULL,
  resp,
  fence_slope = 5,
  fence_difficulty = c(-5, 5),
  start_theta = NULL,
```

```

    max_iter = 100,
    crit = 0.001,
    truncate = FALSE,
    theta_range = c(-4, 4),
    max_change = 1,
    do_Fisher = TRUE
)

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

```

### Arguments

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

do\_Fisher        set TRUE to use Fisher scoring instead of Newton-Raphson method. (default = TRUE)

### Value

`mlef` returns a list containing estimated values.

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

### References

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

### Examples

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

---

output\_Shadow-class    *Class 'output\_Shadow': adaptive assembly solution for one simulee*

---

### Description

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

### Slots

`simulee_id` the numeric ID of the simulee.  
`true_theta` the true theta of the simulee, if was specified.  
`true_theta_segment` the segment number of the true theta.  
`final_theta_est` final theta estimate.  
`final_se_est` the standard error of `final_theta_est`.  
`administered_item_index` item IDs administered at each position.  
`administered_item_resp` item responses from the simulee at each position.  
`administered_item_ncat` the number of categories of each administered item.  
`administered_stimulus_index` stimulus IDs administered at each position.  
`shadow_test_refreshed` TRUE indicates the shadow test was refreshed for the position.  
`shadow_test_feasible` TRUE indicates the MIP was feasible with all constraints.  
`solve_time` elapsed time in running the solver at each position.

interim\_theta\_est interim theta estimates at each position.  
 interim\_se\_est the standard error of the interim estimate at each position.  
 theta\_segment\_index segment numbers of interim theta estimates.  
 prior prior distribution, if was specified.  
 prior\_par prior parameters, if were specified.  
 posterior the posterior distribution after completing test.  
 posterior\_sample posterior samples of interim theta before the estimation of final theta.  $\text{mean}(\text{posterior\_sample})$   
 $\text{== interim\_theta\_est}[\text{test\_length}]$  holds.  
 likelihood the likelihood distribution after completing test.  
 shadow\_test the list containing the item IDs within the shadow test used in each position.  
 max\_cat\_pool the maximum number of response categories the item pool had.  
 ni\_pool the total number of items the item pool had.  
 ns\_pool the total number of stimuli the item pool had.  
 test\_length\_constraints the test length constraint used in assembly.  
 set\_based whether the item pool was set-based.  
 item\_index\_by\_stimulus the list of items by each stimulus the item pool had.

---

output\_Shadow\_all-class

*Class 'output\_Shadow\_all': a set of adaptive assembly solutions*

---

## Description

[output\\_Shadow\\_all](#) is an S4 class to represent a set of adaptive assembly solutions.

## Details

- notations**
- *ni* denotes the number of items in the [item\\_pool](#) object.
  - *ns* denotes the number of stimuli.
  - *nj* denotes the number of participants.

## Slots

output a length-*nj*\* list of [output\\_Shadow](#) objects, containing the assembly results for each participant.  
 final\_theta\_est a length-*nj*\* vector containing final theta estimates for each participant.  
 final\_se\_est a length-*nj*\* vector standard errors of the final theta estimates for each participant.  
 exposure\_rate a matrix containing item-level exposure rates of all items in the pool. Also contains stimulus-level exposure rates if the assembly was set-based.  
 usage\_matrix a *nj*\* by (*ni*\* + *ns*\*) matrix representing whether the item/stimulus was administered to each participant. Stimuli representations are appended to the right side of the matrix.

`true_segment_count` a length-`*nj*` vector containing the how many examinees are now in their segment based on the true theta. This will tend to increase. This can be reproduced with true theta values alone.  
`est_segment_count` a length-`*nj*` vector containing the how many examinees are now in their segment based on the estimated theta. This will tend to increase. This can be reproduced with estimated theta values alone.  
`eligibility_stats` exposure record for diagnostics.  
`check_eligibility_stats` detailed segment-wise exposure record for diagnostics. available when `config_Shadow@exposure_control$diagnostic_stats` is TRUE.  
`no_fading_eligibility_stats` detailed segment-wise exposure record without fading for diagnostics. available when `config_Shadow@exposure_control$diagnostic_stats` is TRUE.  
`freq_infeasible` a table representing the number of times the assembly was initially infeasible.  
`pool` the `item_pool` used in the assembly.  
`config` the `config_Shadow` used in the assembly.  
`constraints` the `constraints` used in the assembly.  
`true_theta` the `true_theta` argument used in the assembly.  
`data` the data argument used in the assembly.  
`prior` the `prior` argument used in the assembly.  
`prior_par` the `prior_par` argument used in the assembly.

---

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

---

## Description

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

## Slots

`MIP` a list containing the result from MIP solver.  
`selected` a `data.frame` containing the selected items and their attributes.  
`obj_value` the objective value of the solution.  
`solve_time` the elapsed time in running the solver.  
`achieved` a `data.frame` containing attributes of the assembled test, by each constraint.  
`pool` the `item_pool` used in the assembly.  
`config` the `config_Static` used in the assembly.  
`constraints` the `constraints` used in the assembly.

---

plot

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

---

### Description

Extension of plot() for objects in TestDesign package

### Usage

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

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

```
)

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

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

## S4 method for signature 'output_Shadow_all'
plot(
  x,
  y,
  type = "audit",
  theta = seq(-3, 3, 0.1),
  info_type = "FISHER",
```



```

plot_sum = TRUE,
select = NULL,
examinee_id = 1,
position = NULL,
theta_range = c(-5, 5),
color = "blue",
z_ci = 1.96,
simple = FALSE,
theta_segment = "Estimated",
color_final = "blue",
...
)

```

## Arguments

x	<p>accepts the following signatures:</p> <ul style="list-style-type: none"> <li>• <code>item_pool</code>: plot information and expected scores.</li> <li>• <code>constraints</code>: plot information range based on the test length constraint.</li> <li>• <code>output_Static</code>: plot information and expected scores based on the fixed assembly solution.</li> <li>• <code>output_Shadow_all</code>: plot audit trail, shadow test chart, and exposure rates from the adaptive assembly solution.</li> <li>• <code>output_Shadow</code>: plot audit trail and shadow test chart from the adaptive assembly solution.</li> </ul>
y	not used, exists for compatibility with <code>plot</code> in the base R package.
type	<p>the type of plot.</p> <ul style="list-style-type: none"> <li>• <code>info</code> plots information from <code>item_pool</code>, <code>output_Static</code>, and <code>output_Shadow_all</code>.</li> <li>• <code>score</code> plots expected scores from <code>item_pool</code> and <code>output_Static</code>.</li> <li>• <code>audit</code> plots audit trail from <code>output_Shadow_all</code> and <code>output_Shadow</code>.</li> <li>• <code>shadow</code> plots shadow test chart from <code>output_Shadow_all</code> and <code>output_Shadow</code>.</li> <li>• <code>exposure</code> plots exposure rates from <code>output_Shadow_all</code>.</li> </ul>
theta	the theta grid to use in plotting. (default = <code>seq(-3, 3, .1)</code> )
info_type	the type of information. Currently accepts FISHER. (default = FISHER)
plot_sum	<p>used in <code>item_pool</code> objects.</p> <ul style="list-style-type: none"> <li>• if TRUE then plot pool-level values.</li> <li>• if FALSE then plot item-level values, and repeat for all items in the pool.</li> <li>• (default = TRUE)</li> </ul>
select	used in <code>item_pool</code> objects. Item indices to subset.
examinee_id	used in <code>output_Shadow</code> and <code>output_Shadow_all</code> with type = 'audit' and type = 'shadow'. The examinee numeric ID to draw the plot.
position	used in <code>output_Shadow_all</code> with type = 'info'. The item position to draw the plot.
theta_range	used in <code>output_Shadow</code> and <code>output_Shadow_all</code> with type = 'audit'. The theta range to plot. (default = <code>c(-5, 5)</code> )

color	the color of the curve.
z_ci	used in <code>output_Shadow</code> and <code>output_Shadow_all</code> with type = 'audit'. The range to use for confidence intervals. (default = 1.96)
simple	used in <code>output_Shadow</code> and <code>output_Shadow_all</code> with type = 'shadow'. If TRUE, simplify the chart by hiding unused items.
theta_segment	used in <code>output_Shadow_all</code> with type = 'exposure'. The type of theta to determine exposure segments. Accepts Estimated or True. (default = Estimated)
color_final	used in <code>output_Shadow_all</code> with type = 'exposure'. The color of item-wise exposure rates, only counting the items administered in the final theta segment as exposed.
...	arguments to pass onto <code>plot</code> .

## Examples

```

subitempool <- itempool_science[1:8]

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

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

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

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

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

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

```

**Description**

(deprecated) Use `plot` with `type = 'audit'` instead.

**Usage**

```
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_all'  
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'  
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'  
plotCAT(  
  object,  
  examinee_id = 1,  
  min_theta = -5,
```

```

    max_theta = 5,
    min_score = 0,
    max_score = 1,
    z_ci = 1.96,
    file_pdf = NULL,
    ...
)

```

### Arguments

object	An output object generated by <a href="#">Shadow</a> .
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

```

## Not run:
config <- createShadowTestConfig()
true_theta <- rnorm(1)
solution <- Shadow(config, constraints_science, true_theta)
plotCAT(solution, 1)

## End(Not run)

```

---

plotExposure *(deprecated) Plot item exposure rates*

---

### Description

(deprecated) Use [plot](#) with type = 'exposure' instead.

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

## S4 method for signature 'output_Shadow_all'
plotExposure(
  object,
  max_rate = 0.25,
  theta_segment = "estimated",
  color = "blue",
  color_final = "blue",
  file_pdf = NULL,
  ...
)

```

### Arguments

object	an output object generated by <a href="#">Shadow</a> .
max_rate	the target exposure rate.
theta_segment	the type of theta to use to create segments. Accepts "estimated" or "true". (default = "estimated")
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

```

## Not run:
true_theta <- runif(10, min = -3.5, max = 3.5)
resp_science <- simResp(itempool_science, true_theta)
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)

## End(Not run)
```

---

plotInfo	<i>(deprecated) Plot item/test/pool-level information</i>
----------	---

---

## Description

(deprecated) Use [plot](#).

## Usage

```
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)
)

## S4 method for signature 'output_Static'
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)
)

## 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	<ul style="list-style-type: none"> <li>• <code>item_pool</code>: plot pool-level or item-level information.</li> <li>• <code>output_Static</code>: plot test-level information of the assembly solution.</li> <li>• <code>constraints</code>: plot attainable information range.</li> </ul>
theta	the theta grid to use on the x-axis. (default = <code>seq(-3, 3, .1)</code> )
info_type	the type of information. Accepts FISHER. (default = FISHER)
plot_sum	used when the object argument is an <code>item_pool</code> object. If TRUE then draw pool-level information, and if FALSE draw item-level information for every item in the pool. (default = TRUE)
select	(optional) used when the object argument is an <code>item_pool</code> object. Items to select from the pool.
color	the color of the curve. (default = blue)
file_pdf	(optional) if supplied a filename, save as a PDF file.
width	the width of the plot. (default = 7)
height	the height of the plot. (default = 6)
mfrow	multi-panel configurations to use. (default = <code>c(2, 4)</code> )

### Examples

```

subitempool <- subsetItemPool(itempool_science, 1:8)
plot(subitempool)

config <- createStaticTestConfig()

```

```
solution <- Static(config, constraints_science)
plot(solution)
```

---

plotShadow	<i>(deprecated) Plot a shadow test chart</i>
------------	--

---

## Description

(deprecated) Use `plot` with `type = 'shadow'` instead.

## Usage

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

## S4 method for signature 'output_Shadow_all'
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

<code>object</code>	An output from <code>Shadow</code> function.
<code>examinee_id</code>	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

```

## Not run:
config <- createShadowTestConfig()
true_theta <- rnorm(1)
solution <- Shadow(config, constraints_science, true_theta)
plotShadow(solution, 1)
plotShadow(solution, 1, simple = TRUE)

## End(Not run)

```

---

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

---

### Description

Extension of print() for objects in TestDesign package

### Usage

```

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

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

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

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### Arguments

x	an object to print.
index_only	if TRUE then only print item indices. If FALSE then print all item attributes. (default = TRUE)
digits	minimal number of <i>*significant*</i> digits. See <a href="#">print.default</a> .

---

p\_1pl

*Calculate item response probability*


---

**Description**

p\_\* and array\_p\_\* are functions to calculate item response probability.

**Usage**

p\_1pl(x, b)

p\_2pl(x, a, b)

p\_3pl(x, a, b, c)

p\_pc(x, b)

p\_gpc(x, a, b)

p\_gr(x, a, b)

array\_p\_1pl(x, b)

array\_p\_2pl(x, a, b)

array\_p\_3pl(x, a, b, c)

array\_p\_pc(x, b)

array\_p\_gpc(x, a, b)

array\_p\_gr(x, a, b)

**Arguments**

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

**Details**

p\_\* functions accept a single theta value, and array\_p\_\* functions accept multiple theta values. Currently supports unidimensional models.

- p\_1pl, array\_p\_1pl: 1PL models
- p\_2pl, array\_p\_2pl: 2PL models
- p\_3pl, array\_p\_3pl: 3PL models
- p\_pc, array\_p\_pc: PC (partial credit) models
- p\_gpc, array\_p\_gpc: GPC (generalized partial credit) models
- p\_gr, array\_p\_gr: GR (graded response) models

## References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

```
x <- 0.5

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

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

array_p_1pl(x, 1)
array_p_2pl(x, 1, 2)
```

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

---

RE *Calculate Relative Errors*

---

### Description

Calculate Relative Errors.

### Usage

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

### Arguments

RMSE\_foc      A vector of RMSE values for the focal group.  
 RMSE\_ref      A vector of RMSE values for the reference group.

---

RMSE *Calculate Root Mean Squared Error*

---

### Description

Calculate Root Mean Squared Error.

### Usage

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

### Arguments

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

---

runAssembly	<i>Run Test Assembly</i>
-------------	--------------------------

---

### Description

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

### Usage

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

### Arguments

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

### Value

A list containing the following entries:

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

### References

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

---

saveOutput	<i>Save or print audit trails</i>
------------	-----------------------------------

---

**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	<i>Run adaptive test assembly</i>
--------	-----------------------------------

---

**Description**

[Shadow](#) is a test assembly function to perform adaptive test assembly based on the generalized shadow-test framework.

**Usage**

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

## S4 method for signature 'config_Shadow'
Shadow(
  config,
```

```

constraints = NULL,
true_theta = NULL,
data = NULL,
prior = NULL,
prior_par = NULL,
excluded_items = NULL,
force_solver = FALSE,
session = NULL
)

```

### Arguments

config	a <code>config_Shadow</code> object. Use <code>createShadowTestConfig</code> for this.
constraints	a <code>constraints</code> object representing test specifications. Use <code>loadConstraints</code> for this.
true_theta	(optional) true theta values to use in simulation. Either <code>true_theta</code> or <code>data</code> must be supplied.
data	(optional) a matrix containing item response data to use in simulation. Either <code>true_theta</code> or <code>data</code> must be supplied.
prior	(optional) prior density at each <code>config@theta_grid</code> . This overrides <code>prior_par</code> . Can be a vector to use the same prior for all $n_j$ participants, or a $n_j$ -row matrix to use a different prior for each participant.
prior_par	(optional) normal distribution parameters $c(\text{mean}, \text{sd})$ to use as prior. Can be a vector to use the same prior for all $n_j$ participants, or a $n_j$ -row matrix to use a different prior for each participant.
excluded_items	(optional) a list containing item names to exclude from selection for each participant.
force_solver	if TRUE, do not check whether the solver is one of recommended solvers for doing set-based assembly. Has no effect on discrete assembly. (default = FALSE)
session	(optional) used to communicate with Shiny app <code>TestDesign</code> .

### Value

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

### References

- van der Linden, W. J., Reese, L. M. (1998). A model for optimal constrained adaptive testing. *Applied Psychological Measurement*, 22, 259–270.
- van der Linden, W. J. (1998). Optimal assembly of psychological and educational tests. *Applied Psychological Measurement*, 22, 195–211.
- van der Linden, W. J. (2000). Optimal assembly of tests with item sets. *Applied Psychological Measurement*, 24, 225–240.
- van der Linden, W. J. (2005). *Linear models for optimal test design*. Springer Science & Business Media.



**Examples**

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

---

show

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

---

**Description**

Extension of show() for objects in TestDesign package

**Usage**

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

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

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

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

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

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

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

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

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

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

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

```
## S4 method for signature 'summary_item_pool'  
show(object)  
  
## S4 method for signature 'summary_item_attrib'  
show(object)  
  
## S4 method for signature 'summary_constraints'  
show(object)  
  
## S4 method for signature 'config_Static'  
show(object)  
  
## S4 method for signature 'config_Shadow'  
show(object)  
  
## S4 method for signature 'output_Static'  
show(object)  
  
## S4 method for signature 'output_Shadow'  
show(object)  
  
## S4 method for signature 'output_Shadow_all'  
show(object)  
  
## S4 method for signature 'summary_output_Static'  
show(object)  
  
## S4 method for signature 'summary_output_Shadow_all'  
show(object)  
  
## S4 method for signature 'exposure_rate_plot'  
show(object)
```

### Arguments

object            an object to display.

---

showConstraints            *(deprecated) Show constraints*

---

### Description

(deprecated) Use [print](#) instead.

### Usage

```
showConstraints(constraints)
```

**Arguments**

constraints     a [constraints](#) object.

---

simResp	<i>Simulate item response data</i>
---------	------------------------------------

---

**Description**

[simResp](#) is a function to simulate item response data.

**Usage**

```
simResp(object, theta)

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

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

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

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

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

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

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

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

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

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

**Arguments**

object            an [item](#) or an [item\\_pool](#) object.  
theta             theta values to use.

## Details

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

## Value

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

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

## References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

## Examples

```

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

sim_item_1 <- simResp(item_1, seq(-3, 3, 1))
sim_item_2 <- simResp(item_2, seq(-3, 3, 1))
sim_item_3 <- simResp(item_3, seq(-3, 3, 1))

```

```
sim_item_4 <- simResp(item_4, seq(-3, 3, 1))
sim_item_5 <- simResp(item_5, seq(-3, 3, 1))
sim_item_6 <- simResp(item_6, seq(-3, 3, 1))
sim_pool  <- simResp(itempool_science, seq(-3, 3, 1))
```

---

Static

*Run fixed-form test assembly*

---

### Description

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

### Usage

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

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

### Arguments

`config` a `config_Static` object. Use `createStaticTestConfig` for this.

`constraints` a `constraints` object representing test specifications. Use `loadConstraints` for this.

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

### Value

`Static` returns a `output_Static` object containing the selected items.

### References

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

### Examples

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

---

st_attrib-class	<i>Load set/stimulus/passage attributes</i>
-----------------	---

---

## Description

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

## Usage

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

## Arguments

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

## Value

`loadStAttrib` returns a `st_attrib` object.

- data a `data.frame` containing stimulus attributes.

## See Also

`dataset_reading` for examples.

## Examples

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

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

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

## End(Not run)
```

## Description

Basic functions for stimulus attribute objects

## Usage

```
## S4 method for signature 'st_attrib,numeric'  
x[i, j, ..., drop = TRUE]  
  
## S4 method for signature 'st_attrib'  
dim(x)  
  
## S4 method for signature 'st_attrib'  
colnames(x)  
  
## S4 method for signature 'st_attrib'  
rownames(x)  
  
## S4 method for signature 'st_attrib'  
names(x)  
  
## S4 method for signature 'st_attrib'  
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

## Arguments

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

## Examples

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

```
as.data.frame(x)
```

---

summary

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

---

## Description

Extension of summary() for objects in TestDesign package

## Usage

```
## S4 method for signature 'item_pool'  
summary(object)  
  
## S4 method for signature 'item_attrib'  
summary(object)  
  
## S4 method for signature 'constraints'  
summary(object)  
  
## S4 method for signature 'output_Static'  
summary(object)  
  
## S4 method for signature 'output_Shadow_all'  
summary(object)
```

## Arguments

object            an object to summarize.

## Examples

```
summary(itempool_science)  
summary(itemattrib_science)  
  
cfg <- createStaticTestConfig()  
solution <- Static(cfg, constraints_science)  
summary(solution)  
  
cfg <- createShadowTestConfig()  
solution <- Shadow(cfg, constraints_science, true_theta = seq(-1, 1, 1))  
summary(solution)
```



---

summary-classes	<i>Summary classes</i>
-----------------	------------------------

---

**Description**

Summary classes

---

test-class	<i>Class 'test': data for test assembly</i>
------------	---

---

**Description**

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

**Slots**

`pool` the `item_pool` object.  
`theta` the theta grid to use as quadrature points.  
`prob` the list containing item response probabilities.  
`info` the matrix containing item information values.  
`true_theta` (optional) the true theta values.  
`data` (optional) the matrix containing item responses.

---

TestDesign	<i>Open TestDesign app</i>
------------	----------------------------

---

**Description**

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

**Usage**

```
TestDesign()
```

**Examples**

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

## End(Not run)
```

---

test_cluster-class	<i>Class 'test_cluster': data for test assembly</i>
--------------------	---

---

### Description

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

### Slots

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

`tests` the list containing `test` objects.

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

---

test_operators	<i>Basic operators for test objects</i>
----------------	---

---

### Description

Create a subset of a `test` object.

### Usage

```
subsetTest(x, i = NULL)
```

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

### Arguments

`x` a `test` object.

`i` item indices to use in subsetting.

`j, drop, ...` not used, exists for compatibility.

---

theta_EAP	<i>Calculate an EAP estimate of theta for one examinee</i>
-----------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

---

theta_EAP_matrix	<i>Calculate EAP estimates of theta for a group of examinees</i>
------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

---

theta_EB	<i>Calculate theta estimates using EB (Empirical Bayes) method</i>
----------	--

---

### Description

theta\_EB\_single and theta\_EB are functions to calculate theta estimates using EB (Empirical Bayes) method.

### Usage

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

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

### Arguments

nx	the number of MCMC draws.
theta_init	initial estimate of theta.
theta_prop	SD of the proposal distribution.
item_parm	a matrix containing item parameters. Each row represents each item.
resp	a vector (or a value if for one item) containing responses on each item.
ncat	a vector (or a value if for one item) containing the number of response categories of each item.
model	a vector (or a value if for one item) indicating item models of each item, using <ul style="list-style-type: none"> <li>• 1: 1PL model</li> </ul>

- 2: 2PL model
  - 3: 3PL model
  - 4: PC model
  - 5: GPC model
  - 6: GR model
- prior            an integer indicating the type of prior distribution, using
- 1: normal distribution
  - 2: uniform distribution
- prior\_parm      a vector containing parameters for the prior distribution.

### Details

theta\_EB\_single is designed for one item, and theta\_EB is designed for multiple items.  
Currently supports unidimensional models.

### References

- Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen: Danish Institute for Educational Research.
- Lord, F. M. (1952). A theory of test scores (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation.
- Birnbaum, A. (1957). *Efficient design and use of tests of mental ability for various decision-making problems* (Series Report No. 58-16. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *On the estimation of mental ability* (Series Report No. 15. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1958). *Further considerations of efficiency in tests of a mental ability* (Series Report No. 17. Project No. 7755-23). Randolph Air Force Base, TX: USAF School of Aviation Medicine.
- Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. M., Novick, M. R. (eds.), *Statistical Theories of Mental Test Scores*, 397–479. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174.
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176.
- Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph*, 17.

### Examples

```
# item parameters
item_parm <- matrix(c(
  1, NA, NA,
  1, 2, NA,
```

```

1, 2, 0.25,
0, 1, NA,
2, 0, 1,
2, 0, 2),
nrow = 6,
byrow = TRUE
)

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

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

```

---

theta\_FB

---

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


---

### Description

Calculate a fully Bayesian estimate of theta for an examinee.

### Usage

```

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

```

### Arguments

nx	The number of MCMC draws.
theta_init	A value for initial estimate of theta.
theta_prop	SD of the proposal distribution.
items_list	A list of item_parm matrices.

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

---

theta_FB_single	<i>Calculate a fully Bayesian estimate of theta for a single item</i>
-----------------	---

---

### Description

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

### Usage

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

### Arguments

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

---

toggleConstraints	<i>Toggle constraints</i>
-------------------	---------------------------

---

**Description**

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

**Usage**

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

**Arguments**

object	a <code>constraints</code> object from <code>loadConstraints</code> .
on	constraint indices to mark as active.
off	constraint indices to mark as inactive.

**Value**

`toggleConstraints` returns the updated `constraints` object.

**Examples**

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

---

updateConstraints	<i>(deprecated) Update constraints</i>
-------------------	--

---

**Description**

Use `toggleConstraints` instead.

**Usage**

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

**Arguments**

object	a <code>constraints</code> object from <code>loadConstraints</code> .
on	constraint indices to mark as active.
off	constraint indices to mark as inactive.



**Value**

`updateConstraints` returns the updated `constraints` object.

**Examples**

```
## Not run:  
constraints_science2 <- updateConstraints(constraints_science, off = 32:36)  
constraints_science3 <- updateConstraints(constraints_science, on = 32:36)  
  
## End(Not run)
```

# Index

## \* datasets

- dataset\_bayes, 36
- dataset\_fatigue, 37
- dataset\_reading, 38
- dataset\_science, 38
- + .item\_pool (item\_pool-operators), 49
- .item\_pool (item\_pool-operators), 49
- == .item\_pool (item\_pool-operators), 49
- == .item\_pool\_cluster
  - (makeItemPoolCluster), 54
- [, constraints, numeric, ANY, ANY-method
  - (constraints-operators), 35
- [, constraints, numeric-method
  - (constraints-operators), 35
- [, item\_attrib, numeric, ANY, ANY-method
  - (item\_attrib-operators), 47
- [, item\_attrib, numeric-method
  - (item\_attrib-operators), 47
- [, item\_pool, numeric, ANY, ANY-method
  - (item\_pool-operators), 49
- [, item\_pool, numeric-method
  - (item\_pool-operators), 49
- [, st\_attrib, numeric, ANY, ANY-method
  - (st\_attrib-operators), 87
- [, st\_attrib, numeric-method
  - (st\_attrib-operators), 87
- [, test, ANY-method (test\_operators), 90
- [, test, numeric, ANY, ANY-method
  - (test\_operators), 90
- app, 4, 4
- array\_info\_1pl (info\_1pl), 43
- array\_info\_2pl (info\_1pl), 43
- array\_info\_3pl (info\_1pl), 43
- array\_info\_gpc (info\_1pl), 43
- array\_info\_gr (info\_1pl), 43
- array\_info\_pc (info\_1pl), 43
- array\_p\_1pl (p\_1pl), 75
- array\_p\_2pl (p\_1pl), 75
- array\_p\_3pl (p\_1pl), 75
- array\_p\_gpc (p\_1pl), 75
- array\_p\_gr (p\_1pl), 75
- array\_p\_pc (p\_1pl), 75
- as.data.frame, item\_attrib-method
  - (item\_attrib-operators), 47
- as.data.frame, st\_attrib-method
  - (st\_attrib-operators), 87
- buildConstraints, 5, 5
- c, constraints-method
  - (constraints-operators), 35
- c, item\_pool-method
  - (item\_pool-operators), 49
- calc\_info, 21
- calc\_info\_EB, 23
- calc\_info\_FB, 23
- calc\_info\_matrix (calc\_info), 21
- calc\_likelihood, 24
- calc\_likelihood\_function
  - (calc\_likelihood), 24
- calc\_log\_likelihood (calc\_likelihood),  
24
- calc\_log\_likelihood\_function
  - (calc\_likelihood), 24
- calc\_MI\_FB, 26
- calc\_posterior, 26
- calc\_posterior\_function, 27
- calc\_posterior\_single, 28
- calcEScore, 6, 6, 7
- calcEScore, item\_1PL, matrix-method
  - (calcEScore), 6
- calcEScore, item\_1PL, numeric-method
  - (calcEScore), 6
- calcEScore, item\_2PL, matrix-method
  - (calcEScore), 6
- calcEScore, item\_2PL, numeric-method
  - (calcEScore), 6
- calcEScore, item\_3PL, matrix-method
  - (calcEScore), 6

- calcEscore,item\_3PL,numeric-method  
(calcEscore), 6
- calcEscore,item\_GPC,matrix-method  
(calcEscore), 6
- calcEscore,item\_GPC,numeric-method  
(calcEscore), 6
- calcEscore,item\_GR,matrix-method  
(calcEscore), 6
- calcEscore,item\_GR,numeric-method  
(calcEscore), 6
- calcEscore,item\_PC,matrix-method  
(calcEscore), 6
- calcEscore,item\_PC,numeric-method  
(calcEscore), 6
- calcEscore,item\_pool,matrix-method  
(calcEscore), 6
- calcEscore,item\_pool,numeric-method  
(calcEscore), 6
- calcEscore,item\_pool\_cluster,numeric-method  
(calcEscore), 6
- calcFisher, 8, 8, 9
- calcFisher,item\_1PL,matrix-method  
(calcFisher), 8
- calcFisher,item\_1PL,numeric-method  
(calcFisher), 8
- calcFisher,item\_2PL,matrix-method  
(calcFisher), 8
- calcFisher,item\_2PL,numeric-method  
(calcFisher), 8
- calcFisher,item\_3PL,matrix-method  
(calcFisher), 8
- calcFisher,item\_3PL,numeric-method  
(calcFisher), 8
- calcFisher,item\_GPC,matrix-method  
(calcFisher), 8
- calcFisher,item\_GPC,numeric-method  
(calcFisher), 8
- calcFisher,item\_GR,matrix-method  
(calcFisher), 8
- calcFisher,item\_GR,numeric-method  
(calcFisher), 8
- calcFisher,item\_PC,matrix-method  
(calcFisher), 8
- calcFisher,item\_PC,numeric-method  
(calcFisher), 8
- calcFisher,item\_pool,matrix-method  
(calcFisher), 8
- calcFisher,item\_pool,numeric-method  
(calcFisher), 8
- calcFisher,item\_pool\_cluster,numeric-method  
(calcFisher), 8
- calcHessian, 11, 11, 12
- calcHessian,item\_1PL,numeric,numeric-method  
(calcHessian), 11
- calcHessian,item\_1PL,numeric-method  
(calcHessian), 11
- calcHessian,item\_2PL,numeric,numeric-method  
(calcHessian), 11
- calcHessian,item\_2PL,numeric-method  
(calcHessian), 11
- calcHessian,item\_3PL,numeric,numeric-method  
(calcHessian), 11
- calcHessian,item\_3PL,numeric-method  
(calcHessian), 11
- calcHessian,item\_GPC,numeric,numeric-method  
(calcHessian), 11
- calcHessian,item\_GPC,numeric-method  
(calcHessian), 11
- calcHessian,item\_GR,numeric,numeric-method  
(calcHessian), 11
- calcHessian,item\_GR,numeric-method  
(calcHessian), 11
- calcHessian,item\_PC,numeric,numeric-method  
(calcHessian), 11
- calcHessian,item\_PC,numeric-method  
(calcHessian), 11
- calcHessian,item\_pool,numeric,numeric-method  
(calcHessian), 11
- calcHessian,item\_pool,numeric-method  
(calcHessian), 11
- calcHessian,item\_pool\_cluster,numeric,list-method  
(calcHessian), 11
- calcHessian,item\_pool\_cluster,numeric-method  
(calcHessian), 11
- calcJacobian, 13, 13, 14
- calcJacobian,item\_1PL,numeric,numeric-method  
(calcJacobian), 13
- calcJacobian,item\_1PL,numeric-method  
(calcJacobian), 13
- calcJacobian,item\_2PL,numeric,numeric-method  
(calcJacobian), 13
- calcJacobian,item\_2PL,numeric-method  
(calcJacobian), 13
- calcJacobian,item\_3PL,numeric,numeric-method  
(calcJacobian), 13
- calcJacobian,item\_3PL,numeric-method

- (calcJacobian), 13
- calcJacobian, item\_GPC, numeric, numeric-method (calcJacobian), 13
- calcJacobian, item\_GPC, numeric-method (calcJacobian), 13
- calcJacobian, item\_GR, numeric, numeric-method (calcJacobian), 13
- calcJacobian, item\_GR, numeric-method (calcJacobian), 13
- calcJacobian, item\_PC, numeric, numeric-method (calcJacobian), 13
- calcJacobian, item\_PC, numeric-method (calcJacobian), 13
- calcJacobian, item\_pool, numeric, numeric-method (calcJacobian), 13
- calcJacobian, item\_pool, numeric-method (calcJacobian), 13
- calcJacobian, item\_pool\_cluster, numeric, list-method (calcJacobian), 13
- calcJacobian, item\_pool\_cluster, numeric-method (calcJacobian), 13
- calcLocation, 15, 16
- calcLocation (calcLocation-methods), 15
- calcLocation, item\_1PL-method (calcLocation-methods), 15
- calcLocation, item\_2PL-method (calcLocation-methods), 15
- calcLocation, item\_3PL-method (calcLocation-methods), 15
- calcLocation, item\_GPC-method (calcLocation-methods), 15
- calcLocation, item\_GR-method (calcLocation-methods), 15
- calcLocation, item\_PC-method (calcLocation-methods), 15
- calcLocation, item\_pool-method (calcLocation-methods), 15
- calcLocation-methods, 15
- calcLogLikelihood, 17, 17
- calcLogLikelihood, item\_pool, matrix, matrix-method (calcLogLikelihood), 17
- calcLogLikelihood, item\_pool, matrix, numeric-method (calcLogLikelihood), 17
- calcLogLikelihood, item\_pool, numeric, matrix-method (calcLogLikelihood), 17
- calcLogLikelihood, item\_pool, numeric, numeric-method (calcLogLikelihood), 17
- calcProb, 10, 16, 18, 19
- calcProb (calcProb-methods), 18
- calcProb, item\_1PL, matrix-method (calcProb-methods), 18
- calcProb, item\_1PL, numeric-method (calcProb-methods), 18
- calcProb, item\_2PL, matrix-method (calcProb-methods), 18
- calcProb, item\_2PL, numeric-method (calcProb-methods), 18
- calcProb, item\_3PL, matrix-method (calcProb-methods), 18
- calcProb, item\_3PL, numeric-method (calcProb-methods), 18
- calcProb, item\_GPC, matrix-method (calcProb-methods), 18
- calcProb, item\_GPC, numeric-method (calcProb-methods), 18
- calcProb, item\_GR, matrix-method (calcProb-methods), 18
- calcProb, item\_GR, numeric-method (calcProb-methods), 18
- calcProb, item\_PC, matrix-method (calcProb-methods), 18
- calcProb, item\_PC, numeric-method (calcProb-methods), 18
- calcProb, item\_pool, matrix-method (calcProb-methods), 18
- calcProb, item\_pool, numeric-method (calcProb-methods), 18
- calcProb, item\_pool\_cluster, numeric-method (calcProb-methods), 18
- calcProb-methods, 18
- checkConstraints, 28
- colnames, item\_attrib-method (item\_attrib-operators), 47
- colnames, st\_attrib-method (st\_attrib-operators), 87
- combineConstraints (constraints-operators), 35
- combineItemPool (item\_pool-operators), 49
- config\_Shadow, 29, 62, 78, 80
- config\_Shadow-class, 29
- config\_Static, 32, 33, 62, 78, 85
- config\_Static-class, 32
- constraint, 34, 35
- constraint-class, 34
- constraints, 5, 28, 34–39, 42, 51, 52, 62, 65,

- 71, 78, 80, 83, 85, 96, 97
- constraints-class, 34
- constraints-operators, 35
- constraints\_bayes (dataset\_bayes), 36
- constraints\_bayes\_data (dataset\_bayes), 36
- constraints\_fatigue (dataset\_fatigue), 37
- constraints\_fatigue\_data (dataset\_fatigue), 37
- constraints\_reading (dataset\_reading), 38
- constraints\_reading\_data (dataset\_reading), 38
- constraints\_science (dataset\_science), 38
- constraints\_science\_data (dataset\_science), 38
- createShadowTestConfig, 29, 78, 80
- createShadowTestConfig (config\_Shadow-class), 29
- createStaticTestConfig, 32, 33, 78, 85
- createStaticTestConfig (config\_Static-class), 32
- data.frame, 35–39, 42, 46, 47, 49, 51–53, 62, 86
- dataset\_bayes, 36, 47, 52, 53
- dataset\_fatigue, 37, 47, 52, 53
- dataset\_reading, 38, 47, 52, 53, 86
- dataset\_science, 38, 47, 52, 53
- dim, item\_attrib-method (item\_attrib-operators), 47
- dim, st\_attrib-method (st\_attrib-operators), 87
- EAP (eap), 39
- eap, 39, 39, 40
- eap, item\_pool-method (eap), 39
- EAP, test-method (eap), 39
- EAP, test\_cluster-method (eap), 39
- find\_segment, 40
- getSolution, 41
- getSolution, list-method (getSolution), 41
- getSolution, output\_Static-method (getSolution), 41
- getSolutionAttributes, 42, 42
- info\_1pl, 43
- info\_2pl (info\_1pl), 43
- info\_3pl (info\_1pl), 43
- info\_gpc (info\_1pl), 43
- info\_gr (info\_1pl), 43
- info\_item (info\_1pl), 43
- info\_pc (info\_1pl), 43
- iparPosteriorSample, 45
- item, 7, 9, 11, 12, 14, 16, 19, 83, 84
- item (item-classes), 45
- item-classes, 45
- item\_1PL, 45, 53
- item\_1PL-class (item-classes), 45
- item\_2PL, 45, 53
- item\_2PL-class (item-classes), 45
- item\_3PL, 45, 53
- item\_3PL-class (item-classes), 45
- item\_attrib, 5, 35–39, 46–48, 51, 86
- item\_attrib-class, 46
- item\_attrib-operators, 47
- item\_GPC, 45, 53
- item\_GPC-class (item-classes), 45
- item\_GR, 45, 53
- item\_GR-class (item-classes), 45
- item\_PC, 45, 53
- item\_PC-class (item-classes), 45
- item\_pool, 5, 7, 9–12, 14, 16, 17, 19, 35–40, 45, 47–55, 57, 59, 61, 62, 65, 71, 83, 84, 89
- item\_pool-class, 48
- item\_pool-operators, 49
- item\_pool\_cluster, 50, 54, 56
- item\_pool\_cluster-class, 50
- itemattrib\_bayes (dataset\_bayes), 36
- itemattrib\_bayes\_data (dataset\_bayes), 36
- itemattrib\_fatigue (dataset\_fatigue), 37
- itemattrib\_fatigue\_data (dataset\_fatigue), 37
- itemattrib\_reading (dataset\_reading), 38
- itemattrib\_reading\_data (dataset\_reading), 38
- itemattrib\_science (dataset\_science), 38
- itemattrib\_science\_data (dataset\_science), 38
- itemcontent\_fatigue\_data (dataset\_fatigue), 37

- itempool\_bayes (dataset\_bayes), 36
- itempool\_bayes\_data (dataset\_bayes), 36
- itempool\_fatigue (dataset\_fatigue), 37
- itempool\_fatigue\_data (dataset\_fatigue), 37
- itempool\_reading (dataset\_reading), 38
- itempool\_reading\_data (dataset\_reading), 38
- itempool\_science (dataset\_science), 38
- itempool\_science\_data (dataset\_science), 38
- itempool\_se\_bayes\_data (dataset\_bayes), 36
- list, 42
- lnHyperPars, 51
- loadConstraints, 28, 51, 51, 52, 78, 80, 85, 96
- loadItemAttrib, 46, 47, 51, 86
- loadItemAttrib (item\_attrib-class), 46
- loadItemPool, 47, 49, 51, 52, 52, 53
- loadStAttrib, 51, 86
- loadStAttrib (st\_attrib-class), 86
- logitHyperPars, 54
- makeItemPoolCluster, 54
- makeItemPoolCluster, item\_pool-method (makeItemPoolCluster), 54
- makeTest, 55, 55
- makeTest, item\_pool-method (makeTest), 55
- makeTestCluster, 56
- makeTestCluster, item\_pool\_cluster, numeric, list-method (makeTestCluster), 56
- makeTestCluster, item\_pool\_cluster, numeric, numeric-method (makeTestCluster), 56
- MLE (mle), 56
- mle, 56, 56, 58
- mle, item\_pool-method (mle), 56
- MLE, test-method (mle), 56
- MLE, test\_cluster-method (mle), 56
- mlef, 58, 58, 60
- mlef, item\_pool-method (mlef), 58
- names, item\_attrib-method (item\_attrib-operators), 47
- names, st\_attrib-method (st\_attrib-operators), 87
- OAT, 4
- OAT (app), 4
- output\_Shadow, 41, 60, 61, 65, 66
- output\_Shadow-class, 60
- output\_Shadow\_all, 61, 65, 66, 80
- output\_Shadow\_all-class, 61
- output\_Static, 41, 62, 65, 71, 85
- output\_Static-class, 62
- p\_1pl, 75
- p\_2pl (p\_1pl), 75
- p\_3pl (p\_1pl), 75
- p\_gpc (p\_1pl), 75
- p\_gr (p\_1pl), 75
- p\_item (p\_1pl), 75
- p\_pc (p\_1pl), 75
- plot, 63, 65–68, 70, 72
- plot, constraints-method (plot), 63
- plot, item\_pool-method (plot), 63
- plot, output\_Shadow-method (plot), 63
- plot, output\_Shadow\_all-method (plot), 63
- plot, output\_Static-method (plot), 63
- plotCAT, 66
- plotCAT, list-method (plotCAT), 66
- plotCAT, output\_Shadow-method (plotCAT), 66
- plotCAT, output\_Shadow\_all-method (plotCAT), 66
- plotExposure, 68
- plotExposure, list-method (plotExposure), 68
- plotExposure, output\_Shadow\_all-method (plotExposure), 68
- plotInfo, 70
- plotInfo, constraints-method (plotInfo), 70
- plotInfo, item\_pool-method (plotInfo), 70
- plotInfo, output\_Static-method (plotInfo), 70
- plotShadow, 72
- plotShadow, list-method (plotShadow), 72
- plotShadow, output\_Shadow\_all-method (plotShadow), 72
- print, 73, 82
- print, config\_Shadow-method (print), 73
- print, config\_Static-method (print), 73
- print, constraints-method (print), 73
- print, exposure\_rate\_plot-method (print), 73
- print, item\_1PL-method (print), 73

- print,item\_2PL-method (print), 73
- print,item\_3PL-method (print), 73
- print,item\_attrib-method (print), 73
- print,item\_GPC-method (print), 73
- print,item\_GR-method (print), 73
- print,item\_PC-method (print), 73
- print,item\_pool-method (print), 73
- print,output\_Shadow-method (print), 73
- print,output\_Shadow\_all-method (print), 73
- print,output\_Static-method (print), 73
- print,st\_attrib-method (print), 73
- print,summary\_constraints-method (print), 73
- print,summary\_item\_attrib-method (print), 73
- print,summary\_item\_pool-method (print), 73
- print,summary\_output\_Shadow\_all-method (print), 73
- print,summary\_output\_Static-method (print), 73
- print.default, 74
- RE, 77
- resp\_fatigue\_data (dataset\_fatigue), 37
- RMSE, 77
- rownames,item\_attrib-method (item\_attrib-operators), 47
- rownames,st\_attrib-method (st\_attrib-operators), 87
- runAssembly, 78, 78
- saveOutput, 79
- Shadow, 5, 28, 52, 55, 68, 69, 72, 78, 79, 79, 80
- Shadow, config\_Shadow-method (Shadow), 79
- show, 81
- show,config\_Shadow-method (show), 81
- show,config\_Static-method (show), 81
- show,constraints-method (show), 81
- show,exposure\_rate\_plot-method (show), 81
- show,item\_1PL-method (show), 81
- show,item\_2PL-method (show), 81
- show,item\_3PL-method (show), 81
- show,item\_attrib-method (show), 81
- show,item\_GPC-method (show), 81
- show,item\_GR-method (show), 81
- show,item\_PC-method (show), 81
- show,item\_pool-method (show), 81
- show,item\_pool\_cluster-method (show), 81
- show,output\_Shadow-method (show), 81
- show,output\_Shadow\_all-method (show), 81
- show,output\_Static-method (show), 81
- show,pool\_cluster-method (show), 81
- show,st\_attrib-method (show), 81
- show,summary\_constraints-method (show), 81
- show,summary\_item\_attrib-method (show), 81
- show,summary\_item\_pool-method (show), 81
- show,summary\_output\_Shadow\_all-method (show), 81
- show,summary\_output\_Static-method (show), 81
- showConstraints, 82
- simResp, 83, 83, 84
- simResp,item\_1PL,numeric-method (simResp), 83
- simResp,item\_2PL,numeric-method (simResp), 83
- simResp,item\_3PL,numeric-method (simResp), 83
- simResp,item\_GPC,numeric-method (simResp), 83
- simResp,item\_GR,numeric-method (simResp), 83
- simResp,item\_PC,numeric-method (simResp), 83
- simResp,item\_pool,numeric-method (simResp), 83
- simResp,item\_pool\_cluster,list-method (simResp), 83
- simResp,item\_pool\_cluster,numeric-method (simResp), 83
- st\_attrib, 5, 35, 38, 51, 86, 87
- st\_attrib-class, 86
- st\_attrib-operators, 87
- Static, 5, 33, 52, 78, 85, 85
- Static,config\_Static-method (Static), 85
- stimattrib\_reading (dataset\_reading), 38
- stimattrib\_reading\_data (dataset\_reading), 38
- subsetConstraints (constraints-operators), 35
- subsetItemPool (item\_pool-operators), 49
- subsetTest (test\_operators), 90

- summary, [88](#)
- summary, constraints-method (summary), [88](#)
- summary, item\_attrib-method (summary), [88](#)
- summary, item\_pool-method (summary), [88](#)
- summary, output\_Shadow\_all-method (summary), [88](#)
- summary, output\_Static-method (summary), [88](#)
- summary-classes, [89](#)
- summary\_constraints-class (summary-classes), [89](#)
- summary\_item\_attrib-class (summary-classes), [89](#)
- summary\_item\_pool-class (summary-classes), [89](#)
- summary\_output\_Shadow\_all-class (summary-classes), [89](#)
- summary\_output\_Static-class (summary-classes), [89](#)
  
- test, [55](#), [89](#), [90](#)
- test-class, [89](#)
- test\_cluster, [40](#), [56](#), [90](#)
- test\_cluster-class, [90](#)
- test\_operators, [90](#)
- TestDesign, [4](#), [80](#), [89](#), [89](#)
- theta\_EAP, [91](#)
- theta\_EAP\_matrix, [91](#)
- theta\_EB, [92](#)
- theta\_EB\_single (theta\_EB), [92](#)
- theta\_FB, [94](#)
- theta\_FB\_single, [95](#)
- toggleConstraints, [96](#), [96](#)
  
- updateConstraints, [96](#), [97](#)