Package 'irt'

November 18, 2020

Type Package

Title Item Response Theory and Computerized Adaptive Testing Functions

Version 0.1.3

Maintainer Emre Gonulates <egonulates@gmail.com>

Description A collection of Item Response Theory (IRT) and Computerized Adaptive Testing (CAT) functions that are used in psychometrics.

License GPL-3

Depends methods

LinkingTo Rcpp

Imports Rcpp (>= 1.0.4), ggplot2, parallel

NeedsCompilation yes

RoxygenNote 7.1.1

Encoding UTF-8

LazyData true

Collate 'Item-class.R' 'Item-class-methods.R' 'Itempool-class.R' 'Itempool-class-methods.R' 'RcppExports.R' 'Testlet-class-methods.R' 'ability_estimation.R' 'bilog.R' 'cat_sim.R' 'cat_sim_helper_functions.R' 'dif.R' 'info.R' 'ipd.R' 'irt.R' 'item_analysis.R' 'item_fit.R' 'misc.R' 'person_fit.R' 'plot_IRT.R' 'prob.R' 'resp_lik.R' 'resp_loglik.R' 'rsss.R' 'sim_resp.R'

Author Emre Gonulates [aut, cre] (<https://orcid.org/0000-0002-3834-3266>)

Repository CRAN

Date/Publication 2020-11-18 08:50:05 UTC

R topics documented:

`add_misc` *Add or change a named value to 'misc' slot of an* `Item-class`*,* `Itempool-class` *or* `Testlet-class` *object.*

Description

Add or change a named value to 'misc' slot of an `Item-class`, `Itempool-class` or `Testlet-class` object.

Usage

```
add_misc(ip, value)

## S4 method for signature 'Item'
add_misc(ip, value)

## S4 method for signature 'Testlet'
add_misc(ip, value)

## S4 method for signature 'Itempool'
add_misc(ip, value)
```

Arguments

`ip`	An `Item-class`, `Testlet-class` or `Itempool-class` object.
`value`	A list where each element should be named. Elements within the list will be added to 'misc' slot.

Value

An object with added 'misc' slot.

Author(s)

Emre Gonulates

Examples

```
item <- item(b = 1)
add_misc(item, list(sympson_hetter_k = .75))
```

as.data.frame.Itempool *Convert an* Itempool-class *object into a* data.frame.

Description

This function converts Itempool-class objects to a data.frame object. All of the items in the item pool should be the same model. If the items belongs to different psychometric models, then, a list of items will be returned instead.

This function converts Item-class objects to a data.frame object.

This function converts Testlet-class objects to a data.frame object. If testlet has an ID, an additional column will be created for the testlet ID.

Usage

```
## S3 method for class 'Itempool'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., include_se = TRUE)

## S3 method for class 'Item'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., include_se = TRUE)

## S3 method for class 'Testlet'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., include_se = TRUE)
```

Arguments

x	An Testlet-class object
row.names	NULL or a character vector giving the row name for the data frame. Missing values are not allowed.
optional	logical. If TRUE, setting row names and converting column names
...	additional arguments
include_se	If TRUE, and items have se_parameters, those will be included in the data frame.

Value

A data frame of items within each row. If all items cannot be coerced to a data.frame, an list of items will be returned and a warning will be raised.

A data frame representation of the item.

A data frame representation of the item.

Author(s)

Emre Gonulates

Emre Gonulates

Emre Gonulates

Examples

```
ip1 <- generate_ip()
as.data.frame(ip1)

ip2 <- generate_ip(n = 10, model = "GRM",
                   content = sample(c("G", "A"), 10, TRUE),
                   id = paste0("grm-i-", 1:10))
as.data.frame(ip2)

t1 <- generate_testlet(n = 3, item_id_preamble = "t1")
t2 <- generate_testlet(n = 2, item_id_preamble = "t2")
ip3 <- c(ip1, t1, t2)
as.data.frame(ip3)

ip4 <- c(ip2, ip3)
as.data.frame(ip4)


item1 <- item(a = 1.12, b = -2.1, c = 0.28)
item2 <- item(a = 2, b = 3.2, c = 0.21)

ip1 <- c(item1, item2)
as.data.frame(ip1)
item1 <- generate_item()
as.data.frame(item1)
testlet1 <- generate_testlet()
as.data.frame(testlet1)
testlet2 <- generate_testlet(id = "T1")
as.data.frame(testlet2)
```

---

as.Itempool *Coerce a given object to* Itempool-class *object*

Description

This function is a wrapper for itempool function. It is recommended to use that function.

Usage

```
as.Itempool(...)
```

Arguments

...	The object that is desired to be converted to an 'Itempool' object. Also additional arguments related to the Itempool.

Value

An Itempool-class object.

Author(s)

Emre Gonulates

See Also

itempool

---

as.list.Itempool *This function converts Itempool objects to a list object*

Description

This function converts Itempool objects to a list object

Usage

```
## S3 method for class 'Itempool'
as.list(x, ...)
```

Arguments

x	an Itempool-class to be coerced to a list object
...	Additional parameters to be passed to the function.

Value

A list object with elements from 'Item' class.

Author(s)

Emre Gonulates

Examples

```
item1 <- item(a = 1.12, b = -2.1, c = 0.28)
item2 <- item(a = 2, b = 3.2, c = 0.21)

ip1 <- c(item1, item2)
as.list(ip1)
```

`biserial` *Calculate biserial correlation*

Description

Calculate biserial correlation

Usage

```
biserial(score, total_score, method = "default")
```

Arguments

`score`	Item scores of each examinee for which biserial correlation will be calculated
`total_score`	Total score of each examinee
`method`	Type of the biserial correlation calculation method.
	"default" The most common way to calculate biserial correlation.
	"point-biserial" Calculate point-biserial correlation.
	"clemans-lord" Modified biserial correlation value based on Clemans (1958) and Lord (1962).
	"brogden" Modified biserial correlation value based on: Brogden (1949)
	"rank" Rank biserial correlation value based on Cureton (1968).

Value

Biserial correlation value

Author(s)

Emre Gonulates

References

Brogden, H. E. (1949). A new coefficient: Application to biserial correlation and to estimation of selective efficiency. Psychometrika, 14, 169-182.

Clemans, W. V. (1958) An index of item-criterion relationship. Educational and Psychological Measurement, 18, 167-172.

Cureton, E. E. (1968). Rank biserial correlation when ties are present. Educational and Psychological Measurement, 28, 77-79.

Kraemer, H. C. (1981). Modified biserial correlation coefficients. Psychometrika, 46(3), 275-282.

Lord, F. M. (1963). Biserial estimates of correlation. Psychometrika, 28, 81–85.

Examples

```
# The example is from Salkind, Rasmussen (2007) Encyclopedia of measurement
# and statistics, pages 94-97
score <- c(rep(0, 16), rep(1, 22))
total_score <- c(87, 90, 94, 94, 97, 103, 103, 104, 106, 108, 109, 109, 109,
                 112, 119, 132, 100, 103, 103, 106, 112, 113, 114, 114, 118,
                 119, 120, 120, 124, 133, 135, 135, 136, 141, 155, 157, 159,
                 162)
# Calculate biserial correlation
biserial(score, total_score)
# Calculate point-biserial correlation
biserial(score, total_score, method = "point-biserial")
# Calculate modified biserial correlation (based on Brogden (1949))
biserial(score, total_score, method = "brogden")
# Calculate modified biserial correlation (Clemans-Lord)
biserial(score, total_score, method = "clemans-lord")
```

`c,Item-method` *Concatenate* `Item`, `Itempool` *or* `Testlet` *objects and return an Itempool object.*

Description

If the elements do not have id fields, function will assign default names.

Usage

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

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

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

Arguments

`x`	A list consist of `Item-class` objects.
`...`	Additional arguments

Value

An `Itempool-class` object.

Author(s)

Emre Gonulates

Examples

```
item1 <- item(a = 1.12, b = -2.1, c = 0.28)
item2 <- item(a = 2, b = 3.2, c = 0.21)

# Concatenate items
c(item1, item2)

ip <- itempool(a = c(1, 1.2), b = c(1, 2), c = c(.2, .4))
# Concatenate items and an Itempool object
c(item1, ip)
c(item1, item2, ip)
c(ip, item1, item2)
```

`calculate_exposure_rates` *Calculate exposure rate of items for CAT*

Description

This function calculates the exposure rate of items for a CAT. It takes a list of `cat_output` objects and `cat_design` object and returns exposure rate of each item.

Usage

```
calculate_exposure_rates(cat_sim_output, cd = NULL, item_ids = NULL)
```

Arguments

`cat_sim_output`	This is a list object containing elements that are "cat_output" class.
`cd`	A `cat_design` object that is created by function `create_cat_design`.
`item_ids`	A vector of Item (or Testlet) ids in the item pool.

Value

This function returns a numeric vector of each item's exposure rate where the names of each exposure rate value is the item's id.

Author(s)

Emre Gonulates

See Also

`cat_sim`

Examples

```
cd <- create_cat_design(ip = generate_ip(n = 30), next_item_rule = 'mfi',
                        termination_rule = 'max_item',
                        termination_par = list(max_item = 10))
cat_data <- cat_sim(true_ability = rnorm(10), cd = cd)
calculate_exposure_rates(cat_data, cd = cd)
```

`calculate_overlap_rates` *Calculate overlap rate of items for CAT*

Description

This function calculates the overlap rate of items for a CAT. It takes a list of `cat_output` objects and `cat_design` object and returns exposure rate of each item.

Usage

```
calculate_overlap_rates(cat_sim_output, cd = NULL, item_ids = NULL)
```

Arguments

`cat_sim_output`	This is a list object containing elements that are "cat_output" class.
`cd`	A `cat_design` object that is created by function `create_cat_design`.
`item_ids`	A vector of item (or Testlet) ids in the item pool.

Value

This function returns a numeric vector of each item's overlap rate where the names of each overlap rate value is the item's id.

Author(s)

Emre Gonulates

See Also

`cat_sim`

Examples

```
cd <- create_cat_design(ip = generate_ip(n = 30), next_item_rule = 'mfi',
                        termination_rule = 'max_item',
                        termination_par = list(max_item = 10))
cat_data <- cat_sim(true_ability = rnorm(10), cd = cd)
calculate_overlap_rates(cat_data, cd = cd)
```

cat_sim *Computerized Adaptive Test (CAT) Simulation*

Description

cat_sim function simulates computerized adaptive test (CAT) for one or more simulees. For long simulations, cat_sim_fast function can be used.

Usage

```
cat_sim(true_ability, cd, verbose = -1)
```

Arguments

true_ability	True ability vector to generate item responses.
cd	A cat_design object that is created by function create_cat_design.
verbose	This is an integer that will print the stage of the test. For example, if the value verbose = 10, a message will be printed at each tenth iteration of the cat_simulation. Default value is -1, where no message will be printed. If the value is 0, only the start time and end time of the simulation will be printed.

Value

If the length of true_ability vector is one a "cat_output" class output will be returned. This is a list containing following elements:

true_ability True ability (theta) value to generate item responses.

est_history A list where each element represent a step of the CAT test. It has following elements:

- **est_before** The estimated ability before the administration of the item.
- **se_before** The standard error of the estimated ability before the administration of the item.
- **testlet** TRUE if the item belongs to a testlet.
- **item** Item-class object that is administered at this step.
- **resp** The simulated response of the simulee for the item administered at this step using simulee's true_ability value.
- **est_after** The estimated ability after the administration of the item.
- **se_after** The standard error of the estimated ability after the administration of the item.

If the length of the true_ability is more than 1, a list of cat_output objects will be returned for each value of true_ability.

Author(s)

Emre Gonulates

See Also

create_cat_design

Examples

```
ip <- generate_ip(n = 50)
# Check the default:
cd <- create_cat_design(ip = ip)
cat_sim(true_ability = rnorm(1), cd = cd)
```

`cat_sim_fast` *Computerized Adaptive Test (CAT) Simulation (Parallel Computing)*

Description

`cat_sim_fast` function simulates computerized adaptive test (CAT) for one or many simulees. This function uses parallel computing, so, for large number of simulees, it might be significantly faster than `cat_sim` function.

Usage

```
cat_sim_fast(true_ability, cd, verbose = -1, n_cores = NULL)
```

Arguments

`true_ability`	True ability vector to generate item responses.
`cd`	A `cat_design` object that is created by function `create_cat_design`.
`verbose`	This is an integer that will print the stage of the test. For example, if the value verbose = 10, a message will be printed at each tenth iteration of the cat_simulation. Default value is `-1`, where no message will be printed. If the value is `0`, only the start time and end time of the simulation will be printed.
`n_cores`	an integer specifying the number of cores to be used. The value should be 1 or larger. The default is `NULL` where the maximum number of cores of the processor will be used.

Value

If the length of `true_ability` vector is one a `"cat_output"` class output will be returned. This is a list containing following elements:

true_ability True ability (theta) value to generate item responses.

est_history A list where each element represent a step of the CAT test. It has following elements:

- **est_before** The estimated ability before the administration of the item.
- **se_before** The standard error of the estimated ability before the administration of the item.
- **testlet** `TRUE` if the item belongs to a testlet.
- **item** `Item-class` object that is administered at this step.
- **resp** The simulated response of the simulee for the item administered at this step using simulee's `true_ability` value.
- **est_after** The estimated ability after the administration of the item.

se_after The standard error of the estimated ability after the administration of the item.

If the length of the `true_ability` is more than 1, a list of `cat_output` objects will be returned for each value of `true_ability`.

Author(s)

Emre Gonulates

See Also

`create_cat_design`

Examples

```
cd <- create_cat_design(ip = generate_ip(n = 30),
                        termination_rule = c('max_item'),
                        termination_par = list(max_item = 7))
cat_sim_fast(true_ability = rnorm(1), cd = cd, n_cores = 1)

cat_sim_fast(true_ability = rnorm(2), cd = cd, n_cores = 1)
```

---

create_cat_design *Computerized Adaptive Test (CAT) Simulation Design*

Description

`create_cat_design` is a helper function for `cat_sim` and `cat_sim_fast` functions. It defines the simulation design.

Ideally, there is a design element for each item. So within this design (which is a list), there are k design elements for each potentially administered item. Each of these sub-design elements are also a list.

Usage

```
create_cat_design(
  ip = NULL,
  title = NULL,
  true_ip = NULL,
  first_item_rule = "fixed_theta",
  first_item_par = list(theta = 0),
  next_item_rule = "mfi",
  next_item_par = NULL,
  ability_est_rule = "eap",
  ability_est_par = NULL,
  final_ability_est_rule = NULL,
  final_ability_est_par = NULL,
```

```
  termination_rule = c("min_item", "min_se", "max_item"),
  termination_par = list(min_item = 10, min_se = 0.33, max_item = 20),
  exposure_control_rule = NULL,
  exposure_control_par = NULL,
  content_bal_rule = NULL,
  content_bal_par = NULL,
  ability_type = "theta"
)
```

Arguments

ip	An Itempool-class object containing item parameters, content information, etc. If ip = NULL this means this is an infinite item pool, where b is on demand, c = 0 and a = 1, D = 1.7. If true_ip argument is NULL, this item pool will be used to generate item responses.
title	A string value representing the title of this CAT design.
true_ip	An Itempool-class object which holds the true values of item pool parameters that will be used to generate item responses. This is an optional argument. If it is NULL and ip is not missing, then, item responses will be generated using ip. **Default**: NULL
first_item_rule	The method how the first item is administered. The main effect of this is to select the first item administered to an examinee. If, for example, first item is desired to be a fixed one or randomly selected from the item pool, then set that rule in next_item_rule. **Default**: 'fixed_theta' Possible values and required parameters: **NULL** If no separate first item selection rule is necessary, the first item will be selected using the next_item_rule and it's parameters next_item_par. **"fixed_theta"** Fixed starting value. Required parameters for first_item_par argument if this rule is selected: **theta** The value of the initial theta estimate. **"theta_range"** An initial theta estimate within min_theta and max_theta will be randomly selected. Required parameters for first_item_par argument if this rule is selected: **min_theta** Minimum theta value of the interval. **max_theta** Maximum theta value of the interval.
first_item_par	Parameters for the first item rule. **Default**: list(theta = 0)
next_item_rule	A vector of length one or length maximum test length which is designating the next item selection rules. **Default**: 'mfi'

Note that, currently, if there are testlets in an item pool and a testlet is selected for administration using one of the methods below, all items within that testlet will be administered regardless of the next item selection rule.

Possible values and required parameters:

random Randomly select items from the item pool. Exposure control rules and parameters will be ignored for this selection rule.
Required parameters: None.

mfi Maximum Fisher Information.
Required parameters: None.

mepv Minimum Expected Posterior Variance.
Required Parameters:

"var_calc_method" Which method to use to calculate the posterior variance. See Equation (4) of Choi and Swartz (2009), Comparison of CAT Criteria for Polytomous Items.
Available options are:

`"eap"` Use the variance from expected a posteriori estimation.

`"owen"` Use the variance from Owen's Bayesian estimation. For `"Rasch"`, `"1PL"`, `"2PL"`, `"3PL"` models this is much faster than `"eap"` option above.

b_optimal Select item which has item difficulty that is close to the current ability estimate.
Required parameters: None.

fixed Administer a fixed set of items from the item pool. This is basically a linear fixed length test where the order of items are predefined. Exposure control rules and parameters will be ignored for this selection rule.
Required Parameters:

item_id A vector of the item IDs that should be administered.

`next_item_par` A list of length one or length maximum test length that sets the parameters of next item selection rules. It can also be `NULL`, in which case no parameters necessary for that next item selection procedure.

Default: `NULL`

`ability_est_rule`

A vector of length one or length maximum test length which is designating the next item selection rules.

Default: `"eap"`

Possible values and required parameters:

"eap" Expected-a-posteriori. Required parameters:

prior_dist Distribution of the prior distribution. Available values:
* `norm` for normal distribution, * `unif` for uniform distribution.
The default value is `norm`.

prior_par A vector of prior parameters.
* For normal distribution `c(0,1)`, see `?dnorm` * For uniform distribution `c(-3,3)`, see `?dunif`
The default value is `c(0,1)`.

min_theta Minimum possible value of theta. It is a lower bound.
The default value is `-4`.

max_theta Maximum possible value of theta. It is an upper bound.
The default value is 4.

no_of_quadrature The number of quadrature, more specifically the number of bins the theta range should be divided. The more bins, the more precise (and slower) the estimates will be.
The default value is 50.

"owen" Owen's Bayesian Estimation Required parameters:

prior_mean Prior mean value. The default value is 0.

prior_var Prior variance value.The default value is 1.

"ml" Maximum likelihood estimation using Newton-Raphson algorithm. If this method is used, the standard error of ability estimates are calculated using the inverse information value at this theta estimate.
Required parameters:

min_theta Minimum possible value of theta. It is a lower bound. The default value is -4.

max_theta Maximum possible value of theta. It is an upper bound. The default value is 4.

criterion This value determines the accuracy of estimates. Smaller values lead more accuracy but the speed of estimation reduces as the value of `criterion` decreases. The default value is 0.001.

"eap_ml" Expected-a-posteriori until an imperfect item response string, then switch to Maximum Likelihood estimation. Required parameters:

prior_dist Distribution of the prior distribution.
Available values:
`norm` for normal distribution,
`unif` for uniform distribution.

prior_par A vector of prior parameters. For normal distribution `c(0,1)`, see `?dnorm` For uniform distribution `c(-3,3)`, see `?dunif`

min_theta Minimum possible value of theta. It is a lower bound.

max_theta Maximum possible value of theta. It is an upper bound.

no_of_quadrature The number of quadrature, more specifically the number of bins the theta range should be divided. The more bins, the more precise (and slower) the estimates will be.

"sum_score" Simple sum score. Required parameters: `NULL`

`ability_est_par`

A list of length one or length maximum test length that sets the parameters of ability estimation rules. It can also be `NULL`.

* If `ability_est_rule = "eap"` then the default is `list(prior_dist = "norm",prior_par = list(mean = 0,sd = 2),min_theta = -4,max_theta = 4)` * If `ability_est_rule = "owen"` then the default is `list(prior_mean = 0,prior_var = 1)`

If it is `NULL`, either no parameters necessary for that ability estimation rule or the defaults of that ability selection rule will be selected.

If it is a list of one, it means that the parameters will be the same throughout the test. The names of the list elements will represent the parameter types.

A list of lists with length of maximum test length designate different parameters for different items in the test progress.

`final_ability_est_rule`

The ability estimation method that will be used to calculate the final ability estimate. The methods and the parameters are the same as `ability_est_rule` and `ability_est_par`. Please see those for details.

Default: `NULL`

`final_ability_est_par`

A list of parameters that will be used for the method designated by the `final_ability_est_rule`.

Default: `NULL`

`termination_rule`

This parameter determines how CAT algorithm decides terminate the test.

The order of termination rules is important. The algorithm will check the rules in that order. If for example `termination_rule = c('min_se','max_item')`, first whether the SE smaller than a certain value checked and if it is smaller, then even the maximum number of items haven't been administered, test will terminate.

The `"min_item"` and `"max_item"` has a special property where, for `"min_item"`, if the number of items administered smaller than `min_item`, then test will not terminate regardless of whether other rules satisfied. Similarly, for `"max_item"`, if the number of items is larger than `max_item`, the test will terminate regardless of whether other conditions satisfied or not. If both `"min_item"` and `"max_item"` are in termination rules, then, test will end when both conditions satisfied, i.e. when the number of items administered is equal to or larger than `max_item` value in `termination_par`.

The "test length" refers to "Item" objects, i.e. individual items not testlets. For example, if an item pool has 10 testlets each having 2 items and 15 standalone items which are not within a testlet, then the test length can go up to 35 (2 x 10 + 15).

Default: `c("min_item","min_se","max_item")`

`"termination_rule"` should be a vector that composed of the following termination rules:

`"min_item"` The minimum number of items should be satisfied. If the number of administered items are equal to or larger than this number test ends.

`"max_item"` The maximum number of items should not be exceeded.. If this is missing, then the item pool size will be set as maximum length.

`"min_se"` If the standard error exceeds `min_se` value, then the test will terminate.

`"sprt"` Sequential Probability Ratio Test (SPRT). SPRT tests two hypotheses:
H_0: Examinee's ability $\hat{\theta} = \theta_0$
H_1: Examinee's ability $\hat{\theta} = \theta_1$
After the administration of each item, the likelihood (or log-likelihood) of the response string is calculated at θ_0 and θ_1. The ratio of this likelihood is then compared to two decision points, A and B.

$$LR = \frac{L(\theta = theta_1)}{\theta = theta_0}$$

In order to calculate the lower (A) and upper (B) decision points, one needs to set α and β. α represents the rate of false positive classification errors

$(0 < \alpha < 1)$, i.e. examinees whose true classification is fail but passed at the end of test. β is the rate of false negative classification errors $(0 < \beta < 1)$, i.e. examinees whose true classification is pass but failed at the end of test. A and B can be calculated as:

$$A = \frac{1-\beta}{\alpha}$$

$$B = \frac{\beta}{1-\alpha}$$

If $LR > A$, examinee passes the test and if $LR < B$ examinee fails the test. If $B < LR < A$, test continues until the maximum number of items reached (or some other test termination criteria satisfied.)

`"sprt"` termination rule needs `termination_par`, where the following parameters should be given in a list:

`"theta_0"` The highest theta value that the test developer is willing to fail an examinee.

`"theta_1"` The lowest theta value that the test developer is willing to pass an examinee.

`"alpha"` The rate of false positive classification errors (0 < `alpha` < 1), i.e. examinees whose true classification is fail but passed at the end of test.

`"beta"` The rate of false negative classification errors (0 < `beta` < 1), i.e. examinees whose true classification is pass but failed at the end of test.

Example: `termination_par = list(sprt = list(theta_0 = -.9, theta_1 = -.1, alpha = 0.05, beta = 0.05))`

`termination_par`

A list of termination rule parameters. This is a named list with length equal to the length of `termination_rule` argument. The names of the list elements should correspond to the elements of `termination_rule` argument.

Default: `list(min_item = 10, min_se = 0.33, max_item = 20)`

`exposure_control_rule`

A vector of length one or length maximum test length which is designating the next item selection rules. It can be `NULL` in which case there won't be any exposure control.

Default: `NULL`, No exposure control will be imposed on item selection.

Possible values and required parameters:

`NULL` No exposure control.

"randomesque" Select one of the most informative `num_items` items.

`num_items` The number of items to select from.

`"sympson-hetter"` The algorithm of Sympson-Hetter exposure control is explained in Sympson and Hetter (1985).

This method does not require any additional "exposure_control_par" but each item/testlet should have a "misc" slot like the following `misc = list(sympson_hetter_k = .75)`.

When using 'sympson-hetter' exposure control rule, please ensure that there are sufficient number of items with 'sympson_hetter_k' values 1. Otherwise, examinees might not get a complete test and an error might be raised by the simulation function.

`exposure_control_par`
: A list of length one or maximum test length designating the exposure control for each item. If there are no parameters it will be `NULL`.

  Default: `NULL`

`content_bal_rule`
: Whether a content balancing is imposed on item selection. Default value is `NULL`, where no content balancing will be imposed on item selection.

  Default: `NULL`

  Possible values and required parameters:

  `NULL` No content balancing.

  max_discrepancy Given a target content distribution, the content with maximum discrepancy with target discrepancy will be administered.
  Required parameters:

  target_dist Target content ratios. For example, suppose there are three content areas: Geometry, Algebra and Arithmetic. If the plan for the test is to include 30 Arithmetic items, then, the `target_dist` should be: c(Geometry = .3, Arithmetic = .2, Algebra = .5). The names in the vector should correspond to the names of the content areas in the item pool. `target_dist` should include each content area within the item pool for it to work properly. If the sum of the `target_dist` is larger than 1, it will be converted to ratios.

`content_bal_par`
: Parameters of `content_bal_rule`. A list, a list of lists or `NULL`.

  Default: `NULL`

`ability_type`
: The type of ability the test is measuring. By default it is IRT based single 'theta'.

  `"theta"` Theta for unidimensional IRT models

  `"multi_theta"` Theta vector for multidimensional IRT models (Not Implemented Yet).

  `"cdm"` An attribute vector (Not Implemented Yet).

  `"raw_score"` Raw score (i.e. total score) of an examinee.

  Default: `"theta"`

Value

A `cat_design` object that holds the test specifications of a CAT.

Author(s)

Emre Gonulates

References

Sympson, J., & Hetter, R. D. (1985). Controlling item-exposure rates in computerized adaptive testing. 973–977.

See Also

`cat_sim`

Examples

```
### Example Designs ###
# Fixed length test IRT test with ability estimation EAP-ML
n_items <- 30
ip <- itempool(data.frame(a = runif(n_items, .5, 1.5), b = rnorm(n_items)))
cd <- create_cat_design(ip = ip, next_item_rule = 'random',
                        termination_rule = 'min_item',
                        termination_par = list('min_item' = n_items))
cd
create_cat_design(ip = ip, next_item_rule = 'random')


n_ip <- 55
ip <- itempool(data.frame(a = runif(n_ip, .5, 1.5), b = rnorm(n_ip)))
# Check the default:
create_cat_design()
create_cat_design(ip = ip)

### Termination Rule ###
create_cat_design(
  termination_rule = c('min_item', 'min_se', 'max_item'),
  termination_par = list(min_item = 10, min_se = .33, max_item = 20))

cd <- create_cat_design(ip = ip, termination_rule = c('min_item', 'min_se'),
                        termination_par = list(min_item = 10, min_se = .33))

### Next Item Rule ###
create_cat_design(ip = ip, next_item_rule = 'random', next_item_par = NULL)
create_cat_design(
  ip = ip, termination_rule = c('min_item', 'max_item'),
  termination_par = list(min_item = 20, max_item = 20),
  next_item_rule = 'fixed',
  next_item_par = list(item_id = ip$id[1:20]))

# Linear test where all of the items in the item pool administered in the
# same order as item pool
ip <- generate_ip(n = 15)
create_cat_design(
  ip = ip, termination_rule = c('max_item'),
  termination_par = list(max_item = 15),
  next_item_rule = 'fixed')

# Generate an item pool with two testlets and three standalone items and
# administer first seven items as a linear test.
ip <- c(generate_testlet(n = 2, id = "t1"), generate_ip(n = 3),
        generate_testlet(n = 5, id = "t2"))
create_cat_design(
  ip = ip, termination_rule = c('max_item'),
  termination_par = list(max_item = 7),
  next_item_rule = 'fixed')
```

```
# A linear test where the item order is predefined.
ip1 <- itempool(data.frame(b = rnorm(5)), id = paste0("i",1:5))
cd <- create_cat_design(
  ip = ip1,
  next_item_rule = 'fixed',
  next_item_par = list(item_id = c("i3", "i2", "i4", "i5", "i1")),
  ability_est_rule = "eap",
  termination_rule = 'max_item', termination_par = list(max_item = 5))

### Ability Estimation Rule ###
create_cat_design(
  ability_est_rule = 'eap',
  ability_est_par = list(prior_dist = 'unif',
                         prior_par = list(min = -2, max = 2),
                         min_theta = -4, max_theta = 4,
                         no_of_quadrature = 31))
create_cat_design(
  ability_est_rule = 'ml',
  ability_est_par = list(min_theta = -4, max_theta = 4, criterion = 0.01))

### Exposure Control ###
create_cat_design(exposure_control_rule = 'randomesque',
                exposure_control_par = list(num_items = 1))

# 5-4-3-2-1 exposure control
create_cat_design(
  exposure_control_rule = 'randomesque',
  exposure_control_par = lapply(c(5:1, rep(1, 15)),
                                function(x) list(num_items = x)))

### Content Balancing ###
create_cat_design(
  content_bal_rule = 'max_discrepancy',
  content_bal_par = list(target_dist = c(
    Geometry = .3, `Rational Numbers` = .2, Algebra = .5)))
```

`dif` *Evaluate Differential Item Functioning (DIF) of a test*

Description

`dif` evaluates Differential Item Functioning (DIF) of a test.

Usage

```
dif(resp, group, focal_name, ip = NULL, type = "mh")
```

Arguments

resp	A vector of item responses.
group	Group membership
focal_name	In the group variable, the value that represents the focal group.
ip	An Itempool-class object.
type	The type of the DIF method.

Value

A data.frame of DIF values.

Author(s)

Emre Gonulates

---

distractor_analysis *Distractor Analysis Function*

Description

Distractor Analysis Function

Usage

```
distractor_analysis(
  raw_resp,
  key,
  criterion = NULL,
  adjusted = FALSE,
  suppress_output = FALSE
)
```

Arguments

raw_resp	A matrix or data.frame containing the item responses.
key	The answer key for the responses
criterion	Provide a continuous criterion variable such as a total raw score, or theta score that will be used in the calculation of correlation calculations. If this value is NULL, the total score will be used.
adjusted	If TRUE, the biserial will be calculated using the total score that is calculated using all of the items except the item that biserial is calculated for.
suppress_output	If TRUE, the function will suppress console output. Default value is FALSE

Value

A list of

'prop' Observed proportions of each choice.

'biserial' Biserial correlation between the examinees selected the choice and the total scores.

'score' Scored response matrix.

Author(s)

Emre Gonulates

Examples

```
n_item <- 10 # sample(8:12, 1)
n_theta <- 50 # sample(100:200, 1)
raw_resp <- matrix(sample(LETTERS[1:4], n_item * n_theta, replace = TRUE),
                   nrow = n_theta, ncol = n_item,
                   dimnames = list(paste0("Examinee-", 1:n_theta),
                                   paste0("Item-", 1:n_item)))
# Add some missing responses
raw_resp[sample(1:length(raw_resp), round(length(raw_resp)*.1))] <- NA
# Prepare answer key
key <- sample(LETTERS[1:4], n_item, replace = TRUE)

# Run distractor analysis:
da <- distractor_analysis(raw_resp = raw_resp, key = key)
```

---

est_ability *Ability Estimation of an examinee*

Description

`est_ability` estimates ability using on various methods such as Owen's Bayesian estimation, Maximum Likelihood estimation, Expected-a-Posteriori.

Usage

```
est_ability(
  ip,
  resp,
  method = "eap",
  ...,
  prior_dist = "norm",
  prior_pars = c(0, 1),
  theta_range = c(-5, 5),
  number_of_quads = 41,
  tol = 1e-06
)
```

Arguments

ip	An Item-class, Itempool-class or a Testlet-class object.
resp	A vector containing examinee responses. If there are missing responses, they will not be included in the ability estimation.
method	The method that will be used to estimate the ability. The default value is "eap". Current methods are: 'sum_score' Basic sum (raw) score of responses. 'owen' Owen's Bayesian Ability Estimation. This estimation method can be used only for dichotomous IRT models, 'Rasch', '1PL', '2PL', '3PL' and '4PL'. Formulas were implemented in Owen (1975) and Vale (1977). Original formulation does not contain D parameter. If D = 1 original solution will be obtained. If D = 1.7 the a parameter will be multiplied with this number. 'ml' Maximum Likelihood Ability Estimation via Newton-Raphson Algorithm 'eap' Expected-a-Posteriori Ability Estimation
...	Additional arguments passed to specific methods
prior_dist	The shape of the prior distribution. Currently following distributions can be specified: **'norm'** Normal distribution **'unif'** Uniform distribution **'t'** t distribution **'cauchy'** Cauchy distribution Default value is 'norm'.
prior_pars	Parameters of the prior distribution. Default value is c(0,1) where 0 is the mean and 1 is the standard deviation of the default prior distribution which is normal distribution. Also, for example, uniform prior parameter can be set as c(a,b) where a is the minimum value and b is the maximum value. For t distribution, prior parameter can be set as df to represent the degree of freedom. For Cauchy distribution, prior parameters can be set as c(location,scale). If method is "owen", provide c(<Prior Mean>,<Prior SD>).
theta_range	The limits of the ability estimation scale. The estimation result will be limited to this interval. The default is c(-5,5).
number_of_quads	Number of quadratures. The default value is 41. As this number increases, the precision of the estimate will also increase. The default value is 41.
tol	The precision level of ability estimate. The final ability estimates will be rounded to remove the precision that is smaller than the tol value. The default value is 1e-06.

Value

est The ability estimated. If the response vector for a subject contains all NAs, then, in order to differentiate all incorrect and all NA, the est returned will be NA.

se The standard error(s) of the ability estimate(s). For "sum_score" method, all of the standard errors will be NA.

Author(s)

Emre Gonulates

References

Owen, R. J. (1975). A Bayesian sequential procedure for quantal response in the context of adaptive mental testing. Journal of the American Statistical Association, 70(350), 351-356.

Vale, C. D., & Weiss, D. J. (1977). A Rapid Item-Search Procedure for Bayesian Adaptive Testing. Research Report 77-4. Minneapolis, MN.

Examples

```
ip <- generate_ip()
resp <- sim_resp(ip, theta = rnorm(1))

# EAP estimation
est_ability(ip, resp)
est_ability(ip, resp, number_of_quads = 81)
est_ability(ip, resp, prior_pars = c(0, 3))
est_ability(ip, resp, prior_dist = 'unif',  prior_pars = c(-3, 3))
est_ability(ip, resp, prior_dist = 't',  prior_pars = 3)
est_ability(ip, resp, prior_dist = 'cauchy',  prior_pars = c(0, 1))

# Maximum Likelihood estimation
est_ability(ip, resp, method = 'ml')
est_ability(ip, resp, method = 'ml', tol = 1e-8)
est_ability(ip, resp = rep(1, length(ip)), method = 'ml')
est_ability(ip, resp = rep(1, length(ip)), method = 'ml',
            theta_range = c(-3, 3))

# Owen's Bayesian ability estimation
est_ability(ip, resp, method = 'owen')
est_ability(ip, resp, method = 'owen', prior_pars = c(0, 3))
```

---

est_bilog *Run BILOG-MG in batch mode*

Description

`est_bilog` runs BILOG-MG in batch mode or reads BILOG-MG output generated by BILOG-MG program. In the first case, this function requires BILOG-MG already installed on your computer under `bilog_exe_folder` directory.

In the latter case, where appropriate BILOG-MG files are present (i.e. "`<analysis_name>.PAR`", "`<analysis_name>.PH1`", "`<analysis_name>.PH2`" and "`<analysis_name>.PH3`" files exist) and `overwrite = FALSE`, there is no need for BILOG-MG program. This function can read BILOG-MG output without BILOG-MG program.

Usage

```
est_bilog(
  x = NULL,
  model = "3PL",
  target_dir = getwd(),
  analysis_name = "bilog_calibration",
  items = NULL,
  id_var = NULL,
  group_var = NULL,
  logistic = TRUE,
  num_of_alternatives = NULL,
  criterion = 0.01,
  num_of_quadrature = 81,
  max_em_cycles = 100,
  newton = 20,
  reference_group = NULL,
  scoring_options = c("METHOD=1", "NOPRINT"),
  calib_options = c("NORMAL"),
  overwrite = FALSE,
  show_output_on_console = TRUE,
  bilog_exe_folder = file.path("C:/Program Files/BILOGMG")
)
```

Arguments

x	Either a data.frame or matrix object. When the data is not necessary, i.e. user only wants to read the BILOG-MG output from the target_dir, then this can be set to NULL.
model	The model of the items. The value is one of the following: "1PL" One-parameter logistic model. "2PL" Two-parameter logistic model. "3PL" Three-parameter logistic model. "CTT" Return only Classical Test theory statistics such as p-values, point-biserial and biserial correlations. The default value is "3PL".
target_dir	The directory/folder where the BILOG-MG analysis and data files will be saved. The default value is the current working directory, i.e. get_wd().
analysis_name	A short file name that will be used for the data files created for the analysis.
items	A vector of column names or numbers of the x that represents the responses. If, in the syntax file, no entry for item names are desired, then, simply write items = "none".
id_var	The column name or number that contains individual subject IDs. If none is provided (i.e. id_var = NULL), the program will check whether the data provided has row names.

group_var
: The column name or number that contains group membership information if multi-group calibration is desired. Ideally, it grouping variable is represented by single digit integers. If other type of data provided, an integer value will automatically assigned to the variables. The default value is NULL, where no multi-group analysis will be performed.

logistic
: A logical value. If TRUE, LOGISTIC keyword will be added to the BILOG-MG command file which means the calibration will assume the natural metric of the logistic response function in all calculations. If FALSE, the logit is multiplied by D = 1.7 to obtain the metric of the normal-ogive model. The default value is TRUE.

num_of_alternatives
: An integer specifying the maximum number of response alternatives in the raw data. 1/num_of_alternatives is used by the analysis as automatic starting value for estimating the pseudo-guessing parameters.

  The default value is NULL. In this case, for 3PL, 5 will be used and for 1PL and 2PL, 1000 will be used.

  This value will be represented in BILOG-MG control file as: NALT = num_of_alternatives.

criterion
: Convergence criterion for EM and Newton iterations. The default value is 0.01.

num_of_quadrature
: The number of quadrature points in MML estimation. The default value is 81. This value will be represented in BILOG-MG control file as: NQPT = num_of_quadrature. The BILOG-MG default value is 20 if there are more than one group, 10 otherwise.

max_em_cycles
: An integer (0, 1, ...) representing the maximum number of EM cycles. This value will be represented in BILOG-MG control file as: CYCLES = max_em_cycles. The default value is 10.

newton
: An integer (0, 1, ...) representing the number of Gauss-Newton iterations following EM cycles. This value will be represented in BILOG-MG control file as: NEWTON = newton.

reference_group
: Represent which group's ability distribution will be set to mean = 0 and standard deviation = 1. For example, if the value is 1, then the group whose code is 1 will have ability distribution with mean 0 and standard deviation 1. When groups are assumed to coming from a single population, set this value to 0.

  The default value is NULL.

  This value will be represented in BILOG-MG control file as: REFERENCE = reference_group.

scoring_options
: A string vector of keywords/options that will be added to the SCORE section in BILOG-MG syntax. Set the value of scoring_options to NULL if scoring of individual examinees is not necessary.

  The default value is c("METHOD=1","NOPRINT") where scale scores will be estimated using Maximum Likelihood estimation and the scoring process will not be printed to the R console (if show_output_on_console = TRUE).

  The main option to be added to this vector is "METHOD=n". Following options are available:

"METHOD=1" Maximum Likelihood (ML)

"METHOD=2" Expected a Posteriori (EAP)

"METHOD=3" Maximum a Posteriori (MAP)

In addition to "METHOD=n" keyword, following keywords can be added:

"NOPRINT": Suppresses the display of the scores on the R console.

"FIT": likelihood ratio chi-square goodness-of-fit statistic for each response pattern will be computed.

"NQPT=(list)", "IDIST=n", "PMN=(list)", "PSD=(list)", "RSCTYPE=n", "LOCATION=(list)", "SCALE=(list)", "INFO=n", "BIWEIGHT", "YCOMMON", "POP", "MOMENTS", "FILE", "READF", "REFERENCE=n", "NFORMS=n"

See BILOG-MG manual for more details about these keywords/options.

`calib_options` A string vector of keywords/options that will be added to the CALIB section in BILOG-MG syntax in addition to the keywords NQPT, CYCLES, NEWTON, CRIT, REFERENCE.

The default value is `c("NORMAL")`.

When "NORMAL" is included in `calib_options`, the prior distributions of ability in the population is assumed to have normal distribution.

When "COMMON" is included in `calib_options`, a common value for the lower asymptote for all items in the 3PL model will be estimated.

Following keywords/options can be added to `calib_options`:

"PRINT=n", "IDIST=n", "PLOT=n", "DIAGNOSIS=n", "REFERENCE=n", "SELECT=(list)", "RIDGE=(a,b,c)", "ACCEL=n", "NSD=n", "COMMON", "EMPIRICAL", "NORMAL", "FIXED", "TPRIOR", "SPRIOR", "GPRIOR", "NOTPRIOR", "NOSPRIOR", "NOGPRIOR", "READPRIOR", "NOFLOAT", "FLOAT", "NOADJUST", "GROUP-PLOT", "RASCH", "NFULL", "CHI=(a,b)".

See BILOG-MG manual for more details about these keywords/options.

NOTE: Do not add any of the following keywords to `calib_options` since they will already be included:

NQPT, CYCLES, NEWTON, CRIT, REFERENCE

`overwrite` If TRUE and there are already a BILOG-MG analysis files in the target path with the same name, these file will be overwritten.

`show_output_on_console`

logical (not NA), indicates whether to capture the output of the command and show it on the R console. The default value is TRUE.

`bilog_exe_folder`

The location of the "blm1.exe", "blm2.exe" and "blm3.exe" files. The default location is `file.path("C:/Program Files/BILOGMG")`.

Value

A list of following objects:

"ip" An Itempool-class object holding the item parameters. This element will not be created when `model = "CTT"`.

"score" A data frame object that holds the number of item examinee has attempted (`tried`), the number of item examinee got right (`right`), the estimated scores of examinees (`ability`), the standard errors of ability estimates (`se`), and the probability of the response string (`prob`). This element will not be created when `model = "CTT"`.

"ctt" The Classical Test Theory (CTT) stats such as p-value, biserial, point-biserial estimated by BILOG-MG. If there are groups, then the CTT statistics for groups can be found in `ctt$group$GROUP-NAME`. Overall statistics for the whole group is at `ctt$overall`.

"failed_items" A data frame consist of items that cannot be estimated.

"syntax" The syntax file.

"converged" A logical value indicating whether a model has been converged or not. If the value is `TRUE`, model has been converged. This element will not be created when `model = "CTT"`.

"neg_2_log_likelihood" -2 Log Likelihood value. This value is `NULL`, when model does not converge. This element will not be created when `model = "CTT"`.

"input" A list object that stores the arguments that are passed to the function.

Author(s)

Emre Gonulates

Examples

```
## Not run:
### Example 1 ###
# Create responses to be used in BILOG-MG estimation
true_ip <- generate_ip(n = 30, model = "2PL")
resp <- sim_resp(true_ip, rnorm(4000))

# The following line will run BILOG-MG, estimate 2PL model and put the
# analysis results under the target directory:
bilog_calib <- est_bilog(x = resp, model = "2PL",
                         target_dir = "C:/Temp/Analysis", overwrite = TRUE)
# Check whether the calibration converged
bilog_calib$converged

# Get the estimated item pool
bilog_calib$ip

# See the BILOG-MG syntax
cat(bilog_calib$syntax)

# See the classical test theory statistics estimated by BILOG-MG:
bilog_calib$ctt

# Get -2LogLikelihood for the model (mainly for model comparison purposes):
bilog_calib$neg_2_log_likelihood


### Example 2 ###
# Get Expected-a-posteriori theta scores:
result <- est_bilog(x = resp, model = "2PL",
```

```
                    scoring_options = c("METHOD=2", "NOPRINT"),
                    target_dir = "C:/Temp/Analysis", overwrite = TRUE)


### Example 3 ###
# Multi-group calibration
ip <- generate_ip(n = 35, model = "3PL", D = 1.7)
n_upper <- sample(1200:3000, 1)
n_lower <- sample(1900:2800, 1)
theta_upper <- rnorm(n_upper, 1.5, .25)
theta_lower <- rnorm(n_lower)
resp <- sim_resp(ip = ip, theta = c(theta_lower, theta_upper))
dt <- data.frame(level = c(rep("Lower", n_lower), rep("Upper", n_upper)), resp)

mg_calib <- est_bilog(x = dt, model = "3PL",
                      group_var = "level",
                      reference_group = "Lower",
                      items = 2:ncol(dt), # Exclude the 'group' column
                      num_of_alternatives = 5,
                      # Use MAP ability estimation.
                      # "FIT": calculate GOF for response patterns
                      scoring_options = c("METHOD=3", "NOPRINT", "FIT"),
                      target_dir = "C:/Temp/Analysis", overwrite = TRUE,
                      show_output_on_console = FALSE)
# Estimated item pool
mg_calib$ip
# Print group means
mg_calib$group_info
# Check Convergence
mg_calib$converged
# Print estimated scores of first five examinees
head(mg_calib$score)


### Example 4 ###
# When user wants to read BILOG-MG output saved in the directory "Analysis/"
# with file names "my_analysis", use the following syntax:
# (The following code does not require an installed BILOG-MG program.)
result <- est_bilog(target_dir = file.path("Analysis/"), model = "3PL",
                    analysis_name = "my_analysis", overwrite = FALSE)


## End(Not run)
```

`generate_ip` *Generate a random* `Itempool` *object*

Description

Generate a random `Itempool` object

Usage

```
generate_ip(
  model = "3PL",
  n = NULL,
  output = "Itempool",
  n_categories = 4,
  se_parameters = NULL,
  ...
)
```

Arguments

model	The model of the item pool
n	The number of items in the item pool.
output	The type of object returned. The default value is "Itempool".
	"Itempool" Return an Itempool-class object.
	"Item" If n = 1 return an Item-class object. If n > 1, returns a list of Item-class object.
	"list" Return a list of item Item-class objects.
n_categories	For polytomous items, designate the number of categories each item should have. It can be a single integer value larger than 1. In this case all of the polytomous items will have this number of categories. It can be a vector of length n designating the categories of each item. For dichotomous items, the values in n_categories will be ignored.
se_parameters	The values of parameter standard errors for each item, i.e. a list object with elements named as parameter names (excluding "D" parameter).
	If the value is TRUE, this function will generate standard error values from a uniform distribution between 0.05 and 0.75 for each parameter of each item.
...	Additional parameters passed to itempool() function.

Value

An Itempool-class object

Author(s)

Emre Gonulates

Examples

```
# By default, a '3PL' model item pool generated
generate_ip()
# Designate the number of items
generate_ip(n = 12)
# Generate item pools for other models
generate_ip(model = "Rasch")
generate_ip(model = "1PL")
```

```
generate_ip(model = "2PL")
generate_ip(model = "4PL")
generate_ip(model = "GRM") # Graded Response Model
generate_ip(model = "GPCM") # Generalized Partial Credit Model
generate_ip(model = "PCM") # Partial Credit Model
generate_ip(model = "GPCM2") # Reparametrized GPCM
# Mixture of models
generate_ip(model = c("4PL", "Rasch"))
generate_ip(model = sample(c("4PL", "GPCM"), 12, TRUE))
generate_ip(model = c("2PL", "GRM", "Rasch"), n = 11)

# Generate parameters standard errors for each item
generate_ip(se_paramters = TRUE)

# Generate an item pool consist of testlets and standalone items
temp_list <- list(ids = paste0("testlet-", 1:7), n = c(2, 3, 4, 2, 3, 4, 2))
ip <- itempool(sample(c(generate_ip(n = 10, output = "list"),
                        sapply(1:length(temp_list$id), function(i)
                          generate_testlet(id = temp_list$id[i],
                                           n = temp_list$item_models[i])))))
```

`generate_item` *Generate a random Item object*

Description

Generate a random Item object

Usage

```
generate_item(model = "3PL", n_categories = 4, se_parameters = NULL, ...)
```

Arguments

`model`	The model of the Item object.
`n_categories`	For polytomous models, the number of categories for an 'item' object.
`se_parameters`	The values of parameter standard errors, i.e. a list object with elements named as parameter names (excluding `"D"` parameter). If the value is `TRUE`, this function will generate standard error values from a uniform distribution between 0.05 and 0.75 for each parameter.
`...`	Additional parameters passed to `itempool()` function.

Value

An `Item-class` object

Author(s)

Emre Gonulates

Examples

```
# By default, a '3PL' model Item generated
generate_item()
# Generate item pools for other models
generate_item("Rasch")
generate_item("1PL")
generate_item("2PL")
generate_item("4PL")
# Polytomous items
generate_item("GRM")
generate_item("GPCM")
generate_item("PCM")
generate_item("GPCM2")
# Different number of categories
generate_item("GRM", n_categories = 2)
generate_item("GPCM", n_categories = 5)

# Generate standard errors for item parameters
generate_item(se_parameters = TRUE)
```

---

generate_testlet *Generate a random Testlet object*

Description

Generate a random Testlet object

Usage

```
generate_testlet(
  model = "BTM",
  n = NULL,
  item_models = "3PL",
  item_id_preamble = NULL,
  n_categories = 4,
  ...
)
```

Arguments

model	The model of the Testlet
n	The number of items in the Testlet.

`item_models` A single model name or a vector of model names with the size of n that represents the models of items in the Testlet object.

`item_id_preamble` The preamble for the item ids within the Testlet.

`n_categories` For polytomous items, designate the number of categories each item should have. It can be a single integer value larger than 1. In this case all of the polytomous items of the testlet will have this number of categories. It can be a vector of length n designating the categories of each item. For dichotomous items, the values in `n_categories` will be ignored.

`...` Additional parameters passed to `testlet()` function.

Value

A `Testlet-class` object

Author(s)

Emre Gonulates

Examples

```
# By default, a Testlet object with '3PL' model items generated
generate_testlet()
# Designate the number of items in the testlet
generate_testlet(n = 12)
# Set the id of the testlet
generate_testlet(id = "my-testlet")
# Designate the id of testlet and preamble for item ids
generate_testlet(id = "my-testlet", item_id_preamble = "mt-")
# Generate item pools for other models
generate_testlet(item_model = "Rasch")
generate_testlet(item_model = "1PL")
generate_testlet(item_model = "2PL")
generate_testlet(item_model = "4PL")
generate_testlet(item_model = "GRM") # Graded Response Model
generate_testlet(item_model = "GPCM") # Generalized Partial Credit Model
generate_testlet(item_model = "PCM") # Partial Credit Model
generate_testlet(item_model = "GPCM2") # Reparametrized GPCM
# Mixture of models
generate_testlet(item_models = c("4PL", "Rasch"))
generate_testlet(model = c("2PL", "GRM", "Rasch"), n = 11)

# Generating multiple testlet objects with custom ids
sapply(paste0("testlet-", 1:4), function(x) generate_testlet(id = x))

# Generate testlet with dichotomous and polytomous with different number of
# categories.
generate_testlet(
  item_models = c("3PL", "GRM", "GPCM", "GRM", "2PL"),
  n_categories = c(2, 3, 6, 7, 2))
```

```
# # Generating multiple testlet objects with custom ids and item models and
# # put them in an item pool:
# temp_list <- list(ids = paste0("testlet-", 1:3),
#                   item_models = c("Rasch", "2PL", "GPCM"))
# itempool(sapply(1:length(temp_list$id), function(i)
#   generate_testlet(id = temp_list$id[i],
#   item_models = temp_list$item_models[i])))
```

`get_cat_administered_items` *Get administered items from a CAT output*

Description

This function returns an item pool object of the administered items using the items in estimate history. If there is one

Usage

```
get_cat_administered_items(cat_sim_output)
```

Arguments

`cat_sim_output` This is a list object containing elements that are "cat_output" class.

Value

For `cat_output` with only one adaptive test, an `Itempool` class object will be returned. For `cat_output` with more than one adaptive tests, a list of `Itempool` class objects will be returned.

Author(s)

Emre Gonulates

Examples

```
cd <- create_cat_design(ip = generate_ip(n = 30), next_item_rule = 'mfi',
                        termination_rule = 'max_item',
                        termination_par = list(max_item = 10))
cat_data <- cat_sim(true_ability = rnorm(10), cd = cd)
get_cat_administered_items(cat_data)
```

`get_cat_response_data` *Extracts the response data of CAT output.*

Description

This function extracts the response data from a single `cat_output` object or a list of `cat_output` objects and gives either a vector (if there is a single `cat_output` object) or a matrix (if there is a lit of `cat_output` objects) of response data.

If `cd`, cat design, object is given, then the item pool in the `cd` will be used.

Usage

```
get_cat_response_data(
  cat_sim_output,
  cd = NULL,
  remove_na = FALSE,
  attach_summary = FALSE
)
```

Arguments

`cat_sim_output`	This is a list object containing elements that are "cat_output" class.
`cd`	A `cat_design` object that is created by function `create_cat_design`.
`remove_na`	If `TRUE`, the columns that are all `NA` will be removed.
`attach_summary`	If `TRUE`, the summary of each CAT will be attached to the beginning of the response string as columns. The default value is `FALSE`.

Value

This function returns a response matrix of adaptive tests. If the input is a list of `cat_output`, then the rows will represent examinees and columns will represent items. For single `cat_output` object the vector names will be the element

Author(s)

Emre Gonulates

See Also

`cat_sim`

Examples

```
n <- 40 # number of items
ip <- generate_ip(n = n,
                  content = sample(c("Algebra", "Arithmetic", "Geometry"),
                                   n, replace = TRUE))
cd <- create_cat_design(ip = ip, next_item_rule = 'mfi',
                        termination_rule = 'max_item',
                        termination_par = list(max_item = 10))
cat_data <- cat_sim(true_ability = rnorm(10), cd = cd)
get_cat_response_data(cat_sim_output = cat_data, cd)
```

---

get_max_possible_total_score

Calculate the maximum score of a set of items

Description

Calculate the maximum score of a set of items

Usage

```
get_max_possible_total_score(ip, resp = NULL)
```

Arguments

ip	An Itempool-class object.
resp	(optional) A response vector or a response matrix. The contents are not important. The function only checks whether an element is missing or not. If an element is missing, then that item will not count towards the maximum possible score. If the maximum score of all items are needed, set resp = NULL.

Value

A vector of numbers showing the maximum possible scores.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(n = 10)
get_max_possible_total_score(ip)
# A mixture of dichotomous and polytomous items
ip <- generate_ip(model = c("3PL", "GRM", "3PL", "GRM", "GRM"),
                  n_categories = c(2, 5, 2, 4, 6))
# 1 + 4 + 1 + 3 + 5 = 14
get_max_possible_total_score(ip)
```

info *Calculates the information of an "Item" object*

Description

This function sets a generic method for calculating the information of a suitable object

Usage

```
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)

## S4 method for signature 'Item'
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)

## S4 method for signature 'Itempool'
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)

## S4 method for signature 'Testlet'
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)

## S4 method for signature 'numMatDfListChar'
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)
```

Arguments

ip	An Item-class, Itempool-class or Testlet-class object.
theta	An vector of ability parameters.
tif	If it is TRUE, function will return total information obtained from each item for a given theta. It simply adds information of individual items.
observed	If TRUE, observed information calculated instead of the default expected information.
resp	A response string (vector or a matrix). Necessary for observed information.

Value

A vector (or matrix) consist of item or test information.

Author(s)

Emre Gonulates

Examples

```
info(ip = generate_item(model = "Rasch"), theta = rnorm(1))
info(ip = generate_item(model = "1PL"), theta = rnorm(1))
info(ip = generate_item(model = "2PL"), theta = rnorm(1))
info(ip = generate_item(model = "3PL"), theta = rnorm(1))
```

```
info(ip = generate_item(model = "4PL"), theta = rnorm(1))
info(ip = generate_item(model = "GRM"), theta = rnorm(1))
info(ip = generate_item(model = "GPCM"), theta = rnorm(1))
info(ip = generate_item(model = "PCM"), theta = rnorm(1))
info(ip = generate_item(model = "GPCM2"), theta = rnorm(1))

info(ip = generate_ip(model = "Rasch"), theta = rnorm(1))
info(ip = generate_ip(model = "1PL"), theta = rnorm(1))
info(ip = generate_ip(model = "2PL"), theta = rnorm(1))
info(ip = generate_ip(model = "3PL"), theta = rnorm(1))
info(ip = generate_ip(model = "4PL"), theta = rnorm(1))
info(ip = generate_ip(model = "GRM"), theta = rnorm(1))
info(ip = generate_ip(model = "GPCM"), theta = rnorm(1))
info(ip = generate_ip(model = "PCM"), theta = rnorm(1))
info(ip = generate_ip(model = "GPCM2"), theta = rnorm(1))

# Multiple Thetas
info(ip = generate_ip(model = "3PL"), theta = rnorm(5))
info(ip = generate_ip(model = "GRM"), theta = rnorm(7))

# Test information function value at theta
info(ip = generate_ip(model = "3PL"), theta = rnorm(5), tif = TRUE)
info(ip = generate_ip(model = "GRM"), theta = rnorm(7), tif = TRUE)

# Information values of an item pool with multiple models
ip <- generate_ip(model = c("2PL", "3PL", "GPCM", "3PL", "GPCM"))
theta <- rnorm(sample(6:10, 1))
info(ip = ip, theta = theta[1])
info(ip = ip, theta = theta)
info(ip = ip, theta = theta, tif = TRUE)

t1 <- generate_testlet(item_models = c("2PL", "3PL", "GRM", "3PL", "GRM"))
theta <- rnorm(sample(6:10, 1))
info(ip = t1, theta = theta[1])
info(ip = t1, theta = theta)
info(ip = t1, theta = theta, tif = TRUE)
```

---

ipd *Item Parameter Drift*

Description

This function detects the unstable (i.e. items whose item parameter values drifted) for a given two sets of items.

Usage

```
ipd(ip1, ip2, method = "robust-z", anchor_item_ids = NULL, alpha = 0.01)
```

Arguments

`ip1`	Itempool object for the first calibration.
`ip2`	Itempool object for the second calibration.
`method`	The method of item parameter drift analysis.
`anchor_item_ids`	Anchor item ids. If `NULL`, all items are assumed to be anchor items.
`alpha`	Two tailed critical value to detect the unstable items. For example if $$alpha = 0.05$$, the critical value is calculated using `qnorm(1-alpha/2)` (= 1.96). Items whose absolute robust z values are larger than this number will be flagged as unstable.
	"robust-z" Robust-Z method based on the Huynh and Meyer (2010).

Value

Return a list depending on the method:

robust-z `output$a$cor` Correlation between two a parameter sets.

- `output$a$sd_ratio` The ratio of the standard deviation of `ip2` to the standard deviation of `ip1`.
- `output$a$robust_z` Robust-z statistic values for each item discrimination parameter.
- `output$a$unstable` Item id's which were flagged if robust z statistic value for a parameters is larger than the absolute value of the critical value (i.e. `qnorm(1-alphe/2)`).
- `output$b$robust_z` Robust-z statistic values for each item difficulty or threshold parameter. If an item has threshold parameters, robust z statistic will be calculated for each threshold.
- `output$b$unstable` Item id's which were flagged if robust z statistic for difficulty/threshold parameters are larger than the absolute value of the critical value (i.e. `qnorm(1-alphe/2)`).

Author(s)

Emre Gonulates

References

Huynh, Huynh and Meyer, Patrick (2010) "Use of Robust z in Detecting Unstable Items in Item Response Theory Models," *Practical Assessment, Research, and Evaluation*: Vol. 15 , Article 2. DOI: https://doi.org/10.7275/ycx6-e864 Available at: https://scholarworks.umass.edu/pare/vol15/iss1/2/

Examples

```
# The example from Huynh and Meyer (2010)
ip1 <- c(itempool(
  a = c(0.729, 0.846, 0.909, 0.818, 0.742, 0.890, 1.741, 0.907, 1.487, 1.228,
        0.672, 1.007, 1.016, 0.776, 0.921, 0.550, 0.624, 0.984, 0.506, 0.594,
        0.687, 0.541, 0.691, 0.843, 0.530, 0.462, 1.007, 0.825, 0.608, 1.177,
        0.900, 0.861, 0.843, 1.404, 0.446, 1.014, 1.632, 0.831, 1.560, 0.798),
```

```
  b = c(1.585, 0.635, -0.378, -0.100, -0.195, 0.749, 1.246, 1.016, -0.234,
        0.537, 0.070, 1.985, 1.101, -0.742, 0.463, -0.060, 0.477, 1.084,
        -2.340, 1.068, -0.055, -1.045, 1.859, 0.645, -0.689, -2.583, 1.922,
        0.709, 0.499, 1.973, 0.104, 0.809, 0.640, 0.247, 0.820, 1.837,
        2.129, 1.012, 1.774, 0.095),
  c = c(0.134, 0.304, 0.267, 0.176, 0.215, 0.194, 0.267, 0.159, 0.095,
        0.197, 0.089, 0.272, 0.229, 0.159, 0.162, 0.100, 0.259, 0.167,
        0.000, 0.242, 0.323, 0.000, 0.196, 0.189, 0.000, 0.000, 0.334,
        0.538, 0.125, 0.511, 0.192, 0.353, 0.103, 0.241, 0.245, 0.118,
        0.155, 0.132, 0.215, 0.148),
  model = "3PL"),
  item(a = 0.561, b = c(0.784, -0.113, 1.166), model = "GPCM"),
  item(a = 0.745, b = c(3.687, 2.506, -0.001), model = "GPCM"))

ip2 <- c(itempool(
  a = c(0.650, 0.782, 0.816, 0.787, 0.611, 0.888, 1.192, 0.589, 1.211,
        0.742, 0.526, 0.690, 0.996, 0.816, 0.781, 0.507, 0.378, 0.976,
        0.473, 0.364, 0.585, 0.566, 0.511, 0.718, 0.354, 1.080, 0.840,
        0.865, 0.528, 0.814, 0.555, 0.701, 0.530, 1.220, 0.344, 0.966,
        1.044, 0.358, 1.192, 0.615),
  b = c(0.676, -0.525, -1.749, -1.092, -1.619, -0.406, -0.132, 0.006,
        -1.352, -0.872, -1.242, 0.873, 0.239, -2.038, -0.487, -1.372,
        -1.492, 0.214, -4.537, 0.220, -0.686, -2.394, 0.747, -0.467,
        -3.629, -5.000, 0.927, 0.305, -0.839, 1.270, -1.618, -0.091,
        -1.228, -1.019, -1.453, 1.090, 1.743, -1.436, 1.024, -1.358),
  c = c(0.110, 0.316, 0.161, 0.149, 0.145, 0.200, 0.243, 0.059, 0.081,
        0.075, 0.028, 0.267, 0.242, 0.189, 0.184, 0.121, 0.000, 0.170,
        0.000, 0.151, 0.383, 0.000, 0.195, 0.177, 0.000, 0.000, 0.352,
        0.647, 0.116, 0.501, 0.000, 0.286, 0.000, 0.248, 0.064, 0.150,
        0.126, 0.000, 0.187, 0.007),
  model = "3PL"),
  item(a = 0.486, b = c(-0.539, -1.489, -0.052), model = "GPCM"),
  item(a = 0.737, b = c(2.599, 1.250, -1.209), model = "GPCM"))
ipd(ip1, ip2)
```

`irt` *A Collection of Item Response Theory (IRT) and Computerized Adaptive Testing (CAT) Functions*

Description

A collection of Item Response Theory (IRT) and Computerized Adaptive Testing (CAT) functions that are used in psychometrics.

Author(s)

Emre Gonulates <egonulates@gmail.com>

`is.Item` *Check whether an object is an* `Item-class`

Description

Check whether an object is an `Item-class`

Check whether an object is an `Itempool-class` object

Check whether an object is a `Testlet-class` object

Usage

```
is.Item(x)

is.Itempool(x)

is.Testlet(x)
```

Arguments

`x`	an object that is checked for being a member of 'Testlet' class

Author(s)

Emre Gonulates

Emre Gonulates

Emre Gonulates

Examples

```
i1 <- item(a = 1, b = 2)
is.Item(i1)
# Alternatively:
is(i1, "Item")

# Not an item:
is.Item("abc")
```

item *Create an* Item *object*

Description

This function is used for creating Item-class objects.

Usage

```
item(
  ...,
  model = NULL,
  id = NULL,
  parameters = NULL,
  se_parameters = NULL,
  content = NULL,
  misc = NULL
)
```

Arguments

...	The item parameter arguments.
model	The model that item parameters represents. Currently model can be: 1PL, 2PL, 3PL, 4PL, M1PL, M2PL and M3PL, GRM, PCM or GPCM. Ideally, a model should be specified for the construction of an Item-class object.
id	Item id. Default value is NULL.
parameters	A list containing numeric vectors that represent item parameters. Depending on the model these can change.
se_parameters	Standard error of item parameters.
content	Content information for item.
misc	This slot is a list where one can put any information about the item. For example, one can enter the id's of the enemies of the current item as misc = list(enemies = c("i1",i2)). Or, one can enter Sympson-Hetter exposure control parameter K: misc = list(sympson_hetter_k = .75).

Value

An Item class object.

Author(s)

Emre Gonulates

Examples

```
# Create 2PL item:
item(a = 1.2, b = -0.94)
item(a = 1.2, b = -0.94, model = "2PL")
# Specify scaling constant D:
item(a = 1.2, b = -0.94, D = 1.7)

# Add additional item specifications:
# Add id
item(a = 1.2, b = -0.94, id = "My-Item-1")
# Add content
item(a = 1.2, b = -0.94, id = "My-Item-1", content = "Geometry")
# Add additional parameter
item(a = 1.2, b = -0.94, misc = list(sympson_hetter_k = 1))
# Add any argument to 'misc' field
i1 <- item(a = 1.2, b = -0.94, id = "item1", content = "Earth Science",
           misc = list(key = "C", operational = TRUE, type = "MC",
                       enemies = c("i2", "i3")))
# Access fields
i1$misc
i1$misc$key
i1$misc$operational
i1$misc$enemies
i1$a
i1$b
i1$D
i1$parameters
i1$id
i1$content


# Rasch Model
item(b = 1.2)
item(b = 1.2, model = "Rasch")

# 1PL model:
item(b = 1.2, model = "1PL")
item(b = 1.2, D = 1)

# 3PL model:
item(a = 0.92, b = 2.7, c = 0.17)
item(a = 0.92, b = 2.7, c = 0.17, model = "3PL")
item(a = 0.92, b = 2.7, c = 0.17, D = 1.7, model = "3PL")

# 4PL model:
item(a = 0.92, b = 2.7, c = 0.17, d = 0.98)
item(a = 0.92, b = 2.7, c = 0.17, d = 0.98, model = "4PL")
item(a = 0.92, b = 2.7, c = 0.17, d = 0.92, D = 1.7, model = "4PL")
item(parameters = list(a = 0.92, b = 2.7, c = 0.17, d = 0.92, D = 1.7),
     model = "4PL")

# Create a GRM model
```

```
item(a = 1.9, b = c(-1, 0.82, 1.5), model = "GRM")
item(parameters = list(a = 1.9, b = c(-1, 2), D = 1), model = "GRM")

# Create a GPCM model
item(a = 1.9, b = c(-1.6, -0.09, 1.25), model = "GPCM")
item(parameters = list(a = 1.9, b = c(-1, 2), D = 1), model = "GPCM")

# Create a GPCM2 model (Reparametrized GPCM model)
item(a = 1.9, b = 0.65, d = c(-1.6, -0.09, 1.25), model = "GPCM2")
item(parameters = list(a = 1.9, b = 0.65, d = c(-1.6, -0.09, 1.25), D = 1.7),
     model = "GPCM2")

# Create a PCM model
item(b = c(-0.7, 0.72, 1.9), model = "PCM")
item(parameters = list(b = c(-1, 2)), model = "PCM")

# Add additional arguments to items
i1 <- item(a = 1.2, b = 2)
i1 <- item(i1, id = "new_item_id", content = "Algebra")
```

`Item-class` *An S4 class to represent an Item*

Description

`Item` is a class to represent an item. An object in Item class should have a `model` name and `parameters`.

Slots

`id` Item id. Default value is `NULL`.

`model` The model that item `parameters` represents. Currently, following models are available:

`"Rasch"` Rasch Model.

Required parameters:

`"b"` Item difficulty parameter.

Probability of correct response at ability estimate θ:

$$P(\theta) = \frac{e^{(\theta - b)}}{1 + e^{(\theta - b)}}$$

Model family: Unidimensional Item Response Theory (UIRT) Models

`"1PL"` Unidimensional One-Parameter Logistic Model.

Required parameters:

`"b"` Item difficulty parameter.

`"D"` Scaling constant. Default value is `1`.

Probability of correct response at ability estimate θ:

$$P(\theta) = \frac{e^{D(\theta-b)}}{1 + e^{D(\theta-b)}}$$

Model family: Unidimensional Item Response Theory (UIRT) Models

`"2PL"` Unidimensional Two-Parameter Logistic Model.

Required parameters:

`"a"` Item discrimination parameter.

`"b"` Item difficulty parameter.

`"D"` Scaling constant. Default value is `1`.

Probability of correct response at ability estimate θ:

$$P(\theta) = \frac{e^{Da(\theta-b)}}{1 + e^{Da(\theta-b)}}$$

Model family: Unidimensional Item Response Theory (UIRT) Models

`"3PL"` Unidimensional Three-Parameter Logistic Model.

Required parameters:

`"a"` Item discrimination parameter.

`"b"` Item difficulty parameter.

`"c"` Pseudo-guessing parameter (lower asymptote).

`"D"` Scaling constant. Default value is `1`.

Probability of correct response at ability estimate θ:

$$P(\theta) = c + (1 - c)\frac{e^{Da(\theta-b)}}{1 + e^{Da(\theta-b)}}$$

Model family: Unidimensional Item Response Theory (UIRT) Models

`"4PL"` Unidimensional Four-Parameter Logistic Model.

Required parameters:

`"a"` Item discrimination parameter.

`"b"` Item difficulty parameter.

`"c"` Pseudo-guessing parameter (lower asymptote).

`"d"` Upper asymptote parameter.

`"D"` Scaling constant. Default value is `1`.

Probability of correct response at ability estimate θ:

$$P(\theta) = c + (d - c)\frac{e^{Da(\theta-b)}}{1 + e^{Da(\theta-b)}}$$

Model family: Unidimensional Item Response Theory (UIRT) Models

`"GRM"` Graded Response Model

Required parameters:

`"a"` Item discrimination parameter.

`"b"` Item threshold parameters (a vector of values). Each value refers to the ability level for which the probability of responding at or above that category is equal to 0.5.

"D" Scaling constant. Default value is 1.

Probability of scoring at or above the category k:

$$P_k^*(\theta) = \frac{e^{Da(\theta - b_k)}}{1 + e^{Da(\theta - b_k)}}$$

Probability of responding at category k where the possible scores are $0, \ldots, m$:

$$P_0(\theta) = 1 - P_1^*(\theta)$$

$$P_1(\theta) = P_1^*(\theta) - P_2^*(\theta)$$

$$\ldots$$

$$P_k(\theta) = P_k^*(\theta) - P_{k+1}^*(\theta)$$

$$\ldots$$

$$P_m(\theta) = P_m^*(\theta)$$

Model family: Polytomous Item Response Theory (PIRT) Models

"GPCM" Generalized Partial Credit Model

Required parameters:

"a" Item discrimination parameter.

"b" Item step difficulty parameters (a vector of values).

"D" Scaling constant. Default value is 1.

Probability of scoring at category k:

$$P_k(\theta) = \frac{exp[\sum_{v=0}^{k} Da(\theta - b_v)]}{\sum_{c=0}^{m-1} exp[\sum_{v=0}^{c} Da(\theta - b_v)]}$$

Model family: Polytomous Item Response Theory (PIRT) Models

"PCM" Partial Credit Model (Masters, 1982)

Required parameters:

"b" Item step difficulty parameters (a vector of values).

Probability of scoring at category k:

$$P_k(\theta) = \frac{exp[\sum_{v=0}^{k} (\theta - b_v)]}{\sum_{c=0}^{m-1} exp[\sum_{v=0}^{c} (\theta - b_v)]}$$

Model family: Polytomous Item Response Theory (PIRT) Models

"GPCM2" An alternative parametrization of Generalized Partial Credit Model "GPCM" where $b_k = b - d_k$. See Muraki (1997), Equation 15 on page 164.

Required parameters:

"a" Item discrimination parameter.

"b" Location parameter.

"d" A vector of threshold parameters.

"D" Scaling constant. Default value is 1.

Probability of scoring at category k:

$$P_k(\theta) = \frac{exp[\sum_{v=0}^{k} Da(\theta - b + d_v)]}{\sum_{c=0}^{m-1} exp[\sum_{v=0}^{c} Da(\theta - b + d_v)]}$$

Model family: Polytomous Item Response Theory (PIRT) Models

A model must be specified for the construction of an `Item` object.

`parameters` A list containing numeric vectors that represent item parameters. Depending on the model these can change.

`se_parameters` Standard error of the item parameters. This should be a list of standard error values. For example, for "2PL", if the parameters are `list(a = 1.2,b = -0.22)`, the standard error values of parameters can be either `NULL` (which is the default value) or `list(a = 0.24,b = 0.42)`. None of the standard error values can be smaller than 0. Individual SE values can be `NA`. For example, `list(a = 0.24,b = NA)` is acceptable, whereas `list(a = 0.24,b = NULL)` is not acceptable.

For models like polytomous items, the SE values should match the parameter values in length. For example, if the parameter values of a `"GPCM"` is `parameters = list(a = 1.4,b = c(-1,0.42,2.1),D = 1.7)`, then the SE values should be like `se_parameters = list(a = .2,b = c(.32,0.34,.3))`. Since the scaling parameter `D` is constant, it does not have a standard error.

`content` Content information for the Item object.

`misc` This slot is a list where one can put any information about the Item object. For example, one can enter the id's of the enemies of the current Item as `misc = list(enemies = c("i1",i2))`. Or, one can enter Sympson-Hetter exposure control parameter K: `misc = list(sympson_hetter_k = .75)`.

References

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

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

`itempool` *Create an* `Itempool` *object*

Description

This method creates a new `Itempool-class` object.

Usage

```
itempool(...)
```

Arguments

`...`	The object that is desired to be converted to an 'Itempool' object. Also additional arguments related to the Itempool.

Value

An Itempool-class object.

Author(s)

Emre Gonulates

Examples

```
# Create an item pool with two 2PL items
itempool(a = c(1, 1.4), b = c(-2, 1))
itempool(a = c(1, 1.4), b = c(-2, 1), model = "2PL")
# Set D parameter
itempool(a = c(1, 1.4), b = c(-2, 1), D = 1.7)
# Set item IDs
itempool(a = c(1, 1.4), b = c(-2, 1), id = c("i1", "i2"))
# Set content
itempool(a = c(1, 1.4), b = c(-2, 1), content = c("Algebra", "Geometry"))

# Create GRM (Graded Response Model) items
# itempool(data.frame(a = rlnorm(10, 0, .3), b1 = rnorm(10), b2 = rnorm(10)),
#          model = "GRM")

# Create a Rasch model item pool
itempool(b = c(-1, 0.2, 1.1), model = "Rasch")

# Add 'misc' field:
ip <- itempool(b = rnorm(2), id = paste0("t1-i", 1:2),
                misc = list(list(sympson_hetter_k = .8),
                            list(sympson_hetter_k = .9)))
ip[[1]]  # First item of the item pool
```

Itempool-class *An S4 class to represent an Itempool*

Description

Itempool-class is a class to represent an item pool. This class is composed of the collection of 'Item' class objects.

Slots

`item_list` The list of items that are 'Item' class

`misc` A list of additional parameters for the item pool. For example, one can put the calibration date of the item pool as `misc = list(calibration_date = as.Date("2020-01-17"))`.

Author(s)

Emre Gonulates

`item_analysis` *Item Analysis Function*

Description

Item Analysis Function

Usage

```
item_analysis(resp, criterion = NULL, suppress_output = FALSE)
```

Arguments

`resp`	A `matrix` or `data.frame` containing the item responses.
`criterion`	Provide a continuous criterion variable such as a total raw score, or theta score that will be used in the calculation of correlation calculations. If this value is `NULL`, the total score will be used.
`suppress_output`	If `TRUE`, the function will suppress console output. Default value is `FALSE`

Value

A list of

'id' Item ID.

'n' Number of examinees responded this item.

'pval' p-value, proportion of examinees correctly answered items.

'pbis' Point biserial correlation.

'bis' Biserial correlation.

'pbis_adj' Point biserial correlation between item and total score without this item.

'bis_adj' Biserial correlation between item and total score without this item.

Author(s)

Emre Gonulates

Examples

```
theta <- rnorm(100)
ip <- generate_ip(n = 20)
resp <- sim_resp(ip = ip, theta = theta, prop_missing = .2)
# Item analysis based on total scores
item_analysis(resp)
# Item analysis based on theta scores
item_analysis(resp, criterion = theta)
```

item_fit *Calculate item-fit indices*

Description

item_fit calculates the fit of an item to a given psychometric model.

Usage

```
item_fit(ip, resp, theta, type = "Q3")
```

Arguments

ip	An Itempool-class object.
resp	A vector of item responses.
theta	An vector containing ability parameters.
type	The type of the item-fit index.

Value

A vector of item-fit index values.

Author(s)

Emre Gonulates

length,Itempool-method *Find the length of an Itempool-class object*

Description

Find the length of an Itempool-class object

Find the length of a Testlet-class object

Usage

```
## S4 method for signature 'Itempool'
length(x)

## S4 method for signature 'Testlet'
length(x)
```

Arguments

x	an Itempool-class object

Author(s)

Emre Gonulates

---

person_fit *Calculate person-fit indices*

Description

person_fit calculates the fit of a person to a given psychometric model.

Usage

```
person_fit(ip, resp, theta, type = "lz")

## S4 method for signature 'Item'
person_fit(ip, resp, theta, type = "lz")

## S4 method for signature 'Itempool'
person_fit(ip, resp, theta, type = "lz")

## S4 method for signature 'Testlet'
person_fit(ip, resp, theta, type = "lz")
```

Arguments

ip	An Item-class, Itempool-class or a Testlet-class object.
resp	A vector of item responses.
theta	An vector containing ability parameters.
type	The type of the person-fit index.

Value

A vector of person-fit index values.

Author(s)

Emre Gonulates

plot.Item *Plot Item Characteristic Curve of an* Item *object*

Description

plot.Item Plots the item characteristic curve.

Usage

```
## S3 method for class 'Item'
plot(
  x,
  theta_range = c(-4, 4),
  title = "",
  suppress_plot = FALSE,
  category_names = FALSE,
  legend_title = NULL,
  ...
)
```

Arguments

x	An Item-class object.
theta_range	The boundaries of x axis.
title	Title of the plot. By default if the item is 1-4PM IRT model then the title will be "Item Characteristic Curve" if the item follows Graded Response Model the title will be "Category Response Functions". Set it NULL to remove it.
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
category_names	If the model used is 'GRM' (Graded Response Model) these names will serve as category names. For example, c("Strongly Disagree", "Disagree", "Agree", "Strongly Agree"). The default is FALSE where the default category scores will be printed. If the value is NULL no legend will be printed but the categories will be printed differently.
legend_title	The title of the plot's legend.
...	Additional arguments that will be passed to geom_line

Value

Depending on the value of suppress_plot function either prints the item characteristic curve or returns the plot object.

Author(s)

Emre Gonulates

Examples

```
plot(x = item(b = 0.3, D = 1))

item <- item(a = 1.2, b = 0.3, c = .2)
plot(item)
plot(item(a = 1.2, b = 0.3, c = .2, d = .89, D = 1))

# Plot Graded Response Model
ip <- item(a = 0.902, b = c(-1.411, 0.385, 1.79), model = "GRM")
plot(ip)
plot(ip, category_names = c("Strongly Disagree", "Disagree", "Agree",
                            "Strongly Agree"))
ip <- item(a = 0.8, b = 1, model = "GRM")
plot(ip, category_names = c("Incorrect", "Correct"), legend_title = "Response")

# # Change the y-axis label
# plot(ip, suppress_plot = TRUE) + ylab("New Label")
```

---

plot.Itempool *Plot Item Characteristic Curves or Test Characteristic Curve of an* Itempool *object*

Description

plot.Itempool plots the item characteristic curves (item response curves) or test characteristic curve of an Itempool-class object.

Usage

```
## S3 method for class 'Itempool'
plot(
  x,
  theta_range = c(-4, 4),
  tcc = FALSE,
  tcc_prop_corr = FALSE,
  title = "",
  suppress_plot = FALSE,
  legend_title = NULL,
  ...
)
```

Arguments

x	An Itempool-class object.
theta_range	The boundaries of x axis.
tcc	If TRUE a test characteristic curve will be plotted.

tcc_prop_corr	If TRUE, test characteristic curve will be show the proportion correct of the test (i.e. the range of y-axis will be 0-1 instead of 0 to the number of items).
title	Title of the plot. Default is NULL. If tcc is TRUE it will be 'Test Characteristic Curve', if FALSE it will be 'Item Characteristic Curve'.
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
legend_title	The title of the plot's legend.
...	Additional arguments that will be passed to geom_line

Value

Depending on the value of suppress_plot function either prints the item characteristic curve or returns the plot object.

Author(s)

Emre Gonulates

Examples

```
n <- sample(10:15,1)
ip <- itempool(a = runif(n, .5, 2), b = rnorm(n), c = runif(n, 0, .3), D = 1)
plot(ip)
# Additional arguments will passed to geom_line
plot(ip, size = .25, alpha = 0.3)
# Test Characteristic Curve
plot(ip, tcc = TRUE)
# Proportion correct for test characteristic curve
plot(ip, tcc = TRUE, tcc_prop_corr = TRUE)
# # Remove the legend altogether
# plot(ip, suppress_plot = TRUE) + theme(legend.position="none")
# # Change the labels:
# plot(ip, suppress_plot = TRUE) + ylab("Probability") + xlab("Ability Score")
```

---

plot_distractor_icc *Plot Empirical Item or Test characteristic curve*

Description

plot_empirical_icc plots empirical item or test characteristic curve.

Usage

```
plot_distractor_icc(
  raw_resp,
  item,
  key,
```

```
  bins = 10,
  ip = NULL,
  theta = NULL,
  x_axis_scale = NULL,
  title = "",
  n_dodge = 1,
  suppress_plot = FALSE,
  ...
)
```

Arguments

raw_resp	Raw response matrix.
item	The column number, column name or the 'id' of the the item that should be plotted.
key	A vector of answer key.
bins	An integer larger than 2 representing of ability groups examinees should be grouped into. The default is 10. The maximum value of bins + 1 is the number of possible total scores.
ip	An Itempool-class object that is needed for some plots. If ip provided and theta is not provided, then ability will be estimated using EAP method with prior mean 0 and prior standard deviation of 1. This is a slower method depending on the size of the data.
theta	A vector of examinee abilities. If theta values provided the bins are formed using them instead of sum scores.
x_axis_scale	Set the scale of the x-axis. The default value is NULL. For if sum score is used scale will be defaulted to "percent", Otherwise if valid theta or ip arguments provided the scale defaults to "theta". "percent" Percent interval. "number" Numbers between 1 and bins. "theta" Theta values equally divided into bins. the middle value of the bin is shown in the x-axis. For example, if bins = 10, the first tick of the x-axis will be the mean of minimum theta value and tenth percentile theta value.
title	Title of the plot
n_dodge	The number of lines the x-axis tick labels should be written to. This is especially useful if the x-axis tick labels overlap with each other. The default value is 1, which means all of the labels are written on the same line.
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
...	Extra parameters that will pass to geom_line.

Value

Depending on the value of suppress_plot function either prints the proportion of examinees in each bin respond to each distractor or returns the plot object.

Author(s)

Emre Gonulates

Examples

```
n_item <- 10 # sample(8:12, 1)
n_theta <- 10000 # sample(100:200, 1)
raw_resp <- matrix(sample(LETTERS[1:4], n_item * n_theta, replace = TRUE),
                   nrow = n_theta, ncol = n_item,
                   dimnames = list(paste0("Examinee-", 1:n_theta),
                                   paste0("Item-", 1:n_item)))
key <- sample(LETTERS[1:4], n_item, replace = TRUE)
plot_distractor_icc(raw_resp, 3, key)
# Change the number of bins
plot_distractor_icc(raw_resp, 3, key, bins = 15)
```

---

plot_empirical_icc *Plot Empirical Item or Test characteristic curve*

Description

plot_emprical_icc plots empirical item or test characteristic curve.

Usage

```
plot_empirical_icc(
  resp,
  item,
  type = "eicc",
  bins = 10,
  ip = NULL,
  theta = NULL,
  title = "",
  suppress_plot = FALSE,
  x_axis_scale = NULL,
  n_dodge = 1,
  ...
)
```

Arguments

resp	Response matrix.
item	The column number, column name or the 'id' of the the item that should be plotted.
type	The type of the graph that will be plotted.

"eicc" Plot empirical item characteristic curve. Examinees will be put into bins based on their total raw scores and the proportion of examinees who correctly answered an item for each bin will be plotted.

"oep" Plot Observed p-values vs. expected p-values grouped into bins based on total raw scores or theta scores. This plot requires an Itempool-class object. Optionally, provide theta vector, otherwise examinee abilities will be estimated by est_ability(...,type = "eap"). This will slow down the plotting function.

bins	An integer larger than 2 representing of ability groups examinees should be grouped into. The default is 10. The maximum value of bins + 1 is the number of possible total scores.
ip	An Itempool-class object that is needed for some plots.
theta	A vector of examinee abilities.
title	Title of the plot
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
x_axis_scale	Set the scale of the x-axis. The default value is NULL. For total score it will be defaulted to "percent".

"percent" Percent interval.

"number" Numbers between 1 and bins

"theta" Theta values equally divided into bins. the middle value of the bin is shown in the x-axis. For example, if bins = 10, the first tick of the x-axis will be the mean of minimum theta value and tenth percentile theta value. This is the only option for type = "oep".

n_dodge	The number of lines the x-axis tick labels should be written to. This is especially useful if the x-axis tick labels overlap with each other. The default value is 1, which means all of the labels are written on the same line.
...	Extra parameters that will pass to geom_line.

Value

Depending on the value of suppress_plot function either prints the empirical item or test characteristic curve or returns the plot object.

Author(s)

Emre Gonulates

Examples

```
# Plot the information function of an item
resp <- sim_resp(ip = generate_ip(model = "3PL", n = 20),
                 theta = rnorm(10000))
plot_empirical_icc(resp, 3)
# Change the number of bins
plot_empirical_icc(resp, 4, bins = 15)
```

plot_info *Plot Item Information Function*

Description

plot_info Plots the item information function.

Usage

```
plot_info(
  ip,
  tif = FALSE,
  theta_range = c(-5, 5),
  title = "",
  suppress_plot = FALSE,
  ...
)
```

Arguments

ip	An Item-class or Itempool-class object.
tif	If TRUE a test information plot will be plotted. The default value is FALSE.
theta_range	The boundaries of x axis.
title	Title of the plot
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
...	Extra parameters that will pass to geom_line.

Value

Depending on the value of suppress_plot function either prints the item information function or returns the plot object.

Author(s)

Emre Gonulates

Examples

```
# Plot the information function of an item
plot_info(item(b = 1))

# Plot information function(s) of an Itempool object
n <- sample(10:20,1)
ip <- itempool(data.frame(a = runif(n, .5, 2), b = rnorm(n),
                          c = runif(n, 0, .3), D = 1))
plot_info(ip)
plot_info(ip, tif = TRUE)
```

plot_resp_loglik *Plot the Log-Likelihood of a response string*

Description

plot_resp_loglik plots the log-likelihood of a response string.

Usage

```
plot_resp_loglik(
  ip,
  resp,
  theta_range = c(-5, 5),
  title = "",
  likelihood = FALSE,
  show_estimate = TRUE,
  suppress_plot = FALSE,
  text_size = 12,
  ...
)
```

Arguments

ip	An Itempool-class class object.
resp	The response string
theta_range	The boundaries of x axis.
title	Title of the Plot
likelihood	If TRUE, likelihood function will be plotted instead of log-likelihood graph. Default value is FALSE.
show_estimate	If TRUE the maximum likelihood ability estimate will be shown. The default value is TRUE.
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
text_size	The overall text size of the axis and titles. The default value is 12.
...	Additional arguments passed to annotate.

Value

Depending on the value of suppress_plot function either prints the Log-likelihood function of the response string or returns the plot object.

To-do

- Make it to plot multiple test information functions. You can input a list each of which contains item parameters. And the name of the test also.

Author(s)

Emre Gonulates

Examples

```
n <- sample(10:50,1)
ip <- itempool(data.frame(a = runif(n, .5, 2), b = rnorm(n),
                          c = runif(n, 0, .3), D = 1.7))
resp <- sim_resp(ip = ip, theta = rnorm(1))
plot_resp_loglik(ip, resp)
plot_resp_loglik(ip, resp, text_size = 9)
# Format the text of the MLE estimate
plot_resp_loglik(ip, resp, size = 3, color = 'blue')
# Suppress the MLE estimate
plot_resp_loglik(ip, resp, show_estimate = FALSE)
```

---

prob *Calculate the probability of a correct response*

Description

prob Returns the probability of correct respond to an item or multiple items with given parameters for a given ability or abilities, i.e. θ. For polytomous models, where there are multiple possible responses, probability of each response category will be returned.

Usage

```
prob(ip, theta, derivative = 0, expected_value = FALSE)

## S4 method for signature 'Item'
prob(ip, theta, derivative = 0, expected_value = FALSE)

## S4 method for signature 'Itempool'
prob(ip, theta, derivative = 0, expected_value = FALSE)

## S4 method for signature 'Testlet'
prob(ip, theta, derivative = 0, expected_value = FALSE)

## S4 method for signature 'numMatDfListChar'
prob(ip, theta, derivative = 0, expected_value = FALSE)
```

Arguments

ip	An Item-class or an Itempool-class object containing the item parameters.
theta	An object containing the ability parameters.
derivative	Whether to calculate the first or second derivative of probability of a response.
	0 No derivative will be calculated. This is the default value.

1 Calculate the first derivative.

2 Calculate the second derivative.

`expected_value` For each possible response value, the probability of that response is calculated and summed to get the expected value at a `theta` value. Default value is `FALSE`.

Value

Item probabilities at given theta will be returned. If `expected_value` is `TRUE`, the expected value(s) of item or item pool at a given `theta` value will be returned.

Author(s)

Emre Gonulates

Examples

```
theta <- rnorm(1)
item1 <- generate_item(model = "2PL")

# Probability of correct response
prob(item1, theta)

# First derivative of probability of correct response:
prob(item1, theta, derivative = 1)

# Second derivative of probability of correct response:
prob(item1, theta, derivative = 2)

# Probability of each response category for Generalized Partial Credit Model
item2 <- generate_item(model = "GPCM", n_categories = 4)
prob(item2, theta)

# First derivative of each response category
prob(item2, theta, derivative = 1)

# Second derivative of each response category
prob(item2, theta, derivative = 2)

# Expected score for a subject with a given theta value
prob(item2, theta, expected_value = TRUE)

# Probability of each response category for Reparametrized Generalized
# Partial Credit Model
item3 <- generate_item(model = "GPCM2", n_categories = 3)
prob(item3, theta)

# Probability of each response category for Graded Response Model
item4 <- generate_item(model = "GRM", n_categories = 5)
prob(item4, theta)

# Multiple theta values
theta_n <- rnorm(5)
```

```
prob(item1, theta_n)
prob(item1, theta_n, derivative = 1)
prob(item1, theta_n, derivative = 2)

prob(item2, theta_n)
prob(item2, theta_n, derivative = 1)
prob(item2, theta_n, derivative = 2)


theta <- rnorm(1)
ip <- generate_ip(model = "3PL")

# Probability of correct response
prob(ip, theta)

# First derivative of probability of correct response:
prob(ip, theta, derivative = 1)

# Second derivative of probability of correct response:
prob(ip, theta, derivative = 2)

# Multiple theta
theta_n <- rnorm(5)
prob(ip, theta_n)
prob(ip, theta_n, derivative = 1)
prob(ip, theta_n, derivative = 2)


# Probability of each response category for Generalized Partial Credit Model
ip <- generate_ip(model = "GPCM", n = 4, n_categories = c(3, 4, 6, 5))
prob(ip, theta)

# First derivative of each response category
prob(ip, theta, derivative = 1)

# Second derivative of each response category
prob(ip, theta, derivative = 2)

# Expected score for a subject with a given theta value for each item
prob(ip, theta, expected_value = TRUE)

# Probability of a mixture of items models
ip <- generate_ip(model = c("GPCM", "2PL", "3PL", "GPCM"),
                  n_categories = c(4, 2, 2, 3))
prob(ip, theta)

theta <- rnorm(1)
t1 <- generate_testlet(model_items = "3PL")

# Probability of correct response
prob(t1, theta)
```

```
# First derivative of probability of correct response:
prob(t1, theta, derivative = 1)

# Second derivative of probability of correct response:
prob(t1, theta, derivative = 2)
```

`resp_lik` *Likelihood of a response string*

Description

`resp_lik` returns the likelihood of a response string for given items and ability.

Usage

```
resp_lik(ip, resp, theta)

## S4 method for signature 'Item'
resp_lik(ip, resp, theta)

## S4 method for signature 'Itempool'
resp_lik(ip, resp, theta)

## S4 method for signature 'Testlet'
resp_lik(ip, resp, theta)
```

Arguments

`ip`	An `Item-class`, `Itempool-class` or a `Testlet-class` object.
`resp`	A vector of item responses.
`theta`	An vector containing ability parameters.

Value

A matrix of likelihood(s)

Author(s)

Emre Gonulates

Examples

```
item <- generate_item(model = "3PL")
theta <- rnorm(6)
resp <- sim_resp(ip = item, theta = theta, prop_missing = .1)
resp_lik(ip = item, resp = resp, theta = theta)
```

```
item <- generate_item(model = "GRM")
resp <- sim_resp(ip = item, theta = theta, prop_missing = .1)
resp_lik(ip = item, resp = resp, theta = theta)
ip <- generate_ip(model = "3PL")
theta <- rnorm(6)
resp <- sim_resp(ip = ip, theta = theta, prop_missing = .1)
resp_lik(ip = ip, resp = resp, theta = theta)

ip <- generate_ip(model = "GRM")
resp <- sim_resp(ip = ip, theta = theta, prop_missing = .1)
resp_lik(ip = ip, resp = resp, theta = theta)
```

`resp_loglik` *Log-likelihood of a Response String*

Description

`resp_loglik` returns the log-likelihood of a response string for given items and ability.

Usage

```
resp_loglik(ip, resp, theta, derivative = 0)

## S4 method for signature 'Item'
resp_loglik(ip, resp, theta, derivative = 0)

## S4 method for signature 'Itempool'
resp_loglik(ip, resp, theta, derivative = 0)

## S4 method for signature 'Testlet'
resp_loglik(ip, resp, theta, derivative = 0)

## S4 method for signature 'numMatDfListChar'
resp_loglik(ip, resp, theta, derivative = 0)
```

Arguments

`ip`	An `Item-class`, `Itempool-class` or a `Testlet-class` object.
`resp`	A vector of item responses.
`theta`	An vector containing ability parameters.
`derivative`	Whether to calculate the first or second derivative of response log-likelihood. `0` No derivative will be calculated. This is the default value `1` Calculate the first derivative of the response log-likelihood `2` Calculate the second derivative of the response log-likelihood

Value

A matrix of log-likelihood(s)

Author(s)

Emre Gonulates

Examples

```
item <- generate_item(model = "3PL")
theta <- rnorm(6)
resp <- sim_resp(ip = item, theta = theta, prop_missing = .1)
resp_loglik(ip = item, resp = resp, theta = theta)

item <- generate_item(model = "GRM")
resp <- sim_resp(ip = item, theta = theta, prop_missing = .1)
resp_loglik(ip = item, resp = resp, theta = theta)
ip <- generate_ip(model = "3PL")
theta <- rnorm(6)
resp <- sim_resp(ip = ip, theta = theta, prop_missing = .1)
resp_loglik(ip = ip, resp = resp, theta = theta)
resp_loglik(ip = ip, resp = resp, theta = theta, derivative = 1)
resp_loglik(ip = ip, resp = resp, theta = theta, derivative = 2)

ip <- generate_ip(model = "GPCM")
resp <- sim_resp(ip = ip, theta = theta, prop_missing = .1)
resp_loglik(ip = ip, resp = resp, theta = theta)
resp_loglik(ip = ip, resp = resp, theta = theta, derivative = 1)
resp_loglik(ip = ip, resp = resp, theta = theta, derivative = 2)
```

---

rsss *Convert raw score to scale score and vice versa*

Description

Convert raw score to scale score and vice versa

Usage

```
rsss(ip, raw_score = NULL, scale_score = NULL, theta_range = c(-5, 5))
```

Arguments

ip	An Itempool-class object.
raw_score	A value (or vector of values) representing raw score(s).
scale_score	A value (or vector of values) representing scale score(s).
theta_range	The limits of the scale score. The default is c(-5,5).

Value

A vector of raw or scale scores.

Author(s)

Emre Gonulates

`sim_resp` *Generate responses for a given model*

Description

`sim_resp` Generate dichotomous (0 or 1) or polytomous responses for given ability and item parameter.

Usage

```
sim_resp(ip, theta, prop_missing = 0)

## S4 method for signature 'Item'
sim_resp(ip, theta, prop_missing = 0)

## S4 method for signature 'Testlet'
sim_resp(ip, theta, prop_missing = 0)

## S4 method for signature 'Itempool'
sim_resp(ip, theta, prop_missing = 0)

## S4 method for signature 'numMatDfListChar'
sim_resp(ip, theta)
```

Arguments

`ip`	An `Item-class`, `Itempool-class`, `Testlet-class` object containing the item parameters.
`theta`	An object containing the subject ability parameters.
`prop_missing`	Proportion of responses that should be missing. Default value is `0`. This argument is valid for only `Itempool-class` and `Testlet-class` objects.

Value

A vector of responses.

Author(s)

Emre Gonulates

Examples

```
## Simulate Responses for an Item object ##
item <- generate_item(model = "3PL")
sim_resp(ip = item, theta = rnorm(1))

item <- generate_item(model = "GPCM")
sim_resp(ip = item, theta = rnorm(1))


item <- generate_item(model = "GRM")
sim_resp(ip = item, theta = rnorm(1))

## Simulate Responses for a Testlet object ##
# Create a testlet
testlet <- testlet(c(item(b = 1), item(a = .8, b = 3.1),
                   item(b = -1:1, model = "PCM")))
sim_resp(ip = testlet, theta = rnorm(1))
## Simulate Responses for an Itempool object ##
# Create 3PL IRT item parameters
ip <- itempool(a = rlnorm(10, 0, 0.3), b = rnorm(10), c = runif(10, 0, .3))
# Simulate responses for one theta:
sim_resp(ip = ip, theta = rnorm(1))
# Simulate responses for eight thetas:
sim_resp(ip = ip, theta = rnorm(8))

# Create Graded Response Model Parameters
ip <- generate_ip(n = 5, model = "GRM", n_categories = c(3, 4, 8, 5, 4))
# Simulate responses for one theta:
sim_resp(ip = ip, theta = rnorm(1))
# Simulate responses for 5 thetas:
sim_resp(ip = ip, theta = rnorm(5))
# Set 10% of the item responses as missing
sim_resp(ip = ip, theta = rnorm(5), prop_missing = .1)
```

`summary.cat_output` *Summarizes the raw output of cat_sim*

Description

This function summarizes a list consist of cat_output objects. It returns a summary data frame of the CAT simulation.

Usage

```
## S3 method for class 'cat_output'
summary(
  object,
  ...,
  cols = c("true_ability", "est_ability", "se", "test_length")
)
```

Arguments

object	This is a cat_output object or a list object containing elements that are "cat_output" class.
...	Additional arguments.
cols	The variables that will be included in the summary. There should be at least one column. Available columns are: **true_ability** True ability of the simulee **est_ability** Ability Estimate **se** Standard Error of the ability estimate **test_length** Test length. **bias** The difference between true ability and ability estimate **mse** Mean squared error

Value

This function returns a summary data frame of adaptive tests. Each row will represent a different adaptive test.

Author(s)

Emre Gonulates

See Also

cat_sim

Examples

```
n <- 100 # number of items
ip <- generate_ip(n = n,
                  content = sample(c("Algebra", "Arithmetic", "Geometry"),
                                   n, replace = TRUE))
cd <- create_cat_design(ip = ip, next_item_rule = 'mfi',
                        termination_rule = 'max_item',
                        termination_par = list(max_item = 10))
cat_data <- cat_sim(true_ability = rnorm(5), cd = cd)
summary(cat_data)
```

---

testlet *Creates a* Testlet-class *object*

Description

Create a Testlet-class object. It is recommended to use this function to create new Testlet-class objects.

Usage

```
testlet(...)
```

Arguments

...	The object that is desired to be converted to a Testlet object. Also additional arguments related to the Testlet.

Value

An Testlet-class object.

Author(s)

Emre Gonulates

Examples

```
ip <- itempool(a = c(1, 1.4), b = c(-2, 1))
testlet(ip, id = "T1")
testlet(ip, id = "T1", content = "Algebra")
# Add misc field to the testlet:
testlet(ip, id = "T1", misc = list(form = "A1", operational = TRUE,
                                   admin_date = as.Date("2020-08-01")))

# Add misc field to the testlet items:
testlet(itempool(b = rnorm(2), id = paste0("t1-i", 1:2),
                 misc = list(list(sympson_hetter_k = .8, form = "B1"),
                             list(sympson_hetter_k = .9))),
        id = "t1")
```

Testlet-class *An S4 class to represent a Testlet*

Description

Testlet is a class to represent an a collection of items. Items that are connected by a common stimulus (for example a reading passage, a graph, etc.) can form a testlet. An object in Testlet class should have a model name and item_list which is an Itempool. object. In fact, a Testlet object is very similar to an Itempool-class object, except, it has a designated model and optional parameters

Slots

id Testlet id. Default value is NULL.

item_list A list of Item objects.

`model` The model that testlet `parameters` represents. Currently model can be: BTM (Basic Testlet Model, this is default testlet model where no parameters necessary and testlet simply connects items), RTM (Rasch Testlet Model), BF (Bifactor Model) (Not implemented yet), 2PTM (Two-parameter testlet model), 3PTM (three-parameter testlet model). A model must be specified for the construction of an `tetlet` object.

`parameters` A list containing numeric vectors that represent testlet parameters. Depending on the model these parameters can change.

`se_parameters` Standard error of testlet parameters.

`content` Content information for testlet.

`misc` A list of additional parameters for the testlet.

Author(s)

Emre Gonulates

`$,Item-method` *Get slots from an* `Item-class` *object.*

Description

Get slots from an `Item-class` object.

Usage

```
## S4 method for signature 'Item'
x$name
```

Arguments

`x`	An `Item-class` object.
`name`	Name of the parameter. Available values: `'id'` Extract `'id'` of an `Item-class` object. `'model'` Extract the `'model'` of an `Item-class` object. `'parameters'` Extract the `'parameters'` of an `Item-class` object. `'se_parameters'` Extract the standard error of parameters of an `Item-class` object. `'content'` Extract the `'content'` slot of an `Item-class` object. `'misc'` Extract the `'misc'` slot of an `Item-class` object. `'max_score'` Extract the maximum possible score of an `Item-class` object. Minimum score is assumed to be 0.

Value

This operation will return the desired slot.

Author(s)

Emre Gonulates

Examples

```
item1 <- item(model =  "3PL", id = 'item23', content = 'Geometry',
              misc = list(enemies = c("item1", "item2")),
              parameters = list(b = 2, c = .12, a = 1.2, D = 1))
# Get individual parameters
item1$a
item1$b
item1$D
# Get item 'model'
item1$model
# Get all parameters
item1$parameters
# Get item 'id'
item1$id
# Get item content
item1$content
# Get misc values
item1$misc
# Get misc values
item1$max_score
```

`$,Itempool-method` *Get slots of the an* `Item-class` *object.*

Description

Get slots of the an `Item-class` object.

Usage

```
## S4 method for signature 'Itempool'
x$name
```

Arguments

`x`	An `Itempool-class` object from which to extract element(s) or in which to replace element(s).
`name`	Name of the parameter. Available values: `'id'` Extract `id`'s of all items and testlets. This will not extract the `id`'s of items within the testlet. `'content'` Extract `content`'s of all items and testlets. This will not extract the `content`'s of items within the testlet. `'model'` Extract `model`'s of all items and testlets. This will not extract the `model`'s of items within the testlet.

'misc' Extract misc parameters of all items and testlets. This will not extract the misc parameters of items within the testlet.

'item_list' Extract individual elements of item pool. If there are testlets in the item pool, a testlet will be an item of the resulting list. If individual items within the testlet is desired to be elements of the list, then use $items.

'items' Extract individual items within the item pool. If there are testlets in the item pool individual elements of the testlet will be extracted. Resulting list will only consist of Item-class objects.

'parameters' Extract parameters's of all items and testlets. This will not extract the parameters's of items within the testlet.

'se_parameters' Extract se_parameters's of all items and testlets. This will not extract the se_parameters's of items within the testlet.

'n' Return a list with three objects: elements the number of standalone items and testlets. testlets the number of Testlet objects. items the sum of the number of items within testlets and standalone items.

'max_score' Returns the maximum possible raw score of the item pool.

'resp_id' Extract id's of all standalone items and items within the testlets. It will not return testlet id's. This is mainly to get the id's of items which has a response.

'resp_content' Extract content's of all standalone items and items within the testlets. It will not return testlet content's. This is mainly to get the content's of items which has a response.

'resp_model' Extract model's of all standalone items and items within the testlets. It will not return testlet model's. This is mainly to get the model's of items which has a response.

'resp_misc' Extract misc fields of all standalone items and items within the testlets. It will not return testlet misc fields.

'resp_item_list' Combine items that are not in a testlet and items within a testlet and return a list object. This list does not contain any Testlet objects. All of the elements are Item objects. If there are no testlets in the item pool, then this argument will be the same as $item_list.

'resp_max_score' Extract the maximum score each standalone item can get.

Value

This operation will return a numeric object.

Author(s)

Emre Gonulates

Examples

```
item1 <- methods::new("Item", model =  "3PL", id = 'item23',
                      content = 'Geometry',
                      parameters = list(b = 2, c = .12, a = 1.2, D = 1))
item1$a
item1$D
```

```
item1$model
item1$id
item1$content
```

`$,Testlet-method` *Access slots of a* `Testlet-class` *object*

Description

Access slots of a `Testlet-class` object

Usage

```
## S4 method for signature 'Testlet'
x$name
```

Arguments

`x`	A `Testlet-class` object from which to extract element(s) or in which to replace element(s).
`name`	Name of the parameter. Available values: `'id'` Get the `id` of the testlet `'content'` Get the `content` of the testlet. `'model'` Get the `model` of the testlet. `'item_models'` Get the `model` of the items within the testlet. `'parameters'` Get the `parameters` of the testlet. `'se_parameters'` Get the `se_parameters` of the testlets. `'item_list'` Get the list of `Item-class` objects of the testlet. Returns a `list` object. `'max_score'` Returns the maximum score obtainable by all of the items within the testlet.

Value

This operation will return the desired slot.

Examples

```
t1 <- testlet(generate_ip(n = 3), id = "my-testlet", content = "Algebra")
t1$model
t1$id
t1$item_list
t1$content
t1$item_models
```

`$<-,Item-method` *Set values to parameters or components of* `Item-class` *object*

Description

Set values to parameters or components of `Item-class` object

Usage

```
## S4 replacement method for signature 'Item'
x$name <- value
```

Arguments

`x`	An `Item-class` object.
`name`	Name of the parameter or component.
`value`	The new value that will be assigned.

Value

This operation will not return anything.

Author(s)

Emre Gonulates

Examples

```
item <- new("Item", model =  "3PL", id = 'item23', content = 'Geometry',
            misc = list(enemies = c("item1", "item2")),
            parameters = list(b = 2, c = .12, a = 1.2, D = 1))
item$a <- 2
item$D <- 1.7
item$id <- "Itm-111"
item$content <- 'Algebra'
item$misc <- list(enemies = c("item5"))
```

$<-,Itempool-method *Set values to parameters or components of 'Item' class.*

Description

Set values to parameters or components of 'Item' class.

Usage

```
## S4 replacement method for signature 'Itempool'
x$name <- value
```

Arguments

x	Itempool-class object.
name	Name of the parameter or component. Currently only misc, id, content, item_list are available.
value	The new value that will be assigned.

- For id, the value should be a list of strings that has the same length as the length of the Itempool-class object. There should not be any duplicated id's.
- For content, the value should be either NULL or a list of strings that has the same length as the length of the Itempool-class object.
- For item_list, the value should be a list of Item-class or Testlet-class objects.
- For misc, the value should be a list.

Value

This operation will return an Itempool-class object.

Author(s)

Emre Gonulates

$<-,Testlet-method *Set values to parameters or components of 'Item' class.*

Description

Set values to parameters or components of 'Item' class.

Usage

```
## S4 replacement method for signature 'Testlet'
x$name <- value
```

Arguments

x	A Testlet-class object.
name	Name of the parameter or component.
value	The new value that will be assigned.

Value

This operation will not return anything.

Author(s)

Emre Gonulates

Index