Package 'irt'

May 8, 2026

Type Package

Title Item Response Theory and Computerized Adaptive Testing Functions

Version 0.2.9

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.

URL https://github.com/egonulates/irt

BugReports https://github.com/egonulates/irt/issues

License AGPL (>= 3)

Depends methods

LinkingTo Rcpp

Imports Rcpp (>= 1.0.1), parallel

NeedsCompilation yes

RoxygenNote 7.3.1

Encoding UTF-8

Language en-US

Collate 'RcppExports.R' 'ability_estimation.R' 'bilog.R' 'cat_sim.R' 'cat_sim_helper_functions.R' 'classification.R' 'item-class.R' 'item-class-methods.R' 'itempool-class.R' 'itempool-class-methods.R' 'dif.R' 'equate_stuirt.R' 'flexMIRT.R' 'response-class.R' 'generate_objects.R' 'info.R' 'ipd.R' 'irtpro.R' 'item_analysis.R' 'item_fit.R' 'kernel_smoothing.R' 'response_set-class.R' 'max_score.R' 'mean.R' 'misc.R' 'package-irt.R' 'person_fit.R' 'plot_cat_output.R' 'plot_distractor_icc.R' 'plot_icc.R' 'plot_itempool.R' 'plot_info.R' 'plot_item.R' 'plot_ks.R' 'plot_resp_loglik.R' 'prob.R' 'resp_lik.R' 'resp_loglik.R' 'response-class-methods.R' 'response_set-class-methods.R' 'rsss.R' 'sim_resp.R' 'testlet-class-methods.R' 'var.R' 'winsteps.R' 'zzz.R'

Config/testthat/edition 3

Suggests ggplot2, markdown, knitr, rmarkdown, tibble, pillar

VignetteBuilder knitr

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

Repository CRAN

Date/Publication 2024-02-20 20:40:02 UTC

Contents

`1PL-class` *One-Parameter Logistic IRT model*

Description

One-Parameter Logistic IRT model

Slots

`b` Item difficulty parameter

`D` Scaling constant

`se_b` Standard error of item difficulty parameter

Author(s)

Emre Gonulates

2PL-class *Two-Parameter Logistic IRT model*

Description

Two-Parameter Logistic IRT model

Slots

- `a` Item discrimination parameter
- `b` Item difficulty parameter
- `D` Scaling constant
- `se_a` Standard error of item discrimination parameter
- `se_b` Standard error of item difficulty parameter

Author(s)

Emre Gonulates

3PL-class *Three-Parameter Logistic IRT model*

Description

Three-Parameter Logistic IRT model

Slots

- `a` Item discrimination parameter
- `b` Item difficulty parameter
- `c` Guessing parameter
- `D` Scaling constant
- `se_a` Standard error of item discrimination parameter
- `se_b` Standard error of item difficulty parameter
- `se_c` Standard error of guessing parameter

Author(s)

Emre Gonulates

4PL-class *Three-Parameter Logistic IRT model*

Description

Three-Parameter Logistic IRT model

Slots

- `a` Item discrimination parameter
- `b` Item difficulty parameter
- `c` Guessing parameter
- `d` Upper-asymptote Parameter
- `D` Scaling constant
- `se_a` Standard error of item discrimination parameter
- `se_b` Standard error of item difficulty parameter
- `se_c` Standard error of guessing parameter
- `se_d` Standard error of upper-asymptote parameter

Author(s)

Emre Gonulates

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))
```

---

area_between_icc *Calculate the area between two ICC curves*

Description

This function calculates the area between two item characteristic curves (ICC) for unidimensional dichotomous IRT models.

There are two types of area calculation methods. The first one is `type = "exact"` where the exact area from negative infinity to positive infinity between the two ICC curves will be calculated. This method implements the approach in Raju's 1988 paper. This method works for 'Rasch', '1PL', '2PL', '3PL' models but when the pseudo-guessing parameters of the items differ for '3PL' model, the area will be infinity. In such cases it is advisable to use `type = "closed"`.

The area can only be calculated for 'Rasch', '1PL', '2PL', '3PL' or '4PL' models.

Usage

```
area_between_icc(
  ...,
  type = c("closed", "exact"),
  theta_range = c(-5, 5),
  signed_area = FALSE
)
```

Arguments

`...`	An `Itempool-class` object; or a combination of `Item-class` and `Testlet-class` objects.
`type`	A string representing the method that will be used to calculate the area between two ICC curves. Available values are: `"exact"` The exact area between the whole theta scale `-Inf` and `Inf`. This method implements Raju's (1988) approach. When the pseudo-guessing parameters of the items differ for '3PL' model, the area will be infinity. See Raju (1988) for details. `"closed"` The area within a closed interval defined by `theta_range` argument will be calculated. This method always returns a finite value. See Kim and Cohen (1991) for details. The default method is `"closed"`.
`theta_range`	A numeric vector of length two with the first element smaller than the second element. The values define the boundaries in which the area between two ICC's will be calculated. The default value is `c(-5, 5)`.
`signed_area`	A logical value for whether the signed or unsigned area between two curves will be calculated. When `signed = TRUE`, the area under the second item is subtracted from the area under the first item. The result can be negative if the first item is mostly under the second item. When `signed = FALSE`, the distance between two ICC curves will be calculated. The default value is `signed = TRUE`.

Value

A matrix where the values in cells are the areas between items. The rows represent the first item and the columns represents the second item and the area of second item is subtracted from the first item when "signed" area is desired. For example, the value corresponding to the cell where row is for "Item_4" and column is for "Item_2", the value in the cell is the area of "Item_4 - Item_2".

Author(s)

Emre Gonulates

References

Kim, S.-H., & Cohen, A. S. (1991). A comparison of two area measures for detecting differential item functioning. Applied Psychological Measurement, 15(3), 269–278.

Raju, N. S. (1988). The area between two item characteristic curves. Psychometrika, 53(4), 495–502.

Examples

```
# Closed area example:
ip <- generate_ip(model = c("3PL", "3PL", "3PL"))
# plot(ip) # See the ICCs
area_between_icc(ip, type = "closed")
area_between_icc(ip, type = "closed", signed_area = TRUE)
```

```
# The result is infinite because 'c' parameters are not equal
area_between_icc(ip, type = "exact")

# Exact area example:
ip <- generate_ip(model = c("2PL", "2PL", "2PL"))
area_between_icc(ip, type = "exact")
area_between_icc(ip, type = "exact", signed_area = TRUE)
# The 'closed' area is very close to the 'exact' area with a wide theta range
area_between_icc(ip, type = "closed", theta_range = c(-10, 10))
```

`as.data.frame.cat_output` *Convert a* `cat_output` *object into a* `data.frame`.

Description

This function converts `cat_output` objects to a `data.frame` object.

Usage

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

Arguments

`x`	An `cat_output` object
`row.names`	`NULL` or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	logical. If `TRUE`, setting row names and converting column names
`...`	additional arguments

Value

A data frame with the following columns:

true_ability True ability of the simulee

est_before Ability estimate before administration of an item.

se_before Standard error before administration of an item.

testlet_id Administered testlet's ID.

item_id Administered item's ID.

resp Response to the item

est_after Ability estimate after the administration of an item.

se_after Standard error after administration of an item.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(n = 40)
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(1), cd = cd)
as.data.frame(cat_data)
```

---

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

Description

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

This function converts Itempool-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 'Item'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., include_se = TRUE)

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

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

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

## S3 method for class 'GPCM2'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., include_se = TRUE)

## S3 method for class 'M2PL'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., include_se = TRUE)

## S3 method for class 'M3PL'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., include_se = TRUE)
```

```
## S3 method for class 'Itempool'
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 representation of the item.

A data frame representation of the GRM item.

A data frame representation of the PCM item.

A data frame representation of the GPCM item.

A data frame representation of the GPCM2 item.

A data frame representation of the M2PL item.

A data frame representation of the M3PL item.

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.

Author(s)

Emre Gonulates

Examples

```
item1 <- generate_item()
as.data.frame(item1)

item2 <- generate_item(model = "Rasch", item_id = "i1",
                       misc = list(type = "MC", op = TRUE, c("i1", "i2")))
as.data.frame(item2)

item3 <- generate_item(model = "GRM")
as.data.frame(item3)

item1 <- generate_item(model = "GRM", item_id = "i1")
as.data.frame(item1)
```

```
item1 <- generate_item(model = "PCM", item_id = "i1")
as.data.frame(item1)
item1 <- generate_item(model = "GPCM", item_id = "i1")
as.data.frame(item1)
item1 <- generate_item(model = "GPCM2", item_id = "i1")
as.data.frame(item1)
item1 <- generate_item(model = "M2PL", item_id = "i1")
as.data.frame(item1)
item1 <- generate_item(model = "M3PL", item_id = "i1")
as.data.frame(item1)
ip1 <- generate_ip()
as.data.frame(ip1)

ip2 <- generate_ip(n = 10, model = "GRM",
                   content = sample(c("G", "A"), 10, TRUE),
                   item_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)
testlet1 <- generate_testlet()
as.data.frame(testlet1)
testlet2 <- generate_testlet(testlet_id = "T1")
as.data.frame(testlet2)
```

`as.data.frame.Response` *Convert a* `Response-class` *object into a* `data.frame`.

Description

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

Usage

```
## S3 method for class 'Response'
as.data.frame(
  x,
```

```
  row.names = NULL,
  optional = FALSE,
  ...,
  attach_unique_misc = TRUE
)
```

Arguments

x	An Response-class object
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
optional	logical. If TRUE, setting row names and converting column names
...	additional arguments
attach_unique_misc	If TRUE, the elements of the misc slot that have lengths one will be attached to the data frame returned. The default is TRUE.

Value

A data frame of item_ids/responses/scores within each row.

Author(s)

Emre Gonulates

Examples

```
resp <- response(examinee_id = "Stu12",
                 item_id = c("Item1", "Item2", "Item3", "Item4"),
                 score = c(0, 1, 1, 1),
                 raw_response = c("B", "A", "D", "Right Angle"),
                 order = c(1L, 2L, 3L, 4L),
                 misc = list(item_role = c("F", "O", "O", "O"),
                             lexile_level = c(1, 4, 3, 1),
                             item_type = c("MC", "MC", "MS", "SA"),
                             test_date = as.Date("2021-11-21"),
                             Form = "Test Form 001",
                             theta = 2.2))
as.data.frame(resp)

# Do not include misc fields whose lengths are not equal to the number of
# items
as.data.frame(resp, attach_unique_misc = FALSE)
```

`as.data.frame.Response_set` *Convert a* `Response_set-class` *object into a long format* `data.frame`

Description

Convert a `Response_set-class` object into a long format `data.frame`

Usage

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

Arguments

`x`	A `Response_set-class` object
`row.names`	`NULL` or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	logical. If `TRUE`, setting row names and converting column names
`...`	additional arguments

Author(s)

Emre Gonulates

`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)
```

`as.list.Response_set` *This function converts Response_set objects to a list object*

Description

This function converts Response_set objects to a list object

Usage

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

Arguments

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

Value

A list object with elements from `Response-class` objects.

Author(s)

Emre Gonulates

`as.matrix,Response_set-method`
Convert a `Response_set-class` *object into a* `matrix`

Description

This function converts `Response_set-class` objects to a `matrix` object.

Usage

```
## S4 method for signature 'Response_set'
as.matrix(x, ..., output = "score", ip = NULL)
```

Arguments

x	A Response_set-class object
...	additional arguments
output	Contents of the matrix. The default value is "score". Other options are: **"score"** Matrix of item scores. **"raw_response"** Matrix of raw responses. **"item_id"** Matrix of item ids. **"testlet_id"** Matrix of testlet ids. **"response_time"** Matrix of response times. **"order"** Matrix of item orders. **misc** If all responses has the same 'misc' field, then the matrix of that misc field can be extracted.
ip	An Itempool-class object to use for adding item_id's as column names. If there are items that are in the item pool but not in the response data, those items will be added and all values will be NA.

Value

A matrix of examinee item scores within each row and items in each column.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(n = 15)
resp_set <- generate_resp_set(ip = ip, theta = rnorm(30), prop_missing = .5)
# Matrix of item scores
as.matrix(resp_set)

# If the item pool object provided, the column names will have the same
# order as the item order in item pool
as.matrix(resp_set, ip = ip)

# Matrix of raw responses
as.matrix(resp_set, output = "raw_response")

# Matrix of item order
as.matrix(resp_set, output = "order")

# Matrix of item ids
as.matrix(resp_set, output = "item_id")
```

`biserial` *Calculate biserial correlation*

Description

Calculate biserial correlation

Usage

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

Arguments

`score`	Item scores of each examinee for which biserial correlation will be calculated
`criterion`	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.

This function concatenates Response and/or Response_set objects and returns a Response_set-class object.

If the elements do not have examinee ID fields, function will assign default ids.

Usage

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

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

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

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

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

Arguments

x	A list consist of Response-class or Response_set-class objects.
...	Additional arguments

Value

An Itempool-class object.

A Response_set-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)
```

---

c.cat_design *Concatenate 'cat_design' objects*

Description

Concatenate 'cat_design' objects

Usage

```
## S3 method for class 'cat_design'
c(x, ...)
```

Arguments

x	A cat_design class object.
...	Remaining cat_design class objects.

Value

A list of cat_design objects.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(n = 20)
cd1 <- create_cat_design(ip = ip,
                         termination_rule = c('max_item'),
                         termination_par = list(max_item = 5))
cd2 <- create_cat_design(ip = ip,
                         termination_rule = c('max_item'),
                         termination_par = list(max_item = 9))
cd <- c(cd1, cd2)
```

---

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)

# Multiple theta, optionally set names to the the vector to give examinee IDs
true_theta <- setNames(c(-2, 0.4, 1.5), c("Jimmy", "Ali", "Mirabel"))
cd <- create_cat_design(
  ip = ip,
  ability_est_rule = 'ml',
  termination_rule = c('min_item', 'min_se', 'max_item'),
  termination_par = list(min_item = 10, min_se = .33, max_item = 20))
cat_sim(true_ability = true_theta, 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)
```

---

classification_agreement_index

Calculate agreement index

Description

Calculate agreement index

Usage

```
classification_agreement_index(
  true_score,
  estimated_score,
  cut_scores,
  cat_labels = NULL
)
```

Arguments

true_score	A numeric vector of examinees' true score values. Values can be in theta scale or summed scores.
estimated_score	A numeric vector of examinees' estimated score values. Values can be in theta scale or summed scores.
cut_scores	A sorted (ascending order) numeric vector of cut score values. Values can be in theta scale or summed scores. Do not include 0 or the maximum possible score of the test.
cat_labels	A string vector representing the labels of the categories. The length of the vector should be one more than the length of the cut scores. The default value is NULL where the categories will be labeled as 1, 2, ..., (number of cut scores plus one). For example, if there are three cut scores category labels can be: c("Unsatisfactory", "Basic", "Mastery", "Advanced").

Value

A list of following elements:

ca_table A classification table where rows are true categories and columns are estimated categories. The values are the number of examinees.

ca_table A classification table where rows are true categories and columns are estimated categories. The values are the proportion of examinees.

ca Classification agreement index

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(model = sample(c("GPCM", "2PL"), 20, TRUE))
n_examinee <- 1000
true_theta <- rnorm(n_examinee)
observed_theta <- true_theta + runif(n_examinee, -.5, .5)
theta_cs <- c(-1, 0, 1.5)
raw_cs <- round(rsss(ip = ip, scale_score = theta_cs))
true_raw_score <- round(rsss(ip = ip, scale_score = true_theta))
observed_raw_score <- round(rsss(ip = ip, scale_score = observed_theta))

# Theta scores
classification_agreement_index(true_score = true_theta,
                               estimated_score = observed_theta,
                               cut_scores = theta_cs)
# Summed scores
classification_agreement_index(true_score = true_raw_score,
                               estimated_score = observed_raw_score,
                               cut_scores = raw_cs)
# Add labels
```

```
classification_agreement_index(true_score = true_theta,
                               estimated_score = observed_theta,
                               cut_scores = theta_cs,
                               cat_labels = c("Unsatisfactory", "Basic",
                                              "Mastery", "Advanced"))
```

`classification_indices` *Calculate classification accuracy and consistency*

Description

Calculate classification accuracy and consistency

Usage

```
classification_indices(
  method = "recursive",
  ip = NULL,
  theta = NULL,
  theta_cs = NULL,
  raw_cs = NULL,
  resp = NULL,
  se = NULL,
  perf_categories = NULL,
  n_theta = 100,
  theta_lower_bound = -6,
  theta_upper_bound = 6,
  cat_labels = NULL
)
```

Arguments

`method` The method of classification accuracy and consistency calculation method. Following methods are available:

- `'rudner'` Rudner (2000, 2005) based classification accuracy and consistency indices.
  Following values should be provided for this method: `theta`, `se`, `theta_cs`.
  Following values can optionally be provided for this method: `perf_categories`, `cat_labels`.
- `'guo'` Guo (2006) based classification accuracy and consistency indices.
  Note that calculation times can be long for this method. The value of `n_theta` can be decreased to speed up the function but this will reduce the accuracy of the index.

Following values should be provided for this method: ip, resp, theta_cs and either one of theta or perf_categories. Following values can optionally be provided for this method: n_theta, theta_lower_bound, theta_upper_bound, cat_labels.

'recursive' Lee (2010) based classification accuracy and consistency indices. Following values should be provided for this method: ip, theta and either one of these theta_cs, raw_cs. Following values can optionally be provided for this method: perf_categories, cat_labels.

ip An Itempool-class object. Item pool parameters can be composed of any combination of unidimensional dichotomous or polytomous items. Required for "guo" and "recursive" methods.

theta A numeric vector representing the abilities of examinees. Required for 'rudner' and 'recursive' method. For "guo" method, this vector will be used to get performance category of each examinee if perf_categories is NULL. The default value is NULL. For method = "guo" either theta or perf_categories should be provided.

theta_cs A sorted (ascending order) numeric vector representing the theta scale cut scores. Do not include -Inf or Inf. Required for 'rudner' and 'guo' method; required for 'recursive' if raw_cs is not provided.

raw_cs A sorted (ascending order) numeric vector of summed-score cut score values. Do not include 0 or the maximum possible score of the test in this vector. Required for 'recursive' method if 'theta_cs' is not provided.

resp A Response_set-class, a matrix or a data.frame object that holds responses. If matrix or a data.frame provided, they will be converted to a Response_set-class. Required for 'guo' method.

se A numeric vector representing the standard errors of ability estimates. Required for 'rudner' method.

perf_categories An integer vector representing the performance categories of examinees. The number 1 should represent the lowest category. For example if there are three cut scores the valid values can only be: 0, 1, 2 and 3. This vector will be used theta is NULL. The default value is NULL. Either theta or perf_categories should be provided. Can optional be provided for all methods.

n_theta An integer representing the number of equally spaced theta points between cut scores. The default value is 100. Use larger values to increase accuracy but larger numbers will also slow the speed of calculation. Can optionally be provided for the 'guo' method.

theta_lower_bound A number representing the lower bound for cut scores. The default value is -6. Can optionally be provided for the 'guo' method.

theta_upper_bound A number representing the upper bound for cut scores. The default value is 6. Can optionally be provided for the 'guo' method.

cat_labels A string vector representing the labels of the categories. The length of the vector should be one more than the length of the cut scores. The default value

is NULL where the categories will be labeled as 1, 2, ..., (number of cut scores plus one). For example, if there are three cut scores category labels can be: c("Unsatisfactory", "Basic", "Mastery", "Advanced"). Can optional be provided for all methods.

Value

A list of following elements:

category_prob A numeric vector representing the performance category classification probabilities of each examinee.

ca Marginal (overall) classification accuracy index

cc Marginal (overall) classification consistency index

ind_cs_ca Individual cut score classification accuracy indices. This value will only be calculated when there are more than one cut score.

ind_cs_ca Individual cut score classification consistency indices. This value will only be calculated when there are more than one cut score.

Author(s)

Emre Gonulates

References

Guo, F. (2006). Expected classification accuracy using the latent distribution. Practical Assessment, Research, and Evaluation, 11(1), 6.

Lee, W. C. (2010). Classification consistency and accuracy for complex assessments using item response theory. Journal of Educational Measurement, 47(1), 1-17.

Rudner, L. M. (2000). Computing the expected proportions of misclassified examinees. Practical Assessment, Research, and Evaluation, 7(1), 14.

Rudner, L. M. (2005). Expected classification accuracy. Practical Assessment, Research, and Evaluation, 10(1), 13.

Wyse, A. E., & Hao, S. (2012). An evaluation of item response theory classification accuracy and consistency indices. Applied Psychological Measurement, 36(7), 602-624.

Examples

```
ip <- generate_ip(model = sample(c("GPCM", "2PL"), 20, TRUE))
n_examinee <- 100

true_theta <- rnorm(n_examinee)
resp_set <- generate_resp_set(ip = ip, theta = true_theta, prop_missing = .2)
theta_est <- est_ability(resp = resp_set, ip = ip, method = "eap")
se <- theta_est$se
theta_est <- theta_est$est
raw_score <- est_ability(resp = resp_set, method = "sum_score")$est

# Cut score
```

```
theta_cs <- c(-1, 0, 1.5)
raw_cs <- round(rsss(ip = ip, scale_score = theta_cs))

# Rudner (2000, 2005) based indices:
classification_indices(method = "rudner", theta = theta_est, se = se,
                       theta_cs = theta_cs)

# Guo (2006) based indices:
classification_indices(method = "guo", ip = ip, resp = resp_set,
                       theta = theta_est, theta_cs = theta_cs)

# Recursive method based indices:
classification_indices(method = "recursive", ip = ip, theta = theta_est,
                       theta_cs = theta_cs)
# Use raw score cut scores with recursive method
classification_indices(method = "recursive", ip = ip, theta = theta_est,
                       raw_cs = raw_cs)
```

`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),
  testlet_rules = NULL,
  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`.

"map" Maximum-a-posteriori (Bayes Modal). Required parameters:

prior_dist Distribution of the prior distribution. Currently only available value is:
* `norm` for normal 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`.

tol The tolerance (precision) level of the estimate.
The default value is `0.00001`.

"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.

"map_ml" Maximum-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,

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

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

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

tol The tolerance (precision) level of the estimate.
The default value is `0.00001`.

"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)

testlet_rules A list containing arguments that specify the rules that will be used within a testlet.

The default value is NULL where the following rules will be applied if there is a testlet: list(next_item_rule = "none", termination_rule = "max_item", termination_par = list(max_item = 999)) where if a testlet is selected all items of this testlet is selected (unless the a testlet has more than 999 items.). Each item is selected with the order it appears in the testlet.

It is assumed that items within testlet are administered together. In other words, an item that does not belong to a selected testlet cannot be administered between two items that belong to the same testlet.

The following list elements are available:

next_item_rule The way item selection is performed within a testlet. Following options are available:

"none" Items are selected with the order of observed in the testlet.

"mfi" Maximum Fisher Information. The most informative unadministered item within the testlet at the current ability estimate is selected.

"termination_rule" The rule that should be satisfied to stop administering items from a testlet. If there are more than one rule, the termination rules will be applied as the order they appear in the termination_rule vector. For example, if termination_rule = c("max_item", "min_se"), then if max_item criteria is met testlet will be terminated without checking for min_se value.

Following options are available:

"max_item" An integer representing the maximum number of items administered for each testlet. The test will terminate when maximum number of items is reached or there are no items left in the testlet.

"min_se" A numeric value representing the standard error of ability estimate value to terminate the test. If the standard error exceeds min_se value, then the testlet will terminate. This testlet termination criteria will only be checked if at least one item from the testlet has already been selected.

"termination_par" The test termination parameters. See the "termination_par" above in the main function for available options.

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$item_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, testlet_id = "t1"), generate_ip(n = 3),
        generate_testlet(n = 5, testlet_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)), item_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)))
```

`cusum_single` *CUSUM based statistics for one examinee*

Description

CUSUM based statistics for one examinee

Usage

```
cusum_single(ip, resp, theta = NULL, method = "T1", initial_theta_est = NULL)
```

Arguments

`ip`	An `Itempool-class` object
`resp`	a response vector, where the order of items represent the administration order.
`theta`	A vector or length 1 or length equal to the number of items administered.
`method`	Method of calculating the CUSUM statistic. Choices are: `"T1"`, `"T2"`, `"T3"`, `"T4"`, `"T5"`, `"T6"`, `"T7"` and `"T8"`. `"T1"` through `"T4"` uses the ability estimate each stage of the test. `"T5"` through `"T8"` uses the final ability estimate and needs only one theta estimate.
`initial_theta_est`	For CAT, the initial theta estimate of an examinee. For CAT, if `theta = NULL`, this initial theta estimate will be used to calculate T_1, C_1^- and C_1^+. For CAT, this value should be provided. By this way for the calculation of T_1, C_1^- and C_1^+, $\hat{\theta}_{n-1}$ will be used. If it's value is `NULL` and `theta = NULL`, then for the calculation of T_1, C_1^- and C_1^+, $\hat{\theta}_n$ will be used. The default value is `NULL`

Value

The function will return a data frame consist of two columns: `Cp` column for C^+ values and `Cn` column for C^- values.

Author(s)

Emre Gonulates

References

van Krimpen-Stoop, E. M. L. A., & Meijer, R. R. (2000). Detecting person-misfit in adaptive testing using statistical process control techniques. In W. J. van der Linden & C. A. W. Glas (Eds.), Computerized adaptive testing: Theory and practice (pp. 210–219). Kluwer.

Xiaofeng Yu & Ying Cheng (2020): A Comprehensive Review and Comparison of CUSUM and Change-Point-Analysis Methods to Detect Test Speededness, Multivariate Behavioral Research, <doi:10.1080/00273171.2020.1809981>

Examples

```
# Example from Table 1 (p.4) of Yu and Cheng (2020):
ip <- itempool(a = c(0.976, 0.973, 0.871, 0.768, 0.94, 1.109, 1.063, 0.888,
                     0.648, 0.733, 0.8, 0.823, 0.611, 0.965, 1.052, 0.937,
                     0.894, 0.72, 0.686, 0.608),
               b = c(-0.693, 0.6, -0.607, -0.637, -1.095, -0.202, -0.679,
                     0.058, -0.822, -0.768, -0.737, -1.158, -0.294, -0.856,
                     -0.833, -0.613, -0.151, -0.614, -0.07, -0.806),
               c = c(0.371, 0.224, 0.159, 0.377, 0.159, 0.146, 0.181, 0.251,
                     0.179, 0.214, 0.312, 0.224, 0.246, 0.225, 0.155, 0.166,
                     0.456, 0.327, 0.112, 0.169),
               D = 1.7)
resp <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1)
theta <- -0.06

cusum_single(ip, resp, theta, method = "T5")
```

`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 matrix 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(resp, key = NULL, ip = NULL, criterion = NULL)
```

Arguments

`resp`	It can be either a `Response_set-class` object with valid raw responses; or, a `matrix` or `data.frame` containing the raw item responses.
`key`	The answer key for the responses. Keys can also be provided via `ip` argument.
`ip`	An `Itempool-class` object that contains the keys of the items. The program will look check whether a `ip$misc$key` is specified for all items. Valid keys should be provided via `ip` if `key` argument is `NULL`.
`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.

Value

A data.frame with following columns

'item_id' Item identifier

'key' Answer key

'option' The selected option

'n' Number of subjects/examinees answered this item

'prop' Observed proportions of the choice.

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

'pbis' Point-biserial correlation between the examinees selected the choice and the total scores.

'bis_adj' Biserial correlation between item and total score without this item. Sum scores will be used in the calculation of 'bis_adj' even 'criterion' is provided.

'pbis_adj' Point-biserial correlation between item and total score without this item. Sum scores will be used in the calculation of 'bis_adj' even 'criterion' is provided.

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(resp = raw_resp, key = key)
```

`equate_stuirt` *IRT Scale Transformation using STUIRT Program*

Description

This function serves as an interface for the STUIRT program (Kim & Kolen, 2004) which offers a range of equating methods including mean-mean, mean-sigma, Haebara, and Stocking-Lord. It is essential to have the STUIRT program installed on your computer for this function to work. You can download the program from the University of Iowa's Center for Advanced Studies in Measurement and Assessment (CASMA) webpage: https://education.uiowa.edu/casma/computer-programs

Usage

```
equate_stuirt(
  new_ip,
  ref_ip,
  method = c("stocking-lord", "haebara", "mean-mean", "mean-sigma"),
  common_item_ids = NULL,
  stuirt_exe_path = "C:/STUIRT/STUIRT.exe",
  target_dir = getwd(),
  analysis_name = "stuirt_analysis",
  add_options = TRUE,
  starting_values = c(1, 0),
  number_of_iterations = NULL,
  new_dist = NULL,
  ref_dist = NULL,
  fs = c("DO", "DO"),
  sy = c("BI", "BI"),
  lm = NULL,
  ko = "SL",
```

```
  show_output_on_console = TRUE
)
```

Arguments

new_ip	An Itempool-class object holding the item parameters of the new form.
ref_ip	An Itempool-class object holding the item parameters of the old form which is the reference form.
method	A string specifying the method to use for equating the new item parameters new_ip to the reference scale (ref_ip). Choose from methods like "stocking-lord", "haebara", "mean-mean","mean-sigma". The default method is "stocking-lord".
common_item_ids	The item IDs of the common items. The default is NULL, assuming that all items are common. Ensure that the same 'item_id's are used in both 'new_ip' and 'ref_ip'.
stuirt_exe_path	The path for the STUIRT executable "STUIRT.exe". Example: "C:/STUIRT/STUIRT.exe".
target_dir	The directory/folder where the STUIRT analysis will be saved. The default value is the current working directory, i.e. get_wd().
analysis_name	A short file name for the data files created for the analysis.
add_options	A logical value. If TRUE, the keyword "OP" will be added to the syntax. This option is useful for detailed program output. Without OP, the section "OPTIONS AND DEFAULTS" will not appear in the main output file.
starting_values	A numeric vector of length two providing starting values for the slope and intercept of the linear transformation in the Haebara and Stocking-Lord methods. The default values are c(1, 0) (i.e., slope = 1 and intercept = 0).
number_of_iterations	An integer indicating the maximum number of iterations to obtain transformation constants that minimize criterion functions for the Haebara and Stocking-Lord methods. The default value is NULL, which lets STUIRT use a maximum of 20 iterations.
new_dist	A list specifying proficiency distribution of new group's distribution. The list should have three named elements: **"type"** A string indication the type of the distribution. The following values can be used: **"GH"** From STUIRT manual "Gauss-Hermite quadrature points and weights. This subkeyword can be used properly, if a continuous distribution of proficiency is known or estimated and the summation of the criterion function could be replaced by integration over the proficiency continuum. In the program, the proficiency distribution is assumed a standard normal one. The possible maximum number of quadrature points is 180. Although more than 180 quadrature points are theoretically possible, the author's experiences suggest that quadrature weights tend to be unstable when trying to obtain more than about 200 quadrature points. According to Zeng and Kolen (1994), even 80 quadrature points seem

to be enough to estimate the slope and intercept of a linear transformation to a satisfactory degree." (p.12)

"PN" From STUIRT manual: "PN stands for a polygonal approximation to a normal distribution. A polygonal approximation is often encountered in finding areas and evaluating integrals. PN can be used to evaluate n proficiency points and their weights to approximate a normal distribution having a value of mean and a value of std, with a left end point and a right end point being mean - multiple-of-std × std and mean + multiple-of-std × std, respectively. More specifically, n proficiency points are equally spaced over the range of a left end point to a right end point. At each proficiency point, the density from N(mean, std) is computed. All the densities are summed and then each density is divided by the sum so that the densities are standardized. The resulting values of densities are used as the weights. These steps are similar to those used in PC-BILOG (Mislevy & Bock, 1986)." (p.12)

"PB" From STUIRT manual: "Third, PB stands for a polygonal approximation to a four-parameter beta distribution , where alpha and beta are two scale parameters and l and u are lower and upper limits. To evaluate n proficiency points and their weights, the same logic used in PN is applied except that the interval [l, u] is divided into n subintervals and then the midpoint of each subinterval is used for a proficiency point." (p.12)

"RN" From STUIRT manual: "Fourth, RN stands for random numbers from a normal distribution. With two real numbers for a mean and a standard deviation, RN generates n pseudo-random proficiency values sampled from a normal distribution, N(mean, std)." (p.12)

"RU" From STUIRT manual: "RU stands for random numbers from a uniform distribution. With two real numbers for a lower limit and an upper limit, RU generates n pseudo-random proficiency values ranging from lower-limit to upper-limit." (p.12)

"ED" From STUIRT manual: "ED stands for equal distance in the intervals between two theta points on the proficiency scale. Users should supply two real numbers for a starting point and an ending point. The theta continuum ranging from the starting point to the ending point is divided into 1 - n intervals with an equal length." (p.12). In this function only "EQ" is available which means same constant weight of 1 is used.

"n" An integer specifying the "the number of proficiency points and should be a number between 1 and 1000, inclusive" (p.11)

"pars" A vector of numbers specifying the parameters used for each distribution. Based on the type of distribution following parameters should be specified.

GH No parameters is needed, it can be NULL

PN mean, std, multiple-of-std

PB alpha, beta, lower-limit, upper-limit

RN mean, std

RU lower-limit, upper-limit

ED starting, ending

The default is NULL. If so the STUIRT default will be used which is according to manual: "The default setting for proficiency values and their weights is that 25 proficiency values, which are equally spaced between -3.0 and 3.0, are used with the same weight 1.0 for all proficiency values." (p.13)

Here are some examples for different distributions:

GH `list(type = "GH", n = 41, pars = NULL)`

PN `list(type = "PN", n = 41, pars = c(0, 1, 4))`

PB `list(type = "PB", n = 51, pars = c(2, 15, 0, 40))`

RN `list(type = "RN", n = 31, pars = c(0, 1))`

RU `list(type = "RU", n = 41, pars = c(0, 3))`

ED `list(type = "ED", n = 25, pars = c(-3, 3))`

`ref_dist` A list representing the proficiency distribution of the reference group. Refer to the description of the 'new_dist' argument for more details.

`fs` A two-element string vector. Each element should be one of the following values: "DO" or "NO".

From the STUIRT manual: "The option keyword FS is used for standardizing the criterion functions for the Haebara and Stocking-Lord methods. The two subkeywords, DO and NO, are used to specify options. The first DO or NO is for the Haebara method and the second DO or NO is for the Stocking- Lord method. DO means that standardization is done and NO means that no standardization is done. What is meant by standardization of the criterion function is that one divides a sum of squared differences between characteristic curves in the criterion function by the number of the squared differences or the sum of weights assigned to the differences. For example, usually, to standardize the criterion function for the Stocking-Lord method, one divides the sum of squared differences between test characteristic curves by the number of proficiency values (or examinees). For more detailed information, refer to Kim and Lee (2004). Theoretically, the standardization of the criterion functions should not affect the solutions of the Haebara and Stocking-Lord methods. However, in practice, it could affect the solutions since the minimization algorithm used for nonlinear problems is affected by the magnitude of the criterion function due to its stopping rules. By default, standardization is conducted for both the Haebara and Stocking-Lord criterion functions." (p.13)

If the value is NULL where this keyword is not added to the syntax and STUIRT will use it's default values.

`sy` A two-element string vector. Both of the elements should be one of the following values: "BI", "ON" or "NO".

From the STUIRT manual: "The option keyword SY is used to define criterion functions as non-symmetric or symmetric. Three subkeywords, BI, NO, and ON, are used to specify options. The first BI, NO, or ON is for the Haebara method and the second BI, NO, or ON is for the Stocking-Lord method. Theoretically, the criterion function for either of the Haebara and Stocking-Lord methods could be defined in three symmetry-related ways. The first is one in which the criterion function is defined only on the old scale as in the typical use of the Stocking-Lord method (new-to-old direction: NO). The second is one

in which the criterion function is defined only on the new scale (old-to-new direction: ON). The third is one in which the criterion function is defined on the both old and new scales as in the use of the Haebara method (new-to-old and old-to-new, i.e., bi-directional: BI) Theoretically, the three ways, BI, NO, and ON, to define the criterion function in question should give the same solutions as far as sampling error and model misfit do not happen. However, with sample data, the three ways will give different solutions for scale transformation. The default setting is in such that SY BI BI." (p.14)

If the value is `NULL` where this keyword is not added to the syntax and STUIRT will use it's default values.

`lm` A six element list with following elements: (1) slope, (2) intercept, (3) number-of-searches, (4) radius, (5) tolerance, and, (6) either "NO" or "IN".

From the STUIRT manual: "The option keyword LM is used to search for possible local minimum solutions for the scale transformation constants after the first solutions for the Haebara and Stocking-Lord methods are obtained. Three subkeywords, NO, IN, and FI are prepared to instruct the program how and where to show the resulting history of local minimum search. If NO is used, no history is shown in an output file. If IN is used, the resulting history is shown within the main output file specified by users or opened by the program. If FI followed by a file name is used, the resulting history is saved separately in the file, which does not need to be located in the folder having the executable file of STUIRT." (p.14)

If the value is `NULL` where this keyword is not added to the syntax and STUIRT will use it's default values.

`ko` A string that specify "input files for the program POLYEQUATE". Available values are `"MM"`, `"MS"`, `"HA"` and `"SL"`.

From the STUIRT manual: "The four subkeywords, MM, MS, HA, and SL stand for the mean/mean, mean/sigma, Haebara, and Stocking-Lord methods, respectively. " (p.15)

If the value is `NULL` where this keyword is not added to the syntax and STUIRT will use it's default values.

`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`.

Author(s)

Emre Gonulates

References

Kim, S., & Kolen, M. J. (2004). STUIRT [Computer software]. Iowa City, IA: Iowa Testing Programs, The University of Iowa

Examples

```
## Not run:
```

```
# ---------- Mixed Models ------------ #
n_item <- 30
models <- sample(c("3PL", "GPCM2"), n_item, TRUE)
new_ip <- generate_ip(model = models, D = 1.702)
old_ip_df <- data.frame(new_ip)
old_ip_df$a <- old_ip_df$a + round(runif(n_item, min = -.2, max = .2), 2)
old_ip_df$b <- old_ip_df$b + round(runif(n_item, min = -.2, max = .2), 2)
old_ip_df$d1 <- old_ip_df$d1 + round(runif(n_item, min = -.2, max = .2), 2)
old_ip_df$d2 <- old_ip_df$d2 + round(runif(n_item, min = -.2, max = .2), 2)
old_ip_df$d3 <- old_ip_df$d3 + round(runif(n_item, min = -.2, max = .2), 2)
ref_ip <- itempool(old_ip_df)

result <- equate_stuirt(new_ip = new_ip,
                        ref_ip = ref_ip,
                        target_dir = "C:/Temp/testthat-stuirt",
                        stuirt_exe_path = "C:/STUIRT/STUIRT.exe",
                        )
result


## End(Not run)
```

`est_ability` *Estimate Examinee Ability*

Description

This function estimates examinee ability using different methods, including Owen's Bayesian estimation, Maximum Likelihood estimation, Maximum-a-Posteriori and Expected-a-Posteriori.

Usage

```
est_ability(
  resp,
  ip = NULL,
  method = c("eap", "ml", "map", "bm", "owen", "sum_score"),
  ...,
  prior_dist = c("norm", "unif", "lnorm", "gamma", "t", "cauchy"),
  prior_pars = c(0, 1),
  theta_range = c(-5, 5),
  number_of_quads = 41,
  tol = 1e-06,
  output_type = c("list", "data.frame", "tibble")
)
```

Arguments

resp	A Response_set-class, matrix or a data.frame object holding responses. Missing responses are excluded from the ability estimation.
ip	An Item-class, Itempool-class or a Testlet-class object. If ip is not an Itempool-class object, the function attempts to convert it. While the default is NULL, this argument is required for all methods except when method = "sum_score".
method	The method used for ability estimation. The default is "eap". Available methods: 'sum_score' Basic sum (raw) score of responses. 'owen' Owen's Bayesian Ability Estimation. This method is suitable for dichotomous IRT models (e.g., 'Rasch', '1PL', '2PL', '3PL' and '4PL'). Testlet groupings are ignored and items within testlets are treated as standalone items. Formulas were implemented in Owen (1975) and Vale (1977). The original formulation does not include the D parameter. If D = 1, the original solution is obtained. If D = 1.7, the a parameter is multiplied by this number. The user needs to provide prior parameters, i.e., prior_pars. These should be a numeric vector of length two, with the first component as the prior mean and the second as the prior standard deviation (not variance). For example, if the prior mean is 0.1 and the prior standard deviation is 2, set the prior parameters as prior_pars = c(0.1, 2). 'ml' Maximum Likelihood Ability Estimation via Newton-Raphson Algorithm. 'eap' Expected-a-Posteriori Ability Estimation. Prior information must be provided for this function. The number of quadrature points can also be specified using the argument number_of_quads. 'map' **or** 'bm' Maximum-a-Posteriori Ability Estimation (or Bayes Modal estimation). Prior information must be provided for this function. Currently, only 'norm' prior distribution is available.
...	Additional arguments passed to specific methods.
prior_dist	The shape of the prior distribution. Available options are: **'norm'** Normal distribution **'unif'** Uniform distribution **'t'** t distribution **'cauchy'** Cauchy distribution The 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 normal prior distribution. 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 bounded within this interval. 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.

`tol` The precision level of ability estimate. The final ability estimates will be rounded to remove precision smaller than the `tol` value. Default is `1e-06`.

`output_type` A string specifying the output type of the function. Default is `"list"`. Options include:

"list" Function returns a `list` object with elements `est` and `se`.

"data.frame" Function returns a `data.frame` object with columns `examinee_id`, `est` and `se`.

"tibble" If the `tibble` package is available, the function returns a `tibble` object with columns `examinee_id`, `est` and `se`.

Value

`est` The estimated examinee abilities. If the response vector for a subject contains all `NA`s, then `est` will be `NA` to differentiate from cases where all answers are incorrect.

`se` The standard errors of the ability estimates. For `"sum_score"` method, all standard errors will be `NA`. For Bayesian methods (like EAP, MAP or Owen's), this value is the square root of the posterior variance.

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(n = 7)
resp <- sim_resp(ip, theta = rnorm(3))

### EAP estimation ###
est_ability(resp, ip)
est_ability(resp, ip, number_of_quads = 81)
# The default prior_dist is 'norm'. prior_pars = c(mean, sd)
est_ability(resp, ip, prior_pars = c(0, 3))
# prior_pars = c(min, max)
est_ability(resp, ip, prior_dist = 'unif',  prior_pars = c(-3, 3))
# prior_pars = c(df)
est_ability(resp, ip, prior_dist = 't',  prior_pars = 3)
```

```
# prior_pars = c(location, scale)
est_ability(resp, ip, prior_dist = 'cauchy',  prior_pars = c(0, 1))


### MAP estimation (Bayes Modal estimation) ###
est_ability(resp, ip, method = "map")
# The default prior_dist is 'norm'. prior_pars = c(mean, sd)
est_ability(resp, ip, method = "map", prior_pars = c(0, 2))


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

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

`est_bilog` *Item Calibration via BILOG-MG*

Description

The function `est_bilog` facilitates item calibration through BILOG-MG. It offers two modes of operation: executing BILOG-MG in batch mode or processing pre-generated BILOG-MG output files. When using the former, ensure BILOG-MG is installed in the directory specified by `bilog_exe_folder`.

In the latter case, if the necessary BILOG-MG files (e.g., "<analysis_name>.PAR", "<analysis_name>.PH1", etc.) exist and `overwrite = FALSE`, there is no need for the BILOG-MG program itself. This function is capable of parsing BILOG-MG output without it.

Both BILOG-MG 3.0 and BILOG-MG 4.0 are supported. Refer to the `bilog_exe_folder` argument for guidance on selecting the desired version.

Usage

```
est_bilog(
  x = NULL,
  model = "3PL",
  target_dir = getwd(),
  analysis_name = "bilog_calibration",
  items = NULL,
  examinee_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,
  fix = NULL,
  scoring_options = c("METHOD=1", "NOPRINT"),
  calib_options = c("NORMAL"),
  prior_ability = NULL,
  prior_ip = NULL,
  overwrite = FALSE,
  show_output_on_console = TRUE,
  bilog_exe_folder = file.path("C:/Program Files/BILOGMG")
)
```

Arguments

x	Either a data.frame, matrix, or a Response_set-class object. Set this to NULL if you only intend to read BILOG-MG output from target_dir.
model	Specifies the item model. Options include: "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 is "3PL".
target_dir	The directory where BILOG-MG analysis and data files will be stored. The default is the current working directory (i.e., get_wd()).
analysis_name	A concise filename (without extension) used for the data files created for the analysis.
items	A vector of column names or numbers in x representing the responses. If no entry for item names is desired in the syntax file, set items = "none".
examinee_id_var	The column name or number containing individual subject IDs. If not provided (i.e., examinee_id_var = NULL), the program will check whether the data provided has row names and use them as subject IDs.
group_var	The column name or number containing group membership information for multi-group calibration. Ideally, the grouping variable should be represented by single-digit integers. If other data types are provided, integer values will be automatically assigned to the variables. The default is NULL, indicating no multi-group analysis will be performed.
logistic	A logical value indicating whether to use logistic calibration.

- If TRUE, the calibration assumes the natural metric of the logistic response function in all calculations.
- If FALSE, the logit is multiplied by a factor of 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. This value is used as an automatic starting value for estimating pseudo-guessing parameters.

The default value is NULL. For 3PL, the default is 5, and for 1PL and 2PL, it's 1000. This value will be represented in the BILOG-MG control file as: NALT = num_of_alternatives.

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

num_of_quadrature

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

max_em_cycles An integer (0, 1, ...) representing the maximum number of EM cycles. This value will be represented in the BILOG-MG control file as: CYCLES = max_em_cycles. The default value is 100.

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

reference_group

A value indicating which group's ability distribution will be set to mean = 0 and standard deviation = 1. For example, if the group_var has values 1 and 2 representing two different groups, setting reference_group = 2 will result in the group with code 2 having an ability distribution with mean 0 and standard deviation 1.

When groups are assumed to come from a single population, set this value to 0.

The default value is 'NULL'.

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

fix Specifies whether the parameters of specific items are free to be estimated or should be held fixed at their starting values. This argument accepts a data.frame with an item_id column, in which items for which the item parameters will be held fixed; a, b, c parameter values. See the examples section for a demonstration.

scoring_options

A string vector of keywords/options to be included in the SCORE section of the BILOG-MG syntax. If scoring individual examinees is not needed, set this to NULL.

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

The primary option to add to this vector is "METHOD=n". The available options are:

"METHOD=1" Maximum Likelihood (ML)

"METHOD=2" Expected a Posteriori (EAP)

"METHOD=3" Maximum a Posteriori (MAP)

Additionally, you can include the following keywords:

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

"FIT": Computes the likelihood ratio chi-square goodness-of-fit statistic for each response pattern.

"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"

Refer to the BILOG-MG manual for detailed explanations of these keywords/options.

`calib_options` A string vector of additional keywords/options for the CALIB section in the BILOG-MG syntax. This is in addition to the keywords NQPT, CYCLES, NEWTON, CRIT, and REFERENCE.

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

Including "NORMAL" in `calib_options` assumes that the prior distributions of ability in the population follow a normal distribution.

Including "COMMON" estimates a common value for the lower asymptote for all items in the 3PL model.

If you're calibrating items using the "RASCH" model, set the argument `model = "Rasch"` instead of adding "RASCH" to `calib_options`.

Additional keywords/options that can be added to `calib_options` include:

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

Refer to the BILOG-MG manual for detailed explanations of these keywords/options.

NOTE: Do not add the following keywords to `calib_options` as they are already included in other arguments: NQPT, CYCLES, NEWTON, CRIT, REFERENCE.

`prior_ability` Prior ability refers to the quadrature points and weights representing the discrete finite distribution of ability for the groups. It should be structured as a list in the following format:

`list(<GROUP-NAME-1> = list(points = ...., weights = ...), <GROUP-NAME-2> = list(points = ...., weights = ...), ... )`

Here, <GROUP-NAME-1> refers to the name of the first group, <GROUP-NAME-2> refers to the name of the second group, and so on.

Please refer to the examples section for a practical implementation.

`prior_ip` Specify prior distributions for item parameters. The default value is NULL, in which case BILOG-MG defaults will be used. To specify priors, provide a list containing one or more of the following elements:

"ALPHA" "'alpha' parameters for the beta prior distribution of lower asymptote (guessing) parameters"

"BETA" "'beta' parameters for the beta prior distribution of lower asymptote (guessing) parameters."

"SMU" prior means for slope parameters

"SSIGMA" prior standard deviations for slope parameters

"TMU" prior means for threshold parameters

"TSIGMA" prior standard deviations for threshold parameters

Quoted descriptions were taken from the BILOG-MG manual.

Examples:

1. A specific set of priors: `list(ALPHA = 4, BETA = 3, SMU = 1, SSIGMA = 1.648, TMU = 0, TSIGMA = 2)`
2. A very strong prior for guessing which almost fixes all guessing parameters at 0.2: `list(ALPHA = 1000000, BETA = 4000000)`
3. Fix guessing at 0.25: `list(ALPHA = 1000000, BETA = 3000000)`

In general, one can adjust the alpha and beta parameters to achieve a desired outcome, considering that the mode of the beta distribution is calculated as:

$$mode = \frac{\alpha - 1}{\alpha + \beta - 2}$$

Additionally, setting SSIGMA or TSIGMA to a very small value effectively fixes the item parameters. For example, TSIGMA = 0.005 or SSIGMA = 0.001. Be aware that this may lead to convergence issues.

Note: A non-null prior_ip value will automatically add the READPRIOR option to the CALIB section.

`overwrite` If set to TRUE, any existing BILOG-MG analysis files with the same name in the target path will be overwritten.

`show_output_on_console`
A logical value indicating whether to capture and display the output of the command on the R console. The default is TRUE.

`bilog_exe_folder`
The directory containing the Bilog-MG executable files. This function supports two versions: BILOG-MG 3 and BILOG-MG 4. For BILOG-MG version 3, the directory should include the files "blm1.exe", "blm2.exe", and "blm3.exe". The default location for version 3 is `file.path("C:/Program Files/BILOGMG")`. If you have version 4 installed, the argument should point to the directory where "BLM64.exe" is located, which is typically "C:/Program Files/BILOG-MG/x64".

Value

A `list` with following elements is returned:

A `list` with the following elements is returned:

"ip" An `Itempool-class` object holding the item parameters. Check `...$converged` to ensure the model has converged before using `ip`. This element is not created when `model = "CTT"`.

"score" A data frame object containing information on examinee scores such as items attempted (`tried`), items answered correctly (`right`), estimated examinee scores (`ability`), standard errors of ability estimates (`se`), and response string probabilities (`prob`). This element is not created when `model = "CTT"`.

"ctt" Classical Test Theory (CTT) statistics, including p-values, biserial, and point-biserial estimates calculated by BILOG-MG. If there are groups, group-specific CTT statistics can be found in `ctt$group$GROUP-NAME`. Overall statistics for the entire group are located at `ctt$overall`.

"failed_items" A data frame containing items that could not be estimated.

"syntax" The syntax file.

"em_cycles" E-M Cycles of the calibration.

"newton_cycles" Newton Cycles of the calibration

"cycle" The number of cycles run before calibration converges or fails to converge.

"largest_change" The largest change observed between the last two cycles.

"neg_2_log_likelihood" -2 Log Likelihood value of the last step of the E-M cycles. See also `$em_cycles`. This value is `NULL` when the model does not converge. This element is not created when `model = "CTT"`.

"posterior_dist" Posterior quadrature points and weights.

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

Author(s)

Emre Gonulates

Examples

```
## Not run:
###############################################
############## Example 1 - 2PL ##############
###############################################
# IRT Two-parameter Logistic Model Calibration

# Create responses to be used in BILOG-MG estimation
true_theta <- rnorm(4000)
true_ip <- generate_ip(n = 30, model = "2PL")
resp <- sim_resp(true_ip, true_theta)

# 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

# Get estimated scores
head(bilog_calib$score)

# Compare true and estimated abilities
plot(true_theta, bilog_calib$score$ability, xlab = "True Theta",
     ylab = "Estimated theta")
abline(a = 0, b = 1, col = "red", lty = 2)

# Compare true item parameters
plot(true_ip$a, bilog_calib$ip$a, xlab = "True 'a'", ylab = "Estimated 'a'")
abline(a = 0, b = 1, col = "red", lty = 2)

plot(true_ip$b, bilog_calib$ip$b, xlab = "True 'b'", ylab = "Estimated 'b'")
abline(a = 0, b = 1, col = "red", lty = 2)

# Note that Bilog-MG centers the ability at mean 0.
mean(bilog_calib$score$ability)

# Quadrature points and posterior weights:
head(bilog_calib$posterior_dist)

###############################################
############## Example 2 - EAP ##############
###############################################
# Getting 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)
head(result$score)


###############################################
############## Example 3 - Rasch ##############
###############################################
# Rasch Model Calibration
true_theta <- rnorm(400)
true_ip <- generate_ip(n = 30, model = "Rasch")
resp <- sim_resp(true_ip, true_theta)

# Run calibration
bilog_calib <- est_bilog(x = resp, model = "Rasch",
                         target_dir = "C:/Temp/Analysis",
                         overwrite = TRUE)
```

```
bilog_calib$ip

plot(true_ip$b, bilog_calib$ip$b, xlab = "True 'b'", ylab = "Estimated 'b'")
abline(a = 0, b = 1, col = "red", lty = 2)

# Note that the 'b' parameters are rescaled so that their arithmetic mean
# equals 0.0.
mean(bilog_calib$ip$b)


###############################################
############## Example 4 - 3PL ################
###############################################
# IRT Three-parameter Logistic Model Calibration

# Create responses to be used in BILOG-MG estimation
true_theta <- rnorm(4000)
true_ip <- generate_ip(n = 30, model = "3PL")
resp <- sim_resp(true_ip, true_theta)

# The following line will run BILOG-MG, estimate 3PL model and put the
# analysis results under the target directory:
bilog_calib <- est_bilog(x = resp, model = "3PL",
                         target_dir = "C:/Temp/Analysis",
                         overwrite = TRUE)
Estimated item pool:
bilog_calib$ip

# Convergence status:
bilog_calib$converged
# Number of EM cycles:
bilog_calib$cycle
# Note that the maximum number of EM cycles were set at:
bilog_calib$input$max_em_cycles
# Largest change at the last cycle (note that convergence criterion is 0.01)
bilog_calib$largest_change
# Estimated Scores:
bilog_calib$score
# CTT stats calculated by BILOG-MG:
bilog_calib$ctt

###############################################
############## Example 5 - 1PL ################
###############################################
# One-Parameter Logistic Model Calibration
true_theta <- rnorm(800)
true_ip <- generate_ip(n = 30, model = "2PL")
# Set 'a' parameters to a fixed number
true_ip$a <- 1.5
resp <- sim_resp(true_ip, true_theta)

# Run calibration
bilog_calib <- est_bilog(x = resp, model = "1PL",
```

```
                        target_dir = "C:/Temp/Analysis",
                        overwrite = TRUE)
# Note that all 'a' parameter values and all 'se_a' values are the same:
bilog_calib$ip

plot(true_ip$b, bilog_calib$ip$b, xlab = "True 'b'", ylab = "Estimated 'b'")
abline(a = 0, b = 1, col = "red", lty = 2)


#############################################################
############## Example 6.1 - Multi-group - 3PL ##############
#############################################################
# Multi-group IRT calibration - 3PL

## Generate Data ##
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))
# Create response data where first column group information
dt <- data.frame(level = c(rep("Lower", n_lower), rep("Upper", n_upper)),
                 resp)

## Run Calibration ##
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)

# Posterior distributions of 'Lower' (in red)  and 'Upper' group
plot(mg_calib$posterior_dist$Upper$point,
     mg_calib$posterior_dist$Upper$weight)
points(mg_calib$posterior_dist$Lower$point,
       mg_calib$posterior_dist$Lower$weight, col = "red")


#############################################################
```

```
############## Example 6.2 - Multi-group - Response_set #####
#################################################################
# Multi-group IRT calibration - Response_set 2PL

## Generate Data ##
ip <- generate_ip(n = 35, model = "2PL", D = 1.7)
n_upper <- sample(1000:2000, 1)
n_lower <- sample(1000:2000, 1)
resp_set <- generate_resp_set(
  ip = ip, theta = c(rnorm(n_lower), rnorm(n_upper, 1.5, .25)))
# Attach the group information
resp_set$mygroup <- c(rep("Lower", n_lower), rep("Upper", n_upper))

## Run Calibration ##
mg_calib <- est_bilog(x = resp_set,
                      model = "2PL",
                      group_var = "mygroup",
                      reference_group = "Lower",
                      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

#################################################################
############## Example 6.3 - Multi-group - 1PL ################
#################################################################
# Multi-group IRT calibration - 1PL

## Generate Data ##
n_item <- sample(30:40, 1)
ip <- generate_ip(n = n_item, model = "2PL", D = 1.7)
ip$a <- 1.25
n_upper <- sample(700:1000, 1)
n_lower <- sample(1200:1800, 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))
# Create response data where first column group information
dt <- data.frame(level = c(rep("Lower", n_lower), rep("Upper", n_upper)),
                 resp)

## Run Calibration ##
mg_calib <- est_bilog(x = dt,
                      model = "1PL",
                      group_var = "level",
                      reference_group = "Lower",
                      items = 2:ncol(dt), # Exclude the 'group' column
                      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 6.4 - Multi-group - Prior Ability ######
################################################################
# Multi-group IRT calibration - 3PL with user supplied prior ability
# parameters
n_item <- sample(40:70, 1)
ip <- generate_ip(n = n_item, model = "3PL", D = 1.7)
n_upper <- sample(2000:4000, 1)
n_lower <- sample(3000:5000, 1)
theta_upper <- rgamma(n_upper, shape = 2, rate = 2)
# hist(theta_upper)
theta_lower <- rnorm(n_lower)
true_theta <- c(theta_lower, theta_upper)
resp <- sim_resp(ip = ip, theta = true_theta, prop_missing = .2)
# Create response data where first column group information
dt <- data.frame(level = c(rep("Lower", n_lower), rep("Upper", n_upper)),
                 resp)

# Set prior ability parameters
points <- seq(-4, 4, .1)
prior_ability = list(
  Lower = list(points = points, weights = dnorm(points)),
  # Also try misspecified prior:
  # Upper = list(points = points, weights = dnorm(points, 1, .25))
  Upper = list(points = points, weights = dgamma(points, 2, 2))
  )
mg_calib <- est_bilog(x = dt,
                      model = "3PL",
                      group_var = "level",
                      reference_group = "Lower",
                      items = 2:ncol(dt), # Exclude the 'group' column
                      calib_options = c("IDIST = 2"),
                      prior_ability = prior_ability,
                      # Use MAP ability estimation.
                      scoring_options = c("METHOD=3"),
                      target_dir = target_dir,
                      overwrite = TRUE,
                      show_output_on_console = FALSE)

# Check whether model has convergence
mg_calib$converged

# Group information
```

```
mg_calib$group_info

# Quadrature points and posterior weights:
head(mg_calib$posterior_dist$Lower)

plot(mg_calib$posterior_dist$Lower$point,
     mg_calib$posterior_dist$Lower$weight,
     xlab = "Quadrature Points",
     ylab = "Weights",
     xlim = c(min(c(mg_calib$posterior_dist$Lower$point,
                    mg_calib$posterior_dist$Upper$point)),
              max(c(mg_calib$posterior_dist$Lower$point,
                    mg_calib$posterior_dist$Upper$point))),
     ylim = c(min(c(mg_calib$posterior_dist$Lower$weight,
                    mg_calib$posterior_dist$Upper$weight)),
              max(c(mg_calib$posterior_dist$Lower$weight,
                    mg_calib$posterior_dist$Upper$weight))))
points(mg_calib$posterior_dist$Upper$point,
       mg_calib$posterior_dist$Upper$weight, col = "red")

# Comparison of true and estimated item parameters
plot(ip$a, mg_calib$ip$a, xlab = "True 'a'", ylab = "Estimated 'a'")
plot(ip$b, mg_calib$ip$b, xlab = "True 'b'", ylab = "Estimated 'b'")
plot(ip$c, mg_calib$ip$c, xlab = "True 'c'", ylab = "Estimated 'c'")

# Ability parameters
plot(true_theta, mg_calib$score$ability,
     xlab = "True Theta",
     ylab = "Estimated Theta")
abline(a = 0, b = 1, col = "red")



######################################################################
############## Example 7 - Read BILOG-MG Output without BILOG-MG ###
######################################################################
# To read BILOG-MG output files saved in the "Analysis/" directory with file
# names like "my_analysis.PH1", "my_analysis.PH2", etc., and without
# performing the calibration (no need for an installed BILOG-MG program on
# your computer), use the following syntax:
result <- est_bilog(target_dir = file.path("Analysis/"), model = "3PL",
                    analysis_name = "my_analysis", overwrite = FALSE)



######################################################################
############## Example 8 - Fixed Item Parameters ###################
######################################################################
# Fixed item calibration involves setting specific item parameters to
# predefined values while allowing other items' parameters to be freely
# estimated.
# If you want to fix all values of a particular item parameter(s), you can
# use strong priors. Refer to the documentation for the "prior_ip" argument
```

```
# for more details.

# Create responses to be used in BILOG-MG estimation
true_theta <- rnorm(3000)
true_ip <- generate_ip(n = 30, model = "3PL")
resp <- sim_resp(true_ip, true_theta)

# Setup the data frame that will hold 'item_id's to be fixed, and the
# item parameters to be fixed.
fix_pars <- data.frame(item_id = c("Item_5", "Item_4", "Item_10"),
                       a = c(1, 1.5, 1.75),
                       b = c(-1, 0.25, 0.75),
                       c = c(.15, .25, .35))

fixed_calib <- est_bilog(x = resp, fix = fix_pars,
                         target_dir = "C:/Temp/Analysis", overwrite = TRUE)
# Check item parameters for  Item_4, Item_5, Item_10:
fixed_calib$ip

######### #########
# If only some of the parameters are supplied, the defaults will be used
# for the missing parameters. For example, for the example below, the
# default 'a' parameter value is 1, and the default 'c' parameter value is
# (1/num_of_alternatives) = (1/5) = 0.2.
fix_pars2 <- data.frame(item_id = c("Item_1", "Item_2", "Item_3"),
                        b = c(-1, 0.25, 0.75))

fixed_calib2 <- est_bilog(x = resp, fix = fix_pars2,
                          target_dir = "C:/Temp/Analysis", overwrite = TRUE)
# Check item parameters for  Item_4, Item_5, Item_10:
fixed_calib2$ip


#####################################################################
############## Example 9 - 3PL with Common Guessing ##############
#####################################################################
# IRT Three-parameter Logistic Model Calibration with Common Guessing

# Create responses to be used in BILOG-MG estimation
true_theta <- rnorm(4000)
true_ip <- generate_ip(n = 30, model = "3PL")
resp <- sim_resp(true_ip, true_theta)

# Run calibration:
bilog_calib <- est_bilog(x = resp, model = "3PL",
                         target_dir = "C:/Temp/Analysis",
                         calib_options = c("NORMAL", "COMMON"),
                         overwrite = TRUE)

# Note the 'c' parameters
bilog_calib$ip

#####################################################################
```

```
############## Example 10 - 3PL with Fixed Guessing ##############
####################################################################
# IRT Three-parameter Logistic Model Calibration with Fixed Guessing
# The aim is to fix guessing parameters of all items to a fixed
# number like 0.25

true_theta <- rnorm(3000)
true_ip <- generate_ip(n = 30, model = "3PL")
true_ip$c <- 0.25
resp <- sim_resp(true_ip, true_theta)
prc1 <- est_bilog(x = resp, model = "3PL", target_dir = "C:/Temp/Analysis",
                  prior_ip = list(ALPHA = 10000000, BETA = 30000000),
                  overwrite = TRUE)



## End(Not run) # end dontrun
```

`est_flexmirt` *Unidimensional Item Calibration via flexMIRT*

Description

`est_flexmirt` runs flexMIRT in batch mode. This function requires flexMIRT program already installed on the Windows machine. Visit <https://vpgcentral.com/software/flexmirt/> for more details about the software. Even though flexMIRT can run various models, only a selected set of unidimensional models can be fitted using `est_flexmirt` function.

Usage

```
est_flexmirt(
  x = NULL,
  model = NULL,
  target_dir = getwd(),
  analysis_name = "flexMIRT_calibration",
  item_ids = NULL,
  D = 1,
  max_em_cycles = c(500, 100),
  quadrature = c(49, 6),
  em_tol = c(1e-04, 1e-09),
  prior = NULL,
  gof = "Basic",
  examinee_id_var = NULL,
  group_var = NULL,
  scoring_method = NULL,
  additional_options = NULL,
  additional_constraints = NULL,
  flexmirt_exe = NULL,
```

```
  overwrite = FALSE,
  show_output_on_console = TRUE
)
```

Arguments

x	A matrix/data.frame/Response_set object including examinee item responses. In it's bare form, it can be a matrix of item responses, where ideally the column names are the item IDs and row names are the examinee IDs (though neither are necessary).
model	The psychometric model(s) of items. The user can provide an input in the following three ways: (a) A vector of length one which represents the model of each item. (b) A vector which has the same length as the number of items that will be calibrated that specifies the model of each item. (c) NULL, the default value, where the program will check the number of categories of each item. Items with two or fewer categories will be calibrated using "3PL" (three-parameter logistic IRT model), items with more than two categories will be calibrated using "GRM" (Graded Response Model). "1PL" One-parameter logistic model. "2PL" Two-parameter logistic model. "3PL" Three-parameter logistic model. "GRM" Graded Response Model "GPCM" Generalized Partial Credit Model
target_dir	The directory/folder where the flexMIRT syntax data files and output will be saved. The default value is the current working directory, i.e. get_wd().
analysis_name	This will be the file names of the data, flexMIRT syntax file and output files. The default value is "flexMIRT_calibration".
item_ids	A vector of column names or numbers of the x that represents the responses. The default value is NULL where all of the columns in the data are assumed to be a response matrix (unless specified by group_var or examinee_id_var arguments).
D	Scaling constant. Default value is 1. If it is not equal to 1, a new line added to constraints to multiply the slope parameter with the D value specified.
max_em_cycles	A numeric vector of length two specifying the maximum number of iterations allowed in E- and M-steps. The default value is c(500, 100) where there will be maximum 500 iterations allowed in E-steps and 100 iterations M-steps,
quadrature	A numeric vector of length two specifying the number of quadrature points and the maximum theta value. The default value is c(49, 6) where there will be 49 rectangular quadrature points over -6 and +6,
em_tol	A numeric vector of length two specifying the convergence criteria for E- and M-steps. The default value is c(1e-4, 1e-9) where convergence criteria for E-steps is 0.0001 and the convergence criteria for M-step is 1e-9.
prior	A data frame that specifies the priors for the estimated item parameters. There are two possible options. Option 1: The same priors will be imposed on all items. The data.frame should have four columns:

"par" The parameter on which prior will be imposed. It can take the following values: "intercept" for location parameters (usually for item difficulty parameters, "slope" for item discrimination parameters, "guessing" for lower asymptote parameter of 3PL.

"dist" The distribution of the prior. It can take the following values: "normal" for normal distribution, "lognormal" for log-normal distribution, and "beta" for Beta distribution.

"v1" A number for the first parameter of the selected distribution. For normal and log-normal distributions this is the mean of the distribution. For Beta distribution, this is the alpha-1 value. So, if the a Beta distribution with alpha = 10 desired, the value of 'v1' should be 11.

"v2" A number for the second parameter of the selected distribution. For normal and log-normal distributions this is the standard deviation of the distribution. For Beta distribution, this is the beta-1 value. So, if the a Beta distribution with beta = 3 desired, the value of 'v2' should be 4.

Here is an example: `prior <- data.frame(par = c("intercept", "slope", "guessing"), dist = c("normal", "lognormal", "beta"), v1 = c(0, 0, 1), v2 = c(1, 0.5, 3))`

Option 2: Different priors will be assigned to individual items. The data.frame should have five columns. In addition to four columns described above, a column specifying item's ID should be added. The column name should be "item_id" and it's values should correspond to the item ID's specified in the response data.

Here is an example: `prior <- data.frame( item_id = c("Item_1", "Item_1", "Item_3", "Item_3", "Item_10"), par = c("intercept", "slope", "intercept", "slope", "slope"), dist = c("normal", "lognormal", "normal", "lognormal", "lognormal"), v1 = c(0, 0, 0, 0, 0), v2 = c(1, 0.5, 2, 0.6, 0.7))`

`gof` A string specifying the extent of Goodness-of-fit indices that will be calculated and reported. The available options are "Basic" (the default value), "Extended" and "Complete".

`examinee_id_var`

If examinee IDs are saved in one of the columns of argument `x`, this will be the name of that column. The default value is `NULL`. When the value is `NULL`, the program will check the row names of the data.frame or matrix. If the row names are not `NULL`, the program will use these values as examinee IDs.

`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.

`scoring_method` A string value representing the method of scoring. The currently available options are: "EAP" and "MAP".

`additional_options`

A vector of strings that will be added to the syntax. For example, when `scoring_method = "ML"`, the minimum and maximum values of theta estimates needs to be specified: `c("MinMLscore = -5;", "MaxMLscore = 5;")`. Or, when estimating "3PL" parameters `"NormalMetric3PL = Yes;"` can be added to get a normal metric

scale. See flexMIRT manual for other options. The default value is NULL, where no additional options will be added to the syntax. Note that following additional options will be added by other arguments of this function, so you don't need to add them in this argument separately: "Quadrature =", "MaxE =", "MaxM =", "Etol =", "Mtol =", "Score ="

additional_constraints

The The default value is NULL, where no additional constraints will be added to the syntax except the constraints that will be added for models such as "Rasch" or "1PL".

Examples of additional constraints can be:

- Guessing parameter. For example, if the items are multiple choice and there are four possible choices, the prior distribution of the pseudo-guessing parameter can be set to Beta(1, 3), where parameters alpha = 2 (2 - 1 = 1) and beta = 4 (4-1 = 3). This will correspond to prior sample size of 4 and prior mode of (1 / (3+1) = 0.25). You can add the following line to the additional_constraints (see Example 2 below.):
  Prior (Item_1-Item_35), Guessing : Beta(1.0,3.0);
  The distribution can be different, for example, for Normal distribution provide mean and standard deviation, for Log-normal distribution provide mean and standard deviation in logarithmic scale.
  For example, in the code below, a normal prior with mean -1.09 and standard deviation 0.5 is imposed on the *logit* of the guessing parameters: Prior (Item_1-Item_35), Guessing : Normal(-1.09,0.5);
- Slope parameters. The following argument will set the item slopes for Item_1 to Item_10 equal: Fix (Item_1-Item_10), Slope;
  Similarly a log-normal prior can be imposed on slope parameters: Prior (Item_1-Item_35), Slope : logNormal(1, 1.6487);

flexmirt_exe This is the executable file to run flexMIRT syntax. On most Windows computers this is the path where "WinFlexMIRT.exe" can be found. For example: "C:\Program Files\flexMIRT3.5.2\WinFlexMIRT.exe". By default the value is NULL, the function will search the relevant locations for "WinFlexMIRT.exe".

overwrite If TRUE and there is already a BILOG-MG data file in the target path with the same name, the 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.

Value

A list containing the calibration results

"ip" An Itempool-class object holding the item parameters. Please check whether model converged (using ...$converged) before interpreting/using ip.

"score" A data frame object that holds examinee IDs, ability estimates and standard error of ability estimates.

"syntax" The syntax file.

"converged" A logical value indicating whether a model has been converged or not. This value is TRUE only when both converged_first_order and converged_second_order are TRUE.

"converged_first_order" A logical value indicating whether first-order test indicates convergence.

"converged_second_order" A logical value indicating whether second-order test indicates convergence.

"convergence_details" A more detailed information about convergence. This element has two values, "First-order test" and "Second-order test". Use this information to further judge the convergence. From flexMIRT user manual (p.11): "the reported first-order test examines if the gradient has vanished sufficiently for the solution to be a stationary point. The second-order test tests if the information matrix is positive definite, a prerequisite for the solution to be a possible maximum. For the second-order test, reporting that the solution is a possible maximum simply means that the program reached a statistically desirable solution. The other possible message that may be printed for the outcome of the second-order test is "Solution is not a maximum; caution is advised." If a warning message is received for either the first- or second-order test, all parameter estimates should be taken a provisional and should not be used as final estimates, for future scoring, etc. but, rather, should be used to diagnose possible issues with the model/items."

"gof" The goodness-of-fit statistics.

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

Author(s)

Emre Gonulates

Examples

```
## Not run:

###############################################
############## Example 1 - 2PL ##############
###############################################
# IRT Two-parameter Logistic Model Calibration

# Create responses to be used in flexMIRT estimation
true_theta <- rnorm(1000)
true_ip <- generate_ip(n = 30, model = "2PL")
resp <- sim_resp(true_ip, true_theta)
# The following line will run flexMIRT, estimate 2PL model and put the
# analysis results in the target directory:
fm_calib <- est_flexmirt(x = resp, model = "2PL",
                         target_dir = "C:/Temp/Analysis", overwrite = TRUE)
# Check whether the calibration converged
fm_calib$converged
fm_calib$convergence_details

# Get the estimated item pool
fm_calib$ip

# See the BILOG-MG syntax
```

```
cat(fm_calib$syntax, sep = "\n")

# Get goodness-of-fit statistics and marginal reliability:
fm_calib$gof

# Get estimated scores
head(fm_calib$score)

# Compare true and estimated abilities
plot(true_theta, fm_calib$score$theta, xlab = "True Theta",
     ylab = "Estimated theta")
abline(a = 0, b = 1, col = "red", lty = 2)

# Compare true item parameters
plot(true_ip$a, fm_calib$ip$a, xlab = "True 'a'", ylab = "Estimated 'a'")
abline(a = 0, b = 1, col = "red", lty = 2)

plot(true_ip$b, fm_calib$ip$b, xlab = "True 'b'", ylab = "Estimated 'b'")
abline(a = 0, b = 1, col = "red", lty = 2)

mean(fm_calib$score$theta)

##############################################
############## Example 2 - 3PL ###############
##############################################
# IRT Three-parameter Logistic Model Calibration with D = 1.7

# Create responses to be used in flexMIRT estimation
true_theta <- rnorm(5000)
true_ip <- generate_ip(n = 35, model = "3PL", D = 1)
resp <- sim_resp(true_ip, true_theta)

# The following line will run 3PL calibration via flexMIRT:
fm_calib <- est_flexmirt(
  x = resp,
  model = "3PL",
  max_em_cycles = c(1000, 200),
  prior = data.frame(par = c("intercept", "slope", "guessing"),
                     dist = c("normal", "lognormal", "beta"),
                     v1 = c(0, 0, 1),
                     v2 = c(1, 0.5, 3)),
  target_dir = "C:/Temp/Analysis",
  overwrite = TRUE)

# Check whether the calibration converged
fm_calib$converged

fm_calib$convergence_details

# Get the estimated item pool
fm_calib$ip

# Get goodness-of-fit statistics and marginal reliability:
```

```
fm_calib$gof

# Get estimated scores
head(fm_calib$score)

# Compare true and estimated abilities
plot(true_theta, fm_calib$score$theta, xlab = "True Theta",
     ylab = "Estimated theta")
abline(a = 0, b = 1, col = "red", lty = 2)

# Compare true item parameters
plot(true_ip$a, fm_calib$ip$a, xlab = "True 'a'", ylab = "Estimated 'a'")
abline(a = 0, b = 1, col = "red", lty = 2)

plot(true_ip$b, fm_calib$ip$b, xlab = "True 'b'", ylab = "Estimated 'b'")
abline(a = 0, b = 1, col = "red", lty = 2)

mean(fm_calib$score$theta)


## End(Not run) # end dontrun
```

`est_irtpro` *Item Calibration via IRTPRO*

Description

`est_irtpro` runs the IRTPRO in batch mode.

This function requires IRTPRO already installed on your computer. The R program is designed to work on IRTPRO 6.0.

NOTE that sometimes IRTPRO requires administrative privileges to run each time it is opened. You can reopen R or RStudio with administrator privileges (right click R or RStudio icon in start menu and select 'More' > 'Run as administrator') to prevent IRTPRO to ask administrator permission each time it is run.

Usage

```
est_irtpro(
  x = NULL,
  model = "3PL",
  target_dir = getwd(),
  D = 1,
  analysis_name = "irtpro_calibration",
  items = NULL,
  examinee_id_var = NULL,
  group_var = NULL,
  reference_group = NULL,
  estimation_method = c("BAEM", "ADQ", "MHRM", "MCMC"),
```

```
 estimation_args = list(`E-Step` = c(500, 1e-05), SE = "S-EM", `M-Step` = c(500, 1e-09),
    Quadrature = c(49, 6), SEM = 0.001, SS = 1e-05),
 scoring_method = c("EAP", "MAP"),
 scoring_args = list(Mean = 0, SD = 1),
 misc_args = list(Decimal = 4, Processors = 1, `Min Exp` = 1),
print_extra = c("StdRes", "CTLD", "M2", "GOF", "Loadings", "P-Nums", "Diagnostic"),
 constraints = NULL,
priors = data.frame(model = c("1PL", "2PL", "2PL", "3PL", "3PL", "3PL"), parameter =
    c("Intercept[0]", "Slope[0]", "Intercept[0]", "Slope[0]", "Intercept[0]",
   "Guessing[0]"), prior_dist = c("Normal", "Lognormal", "Normal", "Lognormal",
  "Normal", "Beta"), prior_par_1 = c(0, 0, 0, 0, 0, 4), prior_par_2 = c(2, 1, 2, 1, 2,
    16)),
 overwrite = FALSE,
 show_output_on_console = TRUE,
 irtpro_exe_dir = file.path("C:/Program Files/IRTPRO 6.0")
)
```

Arguments

x	Either a data.frame, matrix or Response_set-class object. It is assumed that item values start from 0 and goes to number of distinct categories minus one. So, for example, for a polytomous items with four categories, the score values are assumed to be 0, 1, 2, 3. Recode the data to follow this pattern.
model	A string or a vector of strings to specify the psychometric model of the items. Either provide a single model for all items or provide a vector with the same length as the number of items where each value is one of the following: One-parameter logistic model ("1PL"), Two-parameter logistic model ("2PL"), three-parameter logistic model ("3PL"), Generalized Partial Credit model ("GPCM2"), Graded Response Model ("GRM").
target_dir	The directory/folder where the IRTPRO analysis and data files will be saved. The default value is the current working directory, i.e. get_wd().
D	Scaling constant. The default value is 1. If, for "2PL", "3PL" or "GRM" models, the item parameters needs to be converted to the commonly used normal scale where D = 1.7 or D = 1.702, change this value. The item discrimination parameters estimated by IRTPRO will be divided to D to get parameters on the new scale.
analysis_name	A short file name that will be used for the data files created for the analysis.
items	A vector of column names of the x that represents the responses. Default value is NULL where all items in x are assumed to be entering the calibration.
examinee_id_var	The column name or number that contains individual subject IDs. If none is provided (i.e. examinee_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. Currently, this function cannot read multi-

group calibration results. The default value is NULL, where no multi-group analysis will be performed.

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. The default value is NULL.

estimation_method

A string that can take one of the following values: "BAEM" (Bock-Aitkin), "ADQ" (Adaptive Quadrature). The methods "MHRM" (Metropolis-Hastings Robbins-Monro) and "MHRM" are not available at this time via this program.

estimation_args

A list with named arguments that will specify the estimation. Please use one of the following list templates for each estimation method.

"BAEM" list(`E-Step` = c(500, 1e-005), SE = "S-EM", `M-Step` = c(500, 1e-009), Quadrature = c(49, 6), SEM = 0.001, SS = 1e-005)

"ADQ" list(`E-Step` = c(100, 0.001), SE = "S-EM", Quadrature = c(9, "GH"), Adaptation = "EAP", Trust = "Fast")

For "SE" element, the options are "S-EM", "M-Step", "Xpd", and "Sandwich". See the IRTPRO manual for details.

scoring_method A string that can take one of the following values: "EAP" for Expected-a-Posteriori or "MAP" for Maximum-a-Posteriori.

scoring_args A list with named arguments that will specify the scoring. The program will automatically add "Score Persons". Following list elements can also be specified (last two elements are optional): list(Mean = 0, SD = 1, Minimum = <Minimum Score>, Maximum = <Maximum Score>)

misc_args A list with named arguments that will specify the miscellaneous arguments such as the number of decimals for the estimated parameters, the number of processors, etc. The following elements can be changed: list(Decimal = 4, Processors = 1, `Min Exp` = 1)

print_extra A string vector specifying additional results to be printed: 'StdRes' (Print table of standardized residuals) 'CTLD' (Compute Chen-Thissen LD and item fit statistics) 'M2' (Compute limited-information overall model fit statistics) 'GOF' (Print each item's goodness of fit frequency table) 'Loadings' (Print factor loadings) 'P-Nums' (Print parameter numbers) 'Diagnostic' (Print diagnostic information)

constraints A vector of sting commands for constraints section of the syntax. It is usually used to constrain a parameter to a certain value. Usually it has the following format: "Equal = (G1, Item Name, Parameter), (G2, Item Name, Parameter);" Here is an example: c("Equal = (G1, Item_1, Slope[0]), (G2, Item_1, Slope[0]);", "Equal = (G1, Item_1, Intercept[0]), (G2, Item_1, Intercept[0]);", "Equal = (G1, Item_2, Slope[0]), (G2, Item_2, Slope[0]);", "Equal = (G1, Item_2, Intercept[0]), (G2, Item_2, Intercept[0]);") or: c("(Item_1, Slope[0]) = 1.3;", "(Item_1, Intercept[0]) = 2.1;", "(Item_2, Slope[0]) = 0.7;", "(Item_2, Intercept[0]) = -1.2;")

priors
: A list that specifies the prior parameters. There are three possible options.

  The value can be NULL where no prior information will be used.

  The value can be a data frame with the following format: Column names: item_id, parameter, prior_dist, prior_par_1, prior_par_2. item_id column should match item IDs. parameter should be following one of the "Slope[0]", "Intercept[0]", or "Guessing[0]". prior_par column should be one of the following values: "Lognormal", "Normal", "Beta". prior_par_1 and prior_par_2 should be numeric values for the prior parameters. For "Normal" or "Lognormal", prior_par_1 can be 0 (mean) and prior_par_2 can be 1 (standard deviation. For "Beta", prior_par_1 can be 4 (mean) and prior_par_2 can be 16.

  The value can be a data frame with the following format if all items for a model should follow the same priors: Column names:model, parameter, prior_dist, prior_par_1, prior_par_2. The model column should match the model argument of the function. See the model argument's description to see the available options.

overwrite
: If TRUE and there are already an IRTPRO 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.

irtpro_exe_dir
: The location of the "ASCII2SSIG64.exe" and "IRTPROx64.exe". The default location is file.path("C:/Program Files/IRTPRO 6.0").

Author(s)

Emre Gonulates

Examples

```
## Not run:
resp <- sim_resp(generate_ip(n = 15), rnorm(200), prop_missing = .2)
irtpro_calib <- est_irtpro(x = resp, model = "3PL",
                           target_dir = file.path("C:/temp/irtpro1"),
                           overwrite = TRUE)

n_examinee <- 500
resp <- sim_resp(generate_ip(model = sample(c("3PL", "GPCM2"), 20, T)),
                 rnorm(n_examinee), prop_missing = .2)
resp <- cbind.data.frame(examinee_id = paste0("Ex", 1:n_examinee),
                         group = sample(c("A", "B"), n_examinee, TRUE),
                         resp)
irtpro_calib_mixed <- est_irtpro(
  x = resp,
  items = NULL,
  examinee_id_var = "examinee_id",
  group_var = "group",
  target_dir = file.path("C:/temp/irtpro2"),
  overwrite = TRUE)
```

```
## End(Not run)
```

`est_winsteps` *Estimate Rasch Model using Winsteps*

Description

This function serves as an interface to the Winsteps program, allowing for the convenient execution of basic Winsteps calibrations without the need to write Winsteps syntax manually. Please note that a valid installation of Winsteps is necessary for this function to operate. Keep in mind that it is still in beta mode, so exercise caution when using it.

Usage

```
est_winsteps(
  x,
  target_dir = getwd(),
  analysis_name = "winsteps_analysis",
  items = NULL,
  examinee_id_var = NULL,
  additional_vars = NULL,
  anchor_info = NULL,
  overwrite = TRUE,
  winsteps_exe_folder = file.path("C:/Winsteps"),
  verbose = TRUE
)
```

Arguments

`x`	A matrix or data frame that contains both response and person data.
`target_dir`	The directory where the analysis results will be saved. The default value is `getwd()`.
`analysis_name`	A string that will be used for naming the files (data, control, output) and as the title of the analysis. The default is `"winsteps_analysis"`.
`items`	A vector of strings representing item IDs within `x` to be used as response data, or a numeric vector indicating the columns containing response data. The default value is `NULL`, which uses all columns in `x` except those specified in `examinee_id_var` and `additional_vars`.
`examinee_id_var`	A string representing the column name containing examinee/subject IDs, or the column number of examinee/subject IDs. The default value is `NULL`, assuming no examinee/subject IDs.

additional_vars
: A vector of strings or integers representing the column names or numbers to be included in the Winsteps data file. The default value is NULL, meaning no additional columns will be added to the Winsteps data file. Note that if items is NULL, all variables in the dataset will be treated as response data.

anchor_info
: A matrix or data frame containing the sequence number and difficulty values of anchor items. The anchor matrix should have at least two columns: (1) either seq, indicating the column numbers of the anchor items in the response matrix, or item_id column containing the IDs of anchor items, and (2) b, the item difficulty values. The default value is NULL, meaning no anchor items are used.

overwrite
: A logical value. If TRUE, existing control/data files will be overwritten. The default value is TRUE.

winsteps_exe_folder
: The directory containing the Winsteps executable. The default value is file.path("C:/Winsteps").

verbose
: If TRUE, the program will print intermediate steps.

Author(s)

Emre Gonulates

Examples

```
## Not run:
true_theta <- rnorm(300)
ip <- generate_ip(n = 20, model = "Rasch")
resp_set <- generate_resp_set(ip = ip, theta = true_theta, prop_missing = .2)
resp_matrix <- as.matrix(resp_set)

est_pars <- est_winsteps(x = resp_matrix,
                         target_dir = "c:/temp/est_winsteps")

# Relationship between true and estimated item difficulty parameters
plot(x = ip$b, y = est_pars$ip$b, xlab = "True 'b'", ylab = "Estimated 'b'")
cor(x = ip$b, y = est_pars$ip$b)

# Relationship between true and estimated theta parameters
cor(x = true_theta, y = est_pars$raw_person_pars$MEASURE)
plot(x = true_theta, y = est_pars$raw_person_pars$MEASURE,
     xlab = "True 'b'", ylab = "Estimated 'b'")

## 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 = 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	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") # Reparameterized 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$ids), function(i)
    generate_testlet(testlet_id = temp_list$ids[i],
                     n = temp_list$item_models[i],
                     item_id_preamble = paste0("t", i, "-"))))))
```

`generate_item` *Generate a random Item object*

Description

Generate a random Item object

Usage

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

Arguments

`model`	The model of the Item object.
`n_categories`	For polytomous models, the number of categories for an 'item' object.
`se`	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 `item()` 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 = TRUE)
```

---

generate_resp *Generate random item responses (Response object)*

Description

generate_resp Generate dichotomous (0 or 1) or polytomous responses for given ability and item parameter(s). This function returns a Response-class object.

Usage

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

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.

Value

Returns a list of Response-class objects with equal length to the length of theta.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(model = "3PL", n = 15)
generate_resp(ip, theta = rnorm(1))

# A list of Response objects
generate_resp(ip, theta = rnorm(5))

# Set the proportion of missing responses:
generate_resp(ip, theta = rnorm(5), prop_missing = 0.3)
```

---

generate_resp_set *Generate a random item responses (Response_set object)*

Description

`generate_resp_set` Generate dichotomous (0 or 1) or polytomous responses for given ability and item parameter(s). This function returns a `Response_set-class` object,

Usage

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

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`.

Value

Returns a `Response_set-class` object.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(model = "3PL", n = 15)
generate_resp_set(ip, theta = rnorm(5))

# Set the proportion of missing responses:
generate_resp_set(ip, theta = rnorm(7), prop_missing = 0.3)
```

`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(testlet_ = "my-testlet")
# Designate the ID of testlet and preamble for item ids
generate_testlet(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") # Reparameterized 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(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$item_id), function(i)
#   generate_testlet(item_id = temp_list$item_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 returns a `Response_set` object that contains the administered items of each simulee or a matrix or responses.

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,
  output_type = c("Response_set", "matrix"),
  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.
output_type	A string that specifies the output type. Available options are "Response_set" which returns a Response_set object and "matrix" which returns a matrix. If attach_summary = TRUE and output_type = "matrix", a data frame will be returned instead of a matrix. The default value is "Response_set".
remove_na	If TRUE, the columns that are all NA will be removed.
attach_summary	If TRUE and output_type = "matrix", the summary of each CAT will be attached to the beginning of the response string as columns. The default value is FALSE. When output_type = "Response_set", CAT summary will automatically added to each Response object of the output within misc field.

Value

Depending on the output_type, the function returns the 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.

Author(s)

Emre Gonulates

See Also

cat_sim

Examples

```
n <- 40 # number of items
ip <- generate_ip(n = n)
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)
resp_set <- get_cat_response_data(cat_sim_output = cat_data, cd)
resp_set

# Get the examinee_id of third simulee:
resp_set[[3]]$examinee_id
# Extract the true theta of the third examinee:
resp_set[[3]]$true_ability
# Extract the final estimated theta of the third examinee:
resp_set[[3]]$est_ability
# Extract the final standard error of the third examinee:
resp_set[[3]]$se


# Alternatively, output can be a matrix:
```

```
resp_matrix <- get_cat_response_data(cat_sim_output = cat_data,
                                     output_type = "matrix")
resp_matrix

# If cat design provided, the matrix columns will be sorted as the
# item pool used for the simulation:
resp_matrix <- get_cat_response_data(cat_sim_output = cat_data, cd = cd,
                                     output_type = "matrix")
resp_matrix

# Additionally, remove the colums which has all NA values:
resp_matrix <- get_cat_response_data(cat_sim_output = cat_data, cd = cd,
                                     remove_na = TRUE,
                                     output_type = "matrix")
resp_matrix
```

`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)
```

GPCM-class *Generalized Partial Credit Model*

Description

Generalized Partial Credit Model

Slots

- `a` Item discrimination parameter
- `b` A vector of threshold parameters
- `D` Scaling constant
- `se_a` Standard error of item discrimination parameter
- `se_b` A vector of standard error of item threshold parameters

Author(s)

Emre Gonulates

GPCM2-class *Reparameterized Generalized Partial Credit Model*

Description

Reparameterized Generalized Partial Credit Model

Slots

- `a` Item discrimination parameter
- `b` Overall location parameter
- `d` A vector of threshold parameters
- `D` Scaling constant
- `se_a` Standard error of item discrimination parameter
- `se_b` Standard error of overall location parameter
- `se_d` A vector of standard error of item threshold parameters

Author(s)

Emre Gonulates

GRM-class *Graded Response Model*

Description

Graded Response Model

Slots

`a` Item discrimination parameter

`b` A vector of threshold parameters

`D` Scaling constant

`se_a` Standard error of item discrimination parameter

`se_b` A vector of standard error of item threshold parameters

Author(s)

Emre Gonulates

---

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

## S4 method for signature '1PL'
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)

## S4 method for signature '2PL'
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)
```

```
## S4 method for signature '3PL'
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)

## S4 method for signature '4PL'
info(ip, theta, tif = FALSE, observed = FALSE, resp = NULL)

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

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

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

## S4 method for signature 'GPCM2'
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 = "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_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 = "PCM"), 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 = "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 identifies items that have become unstable, meaning their item parameter values have shifted, within two specified sets of items.

Usage

```
ipd(
  ip1,
  ip2,
  method = "robust-z",
  anchor_item_ids = NULL,
  alpha = 0.01,
  iqr_type = 7,
  theta = seq(-4, 4, 0.1),
  weights = stats::dnorm(seq(-4, 4, 0.1))
)
```

Arguments

`ip1`	An Itempool object for the first calibration.
`ip2`	An Itempool object for the second calibration.
`method`	The method for analyzing item parameter drift.

"robust-z" Robust-Z method based on the Huynh and Meyer (2010).

"d2" D2 method assesses item parameter drift using the method outlined in Wells et al. (2014). It involves comparing the Item Characteristic Curves (ICCs) of item parameters from two different item pools. This is also referred to as WRMSD (Weighted Root Mean Squared Difference).
There are no strict thresholds for determining the significance of D2 in identifying item drift. A comprehensive approach considering other measures is recommended. Nevertheless, as a general guideline, for dichotomous items, a D2 value greater than 0.1 may warrant further scrutiny. For polytomous items with two thresholds (or three score categories), a D2 value exceeding 0.15, for those with three thresholds (or four score categories), a D2 value greater than 0.225, for those with four thresholds (or five score categories), a D2 value larger than 0.3, and for items with five thresholds (or six score categories), a D2 value larger than 0.375 may be indicative of item drift and should be investigated further.

`anchor_item_ids`
: A character vector containing the IDs of anchor items. If set to `NULL`, it is assumed that all items are considered anchor items.

`alpha`
: A numeric value ranging from 0 to 1. Only needed when `method = "robust-z"`. The two-tailed critical value is employed to identify unstable items. For instance, if we calculate the critical value using `qnorm(1-alpha/2)` (which equals 1.96 when $alpha = 0.05$), items with absolute robust-z values exceeding this threshold will be marked as unstable.

`iqr_type`
: An integer indicating the choice of quantile algorithm. Refer to the `?quantile` function's `type` argument for more details. For instance, SAS's default quantile algorithm, `QNTLDEF=5`, corresponds to `iqr_type = 2` in R. The default value is `iqr_type = 7`.

`theta`
: A numeric vector containing the quadrature points. Only needed when `method = "d2"`.

`weights`
: A numeric vector containing the weights assigned to the quadrature points. The length of this vector should match the length of the `theta` argument. Only needed when `method = "d2"`

Value

Return a list depending on the method:

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

- `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's discrimination parameter.
- `output$a$unstable` Item IDs that were flagged when the robust-z statistic value for a parameters exceeded the absolute value of the critical value (i.e., `qnorm(1-alpha/2)`).
- `output$b$robust_z` Robust-z statistic values for each item's difficulty or threshold parameter. If an item has multiple threshold parameters, robust z statistics will be calculated for each one.
- `output$b$unstable` Item IDs that were flagged if the robust-z statistic for difficulty/threshold parameters exceeded the absolute value of the critical value (i.e., `qnorm(1-alpha/2)`).

d2 A numeric vector containing the differences between the ICCs of each item.

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:10.7275/ycx6-e864>

Examples

```
##### Robust-z #####
# 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)

##### D2 #####
ip1 <- generate_ip(n = 20)
ip2 <- ip1
# add a small nuisance to item difficulty parameters
ip2$b <- ip1$b + runif(20, -.5, .5)

theta <- seq(-4, 4, 0.2)
weights <- dnorm(theta)
ipd(ip1, ip2, theta = theta, weights = weights)
```

```
# Calculate for only certain items
ipd(ip1, ip2, theta = theta, weights = weights,
  anchor_item_ids = c("Item_2", "Item_6", "Item_9", "Item_13"))

### Polytomous items items
n_item <- 30
models <- sample(c("3PL", "GPCM2"), n_item, TRUE)
new_ip <- generate_ip(model = models, D = 1.702)
old_ip <- data.frame(new_ip)
old_ip$a <- old_ip$a + round(runif(n_item, min = -.5, max = .5), 2)
old_ip$b <- old_ip$b + round(runif(n_item, min = -.75, max = .75), 2)
old_ip$d1 <- old_ip$d1 + round(runif(n_item, min = -.75, max = .75), 2)
old_ip$d2 <- old_ip$d2 + round(runif(n_item, min = -.75, max = .75), 2)
old_ip$d3 <- old_ip$d3 + round(runif(n_item, min = -.75, max = .75), 2)
old_ip <- itempool(old_ip)

ipd(ip1 = old_ip, ip2 = new_ip, theta = theta, weights = weights)
```

`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

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,
  item_id = NULL,
  parameters = NULL,
  se = 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.
`item_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`	A list object containing 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 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 item_id
item(a = 1.2, b = -0.94, item_id = "My-Item-1")
# Add content
item(a = 1.2, b = -0.94, item_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, item_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$parameters
i1$item_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 (Reparameterized 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, item_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`.

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.

Slots

item_id Item ID. Default value is NULL.

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).

Author(s)

Emre Gonulates

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), item_id = c("i1", "i2"))
# Set content
itempool(a = c(1, 1.4), b = c(-2, 1), content = c("Algebra", "Geometry"))

# Create 3PL items from a data frame:
ipdf <- data.frame(a = c(.9, .79, 1.26),
                   b = c(-1, .43, -2.3),
                   c = c(.2, .38, .25))
itempool(ipdf)

# Create GRM (Graded Response Model) items from a data frame
ipdf <- data.frame(a = rlnorm(10, 0, .3), b1 = rnorm(10), b2 = rnorm(10))
itempool(ipdf, 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), item_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,
  ip = NULL,
  stats = c("n", "pval", "pbis", "bis", "pbis_adj", "bis_adj"),
  suppress_output = FALSE
)
```

Arguments

resp	A Response_set-class object, 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.
ip	An Itempool-class object. This will help function in two ways. First, if the resp is a Response_set-class object, the function will help the responses to be arranged in the same order as ip. Second, if there are polytomous items in the data, ip will help finding the maximum values of each item. Otherwise, the maximum values each item can take will be calculated using data, which may be fallible.

`stats` A vector of string containing the columns/statistics to be calculated. `'item_id'` column will be added by default. Some or all of the following columns can be added to the output: `c("n", "pval", "pbis", "bis", "pbis_adj", "bis_adj")`. Please see the 'value' section below to see the details of these columns. By default, all of the columns above will be calculated.

`suppress_output`
If `TRUE`, the function will suppress console output. Default value is `FALSE`

Value

A data.frame with following columns:

'item_id' Item ID.

'n' Number of examinees responded this item.

'pval' p-value, proportion of examinees correctly answered items. If there are polytomous items in the data, p-value will be calculated by dividing the mean of the scores for the item by the maximum possible score of the item.

'pval_unadj' Unadjusted p-value, this is the mean of item scores that is not adjusted for the maximum possible score as `'pval'` column does. For dichotomous items, this will be the same as `'pval'` column.

'pbis' Point biserial correlation.

'bis' Biserial correlation.

'pbis_adj' Point biserial correlation between item and total score without this item. Note that this stat is only available when criterion is `NULL`.

'bis_adj' Biserial correlation between item and total score without this item. Note that this stat is only available when criterion is `NULL`.

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 = NULL, type = "Q1", item_id = NULL, n_groups = NULL)
```

Arguments

ip	An Itempool-class object.
resp	A Response_set-class object, matrix or data.frame containing the item responses.
theta	An vector containing ability parameters. When type = "Q1" and theta = NULL or an invalid theta vector provided, theta values will be estimated using item parameters and responses. In order to speed up the function for large data sets, theta values can be supplied.
type	The type of the item-fit index. Currently the following indices are available: **"Q3"** Yen's Q3 index (Yen, 1984) **"Q1"** Yen's Q1 index (Yen, 1981). Only available for unidimensional dichotomous items. **"G2"** PARSCALE's fit statistic. See DeMars (2005) for details. The default value is "Q1".
item_id	A string vector that is holding the ID's of the item for which item fit should be calculated. The default value is NULL where item fit statistic of all items will be calculated.
n_groups	An integer representing the number of groups of examinees. When type = "Q1" and n_groups = NULL, the default value will be 10 (as specified in Yen (1981)). For example, if there are 900 examinees, when n_groups = 10, first examinees will be sorted according to their theta scores and separated into 10 equally sized groups of approximately 90 examinees each. The same default value is used when type = "G2".

Details

Yen's Q3

The details of Yen's Q3 can be found in Yen (1984). It is mainly used as a measure of local dependence between two set of items.

Yen's Q1

The details of Yen's Q1 can be found in Yen (1981). Please note that Q1 can have inflated Type-I error rates (Orlando & Thissen, 2000).

PARSCALE's G2

PARSCALE's fit statistic G2 is explained in Kang and Chen (2008) and DeMars (2005) in detail. DeMars also detailed the situations when G2 index yields inflated Type-I error rates. Specifically, she did not recommend this index for short tests.

Value

A vector of item-fit index values for `Q1` and `G2`. A correlation matrix will be returned for `Q3`.

Author(s)

Emre Gonulates

References

DeMars, C. E. (2005). Type I error rates for PARSCALE's fit index. Educational and psychological measurement, 65(1), 42-50.

Kang, T., & Chen, T. T. (2008). Performance of the generalized S-X2 item fit index for polytomous IRT models. *Journal of Educational Measurement*, 45(4), 391–406. <doi:10.1111/j.1745-3984.2008.00071.x>

Orlando, M., & Thissen, D. (2000). New item fit indices for dichotomous item response theory models. Applied Psychological Measurement, 24, 50–64.

Yen, W. M. (1981). Using simulation results to choose a latent trait model. *Applied Psychological Measurement*, 5(2), 245–262. <doi:10.1177/014662168100500212>

Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three-parameter logistic model. *Applied Psychological Measurement*, 8(2), 125–145.

Examples

```
ip <- generate_ip(model = "3PL", n = 10)
theta <- rnorm(1000)
resp <- sim_resp(ip = ip, theta = theta, output = "response_set")

### Yen's Q1 ###
# Calculate Yen's Q1 for all items
item_fit(ip = ip, resp = resp, theta = theta, type = "Q1")

# Calculate Yen's Q1 for only selected items
item_fit(ip = ip, resp = resp, theta = theta, type = "Q1",
         item_id = c("Item_3", "Item_5"))

# Change the number of groups examinees will be separated into:
item_fit(ip = ip, resp = resp, theta = theta, type = "Q1", n_groups = 15)
```

kappa_coef *Calculate Cohen's Kappa Coefficient*

Description

This function calculates weighted or unweighted Kappa coefficient for two sets of ratings. Kappa coefficient quantifies the agreement between two sets of ratings (like two raters) beyond what is expected by chance. It can be used as a measure of inter-rater reliability.

If the ratings are ordinal (for example Likert scale), weighted kappa coefficient can be used. Weighted Kappa penalizes the larger discrepancies between raters. More emphasis is put to large differences between rating and small emphasis will be put on smaller differences. The available weighting options are `"linear"` and `"quadratic"`. By default the function calculates `"unweighted"` Kappa coefficient.

Usage

```
kappa_coef(x, weights = "unweighted")
```

Arguments

`x`	A matrix/data.frame with two columns where each column contains a set of ratings. When `weights = "linear"` or `weights = "quadratic"`, each row should be an ordered factor or numbers. The rows with missing values (i.e. `NA`) will be removed from the analysis.
`weights`	Either a string representing the weighting method. Or, a square matrix of weights that will be applied to the cross table (assuming the ratings are ordered factors or numeric). There are three possible weighting methods (aside from the custom weights method): `'unweighted'` This is the original Kappa coefficient where no weighting applied. This is the default method. This method is appropriate for both nominal (i.e. unordered) data or ordinal (i.e. ordered) data. `'linear'` Linear weights applied. `'quadratic'` Quadratic weights applied.

Value

A Kappa coefficient which is a number between -1 and 1. 1 means perfect agreement between ratings. 0 means agreement between rating is no better than agreement one would get merely by chance. Negative values means the agreement is even worse than one would get by chance.

Author(s)

Emre Gonulates

References

Cohen, Jacob (1960). "A coefficient of agreement for nominal scales". Educational and Psychological Measurement. 20 (1): 37–46.

Sim, J., & Wright, C. C. (2005). The kappa statistic in reliability studies: use, interpretation, and sample size requirements. Physical therapy, 85(3), 257-268.

Examples

```
#########  Example 1  #########
# Hypothetical data from Sim and Wright (2005), Table 1
# "Diagnostic Assessments of Relevance of Lateral Shift by 2 Clinicians"
dtf <- data.frame(c1 = c(rep("Relevant", 22), rep("Relevant", 2),
                         rep("Not Relevant", 4), rep("Not Relevant", 11)),
                  c2 = c(rep("Relevant", 22), rep("Not Relevant", 2),
                         rep("Relevant", 4), rep("Not Relevant", 11)))
kappa_coef(dtf)

#########  Example 2  #########
# Hypothetical data from Sim and Wright (2005), p.260, Table 2
pain_raw <- data.frame(t1 = c(rep("No Pain", 15 + 3 + 1 + 1),
                              rep("Mild Pain", 4 + 18 + 3 + 2),
                              rep("Moderate Pain", 4 + 5 + 16 + 4),
                              rep("Severe Pain", 1 + 2 + 4 + 17)),
                       t2 = c(rep("No Pain", 15), rep("Mild Pain", 3),
                              rep("Moderate Pain", 1), rep("Severe Pain", 1),
                              rep("No Pain", 4), rep("Mild Pain", 18),
                              rep("Moderate Pain", 3), rep("Severe Pain", 2),
                              rep("No Pain", 4), rep("Mild Pain", 5),
                              rep("Moderate Pain", 16), rep("Severe Pain", 4),
                              rep("No Pain", 1), rep("Mild Pain", 2),
                              rep("Moderate Pain", 4), rep("Severe Pain", 17))
  )
# Since data is ordinal, convert columns to ordinal factors:
ordered_levels <- c("No Pain", "Mild Pain", "Moderate Pain", "Severe Pain")
pain_ordered <- data.frame(
  t1 = factor(pain_raw$t1, levels = ordered_levels, ordered = TRUE),
  t2 = factor(pain_raw$t2, levels = ordered_levels, ordered = TRUE))
table(pain_ordered)

# Unweighted Kappa Coefficient
kappa_coef(pain_ordered)
# Kappa Coefficient with linear weights
kappa_coef(pain_ordered, weights = "linear")
# Kappa Coefficient with quadratic weights
kappa_coef(pain_ordered, weights = "quadratic")
```

---

ks *Item Characteristic Curve Estimation using Kernel Smoothing*

Description

Item Characteristic Curve Estimation using Kernel Smoothing

Usage

```
ks(
  resp,
  h = NULL,
  kernel_func = "gauss",
  criterion = NULL,
  points = seq(-3, 3, 0.05)
)
```

Arguments

resp	A response matrix where each row is the responses of an examinee and each column represents an item. resp does not necessarily be a matrix. It can be data.frame or any other object that can be convertible to matrix using as.matrix function. resp can contain missing responses.
h	The bandwidth parameter that controls the amount of smoothing. A small value will decrease the bias whereas increase the sampling variability. For a standard normally distributed criterion and Gaussian kernel smoothing function, $h = 0.2$ is recommended for large sample sizes (like 3000), $h = 0.3$ is recommended for medium sample sizes (like 500), and $h = 0.4$ is recommended for small sample sizes (like 100), and The default value is $1.06\sigma(criterion)n_{examinees}^{-1/5}$.
kernel_func	Choice of kernel function. Possible choices are: "gauss" Gaussian kernel. $f(x) = e^{-u^2/2}$. "unif" Uniform kernel. $f(x) = 0.5, \|u\| < 0.5$, else 0. "quadratic" Quadratic kernel. $f(x) = 0.75(1 - u^2), \|u\| < 1$, else 0. **Custom Function** You can provide a custom kernel function object. The function should be maximum at $u = 0$ and gets closer to 0 on either side. The default value is "gauss", i.e. Gaussian kernel function.
criterion	The ability estimates for each examinee. The default is NULL where the abilities will be estimated from the sum scores. First sum scores will be calculated, then the rank of each examinee's sum score will be calculated. These ranks will be divided by the number of examinees plus 1 in order to get values between 0 and 1. Finally, these values will be put on standard normal scale (by inverse CDF).
points	The points at which the item characteristic curve will be calculated. The default value is points = seq(-3, 3, 0.05).

Value

A list with following elements will be returned:

`points` The quadrature points at which ICC is calculated.

`icc` A matrix where each cell represents probability of selecting a response (for dichotomous models, probability of correct response). Items are on columns and quadrature points are on rows.

`se` A matrix of standard errors of each point of `icc`. This matrix has the same dimension as `icc`.

`criterion` The criterion values used for examinees. If `criterion = NULL` these numbers will be based on sum scores.

`h` The bandwidth parameter.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(model = "3PL", n = 50)
true_theta <- rnorm(10000)
resp <- sim_resp(ip = ip, theta = true_theta, prop_missing = 0.3)

kern_output <- ks(resp)

# Plot ICC
i <- 12 # select an item to plot
x <- kern_output$icc[, i]
se <- kern_output$se[, i]
p <- prob(ip = ip[i], theta = kern_output$points)
p <- sapply(p, `[`, 2) # get the probability of correct responses

graph_data <- data.frame(
  theta = kern_output$points,
  icc = x,
  ci_low = sapply(x - qnorm(.975) * se, function(x) max(x, 0)),
  ci_high = sapply(x + qnorm(.975) * se, function(x) min(x, 1)),
  p = p)

## Not run:
p <- ggplot(data = graph_data) +
  geom_line(aes(x = theta, y = icc), color = "blue", alpha = .7, size = 1) +
  geom_line(aes(x = theta, y = p), color = "red", size = 1, alpha = .7) +
  geom_ribbon(data = graph_data,
              aes(x = theta, ymin = ci_low, ymax = ci_high),
              alpha = .25) +
  ylim(0, 1) +
  labs(x = "Theta", y = "Probability",
       title = "Item Characteristic Curve") +
  theme_bw()

p

## End(Not run)
```

length,Itempool-method

Find the length of an Itempool-class *object*

Description

Find the length of an Itempool-class object

Find the length of an Response-class object

Find the length of a Response_set-class object

Find the length of a Testlet-class object

Usage

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

## S4 method for signature 'Response'
length(x)

## S4 method for signature 'Response_set'
length(x)

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

Arguments

x	an Response_set-class object

Author(s)

Emre Gonulates

Examples

```
r <- response(sample(0:1, 22, TRUE))
length(r)
```

M2PL-class *Multidimensional Two-Parameter Logistic Model*

Description

Multidimensional Two-Parameter Logistic Model

Slots

- `a` Slope Parameters
- `d` Intercept Parameter
- `D` Scaling constant
- `se_a` Standard errors of slope parameters
- `se_d` Standard error of intercept parameter

Author(s)

Emre Gonulates

M3PL-class *Multidimensional Three-Parameter Logistic Model*

Description

Multidimensional Three-Parameter Logistic Model

Slots

- `a` Slope Parameters
- `d` Intercept Parameter
- `c` Pseudo-Guessing Parameter
- `D` Scaling constant
- `se_a` Standard errors of slope parameters
- `se_d` Standard error of intercept parameter
- `se_c` Standard error of pseudo-guessing parameter

Author(s)

Emre Gonulates

max_score *Calculate the maximum possible score*

Description

Calculate the maximum possible score

Usage

```
max_score(ip, resp = NULL, sum = TRUE)

## S4 method for signature 'Item'
max_score(ip, resp = NULL, sum = TRUE)

## S4 method for signature 'Itempool'
max_score(ip, resp = NULL, sum = TRUE)
```

Arguments

ip	An Item-class or an Itempool-class object containing the item parameters.
resp	A Response-class or Response_set-class object.
sum	If TRUE, when ip is an Itempool-class object the individual maximum possible scores of items will be summed. This argument will be ignored when resp is not NULL.

Value

Maximum possible score of each item

Author(s)

Emre Gonulates

mean,Item-method *Calculate the expected value of an Item*

Description

mean Returns the expected value of an item for given parameters for a given ability or abilities, i.e. θ.

Usage

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

## S4 method for signature 'Rasch'
mean(x, ...)

## S4 method for signature '1PL'
mean(x, ...)

## S4 method for signature '2PL'
mean(x, ...)

## S4 method for signature '3PL'
mean(x, ...)

## S4 method for signature '4PL'
mean(x, ...)

## S4 method for signature 'GPCM'
mean(x, ...)

## S4 method for signature 'GPCM2'
mean(x, ...)

## S4 method for signature 'GRM'
mean(x, ...)

## S4 method for signature 'PCM'
mean(x, ...)
```

Arguments

x	An Item-class object containing the item parameters.
...	Additional parameters. Specifically theta argument is required. theta should be a numeric vector of ability parameters.

Value

Item expected values at given theta(s) values will be returned.

Author(s)

Emre Gonulates

Examples

```
itm <- generate_item(model = "Rasch")
mean(itm, theta = 1)
```

```
mean(itm, -1.2)

itm <- generate_item(model = "GPCM", n_categories = 5)
mean(itm, theta = 1.5)
mean(itm, 0.2)
```

`mean,Itempool-method` *Calculate the expected value of an Itempool*

Description

`mean` Returns the expected values of each item in an `Itempool-class` object for a given ability or abilities, i.e. θ.

Usage

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

Arguments

`x`	An `Itempool-class` object containing the item parameters.
`...`	Additional parameters. Specifically `theta` argument is required. `theta` should be a numeric vector of ability parameters.

Value

Item expected values at given theta values will be returned.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(model = "2PL")
mean(ip, theta = 1.2)
mean(ip, 1.2)

ip <- generate_ip(model = "GPCM")
mean(ip, theta = -0.37)
mean(ip, -1.55)
```

mean,Testlet-method *Calculate the expected value of an Testlet*

Description

mean Returns the expected values of each item in an Testlet-class object for a given ability or abilities, i.e. θ.

Usage

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

Arguments

x	A Testlet-class object containing the item parameters.
...	Additional parameters. Specifically theta argument is required. theta should be a numeric vector of ability parameters.

Value

Item expected values at given theta values will be returned.

Author(s)

Emre Gonulates

Examples

```
t1 <- generate_testlet()
mean(t1, theta = -1.1)
mean(t1, -1.1)
```

PCM-class *Partial Credit Model*

Description

Partial Credit Model

Slots

b A vector of threshold parameters

se_b A vector of standard error of item threshold parameters

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

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

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

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

Arguments

`resp`	A vector of item responses.
`ip`	An `Item-class`, `Itempool-class` or a `Testlet-class` object.
`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.cat_output *Plot progress of a CAT algorithm for one examinee*

Description

plot.cat_output Plots the progress of CAT for one examinee.

Usage

```
## S3 method for class 'cat_output'
plot(
  x,
  ...,
  plot_b = TRUE,
  se_band = TRUE,
  horizontal_line = "true_theta",
  title = "CAT Progress",
  suppress_plot = FALSE,
  base_r_graph = FALSE
)
```

Arguments

x	A "cat_output" object that is output of cat_sim function for one examinee.
...	Additional arguments.
plot_b	If TRUE, 'b' parameters of the administered items will be plotted along with intermediate theta estimates. The default value is TRUE.
se_band	A logical value. If TRUE, a standard error band is added around the estimated theta values. At each stage one standard error of that stage is added to and subtracted from the ability estimate at that stage. The default value is TRUE.
horizontal_line	An option to add a horizontal line. Provide either one of these or a list of a combination of these (except NULL). **"true_theta"** Add a horizontal line for true theta. Default option. **"final_theta"** Add a horizontal line at final theta (ability) estimate **NULL** No horizontal line added.
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.
base_r_graph	Currently this function only works with package 'ggplot2'. If TRUE function will plot graphs using base R graphics. If FALSE the function will check whether 'ggplot2' package is installed. If it is installed, it will use 'ggplot2' package for the plot. The default value is FALSE.

Value

Depending on the value of `printPlot` function either prints the CAT progress plot or returns the plot object.

Author(s)

Emre Gonulates

Examples

```
cd <- create_cat_design(ip = generate_ip(n = 100))
co <- cat_sim(true_ability = rnorm(1), cd = cd)

plot(co)

# Suppress item difficulties
plot(co, plot_b = FALSE)

# Suppress Standard Error Band
plot(co, se_band = FALSE)

# Add final theta estimate line
plot(co, horizontal_line = "final_theta")
plot(co, horizontal_line = "true_theta")

# Change Title
plot(co, title = "CAT Progress for Examinee ABC")

## Not run:
# Change Text Size
plot(co) + theme(text=element_text(size=20))

# Change x-axis label
plot(co) + xlab("My New X Axis Label")

# Change y limits of the graph
plot(co) + coord_cartesian(ylim = c(-5,5))

# Change legend position
plot(co) + theme(legend.position="none")
plot(co) + theme(legend.position="left")

# Add a horizontal line
plot(co) + geom_hline(yintercept = -1, color = "red", linetype = 5)

## End(Not run)
```

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

Description

plot.Item Plots the item characteristic curve for dichotomous items and category response functions for polytomous items.

Usage

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

Arguments

x	An Item-class object.
theta_range	Either (a) a numeric vector of length two where the values are minimum and maximum theta values, or, (b) a numeric vector of length more than two where values represents the theta values that will be plotted.
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 to NULL to suppress the title.
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE. Function cannot suppress plot when base_r_graph = TRUE, but graph still can be saved in a variable.
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.
base_r_graph	If TRUE function will plot graphs using base R graphics. If FALSE the function will check whether 'ggplot2' package is installed. If it is installed, it will use 'ggplot2' package for the plot. The default value is FALSE.
...	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, model = "1PL"))

itm1 <- item(a = 1.2, b = 0.3, c = .2, model = "3PL")
plot(itm1)
plot(item(a = 1.2, b = 0.3, c = .2, d = .89, D = 1))

# Use base R graphics for the plot
plot(itm1, base_r_graph = TRUE)

# Plot Graded Response Model
itm2 <- item(a = 0.902, b = c(-1.411, 0.385, 1.79), model = "GRM")
plot(itm2)
plot(itm2, category_names = c("Strongly Disagree", "Disagree", "Agree",
                              "Strongly Agree"))

plot(itm2, category_names = c("Strongly Disagree", "Disagree", "Agree",
                              "Strongly Agree"), base_r_graph = TRUE)

# A Graded Response Model item with two categories (i.e. 2PL item):
itm3 <- item(a = 0.8, b = 1, model = "GRM")
plot(itm3, category_names = c("Incorrect", "Correct"),
     legend_title = "Response")

## Not run:
# Change the y-axis label (Only available if 'ggplot2' is installed)
# plot(itm3, suppress_plot = TRUE) + ylab("New Label")

## End(Not run)
```

`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),
  type = "icc",
  tcc_prop_corr = FALSE,
  focus_item = NULL,
  title = "",
  suppress_plot = FALSE,
  legend_title = NULL,
  base_r_graph = FALSE,
  y_lim = NULL,
  ...
)
```

Arguments

x	An Itempool-class object.
theta_range	Either a numeric vector of length two setting the boundaries of x-axis, e.g. c(-4, 4), or, a numeric vector that is includes the theta values that will be plotted, e.g. seq(-3, 3, by = 0.1).
type	The type of the graph. The default value is "icc". Available options are: "icc" Plot item characteristic curve of each item "tcc" Plot test characteristic curve "hist" Plot histograms of item parameters "pars" Plot dot plot of item parameters
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).
focus_item	A character string of the 'item_id' of the item to be focused. If type = "pars", this item will be shown with a red dot to distinguish it from others.
title	Title of the plot. Set title = NULL to suppress the plot title. The default is "". If type = "tcc" and title = "", title will be 'Test Characteristic Curve'. If type = "icc" and title = "", title will be 'Item Characteristic Curve'. If type = "hist" or type = "pars" and title = "", title will be 'Parameter Values'.
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.
base_r_graph	If TRUE function will plot graphs using base R graphics. If FALSE the function will check whether 'ggplot2' package is installed. If it is installed, it will use 'ggplot2' package for the plot. The default value is FALSE.
y_lim	A numeric vector of length two representing the lower and upper bound of y-axis.
...	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

```
ip <- generate_ip(n = sample(10:15,1))
plot(ip)

# Additional arguments will passed to geom_line
plot(ip, size = .25, alpha = 0.3)

# Set the boundaries of the graph
plot(ip, theta_range = c(-2, 2))
# alternatively provide theta values
plot(ip, theta_range = seq(-6, 6, by = 0.25))


# Test Characteristic Curve
plot(ip, type = "tcc")

# Proportion correct for test characteristic curve
plot(ip, type = "tcc", tcc_prop_corr = TRUE)

# Plot histogram of item parameters
plot(ip, type = "hist")


## Not run:
# Item parameter summary
ip <- generate_ip(n = 200)
plot(ip, type = "pars")
plot(ip, type = "pars", dotsize = .75)
plot(ip, type = "pars", focus_item = "Item_22")
# Use base R graphics
plot(ip, type = "pars", base_r_graph = TRUE)

# # Remove the legend altogether
# plot(ip, suppress_plot = TRUE) + ggplot2::theme(legend.position="none")
# # Change the labels:
# plot(ip, suppress_plot = TRUE) +
#   ylab("Probability") + xlab("Ability Score")

## End(Not run)
```

plot.ks_output *Plot Item Fit using Kernel-Smoothing*

Description

Plot Item Fit using Kernel-Smoothing

Usage

```
## S3 method for class 'ks_output'
plot(
  x,
  item_no,
  ip = NULL,
  title = "",
  ci = 0.95,
  base_r_graph = FALSE,
  suppress_plot = FALSE,
  ...
)
```

Arguments

x	The output of ks() function. If this will be provided the function will run much faster.
item_no	The order (i.e. column number) of the item to be plotted.
ip	An Itempool-class or Item-class object if expected probabilities are plotted.
title	Title of the plot. If the value is NULL, the plot title will be suppressed.
ci	It is either a number indicating the confidence interval that will be plotted around the item fit line or NULL if no confidence interval should be plotted. The default value is 0.95, i.e. 95 interval will be plotted.
base_r_graph	If TRUE function will plot graphs using base R graphics. If FALSE the function will check whether 'ggplot2' package is installed. If it is installed, it will use 'ggplot2' package for the plot. The default value is FALSE.
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
...	further arguments.

Author(s)

Emre Gonulates

Examples

```
# Generate responses
ip <- generate_ip()
resp <- sim_resp(ip = ip, theta = rnorm(500), prop_missing = .2)
# Run kernel smoothing
ks_data <- ks(resp)
# Plot first item
plot(ks_data, item_no = 1)
# Plot second item with expected probability value
plot(ks_data, item_no = 2, ip = ip)

plot(ks_data, item = 2, ip = ip[[2]])
```

---

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 = NULL,
  ip = NULL,
  criterion = NULL,
  bins = 10,
  x_axis_scale = NULL,
  add_icc = FALSE,
  title = "",
  n_dodge = 1,
  x_lim = NULL,
  base_r_graph = FALSE,
  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. If key = NULL, the function will check whether the item pool has keys by checking ip$key and raise an error if ip$key is not valid.

ip	An Itempool-class object that is needed for some plots. If ip provided and criterion 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. Also, the key for items can be provided via ip$key.
criterion	A vector of examinee abilities. If criterion values provided the bins are formed using them instead of sum scores.
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.
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 criterion or ip arguments provided the scale defaults to "criterion". "percent" Percent interval. "number" Numbers between 1 and bins. "criterion" Criterion 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 criterion value and tenth percentile criterion value.
add_icc	If TRUE, adds item characteristic curve to the plot. Only available if a valid item pool object (ip) is provided and x_axis_scale = "criterion". The default value is FALSE.
title	Title of the plot. If the value is NULL, the plot title will be suppressed.
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.
x_lim	The limits of x axis in the form c(-4, 4). Only available when x_axis_scale = "criterion". The default value is NULL where the limits will be the minimum and maximum 'criterion' values.
base_r_graph	If TRUE function will plot graphs using base R graphics. If FALSE the function will check whether 'ggplot2' package is installed. If it is installed, it will use 'ggplot2' package for the plot. The default value is FALSE.
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 characteristic curve*

Description

plot_emprical_icc plots empirical item characteristic curve. It plots observed p-values vs. expected p-values grouped into bins based theta scores (or any score supplied). Optionally, provide theta vector, otherwise examinee abilities will be estimated by est_ability(..., type = "eap"). This will slow down the plotting function.

Usage

```
plot_empirical_icc(
  resp,
  item,
  ip,
  theta = NULL,
  bins = 10,
  binwidth = NULL,
  title = "",
  suppress_plot = FALSE,
  base_r_graph = FALSE,
  ...
)
```

Arguments

resp	Response matrix.
item	The column number, column name or the 'ID' of the the item that should be plotted.
ip	An Itempool-class object that is needed for some plots.
theta	A vector of examinee abilities.

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.
binwidth	This determines the width of each bin of the theta scale. Within each bin, there might be different number of examinees.
title	Title of the plot. The default value is "", where title of the plot will be "Trace Plot of <Item ID>". If the value is NULL, the plot title will be suppressed.
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
base_r_graph	If TRUE function will plot graphs using base R graphics. If FALSE the function will check whether 'ggplot2' package is installed. If it is installed, it will use 'ggplot2' package for the plot. The default value is FALSE.
...	Extra parameters that will pass to geom_line.

Value

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

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(model = c("3PL", "GRM"), n = 20)
true_theta <- rnorm(2000)
resp <- generate_resp_set(ip = ip, theta = true_theta)

plot_empirical_icc(resp, "Item_3", ip = ip, theta = true_theta)
plot_empirical_icc(resp, 3, ip = ip, theta = true_theta)
# Change the number of bins
plot_empirical_icc(resp, 3, ip = ip, theta = true_theta, bins = 10)
# Fixed bin width
plot_empirical_icc(resp, 3, ip = ip, theta = true_theta, binwidth = .2)

# Plot GRM item's ICC
plot_empirical_icc(resp, "Item_4", ip = ip, theta = true_theta)
plot_empirical_icc(resp, "Item_4", ip = ip, theta = true_theta, binwidth = .2)
```

---

plot_empirical_icc2 *Plot Empirical Item Characteristic Curve*

Description

plot_emprical_icc plots 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.

Usage

```
plot_empirical_icc2(
  resp,
  item,
  bins = 10,
  binwidth = NULL,
  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.
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.
binwidth	If 'theta' scale is used, the binwidth determines the width of each bin of the theta scale. Within each bin, there might be different number of examinees.
ip	An Itempool-class object needs to be provided if expected ICC desired.
theta	A vector of examinee abilities.
title	Title of the plot. The default value is "".
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.
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

```
ip <- generate_ip(model = c("3PL", "GRM"), n = 20)
true_theta <- rnorm(2000)
resp <- sim_resp(ip = ip, theta = true_theta)

# Provide item ID
plot_empirical_icc2(resp = resp, item = "Item_5")
# Provide item number
plot_empirical_icc2(resp, item = 3)
# Change x-axis scale
plot_empirical_icc2(resp, item = 3, x_axis_scale = "number")
# Change number of bins and x-axis scale
plot_empirical_icc2(resp, item = 3, bins = 11, x_axis_scale = "theta")
# Use bin width
plot_empirical_icc2(resp, item = 3, binwidth = 2)
# Use theta scores instead of raw scores
plot_empirical_icc2(resp, item = 3, binwidth = .2, ip = ip,
                    theta = true_theta)

# A GRM item
plot_empirical_icc2(resp, item = 4)
plot_empirical_icc2(resp, item = 4, x_axis_scale = "percent")
plot_empirical_icc2(resp, item = 4, x_axis_scale = "number")
plot_empirical_icc2(resp, item = 4, binwidth = 4)
# Use raw score and custom binwidth
plot_empirical_icc2(resp, item = 4, x_axis_scale = "percent", binwidth = 4)
# Use theta score
plot_empirical_icc2(resp, item = 4, binwidth = .2, ip = ip,
                    theta = true_theta)
# Add arguments for 'geom_line'
plot_empirical_icc2(resp, item = 4, binwidth = .2, ip = ip,
                    theta = true_theta, size = 1, alpha = .25)
```

---

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),
  focus_item = NULL,
  title = "",
  suppress_plot = FALSE,
  base_r_graph = FALSE,
  separate_testlet = TRUE,
  ...
)
```

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	Either (a) a numeric vector of length two where the values are minimum and maximum theta values, or, (b) a numeric vector of length more than two where values represents the theta values that will be plotted.
focus_item	If one or more items information graphs needed to be focused whereas rest of the items' information functions needed to be on the background, provide item numbers or item ID's to be focused.
title	Title of the plot. If the value is NULL, the plot title will be suppressed.
suppress_plot	If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
base_r_graph	If TRUE function will plot graphs using base R graphics. If FALSE the function will check whether 'ggplot2' package is installed. If it is installed, it will use 'ggplot2' package for the plot. The default value is FALSE.
separate_testlet	A logical value indicating whether to separate items within testlets or not. If TRUE, information values of all items within the testlet are plotted separately. if FALSE, information functions of items within testlets are combined (like test information function) and plotted that way along with standalone items.
...	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 <- generate_ip()
plot_info(ip)
plot_info(ip, tif = TRUE)
plot_info(ip, tif = TRUE, theta_range = c(-3, 3))
# Focus on one item
plot_info(ip, focus_item = "Item_2")

# Base R Graphics
plot_info(ip, base_r_graph = TRUE)
plot_info(ip, focus_item = "Item_2", base_r_graph = TRUE)

# Plot information with focus on a specific item(s)
plot_info(ip, focus_item = "Item_1")
plot_info(ip, focus_item = 3)
# plot_info(ip, focus_item = c(2, 8))
# plot_info(ip, focus_item = c("Item_5", "Item_6"))

plot_info(ip, focus_item = 7, alpha = .7, color = "gray")

plot_info(ip, focus_item = "Item_3", color = "green", base_r_graph = TRUE)


# Information Plots with Testlets
ip <- c(testlet(itempool(b = c(-1, 1), item_id = c("t1-i1", "t1-i2"),
                         D = 1.702), testlet_id = "t1"),
        testlet(itempool(b = c(-2, 0, 2),
                         item_id = c("t2-i1", "t2-i2", "t2-i3"),
                         D = 1.702), testlet_id = "t2"),
        item(b = -1.5, item_id = "i1", D = 1.702),
        item(b = 0.25, item_id = "i2", D = 1.702),
        item(b = 1.5, item_id = "i3", D = 1.702))
plot_info(ip)
plot_info(ip, separate_testlet = FALSE)
```

`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,
  base_r_graph = FALSE,
  suppress_plot = FALSE,
  text_size = 12,
  ...
)
```

Arguments

ip	An Itempool-class class object.
resp	The response string or a Response-class class object.
theta_range	Either (a) a numeric vector of length two where the values are minimum and maximum theta values, or, (b) a numeric vector of length more than two where values represents the theta values that will be plotted.
title	Title of the plot. If the value is NULL, the plot title will be suppressed.
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.
base_r_graph	If TRUE function will plot graphs using base R graphics. If FALSE the function will check whether 'ggplot2' package is installed. If it is installed, it will use 'ggplot2' package for the plot. The default value is FALSE.
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

```
## Not run:
ip <- generate_ip(n = 9)
resp_set <- generate_resp_set(ip = ip, theta = rnorm(10))

# Plot second item's response log-likelihood function
plot_resp_loglik(ip, resp_set[[2]])

# Plot response likelihood function of second item
plot_resp_loglik(ip, resp_set[[2]], likelihood = TRUE)

# Plot using base r graphics
plot_resp_loglik(ip, resp_set[[2]], likelihood = TRUE, base_r_graph = TRUE)

# Suppress the MLE estimate
plot_resp_loglik(ip, resp_set[[4]], show_estimate = FALSE)

## End(Not run)
```

---

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)

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

## S4 method for signature 'Rasch'
prob(ip, theta, derivative = 0)

## S4 method for signature '1PL'
prob(ip, theta, derivative = 0)

## S4 method for signature '2PL'
prob(ip, theta, derivative = 0)

## S4 method for signature '3PL'
prob(ip, theta, derivative = 0)

## S4 method for signature '4PL'
```

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

## S4 method for signature 'GRM'
prob(ip, theta, derivative = 0)

## S4 method for signature 'PCM'
prob(ip, theta, derivative = 0)

## S4 method for signature 'GPCM'
prob(ip, theta, derivative = 0)

## S4 method for signature 'GPCM2'
prob(ip, theta, derivative = 0)

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

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

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

Arguments

ip	An Item-class, or an Itempool-class or Testlet-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.

Value

Item probabilities at given theta will be returned.

Author(s)

Emre Gonulates

Examples

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

# 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)


# Multiple theta values
theta_n <- rnorm(5)

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

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

# 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)


# Multiple theta values
theta_n <- rnorm(5)

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

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)


# Multiple theta values
theta_n <- rnorm(5)

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

```
theta <- rnorm(1)
item1 <- generate_item(model = "3PL")

# 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)


# Multiple theta values
theta_n <- rnorm(5)

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

theta <- rnorm(1)
item1 <- generate_item(model = "4PL")

# 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)


# Multiple theta values
theta_n <- rnorm(5)

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

theta <- rnorm(1)
item1 <- generate_item(model = "GRM")

# Probability of correct response
prob(item1, theta)

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

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

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


item4 <- generate_item(model = "GRM", n_categories = 5)
prob(item4, theta)

# Partial Credit Model
theta <- rnorm(1)
item1 <- generate_item(model = "PCM")

# 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)


# Multiple theta values
theta_n <- rnorm(5)

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


item3 <- generate_item(model = "GPCM2", n_categories = 3)
prob(item3, theta)

theta <- rnorm(1)
item1 <- generate_item(model = "GPCM")

# 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)


# Multiple theta values
theta_n <- rnorm(5)

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

theta <- rnorm(1)
item1 <- generate_item(model = "GPCM2")

# 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)


# Multiple theta values
theta_n <- rnorm(5)

prob(item1, theta_n)
prob(item1, theta_n, derivative = 1)
prob(item1, 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(3)
prob(ip, theta_n)
prob(ip, theta_n, derivative = 1)
prob(ip, theta_n, derivative = 2)

# Extract probabilities of correct response (i.e. response is "1")
sapply(prob(ip, theta_n), `[`, TRUE, "1")
# Probabilities of incorrect response
sapply(prob(ip, theta_n), `[`, TRUE, "0")
```

```
# 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)

# 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)

# Multiple theta
prob(ip, theta_n)

# Extract probabilities of score "2" for each theta value
sapply(prob(ip, theta_n), `[`, TRUE, "2")

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)
```

`prob_sum_score` *Calculate summed-score probabilities*

Description

This function calculates all summed-score probabilities of a given theta value(s) using recursive algorithm described in Thissen, Pommerich, Billeaud and Williams (1995). This function is the extension of the recursive algorithm proposed by Lord and Wingersky (1984) to polytomous items.

Usage

```
prob_sum_score(ip, theta, theta_pdf = NULL)
```

Arguments

ip	An Itempool-class object. Item pool parameters can be composed of any combination of unidimensional dichotomous or polytomous items.
theta	A numeric vector representing the theta values at which the sum score probabilities will be calculated.
theta_pdf	A numeric vector with the same length of theta argument representing the density values of each theta value. The resulting probabilities will be weighted by these values. The default value is NULL where the resulting probabilities will not be weighted.

Value

A matrix containing the probabilities of each possible sum score. Each row represent a sum score and each column represent the theta value provided by theta argument.

Author(s)

Emre Gonulates

References

Kolen, M. J., & Brennan, R. L. (2014). Test equating, scaling, and linking: Methods and practices. Springer Science & Business Media.

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

Thissen, D., Pommerich, M., Billeaud, K., & Williams, V. S. (1995). Item response theory for scores on tests including polytomous items with ordered responses. Applied Psychological Measurement, 19(1), 39-49.

Examples

```
### Example with weighting ###
ip <- generate_ip(model = sample(c("GPCM", "2PL"), 10, TRUE))
theta <- c(-3, -1.2, 0.5, 3)
prob_sum_score(ip, theta = theta)
# Most probable sum scores:
apply(prob_sum_score(ip, theta = theta), MARGIN = 2, which.max) - 1
## Not run:
   plot(ip, type = "tcc", suppress_plot = TRUE) +
     ggplot2::geom_vline(xintercept = theta, lty = "dashed")

## End(Not run)
### Example from Kolen and Brennan (2014) ###
# Item parameters from Kolen and Brennan (2014), p.175, Table 6.1.
ip <- itempool(a = c(1.30, .6, 1.7),
               b = c(-1.30, -.10, .9),
               c = c(.1, .17, .18),
               D = 1.7)
prob(ip, theta = c(-2, 1))
```

```
# IRT observed score distribution using recursive formula from
# Kolen and Brennan (2014), p.200, Table 6.4.
# Numbers are not exactly the same as Kolen and Brennan since due to
# rounding applied to the numbers in the book.
prob_sum_score(ip, theta = -2)


### Example from Thissen, Pommerich, Billeaud and Williams (1995) ###
# Replicating Thissen et al. (1995) example, p.43-44, Table 1.
i1 <- item(a = .5, b = -1)
i2 <- item(a = 1, b = 0)
i3 <- item(a = 1.5, b = 1)
ip <- c(i1, i2, i3) # combine items to form an item pool
theta <- -3:3 # Quadrature points

prob_sum_score(ip, theta)

# Item parameters in Table 2
i1 <- item(a = 1.87, b = c(.65, 1.97, 3.14), model = "GRM")
i2 <- item(a = 2.66, b = c(.12, 1.57, 2.69), model = "GRM")
i3 <- item(a = 1.24, b = c(.08, 2.03, 4.30), model = "GRM")
ip <- c(i1, i2, i3)
delta <- 0.01
theta <- seq(-3, 3, delta)

x <- prob_sum_score(ip = ip, theta = theta, theta_pdf = dnorm(theta))

# Figure 1
plot(x = theta, y = x[2, ], type = "l", ylab = "Posterior Density",
     xlab = "Theta",
     main = paste0("Posterior Distribution for all Examinees Obtaining ",
                   "a Summed Score of 1"))

# Table 3, column "Modeled Score Group Proportion"
rowSums(x)/sum(rowSums(x))
```

---

qip_index *Calculate Quality of Item Pool Index*

Description

The QIP Index can take values between 0 and 1 and indicates an item pool's level of efficiency. A value of 1 signifies an optimum item pool for that examinee group. If one adds redundant items to an item pool that cannot be used by the CAT algorithm, the QIP Index will not increase or will increase minimally. In this sense, the QIP Index is an indicator of the item pools' deficiency, instead of redundancy. However, if an exposure control mechanism is within test specifications, the QIP index can measure whether the redundancy in the item pool supports the exposure control method. See Gonulates (2019) for details.

Note that this function will best work with Rasch or 1PL models. It will not work with polytomous items.

Usage

```
qip_index(cat_sim_output, summary_func = NULL, ...)
```

Arguments

cat_sim_output	This is a list object containing elements that are cat_output class.
summary_func	A string representing the function that will be applied to individual QIP values for a simulee. The default is NULL, where all QIP values of each administered item of a simulee will be returned. Other possible values are: "mean", "median", "min", "max". See examples for demonstrations.
...	Additional arguments that will be passed to the summary_func. For example, if summary_func = "quantile", probability of the 25th quantile can be specified using the argument prob = .25. See examples for demonstrations. Since ... will be passed to sapply function, simplify = FALSE can be passed to function to get results as list elements.

Value

A vector or matrix of QIP values or the summary statistics of QIP values.

Author(s)

Emre Gonulates

References

Gönülateş, E. (2019). Quality of Item Pool (QIP) Index: A Novel Approach to Evaluating CAT Item Pool Adequacy. Educational and Psychological Measurement, 79(6), 1133–1155. <doi:10.1177/0013164419842215>

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_output <- cat_sim(true_ability = rnorm(10), cd = cd)

qip_index(cat_output)

# Return result as list elements
qip_index(cat_output, simplify = FALSE)

# Summarize QIP values:
qip_index(cat_output, summary_func = "mean")
qip_index(cat_output, summary_func = "median")
qip_index(cat_output, summary_func = "min")
qip_index(cat_output, summary_func = "max")
qip_index(cat_output, summary_func = "quantile", prob = .25)
qip_index(cat_output, summary_func = "quantile", prob = c(.25, .5, .75))

qip_index(cat_output, summary_func = "quantile", prob = c(.25, .5, .75),
```

```
            simplify = FALSE)
```

`Rasch-class` *Rasch model*

Description

Rasch model

Slots

`b` Item difficulty parameter

`se_b` Standard error of item difficulty parameter

Author(s)

Emre Gonulates

`response` *Create a Response object from a vector of responses*

Description

Create a Response object from a vector of responses

Usage

```
response(
  score = NULL,
  examinee_id = NULL,
  item_id = NULL,
  raw_response = NULL,
  testlet_id = NULL,
  order = NULL,
  response_time = NULL,
  misc = NULL
)
```

Arguments

score	A numeric vector holding the scores given to items.
examinee_id	Examinee/Subject/Student ID. A character string to identify an examinee.
item_id	A character vector holding the item IDs.
raw_response	A vector of strings holding the raw responses to items.
testlet_id	A character vector holding the testlet IDs that given item belongs. It can be NULL if none of the items belongs to any testlet. Items that do not belong to any testlet should be represented by NA.
order	An integer vector representing the administration order of an item.
response_time	A numeric vector representing the response times. By default, numbers are assumed to represent seconds.
misc	A list that will hold miscellaneous information about the responses. For example, misc = list(item_role = c("O", "O", "O","F")) will hold whether administered item is a field test or an operational test item.

Author(s)

Emre Gonulates

Response-class *An S4 class representing responses of a single examinee*

Description

An S4 class representing responses of a single examinee

Slots

examinee_id Examinee/Subject/Student ID. A string or an integer to identify an examinee.

item_id A character vector holding the item IDs.

testlet_id A character vector holding the testlet IDs that given item belongs. It can be NULL if none of the items belongs to any testlet. Items that do not belong to any testlet should be represented by NA.

score A numeric vector holding the scores given to items.

raw_response A vector of strings holding the raw responses to items.

order An integer vector representing the administration order of an item.

response_time A numeric vector representing the response times. By default, numbers are assumed to represent seconds.

misc A list that will hold miscellaneous information about the responses. For example, misc = list(item_role = c("O", "O", "O", "F")) will hold whether administered item is a field test or an operational test item.

Author(s)

Emre Gonulates

response_set *Create* Response_set-class *object*

Description

This function creates a Response_set-class object from various types of data sets. Currently following scenarios are supported:

Usage

```
response_set(
  x,
  data_format = "wide",
  ip = NULL,
  examinee_id_var = NULL,
  testlet_id_var = NULL,
  item_id_var = NULL,
  score_var = NULL,
  raw_response_var = NULL,
  order_var = NULL,
  response_time_var = NULL,
  misc_var = NULL,
  misc_unique_var = NULL,
  misc = NULL,
  fill_na_score = NULL
)
```

Arguments

x	A matrix or data.frame holding item scores. See the description about the options. Additionally, it can be a list of Response-class objects.
data_format	A string value representing the format of the data x supplied. The default value is "wide". The following options are available: **"wide"** x can be in wide format data where a matrix or data.frame where rows represents examinees and columns represent items. Each row will be converted to a Response-class object. If the columns has names (and an Itempool-class object has not been supplied), then the item_ids will be supplied by the column names. If neither column names nor an Itempool-class object supplied, default item_ids will be given. If rows has names, those will be used as examinee_ids. **"long"** x can be in long format where data.frame with at least three columns: (1) a column for examinee_id, (2) a column for item_id and (3) a column for either scores or raw_responses. Additional columns can be added such as testlet_id, item order, response_time.

ip — Optionally an Itempool-class object that is holding the item parameters can be supplied to check whether Response_set object created is compatible with the Itempool-class object.

examinee_id_var — A string for the column name that holds examinee ids, if x is in long format.

testlet_id_var — A string for the column name that holds testlet ids, if x is in long format.

item_id_var — A string for the column name that holds item ids, if x is in long format.

score_var — A string for the column name that holds examinee scores, if x is in long format.

raw_response_var — A string for the column name that holds raw responses of the examinees, if x is in long format.

order_var — A string for the column name that holds the administration order of items, if x is in long format.

response_time_var — A string for the column name that holds response time information of the items, if x is in long format.

misc_var — A string for the column names that are holding the miscellaneous information of the items. Available only when x is in long format. Within an examinee, if there is additional information for each item (for example, item's type, item's reading level, examinee's raw response to an item, whether an item is operational or not, the date/time item is administered, ratings of multiple raters, etc.), in the dataset, this information can be passed. Later in the code, such information can be extracted by $ operator. See examples.

misc_unique_var — A string for the column names that are holding the miscellaneous information of the items. Different than misc_var, these columns are assumed to be the same within an examinee, so only the unique value of this column within an examinee will be saved. Examples of variables for this column is gender, race, ability score, school of the examinee that will not vary from one item to another within an examinee. The argument is only available when data_format = "long".

misc — A list of miscellaneous variables that needs to be added to the Response_set object.

fill_na_score — If some examinees do not answer all items, the value fill_na_score will be replaced by the scores of unanswered items. If an ip value provided, 'all items' will be all of the items in the item pool. Otherwise, all items will be the list of all unique item_id values.

Currently, this feature only works when x is a data frame or matrix.

Value

A Response_set-class object.

Author(s)

Emre Gonulates

Examples

```
##### Wide format data #####
## Example 1
x_wide <- matrix(sample(0:1, 35, TRUE), nrow = 7, ncol = 5)
response_set(x_wide)

## Example 2
ip <- generate_ip(n = 6)
# simulate responses for 10 examinees
resp_matrix <- sim_resp(ip = ip, theta = rnorm(10), prop_missing = .2,
                        output = "matrix")
# convert it to tibble
resp_wide <- as.data.frame(resp_matrix)
resp_wide$stu_id <- rownames(resp_matrix)
# Create a Response_set object:
resp_set <- response_set(resp_wide, data_format = "wide", ip = ip,
                         examinee_id_var = "stu_id")
# Retrieve examinee ids:
resp_set$examinee_id
# Fourth examinee:
resp_set[[4]]
# Scores of 6th examinee
resp_set[[6]]$score


##### Long format data #####
x_long <- data.frame(examinee_id = c("stu1", "stu1", "stu1", "stu2", "stu2"),
                     item_id = c("i1", "i2", "i4", "i1", "i2"),
                     scr = c(0, 1, 0, 1, 0),
                     rwscore = c("A", "D", "B", "C", "D"),
                     resptime = c(33, 55, 22, 66, 31),
                     # These will be passed to misc
                     item_type = c("MC", "MC", "MS", "SA", "MC"),
                     lexile_level = c(1, 4, 3, 2, 1),
                     word_count = c(123, 442, 552, 342, 666),
                     ability = c(1.1, 1.1, 1.1, -.2, -.2),
                     grade = c("7", "7", "7", "11", "11")
                     )

resp_set <- response_set(x = x_long,
                         data_format = "long",
                         examinee_id_var = "examinee_id",
                         item_id_var = "item_id",
                         score_var = "scr",
                         raw_response_var = "rwscore",
                         response_time_var ="resptime",
                         misc_var = c("item_type", "lexile_level"),
                         misc_unique_var = c("ability", "grade")
                         )

resp_set[[1]]  # Response of the first examinee
resp_set$item_type  # extract item_type of each examinee
```

```
resp_set$grade  # extract grade of each examinee

# Also, additional examinee level miscellaneous information can be added:
resp_set$gender <- c("M", "F")
resp_set[[2]]$gender  # access second examinee's gender.
resp_set$gender

# Fill missing values with 0.
response_set(x = x_long,
             data_format = "long",
             examinee_id_var = "examinee_id",
             item_id_var = "item_id",
             score_var = "scr",
             raw_response_var = "rwscore",
             response_time_var ="resptime",
             misc_var = c("item_type", "lexile_level"),
             fill_na_score = 0
             )
```

`Response_set-class` *An S4 class representing responses of a set of examinees*

Description

An S4 class representing responses of a set of examinees

Slots

`response_list` A list of `Response-class` objects. If the `examinee_id` slots of `Response-class` objects are not `NULL`, there cannot be duplicates.

`item_id` A character vector of `Item` ID's in the `Response-class` objects. The order of this item_id will be used when converting `Response_set-class` objects to a matrix.

`testlet_id` A character vector of `Testlet` ID's in the `Response-class` objects.

`misc` This slot will hold any other information about the response set.

Author(s)

Emre Gonulates

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,ANY'
resp_loglik(ip, resp, theta, derivative = 0)

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

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

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

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

## S4 method for signature 'Itempool,Response_set'
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

`score_info` *Calculate Score Information Function*

Description

This function calculates the score information function of a given CAT test. Ideally, a large number of simulees (say 1,000) will be simulated at each theta level equally spaced along a large theta range (like [-4, 4]). The score information function at each theta will be calculated using the formulas 11-2 and 11-3 presented in Sands, Waters and McBride (1997, pages 127-128). Also see Lord (1980), Eqn. 10-7.

For example if 1000 examinees simulated at each of the following theta values (-3, -2, -1, 0, 1, 2, 3), the function will not calculate score information values at theta = -3 and theta = 3. Score information values at second values to the edges (i.e. theta = -2 and theta = 2) will be calculated using Equation 11-2 of Sands et.al. (1997). The rest of the score information values (at theta = -1, 0, 1) will be calculated using equation 11-3 (page 128).

Usage

```
score_info(true_theta, est_theta, bins = NULL)
```

Arguments

`true_theta`	A vector of true theta values.
`est_theta`	A vector of estimated theta values.
`bins`	The number of bins true theta values should be grouped into. Ideally, this value is `NULL` and equal number of simulees are already in bins, and within each bin `true_theta` values are equal to each other. If these conditions are not satisfied, a bin value can be supplied.

Value

A data frame of true theta values and score information value at each theta value will be returned.

Author(s)

Emre Gonulates

References

Lord, F. M. (1980). Applications of item response theory to practical testing problems. Routledge.

Sands, W. A., Waters, B. K., & McBride, J. R. (1997). Computerized adaptive testing: From inquiry to operation. American Psychological Association.

Examples

```
ip <- generate_ip(n = 30)
cd <- create_cat_design(ip = ip, next_item_rule = 'mfi',
                        termination_rule = 'max_item',
                        termination_par = list(max_item = 10))
# The following true_theta example is not ideal. For more informative score
# score information functions you can use more bins and more simulees like:
# rep(seq(-4, 4, .1), each = 1000)
true_theta <- rep(seq(-3, 3, 1), each = 10)
cat_data <- cat_sim(true_ability = true_theta, cd = cd)
dtf <- summary(cat_data)

s_info <- score_info(true_theta = dtf$true_ability,
                     est_theta = dtf$est_ability)
s_info
```

---

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, output = "matrix")

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

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

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

## S4 method for signature 'numMatDfListChar'
sim_resp(ip, theta, prop_missing = 0, output = "matrix")
```

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.
output	Type of the output. Following options are available: "matrix" A matrix object. "response_set" A Response_set-class object with item pool attached.

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("examinee_id", "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: **examinee_id** Examinee ID's if named true theta vector has been provided to `cat_sim()` function. **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 **mean_qip** Mean of Quality of Item Pool Index. See `qip_index()` function for details. **median_qip** Median of Quality of Item Pool Index. See `qip_index()` function for details. **min_qip** Minimum value of Quality of Item Pool Index. See `qip_index()` function for details. **max_qip** Maximum value of Quality of Item Pool Index. See `qip_index()` function for details.

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)

# Get only selected columns
summary(cat_data, cols = c("examinee_id", "true_ability", "est_ability",
                           "bias"))
summary(cat_data, cols = c("examinee_id", "true_ability", "est_ability",
                           "mean_qip", "median_qip", "min_qip"))
```

---

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, testlet_id = "T1")
testlet(ip, testlet_id = "T1", content = "Algebra")
# Add misc field to the testlet:
testlet(ip, testlet_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), item_id = paste0("t1-i", 1:2),
                 misc = list(list(sympson_hetter_k = .8, form = "B1"),
                             list(sympson_hetter_k = .9))),
        testlet_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

testlet_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

`var,Item-method` *Calculate the variance of an Item*

Description

`var` Returns the variance of an item or multiple items with given parameters for a given ability or abilities, i.e. θ.

Usage

```
## S4 method for signature 'Item'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature 'Rasch'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature '1PL'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature '2PL'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature '3PL'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature '4PL'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature 'GRM'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature 'PCM'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature 'GPCM'
var(x, y = NULL, na.rm = FALSE, use)

## S4 method for signature 'GPCM2'
var(x, y = NULL, na.rm = FALSE, use)
```

Arguments

x	An `Item-class` or an `Itempool-class` object containing the item parameters.

y	A numeric vector containing the ability parameters (i.e. theta).
na.rm	Ignored for var(Item, ...)
use	Ignored for var(Item, ...)

Value

Item variances at given theta will be returned.

Author(s)

Emre Gonulates

---

var,Itempool-method *Calculate the variances of items in an Itempool*

Description

var Returns the variance of each item of an Itempool-class object for a given ability or abilities, i.e. θ.

Usage

```
## S4 method for signature 'Itempool'
var(x, y = NULL, na.rm = FALSE, use)
```

Arguments

x	An Itempool-class object containing the item parameters.
y	A numeric vector containing the ability parameters (i.e. theta).
na.rm	Ignored for var(Itempool, ...)
use	Ignored for var(Itempool, ...)

Value

Item variances at given theta will be returned.

Author(s)

Emre Gonulates

var,Testlet-method *Calculate the variances of items in a Testlet*

Description

Calculate the variances of items in a Testlet

Usage

```
## S4 method for signature 'Testlet'
var(x, y = NULL, na.rm = FALSE, use)
```

Arguments

x	An Testlet-class object containing the item parameters of the testlet.
y	A numeric vector containing the ability parameters (i.e. theta).
na.rm	Ignored for var(Testlet, ...)
use	Ignored for var(Testlet, ...)

Value

Item variances at given theta will be returned.

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: 'item_id' Extract 'item_id' of an Item-class object. 'id' Extract 'item_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", item_id = 'item23', content = 'Geometry',
              misc = list(enemies = c("item1", "item2"), key = "C"),
              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$parameters
# Get item ID
item1$item_id
# Get item content
item1$content
# Get misc values
item1$misc
# Get maximum possible score of item
item1$max_score

# Get elements of misc directly:
item1$misc$key  # "C"
item1$key  # "C"
```

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

Description

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

Usage

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

Arguments

`x`	An `Itempool-class` object.
`name`	Name of the parameter. Available values:

`'id'` Extract `id`'s of all items and testlets. This will not extract the `item_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. Use `$item_model` to extract models of standalone items.

`'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'` Extract `se`'s of all items and testlets. This will not extract the `se`'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.

`'item_id'` **or** `'resp_id'` Extract `item_id`'s of all standalone items and items within the testlets. It will not return `testlet_id`'s. This is mainly to get the `item_id`'s of items which has a response.

`'testlet_id'` Extract `testlet_id`'s of all items within the testlets. If the item is a standalone item, then a `NA` vector will be returned for it's testlet ID value.

'item_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.

'item_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.

'item_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.

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

Value

See the 'name' argument above for possible return values.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(n = 7, model = "3PL", content = c("Geometry", "Algebra"))

ip$a
ip$b
ip$D
ip$model
ip$id
ip$content
```

$,Response-method *Get slots of the an* Response-class *object.*

Description

Get slots of the an Response-class object.

Usage

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

Arguments

x	An Response-class object.
name	Name of the parameter. Available values:
	'examinee_id' Extract Examinee/Subject/Student ID.
	'item_id' Extract item ids
	'testlet_id' Extract testlet IDs, if there is any.
	'score' Extract item scores.
	'raw_response' Extract raw responses.
	'order' Extract item order.
	'response_time' Extract response times.
	'misc' Extract 'misc' field.

Value

See the 'name' argument above for possible return values.

Author(s)

Emre Gonulates

Examples

```
resp <- response(score = c(0, 1, 0), examinee_id = "Ex-412",
                 item_id = c("I1", "I2", "I3"),
                 raw_response = c("B", "D", "A"),
                 order = 1:3,
                 response_time = c(66, 23, 89),
                 misc = list(form = "A1",
                             operational = c(TRUE, TRUE, FALSE))
                 )
resp$score
resp$item_id
resp$examinee_id
resp$raw_response
resp$order
resp$response_time
resp$misc
resp$misc$form
resp$form
```

`$,Response_set-method` *Get slots of the a* `Response_set-class` *object.*

Description

Get slots of the a `Response_set-class` object.

Usage

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

Arguments

`x`	An `Response_set-class` object.
`name`	Name of the parameter. Available values:
	`'response_list'` Extract Response objects as a list.
	`'item_id'` Extract unique list of item IDs that are in the response set.
	`'testlet_id'` Extract unique list of testlet IDs that are in the response set.
	`'misc'` Extract 'misc' field.
	`'score'` Return a score matrix of responses
	`'raw_response'` Return a raw score matrix of responses

Value

See the 'name' argument above for possible return values.

Author(s)

Emre Gonulates

Examples

```
resp <- sim_resp(ip = generate_ip(), theta = rnorm(5),
                 output = "response_set")
resp$response_list
```

`$,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: `'testlet_id'` **or** `'id'` Get the `testlet_id` of the testlet `'content'` Get the `content` of the testlet. `'model'` Get the `model` of the testlet. `'item_models'` Get the `models` of the items within the testlet. `'item_id'` Get the `item_ids` of the items within the testlet. `'misc'` Get the `misc` field of 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), testlet_id = "my-testlet",
              content = "Reading",
              misc = list(paragraph_text = "This is a paragraph."))
t1$model
t1$testlet_id
t1$item_list
t1$item_models
t1$item_id
t1$content
t1$item_models
```

```
t1$misc
t1$paragraph_text
```

`$.cat_output` *Prints the raw output of cat_sim*

Description

This function prints a data frame that shows all of the steps of a CAT for a single examinee.

Usage

```
## S3 method for class 'cat_output'
x$name
```

Arguments

`x`	This is a cat_output object which has `"cat_output"` class.
`name`	Name of the field. Available options: `"ip"` Extract items administered to examinee `"resp"` Extract responses `"testlet"` Extract testlets administered `"est_before"` Extract ability estimate before administration of an item. `"item_id"` Extract administered item IDs. `"est_after"` Extract ability estimate after administration of an item. `"se_before"` Extract standard error before administration of an item. `"se_after"` Extract standard error after administration of an item. `"true_theta"` Extract true theta as a vector `"test_length"` Extract test length of the adaptive test `"final_est"` Extract final ability estimate. `"final_se"` Extract final standard error.

Value

See the 'name' argument above for possible return values.

Author(s)

Emre Gonulates

See Also

`cat_sim`

Examples

```
n <- 20 # number of items
ip <- generate_ip(n = n)
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(1), cd = cd)
cat_data
cat_data$resp # Extract responses to administered items
cat_data$ip # Administered items
cat_data$item_id # Extract administered item IDs
cat_data$est_before # Ability estimates before the administration of an item
cat_data$est_after # Ability estimates after the administration of an item
cat_data$true_theta # True ability that generates examinee responses

# Simulation with more than one simulees
n <- 20 # number of items
ip <- generate_ip(n = n)
cd <- create_cat_design(ip = ip, next_item_rule = 'mfi',
                        termination_rule = 'max_item',
                        termination_par = list(max_item = 10))
n_examinee <- 3
cat_data_list <- cat_sim(true_ability = rnorm(n_examinee), cd = cd)
cat_data_list[[3]]$item_id
cat_data_list[[2]]$item_id
cat_data_list[[3]]$resp
cat_data_list[[2]]$resp
cat_data_list[[2]]$test_length
cat_data_list[[2]]$final_est
cat_data_list[[2]]$final_se
```

$<-,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

```
itm <- new("3PL", item_id = 'item23', content = 'Geometry',
            misc = list(enemies = c("item1", "item2")),
            b = 2, c = .12, a = 1.2, D = 1)
itm$a <- 2
itm$D <- 1.7
itm$item_id <- "Item-111"
itm$content <- 'Algebra'
itm$se_a <- 2.2
# Set all misc fields like this
itm$misc <- list(enemies = c("item5"), strands = c("A4", "C2"))

# Add a misc field
itm$key <- "C"

# Remove a misc field
itm$enemies <- NULL
```

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

Description

Set values to parameters or components of 'Itempool' 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, item_id, id, content, item_list are available.
value	The new value that will be assigned.

'item_id' For item_id, the value should be a list of strings that has the same length as the number of items in the Itempool-class object, i.e. ip$n$items. There should not be any duplicated ID's. If there are Testlet-class objects in the item pool, the items within the testlet elements will be updated.

'id' 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. If there are only Item-class objects, then item ID's will be updated. If there are Testlet-class objects in the item pool, then only the testlet IDs will be updated. Items within the Testlet can be updated using ..$item_id.

'content' 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.

'item_list' For item_list, the value should be a list of Item-class or Testlet-class objects.

'misc' For misc, the value should be a list.

Value

This operation will return an Itempool-class object.

Author(s)

Emre Gonulates

Examples

```
ip <- generate_ip(model = "3PL", n = 5)
ip$a
# Set new values for the a parameters
ip$a <- 2
# Set new values for the b parameters
ip$b <- -2:2
# Set new ids
ip$item_id <- paste0("my-item-", 5:9)

# Set new item content
ip$content <- c("Geometry", "Algebra", "Algebra", "Geometry", "Geometry")

# Add misc field to all items:
ip$difficulty <- c("Easy", "Easy", "Hard", "Hard", "Hard")
ip$difficulty

# Add an overall misc field to itempool:
ip$form_name <- "Frm8"

# Remove the misc field from all items
ip$difficulty <- NULL
ip$difficulty
```

`$<-,Response-method` *Set values to components of 'Response' class objects*

Description

Set values to components of 'Response' class objects

Usage

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

Arguments

x	Response-class object.
name	Name of the parameter or component. Following are available: 'examinee_id' Set Examinee/Subject/Student ID. 'item_id' Set item ids. 'testlet_id' Set testlet IDs. 'score' Set item scores. 'raw_response' Set raw responses. 'order' Set item order. 'response_time' Set response times. 'misc' Set 'misc' field. ... Any value that does not match the names above, will be added to the misc field of the Response.
value	The new value that will be assigned.

Value

This operation will return an Response-class object.

Author(s)

Emre Gonulates

Examples

```
resp <- response(score = c(0, 1, 0))
resp
resp$examinee_id <- "Stu-123"
resp$item_id <- c("i14", "i4", "i9")
resp$raw_response <- c("D", "D", "C")
resp$order <- c(4L, 3L, 1L)
resp$misc <- list(Form = "A1", operational = c(TRUE, TRUE, FALSE))
resp
```

```
# Add any other named element:
resp$content <- c("Alg", "Alg", "Geo")
resp
resp$misc
```

`$<-,Response_set-method` *Set values to components of 'Response_set' class objects*

Description

Set values to components of 'Response_set' class objects

Usage

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

Arguments

`x`	`Response_set-class` object.
`name`	Name of the parameter or component. Currently only `response_list`, `misc` are available.
`value`	The new value that will be assigned.

Value

This operation will return an `Response_set-class` object.

Author(s)

Emre Gonulates

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

Description

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

Usage

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

Arguments

x	An 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

Examples

```
tlt <- generate_testlet()
tlt$testlet_id <- "New-Testlet-ID-111"
tlt$content <- "Algebra"
# Set all misc fields like this
tlt$misc <- list(passage_text = "This is a reading passage.",
                 passage_lexile = 450)

tlt$passage_text
# Add a misc field
tlt$passage_language <- "En-US"

# Remove a misc field
tlt$passage_language <- NULL
```

Index