Package ‘irt’

November 9, 2021

Type Package
Title Item Response Theory and Computerized Adaptive Testing Functions
Version 0.2.5
Maintainer Emre Gonulates <egonulates@gmail.com>
Description A collection of Item Response Theory (IRT) and Computerized Adaptive Testing (CAT) functions that are used in psychometrics.
License AGPL (>= 3)
Depends methods
LinkingTo Rcpp
Imports Rcpp (>= 1.0.1), parallel
NeedsCompilation yes
RoxygenNote 7.1.2
Encoding UTF-8
Collate 'RcppExports.R' 'ability_estimation.R' 'bilog.R' 'cat_sim.R'
  'cat_sim_helper_functions.R' 'item-class.R'
  'item-class-methods.R' 'itempool-class.R'
  'itempool-class-methods.R' 'dif.R' 'response-class.R'
  'generate_objects.R' 'info.R' 'ipd.R' 'irt.R' 'item_analysis.R'
  'item_fit.R' 'kernel_smoothing.R' 'response_set-class.R'
  'max_score.R' 'mean.R' 'misc.R' 'person_fit.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' 'zzz.R'
Config/testthat/edition 3
Suggests ggplot2, markdown, knitr, rmarkdown, tibble, pillar
VignetteBuilder knitr
Author Emre Gonulates [aut, cre] (<https://orcid.org/0000-0002-3834-3266>)
Repository CRAN
Date/Publication 2021-11-09 05:40:02 UTC
R topics documented:

1PL-class ................................................................. 4
2PL-class ................................................................. 4
3PL-class ................................................................. 5
4PL-class ................................................................. 5
add_misc ................................................................. 6
as.data.frame.Item .................................................... 7
as.data.frame.Response ............................................... 9
as.data.frame.Response_set ......................................... 11
as.Itempool ............................................................ 11
as.list.Itempool ....................................................... 12
as.list.Response_set ................................................ 13
as.matrix.Response_set-method .................................... 13
biserial ................................................................. 15
c.Item-method ........................................................ 16
calculate_exposure_rates .......................................... 17
calculate_overlap_rates ............................................. 18
cat_sim ................................................................. 19
cat_sim_fast ........................................................ 20
create_cat_design .................................................... 21
dif ................................................................. 30
distractor_analysis .................................................. 30
est_ability ............................................................ 32
est_bilog ............................................................... 34
generate_ip .......................................................... 47
generate_item ........................................................ 48
generate_resp ........................................................ 49
generate_resp_set ................................................... 50
generate_testlet ..................................................... 51
get_cat_administered_items ....................................... 53
get_cat_response_data ............................................. 53
get_max_possible_total_score ..................................... 55
GPCM-class .......................................................... 56
GPCM2-class ........................................................ 56
GRM-class ............................................................ 57
info ................................................................. 57
ipd ................................................................. 60
irt ................................................................. 62
is.Item ............................................................. 62
item ............................................................... 63
Item-class ............................................................ 65
Itempool ............................................................. 68
Itempool-class ....................................................... 69
item_analysis ....................................................... 70
item_fit ........................................................... 71
ks ................................................................. 73
length,Itempool-method ........................................... 75
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2PL-class</td>
<td>76</td>
</tr>
<tr>
<td>M3PL-class</td>
<td>77</td>
</tr>
<tr>
<td>max_score</td>
<td>77</td>
</tr>
<tr>
<td>mean,Item-method</td>
<td>78</td>
</tr>
<tr>
<td>mean,Itempool-method</td>
<td>79</td>
</tr>
<tr>
<td>mean,Testlet-method</td>
<td>80</td>
</tr>
<tr>
<td>PCM-class</td>
<td>81</td>
</tr>
<tr>
<td>person_fit</td>
<td>81</td>
</tr>
<tr>
<td>plot.Item</td>
<td>82</td>
</tr>
<tr>
<td>plot.Itempool</td>
<td>84</td>
</tr>
<tr>
<td>plot.ks_output</td>
<td>86</td>
</tr>
<tr>
<td>plot_distractor.icc</td>
<td>87</td>
</tr>
<tr>
<td>plot_empirical.icc</td>
<td>89</td>
</tr>
<tr>
<td>plot_empirical.icc2</td>
<td>90</td>
</tr>
<tr>
<td>plot_info</td>
<td>92</td>
</tr>
<tr>
<td>plot_resp_loglik</td>
<td>94</td>
</tr>
<tr>
<td>prob</td>
<td>96</td>
</tr>
<tr>
<td>Rasch-class</td>
<td>102</td>
</tr>
<tr>
<td>response</td>
<td>102</td>
</tr>
<tr>
<td>Response-class</td>
<td>103</td>
</tr>
<tr>
<td>response_set</td>
<td>104</td>
</tr>
<tr>
<td>Response_set-class</td>
<td>107</td>
</tr>
<tr>
<td>resp_lik</td>
<td>108</td>
</tr>
<tr>
<td>resp_loglik</td>
<td>109</td>
</tr>
<tr>
<td>rsss</td>
<td>110</td>
</tr>
<tr>
<td>score_info</td>
<td>111</td>
</tr>
<tr>
<td>sim_resp</td>
<td>112</td>
</tr>
<tr>
<td>summary.cat_output</td>
<td>114</td>
</tr>
<tr>
<td>testlet</td>
<td>115</td>
</tr>
<tr>
<td>Testlet-class</td>
<td>116</td>
</tr>
<tr>
<td>var,Item-method</td>
<td>116</td>
</tr>
<tr>
<td>var,Itempool-method</td>
<td>118</td>
</tr>
<tr>
<td>var,Testlet-method</td>
<td>118</td>
</tr>
<tr>
<td>$,Item-method</td>
<td>119</td>
</tr>
<tr>
<td>$,Itempool-method</td>
<td>120</td>
</tr>
<tr>
<td>$,Response-method</td>
<td>122</td>
</tr>
<tr>
<td>$,Response_set-method</td>
<td>123</td>
</tr>
<tr>
<td>$,Testlet-method</td>
<td>124</td>
</tr>
<tr>
<td>$&lt;-,Item-method</td>
<td>125</td>
</tr>
<tr>
<td>$&lt;-,Itempool-method</td>
<td>126</td>
</tr>
<tr>
<td>$&lt;-,Response-method</td>
<td>127</td>
</tr>
<tr>
<td>$&lt;-,Response_set-method</td>
<td>128</td>
</tr>
<tr>
<td>$&lt;-,Testlet-method</td>
<td>129</td>
</tr>
</tbody>
</table>

Index 130
### 1PL-class

**One-Parameter Logistic IRT model**

#### Description

One-Parameter Logistic IRT model

#### Slots

- \( b \)  Item difficulty parameter
- \( D \)  Scaling constant
- \( \text{se}_b \)  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
- \( \text{se}_a \)  Standard error of item discrimination parameter
- \( \text{se}_b \)  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 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

**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))
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.
onoptional  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
Emre Gonulates
Emre Gonulates
Emre Gonulates
Emre Gonulates
Emre Gonulates
Emre Gonulates
Emre Gonulates
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**

```r
# 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.
as.list.Itempool

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
```r
## 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
```r
## 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

```r
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")
```
Calculate biserial correlation

**Usage**

```r
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**


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, 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 Item-
pool 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

```r
## 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, ...)
```
calculate_exposure_rates

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
Emre Gonulates

Examples

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

# Concatenate items
c(item1, item2)

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

calculate_exposure_rates

*Calculate exposure rate of items for CAT*

Description

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

Usage

```r
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.
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
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

```r
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.

Examples

```r
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)
```
**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**

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

---

**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

```r
cd <- create_cat_design(ip = generate_ip(n = 30),
    termination_rule = c('Var.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)
```

description

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

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

create_cat_design(
  ip = NULL,
  title = NULL,
  true_ip = NULL,
  first_item_rule = "fixed_theta",
  first_item_par = list(theta = 0),
  next_item_rule = "mfi",
  next_item_par = NULL,
  ability_est_rule = "eap",
  ability_est_par = NULL,
  final_ability_est_rule = NULL,
  final_ability_est_par = NULL,
  termination_rule = c("min_item", "min_se", "max_item"),
  termination_par = list(min_item = 10, min_se = 0.33, max_item = 20),
  exposure_control_rule = NULL,
  exposure_control_par = NULL,
  content_bal_rule = NULL,
  content_bal_par = NULL,
  ability_type = "theta"
)

Arguments

ip An Itempool-class object containing item parameters, content information, etc.
If ip = NULL this means this is an infinite item pool, where b is on demand, c = 0 and a = 1, D = 1.7.
If true_ip argument is NULL, this item pool will be used to generate item responses.

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: NULL

First_item_par
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 vari-
ance. 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 abil-
ity estimate.
Required parameters: None.
fixed  Administer a fixed set of items from the item pool. This is basically a
linear fixed length test where the order of items are predefined. Exposure
control rules and parameters will be ignored for this selection rule.
Required Parameters:
item_id  A vector of the item IDs that should be administered.
next_item_par  A list of length one or length maximum test length that sets the parameters of
next item selection rules. It can also be NULL, in which case no parameters
necessary for that next item selection procedure.
Default: NULL
ability_est_rule  A vector of length one or length maximum test length which is designating the
next item selection rules.
Default: "eap"
Possible values and required parameters:

"eap" Expected-a-posteriori. Required parameters:
   prior_dist Distribution of the prior distribution. Available values:
   * norm for normal distribution, * unif for uniform distribution.
   The default value is norm.
   prior_par A vector of prior parameters.
   * For normal distribution c(0,1), see ?dnorm * For uniform distribution c(-3,3), see ?dunif
   The default value is c(0,1).
   min_theta Minimum possible value of theta. It is a lower bound.
   The default value is -4.
   max_theta Maximum possible value of theta. It is an upper bound.
   The default value is 4.
no_of_quadrature The number of quadrature, more specifically the number of bins the theta range should be divided. The more bins, the more precise (and slower) the estimates will be.
   The default value is 50.
"owen" Owen’s Bayesian Estimation Required parameters:
   prior_mean Prior mean value. The default value is 0.
   prior_var Prior variance value. The default value is 1.
"ml" Maximum likelihood estimation using Newton-Raphson algorithm. If this method is used, the standard error of ability estimates are calculated using the inverse information value at this theta estimate.
   Required parameters:
   min_theta Minimum possible value of theta. It is a lower bound. The default value is -4.
   max_theta Maximum possible value of theta. It is an upper bound. The default value is 4.
criterion This value determines the accuracy of estimates. Smaller values lead more accuracy but the speed of estimation reduces as the value of criterion decreases. The default value is 0.001.
"eap_ml" Expected-a-posteriori until an imperfect item response string, then switch to Maximum Likelihood estimation. Required parameters:
   prior_dist Distribution of the prior distribution.
   Available values:
   norm for normal distribution,
   unif for uniform distribution.
   prior_par A vector of prior parameters. For normal distribution c(0,1), see ?dnorm For uniform distribution c(-3,3), see ?dunif
   min_theta Minimum possible value of theta. It is a lower bound.
   max_theta Maximum possible value of theta. It is an upper bound.
no_of_quadrature The number of quadrature, more specifically the number of bins the theta range should be divided. The more bins, the more precise (and slower) the estimates will be.
"sum_score" Simple sum score. Required parameters: NULL

ability_est_par
A list of length one or length maximum test length that sets the parameters of
ability estimation rules. It can also be NULL.
* If ability_est_rule = "eap" then the default is list(prior_dist = "norm", prior_par = list(mean = 0, sd = 2), min_theta = -4, max_theta = 4)
* If ability_est_rule = "owen" then the default is list(prior_mean = 0, prior_var = 1)
If it is NULL, either no parameters necessary for that ability estimation rule or the
defaults of that ability selection rule will be selected.
If it is a list of one, it means that the parameters will be the same throughout the
test. The names of the list elements will represent the parameter types.
A list of lists with length of maximum test length designate different parameters
for different items in the test progress.

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

final_ability_est_par
A list of parameters that will be used for the method designated by the
final_ability_est_rule.
Default: NULL

termination_rule
This parameter determines how CAT algorithm decides terminate the test.
The order of termination rules is important. The algorithm will check the rules
in that order. If for example termination_rule = c(’min_se’, ’max_item’),
first whether the SE smaller than a certain value checked and if it is smaller,
then even the maximum number of items haven’t been administered, test will
terminate.
The "min_item" and "max_item" has a special property where, for "min_item",
if the number of items administered smaller than min_item, then test will not
terminate regardless of whether other rules satisfied. Similarly, for "max_item", if
the number of items is larger than max_item, the test will terminate regardless of
whether other conditions satisfied or not. If both "min_item" and "max_item"
are in termination rules, then, test will end when both conditions satisfied, i.e.
when the number of items administered is equal to or larger than max_item value
in termination_par.
The "test length" refers to "Item" objects, i.e. individual items not testlets. For
example, if an item pool has 10 testlets each having 2 items and 15 standalone
items which are not within a testlet, then the test length can go up to 35 (2 x 10
+ 15).
Default: c("min_item", "min_se", "max_item")
"termination_rule" should be a vector that composed of the following termin-
ation 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 $\theta = \theta_0$

$H_1$: Examinee’s ability $\theta = \theta_1$

After the administration of each item, the likelihood (or log-likelihood) of the response string is calculated at $\theta_0$ and $\theta_1$. 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 $\alpha$ and $\beta$. $\alpha$ 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. $\beta$ 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))
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 expo-
sure control.
**Default:** NULL, No exposure control will be imposed on item selection.
Possible values and required parameters:

NULL  No exposure control.

*randonmesque*  Select one of the most informative num_items items.

num_items  The number of items to select from.

"sympson-hetter"  The algorithm of Sympon-Hetter exposure control is ex-
This method does not require any additional "exposure_control_par" but
each item/testlet should have a "misc" slot like the following misc = list(sympon_hetter_k
= .75).
When using 'sympson-hetter' exposure control rule, please ensure that there
are sufficient number of items with 'sympon_hetter_k' values 1. Other-
wise, 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

calendar_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 maxi-
mum 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.

calendar_bal_par
Parameters of calendar_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 Imple-
mented Yet).
create_cat_design

"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

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'),

cd =>

### 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'),

create_cat_design

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)
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"))
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 <- 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 ###
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))
cat_design(ability_est_rule = 'ml',
    ability_est_par = list(min_theta = -4, max_theta = 4, criterion = 0.01))

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

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

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

dif Evaluate Differential Item Functioning (DIF) of a test

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

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

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

Value
A data.frame of DIF values.

Author(s)
Emre Gonulates

distractor_analysis Distractor Analysis Function

Description
Distractor Analysis Function

Usage
distractor_analysis(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

```r
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)
```
est_ability

Estimate Ability of Examinees

Description

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

Usage

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

Arguments

resp A Response_set-class, matrix or a data.frame object that holds responses. If there are missing responses, they will not be included in the ability estimation.

ip An Item-class, Itempool-class or a Testlet-class object. The default value is NULL. If the value is NULL, resp should be a Response_set-class object with a valid Itempool-class.

method The method that will be used to estimate the ability. The default value is "eap". Current methods are:

'sum_score’ Basic sum (raw) score of responses.

'owen’ Owen’s Bayesian Ability Estimation.

This estimation method can be used only for dichotomous IRT models, 'Rasch', '1PL', '2PL', '3PL' and '4PL'. Formulas were implemented in Owen (1975) and Vale (1977). Original formulation does not contain D parameter. If D = 1 original solution will be obtained. If D = 1.7 the a parameter will be multiplied with this number. User needs to supply prior parameters, i.e. prior_pars. Prior parameters should be a numeric vector of length two. The first component is prior mean and the second component is prior standard deviation (note that it is NOT prior variance). So, for example, if the prior mean is 0.1 and prior variance is 4, 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
'map' 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. Currently following distributions can be
specified:
'norm' Normal distribution
'unif' Uniform distribution
't' t distribution
'cauchy' Cauchy distribution
Default value is 'norm'.

prior_pars Parameters of the prior distribution. Default value is c(0,1) where 0 is the mean
and 1 is the standard deviation of the default prior distribution which is normal
distribution. Also, for example, uniform prior parameter can be set as c(a,b)
where a is the minimum value and b is the maximum value. For t distribution,
prior parameter can be set as df to represent the degree of freedom. For Cauchy
distribution, prior parameters can be set as c(location, scale).
If method is "owen", provide c(<Prior Mean>,<Prior SD> ).

theta_range The limits of the ability estimation scale. The estimation result will be limited
to this interval. The default is c(-5,5).

number_of_quads Number of quadratures. The default value is 41. As this number increases, the
precision of the estimate will also increase. The default value is 41.

tol The precision level of ability estimate. The final ability estimates will be rounded
to remove the precision that is smaller than the to1 value. The default value is
1e-06.

Value

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

se The standard error(s) of the ability estimate(s). For "sum_score" method, all of the standard
effects will be NA. For Bayesian methods (like EAP 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

Research Report 77-4. Minneapolis, MN.
Examples

```r
ip <- generate_ip(n = 7)
resp <- simResp(ip, theta = rnorm(3))

### EAP estimation ###
estAbility(resp, ip)
estAbility(resp, ip, number_of_quads = 81)
# The default prior_dist is 'norm'. prior_pars = c(mean, sd)
estAbility(resp, ip, prior_pars = c(0, 3))
# prior_pars = c(min, max)
estAbility(resp, ip, prior_dist = 'unif', prior_pars = c(-3, 3))
# prior_pars = c(df)
estAbility(resp, ip, prior_dist = 't', prior_pars = 3)
# prior_pars = c(location, scale)
estAbility(resp, ip, prior_dist = 'cauchy', prior_pars = c(0, 1))

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

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

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

Description

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

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

```r
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 `Response_set-class` object. When the data is not necessary, i.e. user only wants to read the BILOG-MG output from the `target_dir`, then this can be set to `NULL`.

- **model**: The model of the items. The value is one of the following:
  
  - "1PL" One-parameter logistic model.
  - "2PL" Two-parameter logistic model.
  - "3PL" Three-parameter logistic model.
  - "CTT" Return only Classical Test theory statistics such as p-values, point-biserial and biserial correlations.

  The default value is "3PL".

- **target_dir**: The directory/folder where the BILOG-MG analysis and data files will be saved. The default value is the current working directory, i.e. `getwd()`.

- **analysis_name**: A short file name that will be used for the data files created for the analysis.

- **items**: A vector of column names or numbers of the x that represents the responses. If, in the syntax file, no entry for item names are desired, then, simply write `items = "none"`. 
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. Ideally, the grouping variable is represented by single digit integers. If other type of data provided, an integer value will automatically assigned to the variables. The default value is NULL, where no multi-group analysis will be performed.

logistic

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

num_of_alternatives

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

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

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

criterion

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

num_of_quadrature

The number of quadrature points in MML estimation. The default value is 81.

This value will be represented in BILOG-MG control file as: NQPT = num_of_quadrature.

The BILOG-MG default value is 20 if there are more than one group, 10 otherwise.

max_em_cycles

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

The default value is 100.

newton

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

reference_group

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

The default value is NULL.

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

fix

This arguments helps to specify whether the parameters of specific items are free to be estimated or are to 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, examples section for a demonstration.
scoring_options
A string vector of keywords/options that will be added to the SCORE section in BILOG-MG syntax. Set the value of scoring_options to NULL if scoring of individual examinees is not necessary.
The default value is c("METHOD=1","NOPRINT") where scale scores will be estimated using Maximum Likelihood estimation and the scoring process will not be printed to the R console (if show_output_on_console = TRUE).
The main option to be added to this vector is "METHOD=n". Following options are available:
"METHOD=1" Maximum Likelihood (ML)
"METHOD=2" Expected a Posteriori (EAP)
"METHOD=3" Maximum a Posteriori (MAP)
In addition to "METHOD=n" keyword, following keywords can be added:
"NOPRINT": Suppresses the display of the scores on the R console.
"FIT": likelihood ratio chi-square goodness-of-fit statistic for each response pattern will be computed.
"NQPT=(list)", "IDIST=n", "PMN=(list)", "PSD=(list)", "RSCTYPE=n", "LOCATION=(list)",
"SCALE=(list)", "INFO=n", "BIWEIGHT", "YCOMMON", "POPC", "MOMENTS", "FILE",
"READF", "REFERENCE=n", "INFORM=n"
See BILOG-MG manual for more details about these keywords/options.

prior_ability
Prior ability is the quadrature points and weights of the discrete finite representations of the prior distribution for the groups. It should be a list in the following form:
list(<GROUP-NAME-1> = list(points = ....,weights = ...),<GROUP-NAME-2>
= list(points = ....,weights = ...),....)
GROUP-NAME-1 is the name of the first group, GROUP-NAME-2 is the name of the second group, etc. See examples section for an example implementation.

prior_ip

Specify priors distributions for item parameters. The default value is NULL, where BILOG-MG defaults will be used. In order to specify priors, a list of one or more of the following elements needs to be provided:

"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 BILOG-MG manual.

Here are couple examples:

\[
\text{list(ALPHA = 4, BETA = 3, SMU = 1, SSIGMA = 1.648, TMU = 0, TSIGMA = 2)}
\]

A very strong prior for guessing which almost fixes all guessing parameters at 0.2:

\[
\text{list(ALPHA = 1000000, BETA = 4000000)}
\]

Fix guessing at 0.25:

\[
\text{list(ALPHA = 1000000, BETA = 3000000)}
\]

More generally, one can play with the alpha and beta parameters to obtain desired number considering the mode of beta distribution is:

\[
\text{mode} = \frac{\alpha - 1}{\alpha + \beta - 2}
\]

Also, one can set SSIGMA or TSIGMA to a very small value to effectively fix the item parameters, for example set TSIGMA = 0.005 or SSIGMA = 0.001 to effectively fix those item parameters. Note that there might be convergence issues with these restrictions.

Note that a non-null prior_ip value will automatically add READPRIOR option to CALIB section.

overwrite

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

show_output_on_console

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

bilog_exe_folder

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

Value

A list of following objects:
"ip" An Itempool-class object holding the item parameters. Please check whether model converged (using .$converged) before interpreting/using ip. This element will not be created when model = "CTT".

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

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

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

"syntax" The syntax file.

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

"cycle" Number of cycles run before calibration converge or fail to converge.

"largest_change" Largest change between the last two cycles.

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

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

Author(s)
Emre Gonulates

Examples

```r
## Not run:
#############################################
############## Example 1 - 2PL ##############
#############################################
# IRT Two-parameter Logistic Model Calibration
# Create responses to be used in BILOG-MG estimation
ture_theta <- rnorm(4000)
ture_ip <- generate_ip(n = 30, model = "2PL")
resp <- sim_resp(ture_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)

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)

# 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)

# 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 2PL 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)

bilog_calib$ip

# 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:

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

```r
ip <- generate_ip(n = 35, model = "3PL", D = 1.7)
n_upper <- sample(1200:3000, 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 ##

```r
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

```r
## Generate Data ##

```r
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 ##

```r
```
est_bilog

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)
# 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),
                 mg_calib$prior_ability$weights[
                 mg_calib$group_info$group == "Lower")
  )

mg_calib$group_info$lower$lower
est_bilog

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 Pars without BILOG-MG ##
##########################################################################
# When user wants to read BILOG-MG output saved in the directory "Analysis/
# with file names "my_analysis.PH1", "my_analysis.PH2", etc.,
use the following syntax to read Bilog output files without running the
# calibration:
# (The following code does not require an installed BILOG-MG program on the
# computer.)
result <- est_bilog(target_dir = file.path("Analysis/"), model = "3PL",
    analysis_name = "my_analysis", overwrite = FALSE)

##########################################################################
## Example 8 - Fixed Item Parameters ##
##########################################################################
# The idea is to fix individual item parameters to certain values.
# If all of values of a certain item parameter(s) need to be fixed,
# then, strong priors can also be used. See the documentation for
# "prior_ip" argument.

# Create responses to be used in BILOG-MG estimation
true_theta <- rnorm(3000)
ttrue_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)
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") # Reparametrized GPCM

# Mixture of models
generate_ip(model = c("4PL", "Rasch"))
generate_ip(model = sample(c("4PL", "GPCM"), 12, TRUE))
generate_ip(model = c("2PL", "GRM", "Rasch"), n = 11)

# Generate parameters standard errors for each item
generate_ip(se_parameters = 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

```r
# 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)
```

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.
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") # Reparametrized GPCM
# Mixture of models
generate_testlet(item_models = c("4PL", "Rasch"))
generate_testlet(model = c("2PL", "GRM", "Rasch"), n = 11)

# Generating multiple testlet objects with custom ids
sapply(paste0("testlet-", 1:4), function(x) generate_testlet(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 gives either a vector (if there is a single cat_output object) or a matrix (if there is a list of cat_output objects) of response data.

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

Usage

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

Arguments

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

cd  A cat_design object that is created by function create_cat_design.

remove_na  If TRUE, the columns that are all NA will be removed.

attach_summary  If TRUE, the summary of each CAT will be attached to the beginning of the
                 response string as columns. The default value is FALSE.

Value

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

Author(s)

Emre Gonulates

See Also

cat_sim

Examples

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

Calculate the maximum score of a set of items

Description

Calculate the maximum score of a set of items

Usage

get_max_possible_total_score(ip, resp = NULL)

Arguments

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

Value

A vector of numbers showing the maximum possible scores.

Author(s)

Emre Gonulates

Examples

```r
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

*Reparametrized Generalized Partial Credit Model*

**Description**

Reparametrized 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
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)
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 inform-
           ation.
 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_item(model = "3PL"), theta = rnorm(5))
info(ip = generate_item(model = "GRM"), theta = rnorm(7))

info(ip = generate_item(model = "3PL"), theta = rnorm(5), tif = TRUE)
info(ip = generate_item(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)

## # Test information function value at theta

## # 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)

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

# 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)

# Information values of an item pool with multiple models
t1 <- generate_testlet(item_models = c("2PL", "3PL", "GRM", "3PL", "GRM"))
theta <- rnorm(sample(6:10, 1))
info(ip = t1, theta = theta[1])
info(ip = t1, theta = theta)
info(ip = t1, theta = theta, tif = TRUE)

ipd

Item Parameter Drift

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

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

Arguments
ip1 Itempool object for the first calibration.
ip2 Itempool object for the second calibration.
method The method of item parameter drift analysis.
anchor_item_ids Anchor item ids. If NULL, all items are assumed to be anchor items.
alpha Two tailed critical value to detect the unstable items. For example if

\[ \alpha = 0.05 \]

, the critical value is calculated using \( qnorm(1-\alpha/2) \) (= 1.96). Items whose absolute robust z values are larger than this number will be flagged as unstable.

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

Value
Return a list depending on the method:

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

Author(s)
Emre Gonulates
References

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

Examples

# The example from Huynh and Meyer (2010)

```r
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, 0.688, 1.177,
         0.900, 0.861, 0.843, 1.404, 0.446, 1.014, 1.632, 0.831, 1.560, 0.790),
  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, 0.164, 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, 0.699, 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, 1.000, 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)
```
A Collection of Item Response Theory (IRT) and Computerized Adaptive Testing (CAT) Functions

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

Author(s)
Emre Gonulates <egonulates@gmail.com>

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

Description
Check whether an object is an Item-class
Check whether an object is an Itempool-class object
Check whether an object is a Testlet-class object

Usage
is.Item(x)
is.Itempool(x)
is.Testlet(x)

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

Author(s)
Emre Gonulates
Emre Gonulates
Emre Gonulates
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 = "i1", 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 (Reparametrized GPCM model)
item(a = 1.9, b = 0.65, d = c(-1.6, -0.09, 1.25), model = "GPCM2")
item(parameters = list(a = 1.9, b = 0.65, d = c(-1.6, -0.09, 1.25), D = 1.7),
    model = "GPCM2")

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

# Add additional arguments to items
i1 <- item(a = 1.2, b = 2)
i1 <- item(i1, 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 \( \theta \):

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

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 $\theta$:

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

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 $\theta$:

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

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 $\theta$:

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

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 $\theta$:

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

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 $0, \ldots, m$:

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

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

$$\cdots$$

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

$$\cdots$$

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

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:

'n' 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


---

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

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

# Add 'misc' field:
ip <- itempool(b = rnorm(2), 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

Description

Item Analysis Function

Usage

item_analysis(resp, criterion = NULL, ip = NULL, 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.
suppress_output  If TRUE, the function will suppress console output. Default value is FALSE

Value

A list of

'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.
'pbis'  Point biserial correlation.
'bis'  Biserial correlation.
'pbis_adj'  Point biserial correlation between item and total score without this item.
'bis_adj'  Biserial correlation between item and total score without this item.

Author(s)
Emre Gonulates

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

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"  PARSACLE’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


Examples

```r
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)
```

ks

Item Characteristic Curve Estimation using Kernel Smoothing

Description

Item Characteristic Curve Estimation using Kernel Smoothing

Usage

```r
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 `1.06σ(criterion)\cdot n^{-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 \( \text{points} = \text{seq}(-3, 3, 0.05) \).

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

```r
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[1], theta = kern_output$points)
p <- sapply(p, `\[`, 2) # get the probability of correct responses
```
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

\texttt{x} \quad \text{an \texttt{Response_set-class} object}

Author(s)

Emre Gonulates  
Emre Gonulates  
Emre Gonulates  

Examples

\begin{verbatim}
r <- response(sample(0:1, 22, TRUE))
length(r)
\end{verbatim}

---

\textbf{M2PL-class \quad Multidimensional Two-Parameter Logistic Model}

Description

Multidimensional Two-Parameter Logistic Model

Slots

\begin{itemize}
  \item \texttt{a} \quad \text{Slope Parameters}
  \item \texttt{d} \quad \text{Intercept Parameter}
  \item \texttt{D} \quad \text{Scaling constant}
  \item \texttt{se_a} \quad \text{Standard errors of slope parameters}
  \item \texttt{se_d} \quad \text{Standard error of intercept parameter}
\end{itemize}

Author(s)

Emre Gonulates
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**

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

## S4 method for signature 'Item'

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

## S4 method for signature 'Itempool'

```r
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. $\theta$.

**Usage**

```r
## 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

```r
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)
```

Description

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

Usage

```r
## 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

```r
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. $\theta$.

Usage

```r
## 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

```r
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**

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

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

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

```r
## 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.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
```r
## 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 NULL to remove it.

- **suppress_plot**  
  If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE. 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

```r
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

Description

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

Usage

```r
## 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,
  ...
)
```

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. Default is NULL. If `tcc` is TRUE it will be 'Test Characteristic Curve', if FALSE it will be 'Item Characteristic Curve'.
- `suppress_plot` If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.
plot.Itempool

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

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.ks_output

# 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**

```r
## S3 method for class 'ks_output'
plot(
  x,
  item_no,
  ip = NULL,
  title = NULL,
  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.
- **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_distractor_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

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 empirical 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

```r
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.
plot_empirical_icc2

<table>
<thead>
<tr>
<th>bins</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>binwidth</td>
<td>This determines the width of each bin of the theta scale. Within each bin, there might be different number of examinees.</td>
</tr>
<tr>
<td>title</td>
<td>Title of the plot</td>
</tr>
<tr>
<td>suppress_plot</td>
<td>If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.</td>
</tr>
<tr>
<td>base_r_graph</td>
<td>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.</td>
</tr>
<tr>
<td>...</td>
<td>Extra parameters that will pass to geom_line.</td>
</tr>
</tbody>
</table>

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

```r
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_empirical_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
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.

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

```r
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**

```r
plot_info(ip,
```

```r
```
plot_info

tif = FALSE,
theta_range = c(-5, 5),
focus_item = NULL,
title = "",
suppress_plot = FALSE,
base_r_graph = FALSE,
...)

Arguments

ip        An Item-class or Itempool-class object.
tif       If TRUE a test information plot will be plotted. The default value is FALSE.
theta_range 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
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 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)

---

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

### Description

plot_resp_loglik plots the log-likelihood of a response string.

### Usage

```r
plotRespLoglik(
  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
### plot_resp_loglik

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>likelihood</td>
<td>If TRUE, likelihood function will be plotted instead of log-likelihood graph. Default value is FALSE.</td>
</tr>
<tr>
<td>show_estimate</td>
<td>If TRUE the maximum likelihood ability estimate will be shown. The default value is TRUE.</td>
</tr>
<tr>
<td>base_r_graph</td>
<td>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.</td>
</tr>
<tr>
<td>suppress_plot</td>
<td>If FALSE the function will print the plot. If TRUE, function will return the plot object. Default value is FALSE.</td>
</tr>
<tr>
<td>text_size</td>
<td>The overall text size of the axis and titles. The default value is 12.</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments passed to annotate.</td>
</tr>
</tbody>
</table>

### 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

```r
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)
```
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. $\theta$. For polytomous models, where there are multiple possible responses, probability of each response category will be returned.

**Usage**

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

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

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

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

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

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

## S4 method for signature `\'4PL\'`
```r
prob(ip, theta, derivative = 0)
```

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

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

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

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

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

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

```r
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)

# 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)

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)

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**

```r
response(
  score = NULL,
  examinee_id = NULL,
  item_id = NULL,
  raw_response = NULL,
  testlet_id = NULL,
  order = NULL,
  response_time = NULL,
  misc = NULL
)
```
Response-class

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>score</td>
<td>A numeric vector holding the scores given to items.</td>
</tr>
<tr>
<td>examinee_id</td>
<td>Examinee/Subject/Student ID. A character string to identify an examinee.</td>
</tr>
<tr>
<td>item_id</td>
<td>A character vector holding the item IDs.</td>
</tr>
<tr>
<td>raw_response</td>
<td>A vector of strings holding the raw responses to items.</td>
</tr>
<tr>
<td>testlet_id</td>
<td>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.</td>
</tr>
<tr>
<td>order</td>
<td>An integer vector representing the administration order of an item.</td>
</tr>
<tr>
<td>response_time</td>
<td>A numeric vector representing the response times. By default, numbers are assumed to represent seconds.</td>
</tr>
<tr>
<td>misc</td>
<td>A list that will hold miscellaneous information about the responses. For example, codemisc = list(item_role = c(&quot;O&quot;, &quot;O&quot;, &quot;O&quot;,&quot;F&quot;)) will hold whether administered item is a field test or an operational test item.</td>
</tr>
</tbody>
</table>

Author(s)

Emre Gonulates

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, codemisc = 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
Description

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

Usage

```r
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 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

```r
x_wide <- matrix(sample(0:1, 35, TRUE), nrow = 7, ncol = 5)
response_set(x_wide)
```

### Example 2

```r
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 #####

```r
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")
)
```

```r
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")
)
```

```r
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.

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

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**

```r
call_for_item
resp_lik(ip, resp, theta)
call_for_itempool
resp_lik(ip, resp, theta)
call_for_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**

```r
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)

## 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

### Description
resp_loglik returns the log-likelihood of a response string for given items and ability.
Value
A matrix of log-likelihood(s)

Author(s)
Emre Gonulates

Examples
```r
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)

test <- generate_item(model = "GRM")
resp <- sim_resp(ip = test, theta = theta, prop_missing = .1)
resp_loglik(ip = test, resp = resp, theta = theta)
	normal <- generate_ip(model = "3PL")
theta <- rnorm(6)
resp <- sim_resp(ip = normal, theta = theta, prop_missing = .1)
resp_loglik(ip = normal, resp = resp, theta = theta, derivative = 1)
resp_loglik(ip = normal, resp = resp, theta = theta, derivative = 2)
	normal <- generate_ip(model = "GPCM")
resp <- sim_resp(ip = normal, theta = theta, prop_missing = .1)
resp_loglik(ip = normal, resp = resp, theta = theta, derivative = 1)
resp_loglik(ip = normal, 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**
```r
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

---

**Calculate Score Information Function**

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**

```r
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


Examples

```r
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**

```r
sim_resp(ip, theta, prop_missing = 0, output = "matrix")
```

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

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

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

# S4 method for signature 'numMatDfListChar'
```r
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

```r
## 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
simResp(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

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

### Arguments

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

### Value

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

### Author(s)

Emre Gonulates

### See Also

- `cat_sim`
Examples

```r
n <- 100 # number of items
ip <- generate_ip(n = n,
    content = sample(c("Algebra", "Arithmetic", "Geometry"),
    n, replace = TRUE))

cd <- create_cat_design(ip = ip, next_item_rule = 'mfi',
    termination_rule = 'max_item',
    termination_par = list(max_item = 10))

cat_data <- cat_sim(true_ability = rnorm(5), cd = cd)
summary(cat_data)
```

---

`testlet`  
*Creates a Testlet-class object*

**Description**

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

**Usage**

`testlet(...)`

**Arguments**

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

**Value**

An `Testlet-class` object.

**Author(s)**

Emre Gonulates

**Examples**

```r
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 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 a testlet 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. $\theta$. 
**Usage**

```r
## 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. $\theta$.

**Usage**

```r
## 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**

```r
## 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

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

```r
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.
'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"))
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.
  - '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
\$\.Response\_set-method

**Examples**

```r
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))
)
```

```r
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**

```r
## 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

```r
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

```r
## S4 method for signature 'Testlet'
x$\text{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.
  - `'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.
$\leftarrow$, Item-method

Examples

t1 <- testlet(generate_ip(n = 3), testlet_id = "my-testlet", content = "Algebra")
t1$model
t1@testlet_id
t1$item_list
t1$item_models
t1$item_id
t1$content
t1$item_models

$\leftarrow$, 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.
$<-,Response-method

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

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

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.
$\leftarrow$.Response_set-method

... 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

```r
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
```

$\leftarrow$.Response_set-method

Set values to components of 'Response_set' class objects

Description

Set values to components of `Response_set` class objects

Usage

```r
## 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 'Item' class.

Description

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

Usage

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

Arguments

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

Value

This operation will not return anything.

Author(s)

Emre Gonulates
Index

* package
  irt, 62
  $, Item-method, 119
  $, Itempool-method, 120
  $, Response-method, 122
  $, Response_set-method, 123
  $, Testlet-method, 124
  $<-, Item-method, 125
  $<-, Itempool-method, 126
  $<-, Response-method, 127
  $<-, Response_set-method, 128
  $<-, Testlet-method, 129
  1PL-class, 4
  2PL-class, 4
  3PL-class, 5
  4PL-class, 5

  add_misc, 6
  add_misc, Item-method(add_misc), 6
  add_misc, Itempool-method(add_misc), 6
  add_misc, Testlet-method(add_misc), 6
  as.data.frame.GPCM
    (as.data.frame.Item), 7
  as.data.frame.GPCM2
    (as.data.frame.Item), 7
  as.data.frame.GRM(as.data.frame.Item),
    7
  as.data.frame.Item, 7
  as.data.frame.Itempool
    (as.data.frame.Item), 7
  as.data.frame.M2PL
    (as.data.frame.Item), 7
  as.data.frame.M3PL
    (as.data.frame.Item), 7
  as.data.frame.PCM(as.data.frame.Item),
    7
  as.data.frame.Response, 9
  as.data.frame.Response_set, 11
  as.data.frame.Testlet
    (as.data.frame.Item), 7
  as.Itempool, 11
  as.list.Itempool, 12
  as.list.Response_set, 13
  as.matrix, Response_set-method, 13
  biserial, 15
  c, Item-method, 16
  c, Itempool-method(c, Item-method), 16
  c, Response-method(c, Item-method), 16
  c, Response_set-method(c, Item-method),
    16
  c, Testlet-method(c, Item-method), 16
  calculate_exposure_rates, 17
  calculate_overlap_rates, 18
  cat_sim, 18, 19, 20, 21, 28, 54, 114
  cat_sim_fast, 19, 20, 21
  create_cat_design, 20, 21, 21
  dif, 30
  distractor_analysis, 30
  est_ability, 32
  est_bilog, 34
  generate_ip, 47
  generate_item, 48
  generate RESP, 49
  generate Resp set, 50
  generate testlet, 51
  get_cat_administered_items, 53
  get_cat_response_data, 53
  get_max_possible_total_score, 55
  GPCM-class, 56
  GPCM2-class, 56
  GRM-class, 57
  info, 57
  info, 1PL-method (info), 57
  info, 2PL-method (info), 57
  info, 3PL-method (info), 57
INDEX

info, 4PL-method (info), 57
info, GPCM-method (info), 57
info, GPCM2-method (info), 57
info, GRM-method (info), 57
info, Item-method (info), 57
info, Itempool-method (info), 57
info, numMatDfListChar-method (info), 57
info, PCM-method (info), 57
info, Rasch-method (info), 57
info, Testlet-method (info), 57
ipd, 60
irt, 62
is.Item, 62
is.Itempool (is.Item), 62
is.Testlet (is.Item), 62
item, 63
Item-class, 6, 7, 62, 65, 119, 125
item_analysis, 70
item_fit, 71
itempool, 11, 12, 68
Itempool-class, 6, 11, 69, 75, 120
ks, 73
length, Itempool-method, 75
length, Response-method
  (length, Itempool-method), 75
length, Response_set-method
  (length, Itempool-method), 75
length, Testlet-method
  (length, Itempool-method), 75
M2PL-class, 76
M3PL-class, 77
max_score, 77
max_score, Item-method (max_score), 77
max_score, Itempool-method (max_score), 77
mean, 1PL-method (mean, Item-method), 78
mean, 2PL-method (mean, Item-method), 78
mean, 3PL-method (mean, Item-method), 78
mean, 4PL-method (mean, Item-method), 78
mean, GPCM-method (mean, Item-method), 78
mean, GPCM2-method (mean, Item-method), 78
mean, GRM-method (mean, Item-method), 78
mean, Item-method, 78
mean, Itempool-method, 79
mean, PCM-method (mean, Item-method), 78
mean, Rasch-method (mean, Item-method), 78
mean, Testlet-method, 80
PCM-class, 81
person_fit, 81
person_fit, ANY, Itempool-method
  (person_fit), 81
person_fit, ANY, Testlet-method
  (person_fit), 81
person_fit, Response_set, Itempool-method
  (person_fit), 81
plot.Item, 82
plot.Itempool, 84
plot.ks_output, 86
plot_distractor ICC, 87
plot_empirical ICC, 89
plot_empirical ICC2, 90
plot_info, 92
plot_resp_loglik, 94
prob, 96
prob, 1PL-method (prob), 96
prob, 2PL-method (prob), 96
prob, 3PL-method (prob), 96
prob, 4PL-method (prob), 96
prob, GPCM-method (prob), 96
prob, GPCM2-method (prob), 96
prob, GRM-method (prob), 96
prob, Item-method (prob), 96
prob, Itempool-method (prob), 96
prob, numMatDfListChar-method (prob), 96
prob, PCM-method (prob), 96
prob, Rasch-method (prob), 96
prob, Testlet-method (prob), 96
Rasch-class, 102
resp_lik, 108
resp_lik, Item-method (resp_lik), 108
resp_lik, Itempool-method (resp_lik), 108
resp_lik, Testlet-method (resp_lik), 108
resp_loglik, 109
resp_loglik, Item, ANY-method
  (resp_loglik), 109
resp_loglik, Itempool, ANY-method
  (resp_loglik), 109
resp_loglik, Itempool, Response-method
  (resp_loglik), 109
resp_loglik, Itempool, Response_set-method
  (resp_loglik), 109
resp_loglik, numMatDfListChar, ANY-method
  (resp_loglik), 109
resp_loglik, Testlet, ANY-method
  (resp_loglik), 109
response, 102
Response-class, 9, 103, 122
response_set, 104
Response_set-class, 11, 13, 104, 107, 123
rsss, 110

score_info, 111
sim_resp, 112
  sim_resp, Item-method (sim_resp), 112
  sim_resp, Itempool-method (sim_resp), 112
  sim_resp, numMatDflistChar-method (sim_resp), 112
  sim_resp, Testlet-method (sim_resp), 112
summary.cat_output, 114
testlet, 115
Testlet-class, 6, 115, 116, 124

var, 1PL-method (var, Item-method), 116
var, 2PL-method (var, Item-method), 116
var, 3PL-method (var, Item-method), 116
var, 4PL-method (var, Item-method), 116
var, GPCM-method (var, Item-method), 116
var, GPCM2-method (var, Item-method), 116
var, GRM-method (var, Item-method), 116
var, Item-method, 116
var, Itempool-method, 118
var, PCM-method (var, Item-method), 116
var, Rasch-method (var, Item-method), 116
var, Testlet-method, 118