Package 'xxIRT'

October 14, 2022

Type Package

Title Item Response Theory and Computer-Based Testing in R

Version 2.1.2

Date 2019-3-21

Author Xiao Luo [aut, cre]

Maintainer Xiao Luo <xluo1986@gmail.com>

Description A suite of psychometric analysis tools for research and operation, including:
(1) computation of probability, information, and likelihood for the 3PL, GPCM, and GRM;
(2) parameter estimation using joint or marginal likelihood estimation method;
(3) simulation of computerized adaptive testing using built-in or customized algorithms;
(4) assembly and simulation of multistage testing.
The full documentation and tutorials are at <https://github.com/xluo11/xxIRT>.

License GPL (>= 3)

Depends R (>= 3.5.0)

URL https://github.com/xluo11/xxIRT

BugReports https://github.com/xluo11/xxIRT/issues

Imports ggplot2, glpkAPI, lpSolveAPI, reshape2, stats

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2019-03-22 10:00:03 UTC

R topics documented:

ata 2
cat_sim 5
cronbach_alpha 8
expected_raw_score_dist 8
freq 9

model_3pl 9
model_gpcm 11
model_grm 13
mst_sim 14
rmse 15
spearman_brown 16

Index **17**

`ata` *Automated Test Assembly (ATA)*

Description

`ata` initiates an ATA model

`ata_obj_relative` adds a relative objective to the model

`ata_obj_absolute` adds an absolute objective to the model

`ata_constraint` adds a constraint to the model

`ata_item_use` limits the minimum and maximum usage for items

`ata_item_enemy` adds an enemy-item constraint to the model

`ata_item_fixedvalue` forces an item to be selected or not selected

`ata_solve` solves the MIP model

Usage

```
ata(pool, num_form = 1, len = NULL, max_use = NULL, ...)

## S3 method for class 'ata'
print(x, ...)

## S3 method for class 'ata'
plot(x, ...)

ata_obj_relative(x, coef, mode = c("max", "min"), tol = NULL,
  negative = FALSE, forms = NULL, collapse = FALSE,
  internal_index = FALSE, ...)

ata_obj_absolute(x, coef, target, equal_tol = FALSE, tol_up = NULL,
  tol_down = NULL, forms = NULL, collapse = FALSE,
  internal_index = FALSE, ...)

ata_constraint(x, coef, min = NA, max = NA, level = NULL,
  forms = NULL, collapse = FALSE, internal_index = FALSE)

ata_item_use(x, min = NA, max = NA, items = NULL)
```

```
ata_item_enemy(x, items)

ata_item_fixedvalue(x, items, min = NA, max = NA, forms)

ata_solve(x, solver = c("lpsolve", "glpk"), as.list = TRUE,
  details = TRUE, time_limit = 10, message = FALSE, ...)
```

Arguments

pool	item pool, a data.frame
num_form	number of forms to be assembled
len	test length of each form
max_use	maximum use of each item
...	options, e.g. group, common_items, overlap_items
x	an ATA object
coef	coefficients of the objective function
mode	optimization mode: 'max' for maximization and 'min' for minimization
tol	the tolerance paraemter
negative	TRUE when the objective function is expected to be negative
forms	forms where objectives are added. NULL for all forms
collapse	TRUE to collapse into one objective function
internal_index	TRUE to use internal form indices
target	the target values of the objective function
equal_tol	TRUE to force upward and downward tolerance to be equal
tol_up	the range of upward tolerance
tol_down	the range of downward tolerance
min	the lower bound of the constraint
max	the upper bound of the constraint
level	the level of a categorical variable to be constrained
items	a vector of item indices, NULL for all items
solver	use 'lpsolve' for lp_solve 5.5 or 'glpk' for GLPK
as.list	TRUE to return results in a list; otherwise, a data frame
details	TRUE to print detailed information
time_limit	the time limit in seconds passed along to solvers
message	TRUE to print messages from solvers

Details

The ATA model stores the definition of a MIP model. `ata_solve` converts the model definition to a real MIP object and attempts to solve it.

`ata_obj_relative`: when mode='max', maximize (y-tol), subject to y <= sum(x) <= y+tol; when mode='min', minimize (y+tol), subject to y-tol <= sum(x) <= y. When `negative` is `TRUE`, y < 0, tol > 0. `coef` can be a numeric vector that has the same length with the pool or forms, or a variable name in the pool, or a numeric vector of theta points. When `tol` is `NULL`, it is optimized; when `FALSE`, ignored; when a number, fixed; when a range, constrained with lower and upper bounds.

`ata_obj_absolute` minimizes y0+y1 subject to t-y0 <= sum(x) <= t+y1.

When `level` is `NA`, it is assumed that the constraint is on a quantitative item property; otherwise, a categorical item property. `coef` can be a variable name, a constant, or a numeric vector that has the same size as the pool.

`ata_solve` takes control options in `...`. For lpsolve, see `lpSolveAPI::lp.control.options`. For glpk, see `glpkAPI::glpkConstants`
Once the model is solved, additional data are added to the model. `status` shows the status of the solution, `optimum` the optimal value of the objective fucntion found in the solution, `obj_vars` the values of two critical variables in the objective function, `result` the assembly results in a binary matrix, and `items` the assembled items

Examples

```
## Not run:
## generate a pool of 100 items
n_items <- 100
pool <- with(model_3pl_gendata(1, nitems), data.frame(id=1:n_items, a=a, b=b, c=c))
pool$content <- sample(1:3, n_items, replace=TRUE)
pool$time <- round(rlnorm(n_items, log(60), .2))
pool$group <- sort(sample(1:round(n_items/3), n_items, replace=TRUE))

## ex. 1: four 10-item forms, maximize b parameter
x <- ata(pool, 4, len=10, max_use=1)
x <- ata_obj_relative(x, "b", "max")
x <- ata_solve(x, timeout=5)
data.frame(form=1:4, b=sapply(x$items, function(x) mean(x$b)))

## ex. 2: four 10-item forms, minimize b parameter
x <- ata(pool, 4, len=10, max_use=1)
x <- ata_obj_relative(x, "b", "min", negative=TRUE)
x <- ata_solve(x, as.list=FALSE, timeout=5)
with(x$items, aggregate(b, by=list(form=form), mean))

## ex. 3: two 10-item forms, mean(b)=0, sd(b)=1
## content = (3, 3, 4), avg. time = 58--62 seconds
constr <- data.frame(name='content',level=1:3, min=c(3,3,4), max=c(3,3,4), stringsAsFactors=F)
constr <- rbind(constr, c('time', NA, 58*10, 62*10))
x <- ata(pool, 2, len=10, max_use=1)
x <- ata_obj_absolute(x, pool$b, 0*10)
x <- ata_obj_absolute(x, (pool$b-0)^2, 1*10)
for(i in 1:nrow(constr))
```

```
  x <- with(constr, ata_constraint(x, name[i], min[i], max[i], level=level[i]))
x <- ata_solve(x, timeout=5)
sapply(x$items, function(x) c(mean=mean(x$b), sd=sd(x$b)))

## ex. 4: two 10-item forms, max TIF over (-1, 1), consider item sets
x <- ata(pool, 2, len=10, max_use=1, group="group")
x <- ata_obj_relative(x, seq(-1, 1, .5), 'max')
x <- ata_solve(x, timeout=5)
plot(x)

## End(Not run)
```

---

cat_sim *Simulation of Computerized Adaptive Testing (CAT)*

Description

cat_sim runs a simulation of CAT. Use theta in options to set the starting value of theta estimate.

cat_estimate_mle is the maximum likelihood estimation rule. Use map_len to apply MAP to the first K items and use map_prior to set the prior for MAP.

cat_estimate_eap is the expected a posteriori estimation rule, using eap_mean and eap_sd option parameters as the prior

cat_estimate_hybrid is a hybrid estimation rule, which uses MLE for mixed responses and EAP for all 1's or 0's responses

cat_stop_default is a three-way stopping rule. When stop_se is set in the options, it uses the standard error stopping rule. When stop_mi is set in the options, it uses the minimum information stopping rule. When stop_cut is set in the options, it uses the confidence interval (set by ci_width) stopping rule.

cat_select_maxinfo is the maximum information selection rule. Use group (a numeric vector) to group items belonging to the same set. Use info_random to implement the random-esque item exposure control method.

cat_select_ccat is the constrained CAT selection rule. Use ccat_var to set the content variable in the pool. Use ccat_perc to set the desired content distribution, with the name of each element being the content code and tue value of each element being the percentage. Use ccat_random to add randomness to initial item selections.

cat_select_shadow is the shadow-test selection rule. Use shadow_id to group item sets. Use constraints to set constraints. Constraints should be in a data.frame with four columns: var (variable name), level (variable level, NA for quantitative variable), min (lower bound), and max (upper bound).

cat_stop_projection is the projection-based stopping rule. Use projection_method to choose the projection method ('info' or 'diff'). Use stop_cut to set the cut score. Use constraints to set the constraints. Constraints should be a data.frame with columns: var (variable name), level (variable level, NA for quantitative varialbe), min (lower bound), max (upper bound)

Usage

```
cat_sim(true, pool, ...)

cat_estimate_mle(len, theta, stats, admin, pool, opts)

cat_estimate_eap(len, theta, stats, admin, pool, opts)

cat_estimate_hybrid(len, theta, stats, admin, pool, opts)

cat_stop_default(len, theta, stats, admin, pool, opts)

cat_select_maxinfo(len, theta, stats, admin, pool, opts)

cat_select_ccat(len, theta, stats, admin, pool, opts)

cat_select_shadow(len, theta, stats, admin, pool, opts)

## S3 method for class 'cat'
print(x, ...)

## S3 method for class 'cat'
plot(x, ...)

cat_stop_projection(len, theta, stats, admin, pool, opts)
```

Arguments

`true`	the true theta
`pool`	the item pool (data.frame)
`...`	option/control parameters
`len`	the current test length
`theta`	the current theta estimate
`stats`	a matrix of responses, theta estimate, information and std error
`admin`	a data frame of administered items
`opts`	a list of option/control parameters
`x`	a `cat` object

Details

`...` takes a variety of option/control parameters for the simulations from users. `min` and max are mandatory for setting limits on the test length. User-defined selection, estimation, and stopping rules are also passed to the simulator via options.
To write a new rule, the function siganiture must be: `function(len, theta, stats, admin, pool, opts)`. See built-in rules for examples.

Value

`cat_sim` returns a `cat` object

an estimation rule should return a theta estimate

a stopping rule should return a boolean: `TRUE` to stop the CAT, `FALSE` to continue

a selection rule should return a list of (a) the selected item and (b) the updated pool

Examples

```
## Not run:
## generate a 100-item pool
num_items <- 100
pool <- with(model_3pl_gendata(1, num_items), data.frame(a=a, b=b, c=c))
pool$set_id <- sample(1:30, num_items, replace=TRUE)
pool$content <- sample(1:3, num_items, replace=TRUE)
pool$time <- round(rlnorm(num_items, mean=4.1, sd=.2))

## MLE, EAP, and hybrid estimation rule
cat_sim(1.0, pool, min=10, max=20, estimate_rule=cat_estimate_mle)
cat_sim(1.0, pool, min=10, max=20, estimate_rule=cat_estimate_eap)
cat_sim(1.0, pool, min=10, max=20, estimate_rule=cat_estimate_hybrid)

## SE, MI, and CI stopping rule
cat_sim(1.0, pool, min=10, max=20, stop_se=.3)
cat_sim(1.0, pool, min=10, max=20, stop_mi=.6)
cat_sim(1.0, pool, min=10, max=20, stop_cut=0)
cat_sim(1.0, pool, min=10, max=20, stop_cut=0, ci_width=2.58)

## maximum information selection with item sets
cat_sim(1.0, pool, min=10, max=20, group="set_id")$admin

## maximum information with item exposure control
cat_sim(1.0, pool, min=10, max=20, info_random=5)$admin

## Constrained-CAT selection rule with and without initial randomness
cat_sim(1.0, pool, min=10, max=20, select_rule=cat_select_ccat,
        ccat_var="content", ccat_perc=c("1"=.2, "2"=.3, "3"=.5))
cat_sim(1.0, pool, min=10, max=20, select_rule=cat_select_ccat, ccat_random=5,
        ccat_var="content", ccat_perc=c("1"=.2, "2"=.3, "3"=.5))

## Shadow-test selection rule
cons <- data.frame(var='content', level=1:3, min=c(3,3,4), max=c(3,3,4))
cons <- rbind(cons, data.frame(var='time', level=NA, min=55*10, max=65*10))
cat_sim(1.0, pool, min=10, max=10, select_rule=cat_select_shadow, constraints=cons)

## Projection-based stopping rule
cons <- data.frame(var='content', level=1:3, min=5, max=15)
cons <- rbind(cons, data.frame(var='time', level=NA, min=60*20, max=60*40))
cat_sim(1.0, pool, min=20, max=40, select_rule=cat_select_shadow, stop_rule=cat_stop_projection,
        projection_method="diff", stop_cut=0, constraints=cons)

## End(Not run)
```

cronbach_alpha *Cronbach's alpha*

Description

cronbach_alpha computes Cronbach's alpha internal consistency reliability

Usage

```
cronbach_alpha(responses)
```

Arguments

responses	the oberved responses, 2d matrix

Examples

```
cronbach_alpha(model_3pl_gendata(1000, 20)$u)
```

expected_raw_score_dist *#' Distribution of Expected Raw Scores*

Description

Calculate the distribution of expected raw scores

Usage

```
expected_raw_score_dist(t, a, b, c)
```

Arguments

t	the ability parameters, 1d vector
a	the item discrimination parameters, 1d vector
b	the item difficulty parameters, 1d vector
c	the item guessing parameters, 1d vector

freq *Frequency Counts*

Description

Frequency counts of a vector

Usage

```
freq(x, values = NULL, rounding = NULL)
```

Arguments

x	a numeric or character vector
values	valid values, NULL to include all values
rounding	round percentage to n-th decimal places

model_3pl *3-parameter-logistic model*

Description

Routine functions for the 3PL model

Usage

```
model_3pl_prob(t, a, b, c, D = 1.702)

model_3pl_info(t, a, b, c, D = 1.702)

model_3pl_lh(u, t, a, b, c, D = 1.702, log = FALSE)

model_3pl_rescale(t, a, b, c, param = c("t", "b"), mean = 0, sd = 1)

model_3pl_gendata(n_p, n_i, t = NULL, a = NULL, b = NULL, c = NULL,
  D = 1.702, t_dist = c(0, 1), a_dist = c(-0.1, 0.2), b_dist = c(0,
  0.7), c_dist = c(5, 46), missing = NULL)

model_3pl_plot(a, b, c, D = 1.702, type = c("prob", "info"),
  total = FALSE, xaxis = seq(-4, 4, 0.1))

model_3pl_plot_loglh(u, a, b, c, D = 1.702, xaxis = seq(-4, 4, 0.1),
  show_mle = FALSE)
```

Arguments

t	ability parameters, 1d vector
a	discrimination parameters, 1d vector
b	difficulty parameters, 1d vector
c	guessing parameters, 1d vector
D	the scaling constant, 1.702 by default
u	observed responses, 2d matrix
log	True to return log-likelihood
param	the parameter of the new scale: 't' or 'b'
mean	the mean of the new scale
sd	the standard deviation of the new scale
n_p	the number of people to be generated
n_i	the number of items to be generated
t_dist	parameters of the normal distribution used to generate t-parameters
a_dist	parameters of the lognormal distribution used to generate a-parameters
b_dist	parameters of the normal distribution used to generate b-parameters
c_dist	parameters of the beta distribution used to generate c-parameters
missing	the proportion or number of missing responses
type	the type of plot: 'prob' for item characteristic curve (ICC) and 'info' for item information function curve (IIFC)
total	TRUE to sum values over items
xaxis	the values of x-axis
show_mle	TRUE to print maximum likelihood estimates

Examples

```
with(model_3pl_gendata(10, 5), model_3pl_prob(t, a, b, c))
with(model_3pl_gendata(10, 5), model_3pl_info(t, a, b, c))
with(model_3pl_gendata(10, 5), model_3pl_lh(u, t, a, b, c))
model_3pl_gendata(10, 5)
model_3pl_gendata(10, 5, a=1, c=0, missing=.1)
with(model_3pl_gendata(10, 5), model_3pl_plot(a, b, c, type="prob"))
with(model_3pl_gendata(10, 5), model_3pl_plot(a, b, c, type="info", total=TRUE))
with(model_3pl_gendata(5, 50), model_3pl_plot_loglh(u, a, b, c, show_mle=TRUE))
```

model_gpcm *Generalized Partial Credit Model*

Description

Routine functions for the GPCM

Usage

```
model_gpcm_prob(t, a, b, d, D = 1.702, insert_d0 = NULL)

model_gpcm_info(t, a, b, d, D = 1.702, insert_d0 = NULL)

model_gpcm_lh(u, t, a, b, d, D = 1.702, insert_d0 = NULL,
  log = FALSE)

model_gpcm_gendata(n_p, n_i, n_c, t = NULL, a = NULL, b = NULL,
  d = NULL, D = 1.702, sort_d = FALSE, t_dist = c(0, 1),
  a_dist = c(-0.1, 0.2), b_dist = c(0, 0.8), missing = NULL)

model_gpcm_rescale(t, a, b, d, param = c("t", "b"), mean = 0, sd = 1)

model_gpcm_plot(a, b, d, D = 1.702, insert_d0 = NULL,
  type = c("prob", "info"), by_item = FALSE, total = FALSE,
  xaxis = seq(-6, 6, 0.1))

model_gpcm_plot_loglh(u, a, b, d, D = 1.702, insert_d0 = NULL,
  xaxis = seq(-6, 6, 0.1), show_mle = FALSE)
```

Arguments

t	ability parameters, 1d vector
a	discrimination parameters, 1d vector
b	item location parameters, 1d vector
d	item category parameters, 2d vector
D	the scaling constant, 1.702 by default
insert_d0	insert an initial category value
u	the observed scores (starting from 0), 2d matrix
log	TRUE to return log-likelihood
n_p	the number of people to be generated
n_i	the number of items to be generated
n_c	the number of score categories
sort_d	TRUE to sort d parameters for each item

t_dist	parameters of the normal distribution used to generate t-parameters
a_dist	parameters of the lognormal distribution parameters of a-parameters
b_dist	parameters of the normal distribution used to generate b-parameters
missing	the proportion or number of missing responses
param	the parameter of the new scale: 't' or 'b'
mean	the mean of the new scale
sd	the standard deviation of the new scale
type	the type of plot, prob for ICC and info for IIFC
by_item	TRUE to combine categories
total	TRUE to sum values over items
xaxis	the values of x-axis
show_mle	TRUE to print maximum likelihood values

Details

Use NA to represent unused category.

Examples

```
with(model_gpcm_gendata(10, 5, 3), model_gpcm_prob(t, a, b, d))
with(model_gpcm_gendata(10, 5, 3), model_gpcm_info(t, a, b, d))
with(model_gpcm_gendata(10, 5, 3), model_gpcm_lh(u, t, a, b, d))
model_gpcm_gendata(10, 5, 3)
model_gpcm_gendata(10, 5, 3, missing=.1)
# Figure 1 in Muraki, 1992 (APM)
b <- matrix(c(-2,0,2,-.5,0,2,-.5,0,2), nrow=3, byrow=TRUE)
model_gpcm_plot(a=c(1,1,.7), b=rowMeans(b), d=rowMeans(b)-b, D=1.0, insert_d0=0)
# Figure 2 in Muraki, 1992 (APM)
b <- matrix(c(.5,0,NA,0,0,0), nrow=2, byrow=TRUE)
model_gpcm_plot(a=.7, b=rowMeans(b, na.rm=TRUE), d=rowMeans(b, na.rm=TRUE)-b, D=1.0, insert_d0=0)
# Figure 3 in Muraki, 1992 (APM)
b <- matrix(c(1.759,-1.643,3.970,-2.764), nrow=2, byrow=TRUE)
model_gpcm_plot(a=c(.778,.946), b=rowMeans(b), d=rowMeans(b)-b, D=1.0, insert_d0=0)
# Figure 1 in Muraki, 1993 (APM)
b <- matrix(c(0,-2,4,0,-2,2,0,-2,0,0,-2,-2,0,-2,-4), nrow=5, byrow=TRUE)
model_gpcm_plot(a=1, b=rowMeans(b), d=rowMeans(b)-b, D=1.0)
# Figure 2 in Muraki, 1993 (APM)
b <- matrix(c(0,-2,4,0,-2,2,0,-2,0,0,-2,-2,0,-2,-4), nrow=5, byrow=TRUE)
model_gpcm_plot(a=1, b=rowMeans(b), d=rowMeans(b)-b, D=1.0, type='info', by_item=TRUE)
with(model_gpcm_gendata(5, 50, 3), model_gpcm_plot_loglh(u, a, b, d))
```

model_grm *Graded Response Model*

Description

Routine functions for the GRM

Usage

```
model_grm_prob(t, a, b, D = 1.702, raw = FALSE)

model_grm_info(t, a, b, D = 1.702)

model_grm_lh(u, t, a, b, D = 1.702, log = FALSE)

model_grm_gendata(n_p, n_i, n_c, t = NULL, a = NULL, b = NULL,
  D = 1.702, t_dist = c(0, 1), a_dist = c(-0.1, 0.2), b_dist = c(0,
  0.8), missing = NULL)

model_grm_rescale(t, a, b, param = c("t", "b"), mean = 0, sd = 1)

model_grm_plot(a, b, D = 1.702, type = c("prob", "info"),
  by_item = FALSE, total = FALSE, xaxis = seq(-6, 6, 0.1),
  raw = FALSE)

model_grm_plot_loglh(u, a, b, D = 1.702, xaxis = seq(-6, 6, 0.1),
  show_mle = FALSE)
```

Arguments

t	ability parameters, 1d vector
a	discrimination parameters, 1d vector
b	item location parameters, 2d matrix
D	the scaling constant, 1.702 by default
raw	TRUE to return P*
u	the observed scores (starting from 0), 2d matrix
log	TRUE to return log-likelihood
n_p	the number of people to be generated
n_i	the number of items to be generated
n_c	the number of score categories
t_dist	parameters of the normal distribution used to generate t-parameters
a_dist	parameters of the lognormal distribution used to generate a-parameters
b_dist	parameters of the normal distribution used to generate b-parameters

missing	the proportion or number of missing responses
param	the parameter of the new scale: 't' or 'b'
mean	the mean of the new scale
sd	the standard deviation of the new scale
type	the type of plot, prob for ICC and info for IIFC
by_item	TRUE to combine categories
total	TRUE to sum values over items
xaxis	the values of x-axis
show_mle	TRUE to print maximum likelihood values

Examples

```
with(model_grm_gendata(10, 5, 3), model_grm_prob(t, a, b))
with(model_grm_gendata(10, 5, 3), model_grm_info(t, a, b))
with(model_grm_gendata(10, 5, 3), model_grm_lh(u, t, a, b))
model_grm_gendata(10, 5, 3)
model_grm_gendata(10, 5, 3, missing=.1)
with(model_grm_gendata(10, 5, 3), model_grm_plot(a, b, type='prob'))
with(model_grm_gendata(10, 5, 3), model_grm_plot(a, b, type='info', by_item=TRUE))
with(model_grm_gendata(5, 50, 3), model_grm_plot_loglh(u, a, b))
```

---

mst_sim *Simulation of Multistage Testing*

Description

mst_sim simulates a MST administration

Usage

```
mst_sim(x, true, rdp = NULL, ...)

## S3 method for class 'mst_sim'
print(x, ...)

## S3 method for class 'mst_sim'
plot(x, ...)
```

Arguments

x	the assembled MST
true	the true theta parameter (numeric)
rdp	routing decision points (list)
...	additional option/control parameters

Examples

```
## Not run:
## assemble a MST
nitems <- 200
pool <- with(model_3pl_gendata(1, nitems), data.frame(a=a, b=b, c=c))
pool$content <- sample(1:3, nrow(pool), replace=TRUE)
x <- mst(pool, "1-2-2", 2, 'topdown', len=20, max_use=1)
x <- mst_obj(x, theta=-1, indices=1)
x <- mst_obj(x, theta=0, indices=2:3)
x <- mst_obj(x, theta=1, indices=4)
x <- mst_constraint(x, "content", 6, 6, level=1)
x <- mst_constraint(x, "content", 6, 6, level=2)
x <- mst_constraint(x, "content", 8, 8, level=3)
x <- mst_stage_length(x, 1:2, min=5)
x <- mst_assemble(x)

## ex. 1: administer the MST using fixed RDP for routing
x_sim <- mst_sim(x, .5, list(stage1=0, stage2=0))
plot(x_sim)

## ex. 2: administer the MST using the max. info. for routing
x_sim <- mst_sim(x, .5)
plot(x_sim, ylim=c(-5, 5))

## End(Not run)
```

`rmse` *Root Mean Squared Error*

Description

Root mean squared error (RMSE) of two numeric vectors/matrices

Usage

```
rmse(x, y)
```

Arguments

x	a numeric vector/matrix
y	a numeric vector/matrix

spearman_brown *Spearman Brown Prophecy*

Description

Use Spearman-brown formula to compute the predicted reliability when the test length is extened to n-fold or reversely the n-fold extension of test length in order to reach the targeted reliability

Usage

```
spearman_brown(n, rho)

spearman_brown_reverse(rho, target)
```

Arguments

n	extend the test length to n-fold
rho	the reliability of current test
target	the targeted reliability

Examples

```
spearman_brown(2, .70)
spearman_brown_reverse(.70, .85)
```

Index

ata, 2
ata_constraint *(ata)*, 2
ata_item_enemy *(ata)*, 2
ata_item_fixedvalue *(ata)*, 2
ata_item_use *(ata)*, 2
ata_obj_absolute *(ata)*, 2
ata_obj_relative *(ata)*, 2
ata_solve *(ata)*, 2

cat_estimate_eap *(cat_sim)*, 5
cat_estimate_hybrid *(cat_sim)*, 5
cat_estimate_mle *(cat_sim)*, 5
cat_select_ccat *(cat_sim)*, 5
cat_select_maxinfo *(cat_sim)*, 5
cat_select_shadow *(cat_sim)*, 5
cat_sim, 5
cat_stop_default *(cat_sim)*, 5
cat_stop_projection *(cat_sim)*, 5
cronbach_alpha, 8

expected_raw_score_dist, 8

freq, 9

model_3pl, 9
model_3pl_gendata *(model_3pl)*, 9
model_3pl_info *(model_3pl)*, 9
model_3pl_lh *(model_3pl)*, 9
model_3pl_plot *(model_3pl)*, 9
model_3pl_plot_loglh *(model_3pl)*, 9
model_3pl_prob *(model_3pl)*, 9
model_3pl_rescale *(model_3pl)*, 9
model_gpcm, 11
model_gpcm_gendata *(model_gpcm)*, 11
model_gpcm_info *(model_gpcm)*, 11
model_gpcm_lh *(model_gpcm)*, 11
model_gpcm_plot *(model_gpcm)*, 11
model_gpcm_plot_loglh *(model_gpcm)*, 11
model_gpcm_prob *(model_gpcm)*, 11
model_gpcm_rescale *(model_gpcm)*, 11
model_grm, 13
model_grm_gendata *(model_grm)*, 13
model_grm_info *(model_grm)*, 13
model_grm_lh *(model_grm)*, 13
model_grm_plot *(model_grm)*, 13
model_grm_plot_loglh *(model_grm)*, 13
model_grm_prob *(model_grm)*, 13
model_grm_rescale *(model_grm)*, 13
mst_sim, 14

plot.ata *(ata)*, 2
plot.cat *(cat_sim)*, 5
plot.mst_sim *(mst_sim)*, 14
print.ata *(ata)*, 2
print.cat *(cat_sim)*, 5
print.mst_sim *(mst_sim)*, 14

rmse, 15

spearman_brown, 16
spearman_brown_reverse *(spearman_brown)*, 16