

Package ‘PROsetta’

July 20, 2021

Type Package

Title Linking Patient-Reported Outcomes Measures

Version 0.3.2

Date 2021-07-19

Description Perform scale linking to establish relationships between instruments that measure similar constructs according to the PROsetta Stone methodology, as in Choi, Schalet, Cook, & Cella (2014) <[doi:10.1037/a0035768](https://doi.org/10.1037/a0035768)>.

URL <http://prosettastone.org> (project description),
<https://choi-phd.github.io/PROsetta/> (documentation)

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

Imports equate, lavaan, mirt, plink, psych, methods, mvnfast

Depends R (>= 3.5.0)

Suggests shiny, shinythemes, shinyWidgets, shinyjs, DT, knitr,
kableExtra, testthat (>= 2.1.0), rmarkdown, dplyr, pkgdown

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

Collate 'import.R' 'configPROsetta.R' 'post_functions.R'
'core_functions.R' 'datasets.R' 'example.R'
'helper_functions.R' 'linking_functions.R' 'plot_functions.R'
'runshiny.R'

NeedsCompilation no

Author Seung W. Choi [aut, cre] (<<https://orcid.org/0000-0003-4777-5420>>),
Sangdon Lim [aut] (<<https://orcid.org/0000-0002-2988-014X>>),
Benjamin D. Schalet [ctb],
Aaron J. Kaat [ctb],
David Cella [ctb]

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

Repository CRAN

Date/Publication 2021-07-20 15:50:05 UTC

R topics documented:

checkFrequency	2
compareScores	3
dataset_asq	3
dataset_dep	4
getCompleteData	5
getEscore	6
getItemNames	6
getResponse	7
getRSSS	8
getScaleSum	8
getTheta	9
loadData	10
plot,PROsetta_data,ANY-method	11
plotInfo	12
PROsetta	13
runCalibration	13
runCFA	15
runClassical	16
runDescriptive	16
runEquateObserved	17
runFrequency	19
runLinking	19
runRSSS	20
Index	22

checkFrequency	<i>Check frequency table for unobserved response categories</i>
----------------	---

Description

[checkFrequency](#) is a descriptive function to check whether all response categories in a frequency table have a frequency of at least 1.

Usage

```
checkFrequency(data)
```

Arguments

data a [PROsetta_data](#) object. See [loadData](#) for loading a dataset.

Value

If all response categories have a frequency of at least 1, the value is TRUE.
Otherwise, the value is FALSE.

compareScores	<i>Compare two sets of scores</i>
---------------	-----------------------------------

Description

`compareScores` is a helper function to compare two sets of scores.

Usage

```
compareScores(left, right, type = c("corr", "mean", "sd", "rmsd", "mad"))
```

Arguments

<code>left</code>	scores on the left side of comparison.
<code>right</code>	scores on the right side of comparison. This is subtracted from 'left'.
<code>type</code>	type of comparisons to include. Accepts 'corr', 'mean', 'sd', 'rmsd'. Defaults to all four.

Value

`compareScores` returns a `data.frame` containing the comparison results.

dataset_asq	<i>ASQ dataset</i>
-------------	--------------------

Description

This dataset is associated with the following objects:

Details

- `response_asq` a `data.frame` containing raw response data of 751 participants and 41 variables.
 - `prosettaid`. participant IDs.
 - `EDANX01 --MASQ11`. response to items.
- `itemmap_asq` a `data.frame` containing the item map, describing the items in each instrument.
 - `item_order` item numeric IDs. This column refers to the column `item_order` in anchor item attributes.
 - `instrument` the instrument ID that each item belongs to.

- `item_id` item ID strings. This column refers to column names in raw response data, excluding the participant ID column.
- `item_name` new item ID strings to be used in the combined scale.
- `ncat` the number of response categories.
- `min_score` the minimum score of the item.
- `reverse` whether the item data has been reverse-scored. 1 indicates the item has been reverse-scored, and 0 indicates the item has not been reverse-scored.
- `scores` a comma-separated string representing all possible score values from the item.
- `anchor_asq` a `data.frame` containing anchor item parameters for 29 items.
 - `item_order` item numeric IDs.
 - `item_id` item ID strings. This column refers to column names in raw response data, excluding the participant ID column.
 - `a` the discrimination (slope) parameter for the graded response model.
 - `cb1 -cb4` the boundaries between each category-pair for the graded response model.
 - `ncat` the number of response categories.
- `data_asq` a `PROsetta_data` object containing the datasets above. See `loadData` for creating `PROsetta_data` objects.

Examples

```
## load datasets into a PROsetta_data object
data_asq <- loadData(
  response = response_asq,
  itemmap  = itemmap_asq,
  anchor   = anchor_asq
)

## run descriptive statistics
runDescriptive(data_asq)

## run item parameter calibration on the response data, linking to the anchor item parameters
runLinking(data_asq, method = "FIXEDPAR")
```

dataset_dep

DEP dataset

Description

This dataset is associated with the following objects:

Details

- `response_dep` a `data.frame` containing raw response data of 747 participants and 49 variables.
 - `prosettaid`. participant IDs.

- EDDEP04 --CESD20. response to items.
- `itemmap_dep` a `data.frame` containing the item map, describing the items in each instrument.
 - `item_order` item numeric IDs. This column refers to the column `item_order` in anchor item parameters.
 - `instrument` the instrument ID that each item belongs to.
 - `item_id` item ID strings. This column refers to column names in raw response data, excluding the participant ID column.
 - `item_name` new item ID strings to be used in the combined scale.
 - `ncat` the number of response categories.
 - `min_score` the minimum score of the item.
 - `reverse` whether the item data has been reverse-scored. 1 indicates the item has been reverse-scored, and 0 indicates the item has not been reverse-scored.
 - `scores` a comma-separated string representing all possible score values from the item.
- `anchor_dep` a `data.frame` containing anchor item parameters for 28 items.
 - `item_order` item numeric IDs.
 - `item_id` item ID strings. This column refers to column names in raw response data, excluding the participant ID column.
 - `a` the discrimination (slope) parameter for the graded response model.
 - `cb1 -cb4` the boundaries between each category-pair for the graded response model.
 - `ncat` the number of response categories.
- `data_dep` a `PROsetta_data` object containing the datasets above. See `loadData` for creating `PROsetta_data` objects.

Examples

```
## load datasets into a PROsetta_data object
data_dep <- loadData(
  response = response_dep,
  itemmap  = itemmap_dep,
  anchor   = anchor_dep
)

## run descriptive statistics
runDescriptive(data_dep)

## run item parameter calibration on the response data, linking to the anchor item parameters
runLinking(data_dep, method = "FIXEDPAR")
```

<code>getCompleteData</code>	<i>Get complete data</i>
------------------------------	--------------------------

Description

`getCompleteData` is a helper function to perform casewise deletion of missing values.

Usage

```
getCompleteData(data, scale = NULL)
```

Arguments

`data` a [PROsetta_data](#) object.

`scale` the index of the scale to perform casewise deletion. Leave empty or set to "combined" to perform on all scales.

getEScore	<i>Calculate expected scores at theta</i>
-----------	---

Description

[getEScore](#) is a helper function to calculate expected scores at supplied thetas.

Usage

```
getEScore(ipar, model, theta, is_minscore_0)
```

Arguments

`ipar` item parameters.

`model` item model to use.

`theta` theta values.

`is_minscore_0` if TRUE the score begins from 0 instead of 1.

Value

[getEScore](#) returns a vector of expected scores.

getItemNames	<i>Get item names</i>
--------------	-----------------------

Description

[getItemNames](#) is a helper function to extract item names for a specified scale from a [PROsetta_data](#) object.

Usage

```
getItemNames(d, scale_id)
```

Arguments

`d` a [PROsetta_data](#) object.
`scale_id` scale IDs to extract item names.

Value

[getItemNames](#) returns a vector containing item names.

Examples

```
idx <- getItemNames(data_asq, 1)
data_asq@response[, idx]
```

getResponse	<i>Extract scale-wise response</i>
-------------	------------------------------------

Description

[getResponse](#) is a helper function to extract scale-wise response from a [PROsetta_data](#) object.

Usage

```
getResponse(d, scale_id = "all", person_id = FALSE)
```

Arguments

`d` a [PROsetta_data](#) object.
`scale_id` scale IDs to extract response. If `all`, use all scale IDs. (default = `all`)
`person_id` if `TRUE`, also return person IDs. (default = `FALSE`)

Value

[getResponse](#) returns a `data.frame` containing scale-wise response.

Examples

```
getResponse(data_asq)
getResponse(data_asq, 1)
getResponse(data_asq, 2)
getResponse(data_asq, c(1, 2))
getResponse(data_asq, c(2, 1))
getResponse(data_asq, c(1, 2), person_id = TRUE)
```

getRSSS *Compute a Crosswalk Table*

Description

[getRSSS](#) is a function to generate a raw-score to standard-score crosswalk table.

Usage

```
getRSSS(ipar, theta_grid, is_minscore_0, prior_mu_sigma)
```

Arguments

ipar an item parameter matrix for graded response items. Accepts both a/b and a/d format parameters. Accepts multidimensional item parameters.

theta_grid the theta grid to use.

is_minscore_0 if TRUE, the scores of each item begins from 0. if FALSE, the scores of each item begins from 1.

prior_mu_sigma a named list containing prior distribution parameters:

- mu means
- sigma the covariance matrix
- sd standard deviations
- corr the correlation matrix

getScaleSum *Calculate raw sum scores of a scale*

Description

[getScaleSum](#) is a helper function to calculate raw sum scores of a scale.

Usage

```
getScaleSum(data, scale_idx)
```

Arguments

data a [PROsetta_data](#) object.

scale_idx the index of the scale to obtain the raw sum scores.

getTheta	<i>Obtain EAP estimates</i>
----------	-----------------------------

Description

`getTheta` is a helper function to calculate EAP estimates.

Usage

```
getTheta(  
  data,  
  ipar,  
  scale = "combined",  
  model = "grm",  
  theta_grid = seq(-4, 4, 0.1),  
  prior_dist = "normal",  
  prior_mean = 0,  
  prior_sd = 1  
)
```

Arguments

<code>data</code>	a <code>PROsetta_data</code> object.
<code>ipar</code>	a <code>data.frame</code> containing item parameters.
<code>scale</code>	the index of the scale to use. Set to 'combined' to use the combined scale.
<code>model</code>	the item model to use. Accepts 'grm' or 'gpcm'.
<code>theta_grid</code>	the theta grid to use in calculating EAP estimates.
<code>prior_dist</code>	the type of prior distribution. Accepts 'normal' or 'logistic'.
<code>prior_mean</code>	mean of the prior distribution.
<code>prior_sd</code>	SD of the prior distribution.

Value

`getTheta` returns a `list` containing EAP estimates.

loadData	<i>Load data from supplied config</i>
----------	---------------------------------------

Description

`loadData` is a data loading function to create a `PROsetta_data` object, for scale linking/equating with 'PROsetta' package.

Usage

```
loadData(  
  response,  
  itemmap,  
  anchor,  
  item_id = NULL,  
  person_id = NULL,  
  scale_id = NULL,  
  input_dir = getwd()  
)
```

Arguments

<code>response</code>	response data containing case IDs and item responses. This can be a <code>.csv</code> filename or a <code>data.frame</code> object.
<code>itemmap</code>	an item map containing item IDs and scale IDs. This can be a <code>.csv</code> filename or a <code>data.frame</code> object.
<code>anchor</code>	anchor data containing item parameters for anchor items. This can be a <code>.csv</code> filename or a <code>data.frame</code> object.
<code>item_id</code>	the column name to look for item IDs. Automatically determined if not specified.
<code>person_id</code>	the column name to look for case IDs. Automatically determined if not specified.
<code>scale_id</code>	the column name to look for scale IDs. Automatically determined if not specified.
<code>input_dir</code>	the directory to look for the files.

Value

`loadData` returns a `PROsetta_data` object containing the loaded data.

plot,PROsetta_data,ANY-method
Plot frequency distribution

Description

This is an extension of `plot` to visualize frequency distribution from `PROsetta_data` object.

Usage

```
## S4 method for signature 'PROsetta_data,ANY'  
plot(  
  x,  
  y,  
  scale_id = "combined",  
  filename = NULL,  
  title = NULL,  
  xlim = NULL,  
  color = "blue",  
  nbar = 20,  
  rug = FALSE,  
  filetype = "pdf",  
  savefile = FALSE,  
  bg = "white",  
  width = 6,  
  height = 6,  
  pointsize = 12  
)
```

Arguments

<code>x</code>	a <code>PROsetta_data</code> object.
<code>y</code>	unused argument, exists for compatibility with <code>plot</code> in the base R package.
<code>scale_id</code>	scale ID to plot. <code>combined</code> (default) represents the combined scale.
<code>filename</code>	filename to write if <code>'savefile'</code> argument is TRUE.
<code>title</code>	the title of the figure.
<code>xlim</code>	the range of scores to plot.
<code>color</code>	the color to fill the histogram.
<code>nbar</code>	the number of histogram bars.
<code>rug</code>	if TRUE, display the actual distribution of scores below each bar.
<code>filetype</code>	the type of file to write if <code>'savefile'</code> argument is TRUE. Accepts <code>'pdf'</code> , <code>'jpeg'</code> , <code>'png'</code> , and <code>'tiff'</code> .
<code>savefile</code>	if TRUE, save the figure as a file.
<code>bg</code>	the background color of the plot.

width the width of the plot.
 height the height of the plot.
 pointsize point size to pass onto file writing functions.

Examples

```
plot(data_asq)
plot(data_asq, scale_id = 1)
plot(data_asq, scale_id = 2)
```

plotInfo *Plot scale information*

Description

[plotInfo](#) is a plotting function to visualize scale-level information.

Usage

```
plotInfo(
  object,
  data,
  theta = seq(-4, 4, 0.1),
  t_score = FALSE,
  scale_label = c(1, 2, "Combined"),
  color = c("red", "blue", "black"),
  lty = c(3, 2, 1)
)

## S4 method for signature 'SingleGroupClass'
plotInfo(
  object,
  data,
  theta = seq(-4, 4, 0.1),
  t_score = FALSE,
  scale_label = c(1, 2, "Combined"),
  color = c("red", "blue", "black"),
  lty = c(3, 2, 1)
)
```

Arguments

object a [SingleGroupClass](#) object from [runCalibration](#).
 data a [PROsetta_data](#) object.
 theta theta values to plot on the x-axis.

t_score	set to TRUE to convert thetas into T-scores.
scale_label	names of each scale.
color	line colors to plot.
lty	line types to plot.

Examples

```
out_calib = runCalibration(data_asq, technical = list(NCYCLES = 1000))
plotInfo(out_calib, data_asq)
```

PROsetta

PROsetta

Description

[PROsetta](#) is a caller function to launch a Shiny app locally.

Usage

```
PROsetta()
```

```
guiPROsetta()
```

Examples

```
if (interactive()) {
  PROsetta()
}
```

runCalibration

Run Calibration

Description

[runCalibration](#) is a function to perform item calibration on the response data.

Usage

```
runCalibration(
  data,
  dimensions = 1,
  fix_method = "free",
  fixedpar = NULL,
  ignore_nonconv = FALSE,
  ...
)
```

Arguments

<code>data</code>	a <code>PROsetta_data</code> object. See <code>loadData</code> for loading a dataset.
<code>dimensions</code>	number of dimensions to use. Must be 1 or 2. If 1, use one underlying dimension for all instruments combined. If 2, use each dimension separately for the anchor instrument and the developing instrument. Covariance between dimensions is freely estimated. (default = 1)
<code>fix_method</code>	the type of constraints to impose. (default = free) <ul style="list-style-type: none"> • <code>item</code> for fixed parameter calibration using anchor item parameters • <code>theta</code> for using the mean and the variance obtained from a unidimensional calibration of anchor items • <code>free</code> for free calibration
<code>fixedpar</code>	this argument exists for reproducibility. TRUE is equivalent to <code>fix_method = "item"</code> , and FALSE is equivalent to <code>fix_method = "free"</code> .
<code>ignore_nonconv</code>	if TRUE, return results even when calibration does not converge. If FALSE, raise an error when calibration does not converge. (default = FALSE)
<code>...</code>	additional arguments to pass onto <code>mirt</code> in <code>'mirt'</code> package.

Value

`runCalibration` returns a `SingleGroupClass` object containing item calibration results.

This object can be used in `coef`, `itemfit`, `itemplot` in `'mirt'` package to extract wanted information.

Examples

```
## Not run:
out_calib <- runCalibration(data_asq) # errors

## End(Not run)

out_calib <- runCalibration(data_asq, technical = list(NCYCLES = 1000))

mirt::coef(out_calib, IRTpars = TRUE, simplify = TRUE)
mirt::itemfit(out_calib, empirical.plot = 1)
mirt::itemplot(out_calib, item = 1, type = "info")
mirt::itemfit(out_calib, "S_X2", na.rm = TRUE)
```

runCFA	<i>Run a confirmatory factor analysis</i>
--------	---

Description

`runCFA` is a function to perform a one-factor confirmatory factor analysis (CFA) to test unidimensionality.

Usage

```
runCFA(data, estimator = "WLSMV", std.lv = TRUE, scalewise = FALSE, ...)
```

Arguments

<code>data</code>	a <code>PROsetta_data</code> object. See <code>loadData</code> for loading a dataset.
<code>estimator</code>	the estimator to be used. Passed onto <code>cfa</code> in <code>'lavaan'</code> package. (default = WLSMV)
<code>std.lv</code>	if TRUE, the metric of the latent variable is determined by fixing their (residual) variances to 1.0. If FALSE, the metric of each latent variable is determined by fixing the factor loading of the first indicator to 1.0. Passed onto <code>cfa</code> . (default = TRUE)
<code>scalewise</code>	if TRUE, run analysis for each scale as well as for the combined scale. If FALSE, run analysis only for the combined scale. (default = FALSE)
<code>...</code>	additional arguments to pass onto <code>cfa</code> .

Value

`runCFA` returns a list containing the CFA results.

Examples

```
out_cfa <- runCFA(data_asq, scalewise = TRUE)
lavaan::summary(out_cfa$`1` , fit.measures = TRUE, standardized = TRUE, estimates = FALSE)
lavaan::summary(out_cfa$`2` , fit.measures = TRUE, standardized = TRUE, estimates = FALSE)
lavaan::summary(out_cfa$`combined`, fit.measures = TRUE, standardized = TRUE, estimates = FALSE)
```

runClassical	<i>Run CTT-based reliability analysis</i>
--------------	---

Description

`runClassical` is a function to perform Classical Test Theory (CTT) based reliability analysis.

Usage

```
runClassical(data, omega = FALSE, scalewise = TRUE, ...)
```

Arguments

<code>data</code>	a <code>PROsetta_data</code> object. See <code>loadData</code> for loading a dataset.
<code>omega</code>	if TRUE, also obtain McDonald's omega using <code>omega</code> in <code>psych</code> package. (default = FALSE)
<code>scalewise</code>	if TRUE, run analysis for each scale as well as for the combined scale. If FALSE, run analysis only for the combined scale. (default = TRUE)
<code>...</code>	additional arguments to pass onto <code>omega</code> .

Value

`runClassical` returns a `list` containing reliability analysis results.

Examples

```
out_alpha <- runClassical(data_asq)
out_omega <- runClassical(data_asq, omega = TRUE) # also obtain omega
```

runDescriptive	<i>Obtain a descriptive statistics table</i>
----------------	--

Description

`runDescriptive` is a descriptive function to obtain descriptive statistics for each item in the dataset.

Usage

```
runDescriptive(data = NULL)
```

Arguments

<code>data</code>	a <code>PROsetta_data</code> object. See <code>loadData</code> for loading a dataset.
-------------------	---

Value

`runDescriptive` returns a `data.frame` containing descriptive statistics (mean, standard deviation, median, ...) of the items in the dataset. These are calculated with `describe` in `'psych'` package.

Examples

```
out_desc <- runDescriptive(data_asq)
```

runEquateObserved *Run Test Equating*

Description

`runEquateObserved` is a function to perform equipercntile test equating between two scales. A concordance table is produced, mapping the observed raw scores from one scale to the scores from another scale.

Usage

```
runEquateObserved(
  data,
  scale_from = 2,
  scale_to = 1,
  type_to = "raw",
  rsss = NULL,
  eq_type = "equipercntile",
  smooth = "loglinear",
  degrees = list(3, 1),
  boot = TRUE,
  reps = 100,
  ...
)
```

Arguments

<code>data</code>	a <code>PROsetta_data</code> object. See <code>loadData</code> for loading a dataset.
<code>scale_from</code>	the scale ID of the input scale. References to <code>itemmap</code> in <code>data</code> argument. (default = 2)
<code>scale_to</code>	the scale ID of the target scale to equate to. References to <code>itemmap</code> in <code>data</code> argument. (default = 1)
<code>type_to</code>	the type of score to use in the target scale frequency table. Accepts <code>raw</code> , <code>tscore</code> , and <code>theta</code> . <code>tscore</code> and <code>theta</code> require argument <code>rsss</code> to be supplied. (default = <code>raw</code>)
<code>rsss</code>	the RSSS table to use to map each raw score level onto a t-score or a theta. See <code>runRSSS</code> .

eq_type	the type of equating to be passed onto <code>equate</code> in 'equate' package. (default = equipercentile)
smooth	the type of smoothing method to be passed onto <code>presmoothing</code> in 'equate' package. (default = loglinear)
degrees	the degrees of smoothing to be passed onto <code>presmoothing</code> . (default = list(3,1))
boot	performs bootstrapping if TRUE. (default = TRUE)
reps	the number of replications to perform in bootstrapping. (default = 100)
...	other arguments to pass onto <code>equate</code> .

Value

`runEquateObserved` returns an `equate` object containing the test equating result.

The printed summary statistics indicate the distributional properties of the two supplied scales and the equated scale.

- x corresponds to `scale_from`.
- y corresponds to `scale_to`.
- yx corresponds to `scale_from` after equating to `scale_to`.

See `equate` for details.

The concordance table is stored in `concordance` slot.

Examples

```
out_eq_raw <- runEquateObserved(data_asq,
  scale_to = 1, scale_from = 2,
  eq_type = "equipercentile", smooth = "loglinear"
)
out_eq_raw$concordance

out_link <- runLinking(data_asq, method = "FIXEDPAR")
out_rsss <- runRSSS(data_asq, out_link)
out_eq_tscore <- runEquateObserved(data_asq,
  scale_to = 1, scale_from = 2,
  type_to = "tscore", rsss = out_rsss,
  eq_type = "equipercentile", smooth = "loglinear"
)
out_eq_tscore$concordance
```

runFrequency	<i>Obtain a frequency table</i>
--------------	---------------------------------

Description

`runFrequency` is a descriptive function to obtain a frequency table from the dataset.

Usage

```
runFrequency(data, check_frequency = TRUE)
```

Arguments

`data` a `PROsetta_data` object. See `loadData` for loading a dataset.

`check_frequency` Logical. If TRUE, check the frequency table for missing response categories, and display warning message if any is missing. (default = TRUE)

Value

`runFrequency` returns a `data.frame` containing the frequency table.

Examples

```
freq_asq <- runFrequency(data_asq)
freq_dep <- runFrequency(data_dep)
```

runLinking	<i>Run Scale Linking</i>
------------	--------------------------

Description

`runLinking` is a function to obtain item parameters from the response data, and perform scale linking onto the metric of supplied anchor item parameters.

Usage

```
runLinking(data, method, ...)
```

Arguments

data	a <code>PROsetta_data</code> object. See <code>loadData</code> for loading a dataset.
method	the type of linking to perform. Accepts: <ul style="list-style-type: none"> • MM for mean-mean • MS for mean-sigma • HB for Haebara method • SL for Stocking-Lord method • FIXEDPAR for fixed parameter calibration • CP for calibrated projection using fixed parameter calibration on the anchor dimension • CPLA for linear approximation of calibrated projection. This is identical to 'CP' in <code>runLinking</code> but uses approximation in <code>runRSSS</code> • CPFIXEDDIM for calibrated projection using mean and variance constraints on the anchor dimension <p>Linear transformation methods are performed with <code>plink</code> in 'plink' package.</p>
...	additional arguments to pass onto <code>mirt</code> in 'mirt' package.

Value

`runLinking` returns a `list` containing the scale linking results.

- constants linear transformation constants. NA if method argument was FIXEDPAR.
- ipar_linked item parameters calibrated to the response data, and linked to the anchor item parameters.
- ipar_anchor anchor item parameters used in linking.

Examples

```
out_link <- runLinking(data_asq, "SL", technical = list(NCYCLES = 1000))
out_link$constants # transformation constants
out_link$ipar_linked # item parameters linked to anchor
out_link <- runLinking(data_asq, "FIXEDPAR")
out_link$ipar_linked # item parameters linked to anchor
```

runRSSS

Compute Crosswalk Tables

Description

`runRSSS` is a function to generate raw-score to standard-score crosswalk tables from supplied calibrated item parameters.

Usage

```
runRSSS(  
  data,  
  ipar_linked,  
  prior_mean = 0,  
  prior_sd = 1,  
  min_theta = -4,  
  max_theta = 4,  
  inc = 0.05,  
  min_score = 1  
)
```

Arguments

<code>data</code>	a PROsetta_data object. See loadData for loading a dataset.
<code>ipar_linked</code>	an object returned from runLinking or runCalibration .
<code>prior_mean</code>	prior mean. (default = 0.0)
<code>prior_sd</code>	prior standard deviation. (default = 1.0)
<code>min_theta</code>	the lower limit of theta grid. (default = -4)
<code>max_theta</code>	the upper limit of theta grid. (default = 4)
<code>inc</code>	the increment to use in theta grid. (default = 0.05)
<code>min_score</code>	minimum item score (0 or 1) for each scale (1, 2, and combined). If a single value is supplied, the value is applied to all scales. (default = 1)

Value

[runRSSS](#) returns a [list](#) containing crosswalk tables.

Examples

```
out_link <- runLinking(data_asq, method = "FIXEDPAR")  
score_table <- runRSSS(data_asq, out_link)
```

Index

* datasets

dataset_asq, 3
dataset_dep, 4

anchor_asq, 4
anchor_asq (dataset_asq), 3
anchor_dep, 5
anchor_dep (dataset_dep), 4

cfa, 15
checkFrequency, 2, 2
coef, 14
compareScores, 3, 3

data.frame, 3–5, 7, 9, 10, 17, 19
data_asq, 4
data_asq (dataset_asq), 3
data_dep, 5
data_dep (dataset_dep), 4
dataset_asq, 3
dataset_dep, 4
describe, 17

equate, 18

getCompleteData, 5, 5
getEscore, 6, 6
getItemNames, 6, 6, 7
getResponse, 7, 7
getRSSS, 8, 8
getScaleSum, 8, 8
getTheta, 9, 9
guiPROsetta (PROsetta), 13

itemfit, 14
itemmap_asq, 3
itemmap_asq (dataset_asq), 3
itemmap_dep, 5
itemmap_dep (dataset_dep), 4
itemplot, 14

list, 9, 16, 20, 21
loadData, 2, 4, 5, 10, 10, 14–17, 19–21

mirt, 14, 20

omega, 16

plink, 20
plot, 11
plot, PROsetta_data, ANY-method, 11
plotInfo, 12, 12
plotInfo, SingleGroupClass-method
(plotInfo), 12
presmoothing, 18
PROsetta, 13, 13
PROsetta_data, 2, 4–12, 14–17, 19–21
PROsetta_data-class (loadData), 10

response_asq, 3
response_asq (dataset_asq), 3
response_dep, 4
response_dep (dataset_dep), 4
runCalibration, 12, 13, 13, 14, 21
runCFA, 15, 15
runClassical, 16, 16
runDescriptive, 16, 16, 17
runEquateObserved, 17, 17, 18
runFrequency, 19, 19
runLinking, 19, 19, 20, 21
runRSSS, 17, 20, 20, 21

SingleGroupClass, 12, 14