Package ‘PROsetta’

February 3, 2023

**Type**  Package

**Title**  Linking Patient-Reported Outcomes Measures

**Version**  0.4.1

**Date**  2023-2-3

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

**URL**  https://www.prosettastone.org/ (project description), https://choi-phd.github.io/PROsetta/ (documentation)

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

**Imports**  Rcpp (>= 1.0.0), equate, lavaan, mirt, plink, psych, methods, mvnfast, TestDesign (>= 1.5.1)

**SystemRequirements**  C++17

**Depends**  R (>= 3.5.0)

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

**LinkingTo**  Rcpp, RcppArmadillo

**License**  GPL-3

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.2.3

**VignetteBuilder**  knitr

**Config/testthat/edition**  3

**Collate**  'RcppExports.R' 'import.R' 'loading_functions.R'
'preanalysis_functions.R' 'linking_functions.R' 'example.R'
'helper_functions.R' 'post_functions.R' 'core_functions.R' 'datasets.R'
'plot_functions.R' 'runshiny.R' 'runanalysis_functions.R'

**NeedsCompilation**  yes
checkFrequency

**Description**

`checkFrequency` is a descriptive function for checking whether all response categories in a frequency table have a frequency of at least 1.
compareScores

Usage

checkFrequency(data)

Arguments

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

Value

checkFrequency returns TRUE if all response categories have a frequency of at least 1, and FALSE if not.

Examples

checkFrequency(data_asq) # TRUE

## Not run:
data_asq@response$EDANX01[data_asq@response$EDANX01 == 4] <- 3
checkFrequency(data_asq) # FALSE

## End(Not run)

compareScores

Compare two sets of scores

Description

compareScores is a helper function for comparing two sets of scores.

Usage

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

Arguments

defaults to all types.

Value

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

```r
set.seed(1)
true_theta <- rnorm(100)
theta_est <- true_theta + rnorm(100, 0, 0.3)
compareScores(theta_est, true_theta)
```

---

**dataset_asq**

**ASQ dataset**

**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.
- **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.
- **data_asq** a `PROsetta_data` object containing the datasets above. See `loadData` for creating `PROsetta_data` objects.

**Examples**

```r
## load datasets into a PROsetta_data object
data_asq <- loadData(
  response = response_asq,
  itemmap = itemmap_asq,
```
dataset_dep

    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.
- **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.
- **data_dep** a PROsetta_data object containing the datasets above. See `loadData` for creating PROsetta_data objects.
getCompleteData

Examples

```r
## 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")
```

---

**getCompleteData**

*Get complete data*

**Description**

`getCompleteData` is a helper function for performing casewise deletion of missing values.

**Usage**

```r
getCompleteData(data, scale = NULL, verbose = FALSE)
```

**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.
- `verbose`: if TRUE, print status messages. (default = FALSE)

**Value**

`getCompleteData` returns an updated `PROsetta_data` object.

**Examples**

```r
d <- getCompleteData(data_asq, verbose = TRUE)
```
**getEscore**

*Calculate expected scores at theta*

**Description**

`getEscore` is a helper function for obtaining expected scores at supplied thetas.

**Usage**

```r
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**

*Get item names*

**Description**

`getItemNames` is a helper function for extracting item names for a specified scale from a `PROsetta_data` object.

**Usage**

```r
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.
**getResponse**

**Extract scale-wise response**

**Description**

`getResponse` is a helper function for extracting scale-wise response from a `PROsetta_data` object.

**Usage**

```r
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**

```r
getResponse(data_asq)
ggetItemNames(data_asq, 1)
ggetItemNames(data_asq, 2)
ggetItemNames(data_asq, c(1, 2))
ggetItemNames(data_asq, c(2, 1))
ggetResponse(data_asq, c(1, 2), person_id = TRUE)
```
getRSSS

Compute a Crosswalk Table

Description

getRSSS is a function for generating 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 for numerical integration.

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

prior_mu_sigma a named list containing prior distribution parameters. All values must be in the theta metric.
  • mu the prior means
  • sigma the covariance matrix
  • sd the prior standard deviations
  • corr the correlation matrix

Examples

## Free calibration without using anchor

o <- runCalibration(data_asq, technical = list(NCYCLES = 1000))
ipar <- mirt::coef(o, IRTpars = TRUE, simplify = TRUE)$items
items <- getItemNames(data_asq, 2)

getRSSS(
  ipar = ipar[items, ],
  theta_grid = seq(-4, 4, .1),
  is_minscore_0 = TRUE,
  prior_mu_sigma = list(mu = 0, sigma = 1)
)
getScaleSum  
*Calculate raw sum scores of a scale*

---

**Description**

`getScaleSum` is a helper function for calculating instrument-wise raw sum scores from response data.

**Usage**

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

**Arguments**

- `data`: a `PROsetta_data` object.
- `scale_idx`: the instrument index to obtain the raw sum scores.

**Examples**

```r
getScaleSum(data_asq, 1)
getScaleSum(data_asq, 2)
```

---

getTheta  
*Obtain theta estimates*

---

**Description**

`getTheta` is a helper function for obtaining theta estimates. Estimates are obtained using an expected a posteriori (EAP) method.

**Usage**

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

Arguments

data  a PROsetta_data object.
ipar  a data.frame containing item parameters.
scale  the index of the scale to use. combined refers to the combined scale. (default = combined)
model  the item model to use. Accepts grm or gpcm. (default = grm)
theta_grid  the theta grid to use for numerical integration. (default = seq(-4, 4, .1))
prior_dist  the type of prior distribution. Accepts normal or logistic. (default = normal)
prior_mean  mean of the prior distribution. (default = 0.0)
prior_sd  SD of the prior distribution. (default = 1.0)

Value

getTheta returns a list containing EAP estimates.

Examples

x <- runLinking(data_asq, method = "FIXEDPAR")
o <- getTheta(data_asq, x$ipar_linked, scale = 1)
o$theta
o$item_idx

o <- getTheta(data_asq, x$ipar_linked, scale = 2)
o$theta
o$item_idx

o <- getTheta(data_asq, x$ipar_linked, scale = "combined")
o$theta
o$item_idx

loadData  Load data from supplied config

Description

loadData is a data loading function for creating a PROsetta_data object, for performing scale linking/equating in the 'PROsetta' package. loadData assumes the response data has been reverse-coded for applicable items.
Usage

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

Arguments

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

Value

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

Description

This is an extension of `plot` to visualize frequency distribution from `PROsetta_data` object.
### plot.PROsetta_data,ANY-method

#### Usage

```r
## 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

- **x**
  - a `PROsetta_data` object.
- **y**
  - unused argument, exists for compatibility with `plot` in the base R package.
- **scale_id**
  - scale ID to plot. combined represents the combined scale. (default = combined)
- **filename**
  - the filename to write if savefile argument is TRUE.
- **title**
  - the title of the figure.
- **xlim**
  - the range of scores to plot.
- **color**
  - the color to fill the histogram. (default = blue)
- **nbar**
  - the number of histogram bars. (default = 20)
- **rug**
  - if TRUE, display the actual distribution of scores below each bar. (default = FALSE)
- **filetype**
  - the type of file to write if savefile argument is TRUE. Accepts pdf, jpeg, png, and tiff. (default = pdf)
- **savefile**
  - if TRUE, save the figure as a file. (default = FALSE)
- **bg**
  - the background color to use when saving the figure as a file. (default = white)
- **width**
  - the width of the plot. (default = 6)
- **height**
  - the height of the plot. (default = 6)
- **pointsize**
  - the point size to use when saving the figure as a file. (default = 12)

#### Examples

```r
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 (optional) theta values to plot on the x-axis. (default = seq(-4, 4, .1))
t_score (optional) set to TRUE to plot T-scores on the x-axis instead of thetas. (default = FALSE)
scale_label (optional) names of each scale. This must have three values. (default = c(1, 2, "Combined"))
color (optional) line colors to plot. This must have three values. (default = c("red", "blue", "black"))
lty (optional) line types to plot. This must have three values. (default = c(3, 2, 1))
PROsetta

Examples

```r
o <- runCalibration(data_asq, technical = list(NCYCLES = 1000))
plotInfo(o, data_asq)
```

Description

**PROsetta** is a caller function for launching a Shiny app locally.

Usage

```r
PROsetta()
```

Examples

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

runCalibration

Description

**runCalibration** is a function for performing item parameter calibration on the response data.

Usage

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

- **data**: a PROsetta_data object. See `loadData` for loading a dataset.
- **dimensions**: the 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)
- **fix_method**: the type of constraints to impose. (default = free)
  - item: for fixed parameter calibration using anchor item parameters
  - theta: for using the mean and the variance obtained from a unidimensional calibration of anchor items
  - free: for free calibration
- **fixedpar**: this argument exists for backward compatibility. TRUE is equivalent to `fix_method` = "item", and FALSE is equivalent to `fix_method` = "free".
- **ignore_nonconv**: if TRUE, return results even when calibration does not converge. If FALSE, raise an error when calibration does not converge. (default = FALSE)
- **verbose**: if TRUE, print status messages. (default = FALSE)
- **...**: additional arguments to pass onto `mirt` in 'mirt' 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

```r
## 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

Run a confirmatory factor analysis

Description

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

Usage

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

Arguments

data a PROsetta_data object. See loadData for loading a dataset.
estimator the estimator to be used. Passed onto cfa in 'lavaan' package. (default = WLSMV)
std.lv 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 cfa. (default = TRUE)
scalewise if TRUE, run analysis for each instrument as well as for the combined instrument. If FALSE, run analysis only for the combined instrument. (default = FALSE)
... additional arguments to pass onto cfa.

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

**Run CTT-based reliability analysis**

**Description**

runClassical is a function for performing a Classical Test Theory (CTT) based reliability analysis.

**Usage**

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

**Arguments**

- **data**: a PROsetta_data object. See loadData for loading a dataset.
- **omega**: if TRUE, also obtain McDonald’s omega using omega in psych package. (default = FALSE)
- **scalewise**: if TRUE, run analysis for each instrument as well as for the combined instrument. If FALSE, run analysis only for the combined instrument. (default = TRUE)
- **...**: additional arguments to pass onto omega.

**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

**Obtain a descriptive statistics table**

**Description**

runDescriptive is a descriptive function for obtaining descriptive statistics for each item in the dataset.

**Usage**

runDescriptive(data = NULL)

**Arguments**

- **data**: a PROsetta_data object. See loadData 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

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

---

runEquateObserved  
**Run Test Equating**

---

Description

runEquateObserved is a function for performing equipercentile equating between two scales. runEquateObserved also produces a concordance table, mapping the observed raw scores from one scale to the scores from another scale.

Usage

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

Arguments

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

scale_from  
the scale ID of the input scale. References to itemmap in data argument. (default = 2)

scale_to  
the scale ID of the target scale to equate to. References to itemmap in data argument. (default = 1)

type_to  
the type of score to use in the target scale frequency table. Accepts raw, tscore, and theta. tscore and theta require argument rsss to be supplied. (default = raw)
runEquateObserved

rsss  the RSSS table to use to map each raw score level onto a t-score or a theta. See runRSSS.

eq_type  the type of equating to be passed onto equate in 'equate' package. (default = equipercentile)

smooth  the type of smoothing method to be passed onto presmoothing in 'equate' package. (default = loglinear)

degrees  the degrees of smoothing to be passed onto presmoothing. (default = list(3, 1))

boot  performs bootstrapping if TRUE. (default = TRUE)

reps  the number of replications to perform in bootstrapping. (default = 100)

verbose  if TRUE, print status messages. (default = FALSE)

...  other arguments to pass onto equate.

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**

---

### Description

**runFrequency** is a descriptive function for obtaining a frequency table from the dataset.

### Usage

```r
class.runFrequency <- runFrequency(data, check_frequency = TRUE)
```

### Arguments

- `data`: a PROsetta_data object. See `loadData` for loading a dataset.
- `check_frequency`: 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

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

---

**runLinking**

---

### Description

**runLinking** is a function for obtaining item parameters from the response data in the metric of supplied anchor item parameters.

### Usage

```r
runLinking(data, method, verbose = FALSE, ...)
```

### Description

**runLinking** is a function for obtaining item parameters from the response data in the metric of supplied anchor item parameters.

### Usage

```r
runLinking(data, method, verbose = FALSE, ...)
```
Arguments

- **data**: a `PROsetta_data` object. See `loadData` for loading a dataset.
- **method**: the linking method to use. Accepts:
  - `MM` for mean-mean method
  - `MS` for mean-sigma method
  - `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' but uses approximation in `runRSSS`
  - `CPFIXEDDIM` for calibrated projection using mean and variance constraints on the anchor dimension

Linear transformation methods (i.e., MM, MS, HB, SL) are performed with `plink` in 'plink' package.

- **verbose**: if `TRUE`, print status messages. (default = `FALSE`)
- **...**: additional arguments to pass onto `mirt` in 'mirt' package.

Value

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

- `constants`: linear transformation constants. Only available when linear transformation methods were used (i.e., MM, MS, HB, SL).
- `ipar_linked`: item parameters calibrated to the response data, and linked to the metric of anchor item parameters.
- `ipar_anchor`: anchor item parameters used in linking.

Examples

```r
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 for generating raw-score to standard-score crosswalk tables from supplied calibrated item parameters.

Usage

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

Arguments

data  a PROsetta_data object. See `loadData` for loading a dataset.
ipar_linked an object returned from `runLinking` or `runCalibration`.
prior_mean prior mean. (default = 0.0)nprior_sd prior standard deviation. (default = 1.0)nmn_theta the lower limit of theta quadratures for numerical integration. (default = -4)nmx_theta the upper limit of theta quadratures for numerical integration. (default = 4)ning the increment between theta quadratures for numerical integration. (default = 0.05)nmn_score 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

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

* datasets
  dataset_asq, 4
dataset_dep, 5
anchor_asq, 4
anchor_asq(dataset_asq), 4
anchor_dep, 5
anchor_dep(dataset_dep), 5
cfa, 17
checkFrequency, 2, 2, 3
test, 16
coeff, 16
coefficients, 16
compareScores, 3, 3
data.frame, 3–5, 8, 11, 12, 19, 21
data_asq, 4
data_asq(dataset_asq), 4
data_dep, 5
data_dep(dataset_asq), 5
data_dep(dataset_dep), 5
dataset, 5
dataset.asq, 4
dataset.asq(dataset_asq), 4
dataset.dep, 5
dataset.dep(dataset_dep), 5
describe, 19
equate, 20
getCompleteData, 6, 6
getEscore, 7, 7
getItemNames, 7, 7
getResponse, 8, 8
getRSSS, 9, 9
getScaleSum, 10, 10
getTheta, 10, 10, 11
itemfit, 16
itemmap.asq, 4
itemmap.asq(dataset_asq), 4
itemmap.dep, 5
itemmap.dep(dataset_dep), 5
itemplot, 16
list, 11, 18, 22, 23
loadData, 3–5, 11, 11, 12, 16–19, 21–23
mirt, 16, 22
omega, 18
plink, 22
plot, 12, 13
plot, PROsetta_data, ANY-method, 12
plotInfo, 14, 14
plotInfo, SingleGroupClass-method
  (plotInfo), 14
preSmoothing, 20
PROsetta, 15, 15
PROsetta_data, 3–8, 10–14, 16–19, 21–23
PROsetta_data-class (loadData), 11
response.asq, 4
response.asq(dataset_asq), 4
response.dep, 5
response.dep(dataset_dep), 5
runCalibration, 14, 15, 15, 16, 23
runCFA, 17, 17
runClassical, 18, 18
runDescriptive, 18, 18, 19
runEquateObserved, 19, 19, 20
runFrequency, 21, 21
runLinking, 21, 21, 22, 23
runRSSS, 20, 22, 23, 23
SingleGroupClass, 14, 16