Package 'PROsetta'

October 12, 2022

Type Package

Title Linking Patient-Reported Outcomes Measures

Version 0.3.5

Date 2021-12-28

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>.

URL https://www.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 7.1.2

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 2022-01-04 00:10:15 UTC

R topics documented:

checkFrequency . . . 2
compareScores . . . 3
dataset_asq . . . 3
dataset_dep . . . 4
getCompleteData . . . 5
getEscore . . . 6
getItemNames . . . 6
getResponse . . . 7
getRSSS . . . 7
getScaleSum . . . 8
getTheta . . . 8
loadData . . . 9
plot,PROsetta_data,ANY-method . . . 10
plotInfo . . . 11
PROsetta . . . 12
runCalibration . . . 13
runCFA . . . 14
runClassical . . . 15
runDescriptive . . . 16
runEquateObserved . . . 16
runFrequency . . . 18
runLinking . . . 19
runRSSS . . . 20

Index **22**

`checkFrequency` *Check frequency table for unobserved response categories*

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 *Compare two sets of scores*

Description

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

Usage

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

Arguments

left	scores on the left side of comparison.
right	scores on the right side of comparison. This is subtracted from 'left'.
type	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 *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.
  - 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.
- 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")
```

---

getCompleteData *Get complete data*

Description

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

Usage

```
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)

getEscore *Calculate expected scores at theta*

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 *Get item names*

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 *Extract scale-wise response*

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 *Obtain EAP estimates*

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

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

Value

getTheta returns a list containing EAP estimates.

---

loadData *Load data from supplied config*

Description

loadData is a data loading function to create a PROsetta_data object, for scale linking/equating with 'PROsetta' package. Response data is assumed to be reverse-coded for applicable items.

Usage

```
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.

---

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

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 (default) represents the combined scale.
filename	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.
nbar	the number of histogram bars.
rug	if TRUE, display the actual distribution of scores below each bar.
filetype	the type of file to write if 'savefile' argument is TRUE. Accepts 'pdf', 'jpeg', 'png', and 'tiff'.
savefile	if TRUE, save the figure as a file.
bg	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,
  verbose = FALSE,
  ...
)
```

Arguments

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

```
## 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 to perform 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 scale as well as for the combined scale. If FALSE, run analysis only for the combined scale. (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 to perform Classial 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 scale as well as for the combined scale. If FALSE, run analysis only for the combined scale. (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 to obtain 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

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

runEquateObserved *Run Test Equating*

Description

runEquateObserved is a function to perform equipercentile 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 = "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)
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 *Obtain a frequency table*

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 *Run Scale Linking*

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, verbose = FALSE, ...)
```

Arguments

data	a PROsetta_data object. See loadData for loading a dataset.
method	the type of linking to perform. Accepts: • 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 runLinking but uses approximation in runRSSS • CPFIXEDDIM for calibrated projection using mean and variance constraints on the anchor dimension Linear transformation methods 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. 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

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)
prior_sd	prior standard deviation. (default = 1.0)
min_theta	the lower limit of theta grid. (default = -4)
max_theta	the upper limit of theta grid. (default = 4)
inc	the increment to use in theta grid. (default = 0.05)
min_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

```
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, *14*
checkFrequency, *2*, 2
coef, *13*
compareScores, *3*, 3

data.frame, *3–5*, *7*, *9*, *16*, *18*
data_asq, *4*
data_asq *(*dataset_asq*)*, 3
data_dep, *5*
data_dep *(*dataset_dep*)*, 4
dataset_asq, 3
dataset_dep, 4
describe, *16*

equate, *17*

getCompleteData, *5*, 5
getEscore, *6*, 6
getItemNames, *6*, 6
getResponse, *7*, 7
getRSSS, *7*, 7
getScaleSum, *8*, 8
getTheta, *8*, 8, *9*
guiPROsetta *(*PROsetta*)*, 12

itemfit, *13*
itemmap_asq, *3*
itemmap_asq *(*dataset_asq*)*, 3
itemmap_dep, *4*
itemmap_dep *(*dataset_dep*)*, 4
itemplot, *13*

list, *9*, *15*, *19*, *20*
loadData, *2*, *4*, *5*, *9*, 9, *10*, *13–20*

mirt, *13*, *19*

omega, *15*

plink, *19*
plot, *10*
plot,PROsetta_data,ANY-method, 10
plotInfo, *11*, 11
plotInfo,SingleGroupClass-method
        *(*plotInfo*)*, 11
presmoothing, *17*
PROsetta, *12*, 12
PROsetta_data, *2*, *4–10*, *12–20*
PROsetta_data-class *(*loadData*)*, 9

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

SingleGroupClass, *12*, *13*