Package: pairwise (via r-universe)

October 31, 2026

Type Package

Maintainer Joerg-Henrik Heine <jhheine@googlemail.com>

Date 2026-10-30

Version 0.6.4-0

Encoding UTF-8

License GPL-3

Imports graphics, stats, methods

Depends R (>= 3.5.0)

Title Rasch Model Parameters by Pairwise Algorithm

Description Performs the explicit calculation -- not estimation! -- of the Rasch item parameters for dichotomous and polytomous item responses, using a pairwise comparison approach. Person parameters (WLE) are calculated according to Warm's weighted likelihood approach.

Suggests psych, exhaustiveRasch

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Joerg-Henrik Heine [aut, cre, ctb], Christian Grebe [ctb]

Repository https://cran.r-universe.dev

Date/Publication 2026-10-31 01:00:02 UTC

RemoteUrl https://github.com/cran/pairwise

RemoteRef HEAD

RemoteSha 785397f6399bedd206f6216f2290151162526282

Contents

andersentest.pers 3
bfi_cov 4
bfiN 5
bfiN_miss 7

catprob . . . 8
cog . . . 9
cogBOOKLET . . . 9
deltapar . . . 10
DEU_PISA2012 . . . 11
esc . . . 12
ftab . . . 14
gif . . . 15
grm . . . 17
iff . . . 20
KCT . . . 22
kft5 . . . 23
logLik.pers . . . 24
lrtest.pers . . . 24
make.incidenz . . . 25
Neoffi5 . . . 26
pair . . . 27
pairSE . . . 30
pairwise.item.fit . . . 31
pairwise.person.fit . . . 32
pairwise.S . . . 33
pairwise.SepRel . . . 36
pairwise_PACKAGE . . . 37
pers . . . 39
plot.grm . . . 41
plot.pair . . . 43
plot.pairSE . . . 44
plot.pers . . . 45
plot.rfa . . . 46
ptbis . . . 47
Q . . . 48
q3 . . . 49
Qtest . . . 50
residuals.pers . . . 53
rfa . . . 53
sim200x3 . . . 55
simra . . . 56
summary.grm . . . 57
summary.pair . . . 58
summary.pairS . . . 58
summary.pairSE . . . 59
summary.pairwiseSepRel . . . 59
summary.pers . . . 60
summary.pifit . . . 60
summary.ppfit . . . 61
summary.q3 . . . 62
summary.rfa . . . 63
tff . . . 63

Index **67**

`andersentest.pers` *Andersen's Likelihood Ratio Test for Object of class "pers"*

Description

The Andersen likelihood ratio test is based on splitting the dataset into subgroups of persons. One can argue that it is a significance testable version of the more descriptive graphical model check - see `grm`.

Usage

```
andersentest.pers(
  pers_obj,
  split = "median",
  splitseed = "no",
  pot = NULL,
  zerocor = NULL
)
```

Arguments

`pers_obj`	an object of class`"pers"` - see function `pers`.
`split`	Specifies the splitting criterion. Basically there are three different options available - each with several modes - which are controlled by passing the corresponding character expression to the argument. 1) Using the rawscore for splitting into subsamples with the following modes: `split = "median"` median raw score split - high score group and low score group; `split = "mean"` mean raw score split - high score group and low score group. Finaly `split = "score"` that is splitting `daten` into as many subsamples as there are raw score groups - discarding min and max (theoretical) score group - which matches the concept proposed by Andersen (1973). 2) Dividing the persons in `daten` into subsamples with equal size by random allocation with the following modes: `split = "random"` (which is equivalent to `split = "random.2"`) divides persons into two subsamples with equal size. In general the number of desired subsamples must be expressed after the dot in the character expression - e.g. `split = "random.6"` divides persons into 6 subsamples (with equal size) by random allocation etc. 3) The third option is using a manifest variable as a splitting criterion. In this case a vector with the same length as number of cases in `daten` must be passed to the argument grouping the data into subsamples. This vector should be coded as `"factor"` or a `"numeric"` integer vector with min = 1.
`splitseed`	numeric, used for `set.seed(splitseed)` for random splitting - see argument `split`.
`pot`	optional argument, at default (`pot=NULL`) setting is read from `pers_obj` - see description for `pair`.

`zerocor` optional argument, at default (`zerocor=NULL`) setting is read from `pers_obj` - see description for `pair`.

pot=pers_obj$pair$fuargs$pot, zerocor=pers_obj$pair$fuargs$zerocor

Details

Andersen (1973) proposed to split the dataset by [raw] score groups, which can be achieved setting the argument `split = "score"`. However as pointed out by Rost (2004) there might be several different splitting criteria for testing subsample invariance of the raschmodel. Thus the argument `split` provides some other options for splitting the data - see description of arguments.

Value

A (list) object of class `"andersentest.pers"` ...

References

Andersen, E. B. (1973). A goodness of fit test for the rasch model. *Psychometrika, 38*(1), 123–140.

Rost, J. (2004). *Lehrbuch Testtheorie - Testkonstruktion* (2 nd Ed.) Huber: Bern.

Examples

```
## Not run:
data(bfiN) # loading example data set
data(bfi_cov) # loading covariates to bfiN data set
model <- pers(pair(bfiN,m=6))
andersentest.pers(model, split = bfi_cov$gender)
andersentest.pers(model, split = "random")
andersentest.pers(model, split = "median")
### unsing simulated data:
data("sim200x3")
model2 <- pers(pair(sim200x3))
andersentest.pers(model2, split = "median")

## End(Not run)
```

---

bfi_cov *Covariates to the bfiN Data*

Description

Covaraites to the data from 2800 subjects answering to 5 neuroticism items of the bfi dataset originally included in the R-package {psych} - see https://cran.r-project.org/package=psych.

Usage

```
data(bfi_cov)
```

Format

A "data.frame" containing 3 variables (gender, education, and age) for 2800 obsevations.

Details

The covariates are in the same row (person) order as the responses to the 5 neuroticism items in the seperate datasets bfiN and bfiN_miss. The coding is as follows:

gender Males = 1, Females =2

education 1 = HS, 2 = finished HS, 3 = some college, 4 = college graduate 5 = graduate degree

age age in years

Source

https://cran.r-project.org/package=psych

References

Revelle, William (2015), psych: Procedures for Psychological, Psychometric, and Personality Research.*R package version 1.5.1*

Examples

```
data(bfi_cov)
dim(bfi_cov)
#############################################################
names(bfi_cov) # show all variable names of data
```

---

bfiN *5 polytomous personality items*

Description

Data from 2800 subjects answering to 5 neuroticism items with 6 answer categories (0-5) of the bfi dataset originally included in the R-package {psych} - see https://cran.r-project.org/package=psych.

Usage

```
data(bfiN)
```

Format

A "data.frame" containing 5 variables and 2800 obsevations.

Details

The other variables from the original `bfi` dataset were skipped and the categories are 'downcoded' to '0,1,2,3,4,5' to have a simple, ready to use example data frame. For further Information on the original dataset see R-package `{psych}`.

The category meanings (after downcoding) are as follows:

`score 0` Very Inaccurate

`score 1` Moderately Inaccurate

`score 2` Slightly Inaccurate

`score 3` Slightly Accurate

`score 4` Moderately Accurate

`score 5` Very Accurate

The Item meanings are as follows:

`N1` Get angry easily.

`N2` Get irritated easily.

`N3` Have frequent mood swings.

`N4` Often feel blue.

`N5` Panic easily.

The covariates like gender, education and age are in a seperate dataset `cov_bfi`

Source

https://cran.r-project.org/package=psych

References

Revelle, William (2015), psych: Procedures for Psychological, Psychometric, and Personality Research.*R package version 1.5.1*

Examples

```
data(bfiN)
dim(bfiN)
################################################################
names(bfiN) # show all variable names of data.frame bfiN
range(bfiN,na.rm=TRUE) # checking the valid response range
```

`bfiN_miss` *5 polytomous personality items*

Description

Data from 2800 subjects answering to 5 neuroticism items with 6 answer categories (0-5) of the bfi dataset originally included in the R-package {psych} with artificial missing data (see details).

Usage

```
data(bfiN_miss)
```

Format

A "data.frame" containing 5 variables and 2800 obsevations.

Details

This dataset is the same like the dataset {bfiN} included in this package, exept for the amount of missing data, which were additional created in that way, having aprox. 15% missing for each of the 5 variables by random.

The other variables from the original bfi dataset were skipped and the categories are 'downcoded' to '0,1,2,3,4,5' to have a simple, ready to use example data frame. For further Information on the original dataset see R-package {psych}. The covariates like gender, education and age are in a seperate dataset `cov_bfi`

Source

https://cran.r-project.org/package=psych

References

Revelle, William (2015), psych: Procedures for Psychological, Psychometric, and Personality Research.*R package version 1.5.1*

Examples

```
data(bfiN_miss)
dim(bfiN_miss)
#############################################################
names(bfiN_miss) # show all variable names of data.frame bfiN_miss
range(bfiN_miss,na.rm=TRUE) # checking the valid response range
colSums(is.na(bfiN_miss))/dim(bfiN_miss)[1] # percentage of missing per variable
```

catprob *Category Probability Plots*

Description

plotting function for plotting category probability curves.

Usage

```
catprob(pair_obj, itemnumber = 1, ra = 4, plot = TRUE, ...)
```

Arguments

pair_obj	an object of class "pair" as a result from function pair.
itemnumber	an integer, defining the number of the item to plot the respective category probability for. This is set to an arbitrary default value of itemnumber = 1 to avoid error messages when you forget to choose an item to plot the expected score curves for.
ra	an integer, defining the (logit) range for x-axis
plot	a logical (default plot = TRUE), defining wether to supress plotting an just return a matrix of category probabilities
...	arguments passed to plot

Details

no details in the moment.

Value

a plot or a matrix with category probabilities.

Examples

```
########
data(sim200x3)
result <- pair(sim200x3)
catprob(pair_obj = result, itemnumber = 2 )
data(bfiN)
result <- pair(bfiN)
catprob(pair_obj = result, itemnumber = 3 )
```

cog *Math PISA (2003) data*

Description

Data from the german sample of the PISA 2003 survey, containing 31 dichotomous items from the math task.

Usage

```
data(cog)
```

Format

A data frame containing 34 variables and 4660 obsevations.

Details

The first 3 variables are ID variables. For further Information on variables and their meaning see the codebook PDF file available at `https://www.oecd.org/pisa/pisaproducts/PISA12_cogn_codebook.pdf`

References

Database - PISA 2003, *Downloadable Data*, `https://www.oecd.org/pisa/data/pisa2012database-downloadabledata`

Examples

```
data(cog)
dim(cog)
#############################################################
names(cog) # show all variable names of data.frame cog
names(cog[,4:34]) # show the variable names of the math items
names(cog[,1:3]) # show the variable names of the ID variables
```

cogBOOKLET *Booklet allocation table for Math PISA (2003) data*

Description

a `data.frame` containing a booklet allocation table for the cognitive Data `cog` in this package, which holds 31 dichotomous items from the math task from the german sample of the PISA 2003 survey.

Usage

```
data(cogBOOKLET)
```

Format

A data.frame containing 31 rows.

Details

For further Information on variables and their meaning see the codebook PDF file available at https://www.oecd.org/pisa/pisaproducts/PISA12_cogn_codebook.pdf

References

Database - PISA 2003, *Downloadable Data*, https://www.oecd.org/pisa/data/pisa2012database-downloadabledata

Examples

```
data(cogBOOKLET)
cogBOOKLET
```

---

deltapar *Compute delta parameters from thurstonian thresholds*

Description

Calculation of delta parameters or rather item step parameters from thurstonian threshold parameters returned by the function pair.

Usage

```
deltapar(object, sigma = TRUE)
```

Arguments

object	an object of class "pair" as resulting from item parameter calculation using the function pair.
sigma	a logical whether to return item difficulties (sigma) or not

Details

The "Thurstone threshold" or rather thurstonian threshold for a category corresponds to a point on the latent variable at which the probability of being observed in that category or above equals that of being observed in the categories below. Thus these thurstonian threshold parameters can be interpreted in an strait forward and easy way. However, some other computer programs related to Rasch analysis don't return thurstonian threshold parameters from their estimation procedure, but rather so called delta parameters for the item steps. The later are also known as "step measures", "step calibrations", "step difficulties", "tau parameters", and "Rasch-Andrich thresholds". For a better comparability between different Rasch software and estimation procedures the thurstonian threshold parameters can be converted into delta or rather items step parameters.

Value

If `sigma=TRUE` an object of class `c("data.frame", "deltapar")` containing delta parameters for items and their difficultie (first column). Otherwise a matrix containing only the delta parameters.

References

Linacre J.M. (1992). Rasch-Andrich Thresholds and Rasch-Thurstone Thresholds. *Rasch Measurement Transactions*, 5:4, 191. https://www.rasch.org/rmt/rmt54r.htm

Linacre J.M. (2001). Category, Step and Threshold: Definitions & Disordering. *Rasch Measurement Transactions*, 15:1, 794. https://www.rasch.org/rmt/rmt151g.htm

Adams, R. J., Wu, M. L., & Wilson, M. (2012). The Rasch Rating Model and the Disordered Threshold Controversy. *Educational and Psychological Measurement, 72*(4), 547–573. https://doi.org/10.1177/0013164411

Linacre J.M. (2006). Item Discrimination and Rasch-Andrich Thresholds. *Rasch Measurement Transactions*, 20:1, 1054. https://www.rasch.org/rmt/rmt201k.htm

Examples

```
######################
data(sim200x3) # loading reponse data
ip <- pair(sim200x3,m = c(2,3,3)) # compute item parameters
summary(ip) # have a look at the results (thurstonian thresholds)
deltapar(ip) # compute delta parameters from these
```

DEU_PISA2012 *Data from PISA 2012 - German Sample*

Description

Selectetd data for 5001 'subjects' who participated in the PISA 2012 survey.

Usage

```
data(DEU_PISA2012)
```

Format

A list containing ... .

Details

The data is based on freely down loadable data on the official OECD page - see source. The general structure of the data in list format, is described in an PDF document available in the User guides, package vignettes and other documentation section.

References

Database - PISA 2003, *Downloadable Data*, `https://www.oecd.org/pisa/data/pisa2012database-downloadabledata`

Examples

```
############################################################
data(DEU_PISA2012)
str(DEU_PISA2012)
```

---

esc *Expected score curves for one or several items*

Description

plotting function for plotting model-implied expected score curves from a Rasch or Partial Credit Model together with empirical item-score means. Several items can be drawn in one common coordinate system.

Usage

```
esc(
  pers_obj,
  itemnumber = NULL,
  integ = 6,
  group.method = c("auto", "theta_levels", "equal_n"),
  ra = 4,
  nodes = 100,
  lwd = 2,
  lty = 1,
  col = NULL,
 bw.lty = c("solid", "dashed", "dotted", "dotdash", "longdash", "twodash", "44", "13",
    "1343", "73", "2262", "12223242"),
  empirical = TRUE,
  empirical.lty = 2,
  empirical.lwd = lwd + 0.3,
 pch = c(1, 2, 0, 5, 6, 3, 4, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
    23, 24, 25),
  show.legend = TRUE,
  legend.position = "topleft",
  type.legend.position = "bottomright",
  main = NULL,
  sub = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

pers_obj	Object of class "pers", returned by pairwise::pers().
itemnumber	NULL, a numeric vector of item positions, or a character vector of item names. The default NULL selects all items contained in pers_obj$pair$threshold.

integ	Positive integer giving the requested number of empirical ability groups, or "all" for the original score-wise display.
group.method	Method for numeric integ: "auto" chooses "theta_levels" when ties are substantial and otherwise "equal_n"; "theta_levels" groups adjacent distinct WLE values without splitting ties; "equal_n" forms approximately equally sized person groups.
ra	Positive number defining the theoretical theta range -ra to ra.
nodes	Number of theta evaluation points.
lwd	Line width of model-implied curves.
lty	Line type(s) of model-implied curves in colour plots.
col	Item colour(s). If NULL, colours are generated automatically. If all selected items are drawn with col = "black", the items are automatically distinguished by different line types.
bw.lty	Line types used to distinguish items in a black-and-white plot. They are recycled when more items than line types are selected.
empirical	Logical; add empirical curves?
empirical.lty	Line type of empirical curves.
empirical.lwd	Line width of empirical curves.
pch	Plotting symbols used to distinguish items at the empirical group means. The vector is recycled when necessary.
show.legend	Logical; draw legends?
legend.position	Position of item legend.
type.legend.position	Position of curve-type legend.
main, sub	Optional plot title and subtitle.
verbose	Logical; print information about the empirical grouping?
...	Further arguments passed to plot().

Details

The empirical ability groups are formed once for all selected items. By default, ties in WLE estimates are not split across groups. This is important for Rasch WLEs, which often take only a small number of distinct values.

Value

Invisibly returns a list containing grouping information and all theoretical and empirical plotting values.#'

Examples

```
data(bfiN)
pers_obj <- pers(pair(bfiN))
esc(pers_obj, integ = 6)             # plot all items
esc(pers_obj, itemnumber = 1:5, integ = 6)
esc(pers_obj, itemnumber = 1:5, integ = 6, group.method = "theta_levels")
esc(pers_obj, itemnumber = 1:3, integ = "all")
esc(pers_obj, col = "black")             # black-and-white
```

---

ftab *Tabulating Answer Categories in Data*

Description

function tabulating (answer) categories in X.

Usage

```
ftab(X, catgories = NULL, na.omit = FALSE)
```

Arguments

X	Data as a "matrix", a "data.frame" or even a "vector" or "factor". "vector" or "factor" are coerced to a "data.frame" with one column.
catgories	optional a vector ("numeric" or "character") containig the categories to tabulate. At default (catgories=NULL) the fuction looks for unique categories in X.
na.omit	logical (default: na.omit=FALSE) wether to return frequencies for missing values, NAs.

Details

X can either be a ("numeric" or "character") "matrix" containing response vectors of persons (rows) or a "data.frame" containing "numeric", "character" or "factor" variables (columns).

Value

a "matrix" with category frequencies

Examples

```
########
data(bfiN)
ftab(bfiN)
data(sim200x3)
ftab(sim200x3)
```

gif *Graphical item-fit plots for one or several items*

Description

Plots model-implied category probability curves from a Rasch or Partial Credit Model together with empirical category probabilities. Several items can be drawn in one common coordinate system.

Usage

```
gif(
  pers_obj,
  itemnumber = NULL,
  ra = 4,
  nodes = 100,
  integ = "raw",
  group.method = c("auto", "theta_levels", "equal_n"),
  kat = "all",
  col = NULL,
 item.lty = c("solid", "dashed", "dotted", "dotdash", "longdash", "twodash", "44", "13",
    "1343", "73", "2262", "12223242"),
  category.lty = 1:6,
 item.pch = c(1, 2, 0, 5, 6, 3, 4, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
    22, 23, 24, 25),
  category.pch = 1:25,
  model.marker.every = 15,
  model.marker.cex = 0.55,
  model.lwd = 1.5,
  empirical.lwd = 1.5,
  empirical = TRUE,
  show.legend = TRUE,
  item.legend.position = "topleft",
  category.legend.position = "topright",
  type.legend.position = "bottomright",
  label.model.categories = NULL,
  label.empirical.categories = NULL,
  main = NULL,
  sub = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

pers_obj	an object of class "pers" as a result from function pers.
itemnumber	NULL, a numeric vector of item positions, or a character vector of item names. The default NULL selects all items contained in pers_obj$pair$threshold.

ra	Positive number defining the theoretical theta range -ra to ra.
nodes	Number of theta evaluation points for the model curves.
integ	Either a positive integer giving the requested number of common empirical ability groups, or "raw" for grouping by the raw score stored in pers_obj$pers$raw.
group.method	Method used when ‘integ‘ is numeric: "auto", "theta_levels", or "equal_n".
kat	Either "all" or a numeric vector of category numbers. For compatibility with the original function, category number 1 denotes the first response category (score 0), category number 2 denotes score 1, etc.
col	Item colour(s). If NULL, colours are generated automatically. If all selected items are drawn with col = "black", items are automatically distinguished by different line types.
item.lty	Line types used to distinguish items in a black-and-white plot. They are recycled when necessary.
category.lty	Line types used to distinguish response categories in colour plots. In black-and-white plots, categories are distinguished by plotting symbols instead.
item.pch	Plotting symbols used for the empirical points of the selected items. The vector is recycled when necessary.
category.pch	Plotting symbols used to distinguish response categories on model curves in black-and-white plots.
model.marker.every	Approximate spacing, in theta nodes, between category symbols placed on model curves in black-and-white plots.
model.marker.cex	Size of those model-curve category symbols.
model.lwd	Line width of model-implied probability curves.
empirical.lwd	Line width of empirical probability curves.
empirical	Logical; add empirical category probabilities?
show.legend	Logical; draw legends?
item.legend.position	Position of the item legend.
category.legend.position	Position of the response-category legend.
type.legend.position	Position of the model/empirical legend.
label.model.categories	NULL or logical. If NULL, category labels at the model-curve maxima are shown for a single item and omitted for several items.
label.empirical.categories	NULL or logical. If NULL, labels at the empirical points are shown for a single item and omitted for several items.
main, sub	Optional plot title and subtitle.
verbose	Logical; print information about the empirical grouping?
...	Further arguments passed to plot().

Details

For several items in colour, colour and the empirical plotting symbol identify the item, whereas line type identifies the response category. With `col = "black"`, line type and the empirical plotting symbol identify the item; sparse symbols on the model curves identify the response category. Model-implied probabilities are shown as lines; empirical probabilities are superimposed as lines with item-specific points. The empirical ability groups are formed once for all selected items. For a numeric value of `integ`, ties in WLE estimates are not split across groups when `group.method = "theta_levels"` is used. The default `"auto"` chooses this method when the WLE distribution contains substantial ties.

Value

Invisibly returns a list containing grouping information and all theoretical and empirical plotting values.

Examples

```
########
data(bfiN)
pers_obj <- pers(pair(bfiN))
#### plot
# gif(pers_obj)                                  # all items
# gif(pers_obj, itemnumber = 1:4)               # selected items
# gif(pers_obj, itemnumber = c("i1", "i2"))   # item names
# gif(pers_obj, itemnumber = 1:4, integ = 6)
# gif(pers_obj, itemnumber = 1:4, integ = 6,
#     group.method = "theta_levels", kat = 1)
# gif(pers_obj, itemnumber = 1:4, col = "black")
#################################################
```

`grm` *Graphical Model Check*

Description

This function makes the basic calculations for the graphical model check for dicho- or polytomous item response formats. It is more or less a wraper function, internaly calling the function `pairSE`. Several splitting options are available (see arguments).

Usage

```
grm(
  daten,
  m = NULL,
  w = NULL,
  split = "random",
  splitseed = "no",
  verbose = FALSE,
```

```
  ...
)
```

Arguments

daten	a data.frame or matrix with optionaly named colums (names of items), potentially with missing values, comprising polytomous or dichotomous (or mixed category numbers) responses of n respondents (rows) on k items (colums) coded starting with 0 for lowest category to *m*-1 for highest category, with *m* beeing a vector (with length k) with the number of categories for the respective item.
m	an integer (will be recycled to a vector of length k) or a vector giving the number of response categories for all items - by default m = NULL, m is calculated from data, assuming that every response category is at least once present in data. For sparse data it is strongly recomended to explicitly define the number of categories by defining this argument.
w	an optional vector of case weights.
split	Specifies the splitting criterion. Basically there are three different options available - each with several modes - which are controlled by passing the corresponding character expression to the argument. 1) Using the rawscore for splitting into subsamples with the following modes: split = "median" median raw score split - high score group and low score group; split = "mean" mean raw score split - high score group and low score group. 2) Dividing the persons in daten into subsamples with equal size by random allocation with the following modes: split = "random" (which is equivalent to split = "random.2") divides persons into two subsamples with equal size. In general the number of desired subsamples must be expressed after the dot in the character expression - e.g. split = "random.6" divides persons into 6 subsamples (with equal size) by random allocation etc. 3) The third option is using a manifest variable as a splitting criterion. In this case a vector with the same length as number of cases in daten must be passed to the argument grouping the data into subsamples. This vector should be coded as "factor" or a "numeric" integer vector with min = 1.
splitseed	numeric, used for set.seed(splitseed) for random splitting - see argument split.
verbose	logical, if verbose = TRUE (default) a message about subsampling is sent to console when calculating standard errors.
...	additional arguments nsample, size, seed, pot for caling pairSE are passed through - see description for pairSE.

Details

The data is splitted in two or more subsamples and then item thresholds, the parameter (Sigma) and their standard errors (SE) for the items according the PCM are calculated for each subsample. Additional arguments (see description of function pairSE) for parameter calculation are passed through.

WARNING: When using data based on booklet designs with systematically missing values (by design) you have to ensure that in each of the booklet the maximum raw value to reach is equal while using the raw value as splitting criterion.

Value

A (list) object of class `c("grm","list")` containing the item difficulty parameter sigma and their standard errors for two or more subsamples.

A note on standard errors

Estimation of standard errors is done by repeated calculation of item parameters for subsamples of the given data. This procedure is mainly controlled by the arguments `nsample` and `size` (see arguments). With regard to calculation time, the argument `nsample` is the 'time killer'. On the other hand, things (estimation of standard errors) will not necessarily get better when choosing large values for `nsample`. For example choosing `nsample=400` will only result in minimal change for standard error estimation in comparison to (`nsample=30`) which is the default setting (see examples).

References

description of function `pairSE{pairwise}`.

Examples

```
data(bfiN) # loading example data set

data(bfi_cov) # loading covariates to bfiN data set

# calculating itemparameters and SE for two random allocated subsamples
grm_gen <- grm(daten=bfiN, split = bfi_cov$gender)
summary(grm_gen)
#### plot(grm_gen)

grm_med <- grm(daten=bfiN, split = "median")
summary(grm_med)
#### plot(grm_med)

grm_ran<-grm(daten=bfiN, split = "random")

summary(grm_ran)

# some examples for plotting options
# plotting item difficulties for two subsamples against each other
# with elipses for a CI = 95% .
#### plot(grm_ran)

# using triangles as plotting pattern
#### plot(grm_ran,pch=2)

#plotting without CI ellipses
#### plot(grm_ran,ci=0,pch=2)
```

```
# plotting with item names
#### plot(grm_ran,itemNames=TRUE)

# Changing the size of the item names
#### plot(grm_ran,itemNames=TRUE, cex.names = 1.3)

# Changing the color of the CI ellipses
plot(grm_ran,itemNames=TRUE, cex.names = .8, col.error="green")

###### example from details section 'Some Notes on Standard Errors' ########
## Not run:
grm_def<-grm(daten=bfiN, split = "random",splitseed=13)
plot(grm_def)
######
grm_400<-grm(daten=bfiN, split = "random", splitseed=13 ,nsample=400)
plot(grm_400)

## End(Not run)
```

`iff` *Item information function*

Description

plotting function for plotting the Item information function(IIF). Item information function using the Rasch/PCM Fisher information.

Usage

```
iff(
  pair_obj,
  itemnumber = 1,
  x = NULL,
  ra = 4,
  nodes = 161,
  plot = TRUE,
  cat = FALSE,
  lwd = 2,
  col = 1,
  lty = 1,
  main = NULL,
  xlab = "Theta (ability logits)",
  ylab = "Information",
  ylim = NULL,
  bty = "n",
  ...
)
```

Arguments

pair_obj	an object of class "pair" as a result from function pair.
itemnumber	One numeric item position or one item name.
x	Numeric theta values. If NULL, 'nodes' points from -'ra' to 'ra' are used. A single theta suppresses plotting.
ra, nodes	Theta range and number of evaluation points when x is NULL.
plot	Logical; draw the plot?
cat	Logical; plot or return category-information contributions?
lwd, col, lty	Graphical parameters for the item-information curve.
main, xlab, ylab, ylim, bty	Plot settings.
...	Further arguments passed to plot().

Details

This version of the funktion uses the conditional variance of the item score. For a dichotomous Rasch item, I_i(theta) = p(theta) * (1 - p(theta)); for a Partial Credit item, I_i(theta) = Var(X_i | theta).

Value

Item information, or category-information contributions if cat=TRUE.

Examples

```
########
data(sim200x3)
result <- pair(sim200x3)
iff(pair_obj = result)
#' # IFF plot for Item No. 2
iff(pair_obj = result, itemnumber = 2 )
# IFF plot for Categories of Item No. 2
iff(pair_obj = result, itemnumber = 2 ,cat=TRUE)
# IFF at theta=0 for Item No. 2
iff(pair_obj = result, itemnumber = 2 ,x=0)
# IFF at theta=0 for Categories of Item No. 2
iff(pair_obj = result, itemnumber = 2 ,x=0,cat=TRUE)
# IFF of Item No. 2 for a given range of thetas
iff(pair_obj = result, itemnumber = 2 ,x=seq(0,4,.1))
# ... etc.
iff(pair_obj = result, itemnumber = 2 ,x=seq(0,4,.1),cat=TRUE)
##### examples with other data ...
data(bfiN)
result <- pair(bfiN)
iff(pair_obj = result, itemnumber = 3 )
iff(pair_obj = result, itemnumber = 3 ,cat=TRUE)
```

KCT *Knox Cube Test Data from Wright & Stone (1979)*

Description

Data from the Book 'Best Test Design' from Wright & Stone (1979, p. 31, table 2.3.1) comprising responses from 35 subjects scored in 18 dichotomous items.

Usage

```
data(KCT)
```

Format

A "data.frame" containing 18 numeric variables (coded 0,1) and 35 obsevations.

Details

The so called 'Knox Cube Test' was initially developed as a cube imitation test around 1913 by Howard A. Knox as a nonverbal test of intelligence to screen and identify potential immigrants with mental deficits at the Ellis Island immigration station in New York Harbor – see Richardson (2005) for a historical review.

Quoted from Wright & Stone (1979):

"*Success on this subtest requires the application of visual attention and short-term memory to a simple sequencing task. It appears to be free from school-related tasks and hence to be an indicator of nonverbal intellectual capacity.*" (Wright & Stone 1979, p. 28).

References

Wright, B. D. & Stone, M. H. (1979). *Best Test Design: Rasch Measurement*. Chicago: MESA Press.

Richardson, J. T. E. (2005). Knox's cube imitation test: A historical review and an experimental analysis. *Brain and Cognition, 59*(2), 183–213. https://doi.org/10.1016/j.bandc.2005.06.001

Examples

```
data(KCT)
dim(KCT)
############# some item calibrations ###############
data(KCT)
IP_pair <- pair(daten = KCT[,4:17], m = 2)
summary(IP_pair)
#####################################################
#####################################################
#########MIKE error message never received##########
#####################################################
#####################################################
```

kft5 *Dichotomous example data in Rost 2004*

Description

Data for 300 subjects answering to 5 dichotomous items out of 'Kognitiver Fähigkeits Test' [Cognitive Skills Test] (KFT - Gaedike & Weinläder, 1976) . This data is used as an example in the textbook by J. Rost (2004) to demonstrate some principles of rasch measurement.

Usage

```
data(kft5)
```

Format

A "matrix" containing 5 columns (variables) and 300 rows (obsevations).

Details

The instrument KFT and the data are described in Rost (2004) at page 95.

References

Rost, J. (2004). *Lehrbuch Testtheorie - Testkonstruktion* (2 nd Ed.) Huber: Bern.

Heller, K, Gaedike, A.-K & Weinläder, H. (1976). *Kognitiver Fähigkeits-Test (KFT 4-13).* Weinheim: Beltz.

Examples

```
data(kft5)
dim(kft5)
###########
# frequencies
ftab(kft5)
# Itemparameter to be compared with Rost (2004), page 120.
summary(pair(kft5))
# Itemparameter to be compared with Rost (2004), page 120.
summary(pers(pair(kft5)))
```

logLik.pers *S3 logLik for Object of class "pers"*

Description

S3 logLik method to extract the log-likelihood for object of class"pers"

Usage

```
## S3 method for class 'pers'
logLik(object, sat = FALSE, p = FALSE, ...)
```

Arguments

object	object of class"pers"
sat	a "logical" with default set to sat=FALSE to return the Log-Likelihood of the data for the unrestricted modell based on parameters estimated with function pers. If set to sat=TRUE the Log-Likelihood of the saturated model is returned instead.
p	a "logical" with default set to p=FALSE to return the category propabilities for the empirical data.
...	not used jet.

lrtest.pers *Likelihood Ratio Test for Object of class "pers"*

Description

Function to perform a likelihood ratio test for the estimated model 'against' the saturated model for object of class"pers".

Usage

```
lrtest.pers(object, ...)
```

Arguments

object	an object of class"pers" - see function pers.
...	not used jet.

make.incidenz *Converting a booklet allocation table into a incidence matrix*

Description

This function converts a booklet allocation table (like in cogBOOKLET) into a incidence matrix used in the function pers.

Usage

```
make.incidenz(tab, bookid, item_order = NULL, info = FALSE)
```

Arguments

tab	a booklet allocation table as a data.frame. The first column is assumed to contain the item names as a character vector (not a factor!) the other columns must be integer vectors containing the information in which booklet(s) the respective item is allocated.
bookid	a integer vector with the same length as the number of persons in the response data giving the information which booklet was assigned to each person.
item_order	optional a character vector with the item names in the order of the items in the response data (from first to last column in the response data). By default it is assumend that the item order in the booklet allocation table is already the same as in the response data.
info	logical default: info=FALSE to return just the incidence matrix. If set to info=TRUE more detailed information about the booklet design ist returned.

Details

It is assumed that there is an equal replicate factor for each item used, when constructing the bookletdesign - so every items occures with the same frequency over all booklets of the entire set of booklets.

Value

an incidence matrix as an object of class "matrix" with 0,1 coding or a "list" with detailed information.

Examples

```
########################
data(cog);data(cogBOOKLET) # loading reponse and allocation data
table(cog$BOOKID)# show n persons per booklet
names(table(c(as.matrix(cogBOOKLET[,2:5])))) # show booklets in allocation data
d<-(cog[cog$BOOKID!=14,]) # skip persons which got booklet No.14.
inc<-make.incidenz(tab=cogBOOKLET, bookid=d$BOOKID) # make just the incidence matrix
inc
```

```
make.incidenz(tab=cogBOOKLET, bookid=d$BOOKID, info=TRUE) # get some info too
# in this case not necessary but just to show
# using the (item) names in cog to secure the item order in incidence matrix:
make.incidenz(tab=cogBOOKLET, bookid=d$BOOKID, item_order=names(cog)[4:34])
########################
```

`Neoffi5` *Polytomous example data in Rost 2004*

Description

Data for 1000 subjects answering to 5 polytomous items assessing neuroticism contained in the german version of the NEO–five–factor–inventory (NEOFFI) by Borkenau and Ostendorf (1991). This data is used as an example in the textbook by J. Rost (2004) to demonstrate some principles of rasch measurement.

Usage

```
data(Neoffi5)
```

Format

A `"matrix"` containing 5 columns (variables) and 1000 rows (obsevations).

Details

An detailed description of the data can be found in Rost (2004) at page 202.

References

Rost, J. (2004). *Lehrbuch Testtheorie - Testkonstruktion* (2 nd Ed.) Huber: Bern.

Borkenau. P. & Ostendorf F. (1991). Ein Fragebogen zur Erfassung fünf robuster Persönlichkeitsfaktoren. *Diagnostica, 37*, (1), 29–41.

Examples

```
data(Neoffi5)
dim(Neoffi5)
###########
# frequencies
ftab(Neoffi5)
# Itemparameter to be compared with Rost (2004), page 211.
summary(pair(Neoffi5))
# Itemparameter to be compared with Rost (2004), page 213.
summary(pers(pair(Neoffi5)))
```

pair *Item Parameter Calculation*

Description

This is the (new) main function for calculation of the item parameter for the dichotomous Rasch Model (Rasch, 1960) and its extension for polytomous items (thurstonian thresholds) according to the Partial Credit Model (Masters, 1982).

The function implements a generalization (see Heine & Tarnai, 2015) of the pairwise comparison approach, that is based on the principle for item calibration introduced by Choppin (1968, 1985) – see also (Wright & Masters, 1982). The number of (response) categories may vary across items.

Missing values up to an high amount in data are allowed, as long as items are proper linked together.

Usage

```
pair(
  daten,
  m = NULL,
  w = NULL,
  pot = TRUE,
  zerocor = TRUE,
  ccf = FALSE,
  likelihood = NULL,
  pot2 = 2,
  delta = TRUE,
  conv = 1e-04,
  maxiter = 3000,
  progress = TRUE,
  init = NULL,
  zerosum = TRUE,
  ...
)
```

Arguments

daten	a single `data.frame` or `matrix` or a `list` with every list entry holding a `data.frame` or `matrix`. Row and column names may represent the names of items (columns) and persons (rows). Missing values (`NA`) are allowed. Item responses can be polytomous or dichotomous (or a mixture of a varying number of categories across all items). Responses of `n` respondents (rows) on `k` items (colums) must be coded (scored) starting with 0 for lowest category to *m*-1 for highest category. If a list is assigned to this argument each `data.frame` or `matrix` as respective list entry must have the same dimensionality. See details and examples.
m	an integer (will be recycled to a vector of length k) or a vector giving the number of response categories for all items - by default (`m = NULL`), `m` is calculated from data, assuming that every response category is at least once present in the data.

For *'sparse' data* it is *strongly recomended* to explicitly *define the number of categories* by defining this argument.

`w` an optional numeric vector of non-negative case weights. Its length must equal the number of persons. For list input, a vector with one value per person is repeated for every rater/time point; alternatively, one value per row of the internally stacked response matrix can be supplied.

`pot` either a logical or an integer >= 2 defining the power to compute of the pairwise comparison matrix. If TRUE (default) a power of three of the pairwise comparison matrix is used for further calculations. If FALSE no powers are computed.

`zerocor` either a logical or an numeric value between >0 and <=1. If (in case of a logical) zerocor is set to TRUE (default) unobserved combinations (1-0, 0-1) in the data for each pair of items are given a frequency of one conf. proposal by Alexandrowicz (2011, p.373). As an alternative option a numeric value between >0 and <=1 can be assigned to unobserved combinations (1-0, 0-1) in the data for each pair of items (conf. to personal communication with A. Robitzsch; 29-03-2021).

`ccf` logical with default `ccf=FALSE` to perform normal item parameter calculation, if set to `ccf=TRUE` just the conditional item (category) frequencies are returned.

`likelihood` either NULL (default) or a a character expression defining a likelihood estimation approach based on the pairwise comparison matrix. Currently only the so called MINCHI approach as described in Fischer (2006) is implemented , which can be selected by setting `likelihood="minchi"`.

`pot2` ignored when `likelihood=NULL` (default).

`delta` ignored when `likelihood=NULL` (default).

`conv` ignored when `likelihood=NULL` (default).

`maxiter` ignored when `likelihood=NULL` (default).

`progress` ignored when `likelihood=NULL` (default).

`init` ignored when `likelihood=NULL` (default).

`zerosum` ignored when `likelihood=NULL` (default).

`...` additional parameters passed through.

Details

Parameter calculation is based on the construction of a paired comparison matrix M*nicjc* with entries f*icjc* representing the number of respondents who answered to item *i* in category *c* and to item *j* in category *c-1* widening Choppin's (1968, 1985) conditional pairwise algorithm to polytomous item response formats. This algorithm is simply realized by matrix multiplication.

To avoid numerical problems with off diagonal zero's when constructing the pairwise comparison matrix M*nij*, powers of the M*nicjc* matrix, can be used (Choppin, 1968, 1985). Using powers *k* of M*nicjc* - argument `pot=TRUE` (default), replaces the results of the direct comparisons between *i* and *j* with the sum of the indirect comparisons of *i* and *j* through an intermediate *k*. In general, it is recommended to use the argument with default value `pot=TRUE`.

If a list object is assigned to the argument `data`, the list entries (matrix or data.frame) must all have the same dimensionality. The individual list entries represent either *r* measurement times or raters.

If such a list object is used, first the item parameters are calculated across all *r* measurement points or raters and additionally a threshold parameter is given for each of the *r* measurement points or raters (e.g. rater severity or overal item shift).

For a graphic representation of the item 'estimates' the plotting S3 method `plot.pair` is available. For plotting the item category probabilities the function `catprob` can be used.

Value

A (list) object of class `"pair"` containing the item category thresholds, difficulties sigma, response data and the case weights used. The new `weights` component is `NULL` for an unweighted fit and a named numeric vector aligned with the rows of `resp` otherwise. Existing components and their order are unchanged; `weights` is appended.

References

Choppin, B. (1968). Item Bank using Sample-free Calibration. *Nature, 219*(5156), 870-872.

Choppin, B. (1985). A fully conditional estimation procedure for Rasch model parameters. *Evaluation in Education, 9*(1), 29-42.

Heine, J. H. & Tarnai, Ch. (2015). Pairwise Rasch model item parameter recovery under sparse data conditions. *Psychological Test and Assessment Modeling, 57*(1), 3–36.

Alexandrowicz, R. W. (2011). 'GANZ RASCH': A Free Software for Categorical Data Analysis. *Social Science Computer Review, 30*(3), 369-379.

Masters, G. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149–174.

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danmarks pædagogiske Institut.

Wright, B. D., & Masters, G. N. (1982). *Rating Scale Analysis.* Chicago: MESA Press.

Fischer, Gerhard H. 2006. "Rasch Models". Pp. 515–85 in Handbook of statistics (26): Psychometrics. Vol. 26, edited by C. R. Rao and S. Sinharay. Amsterdam: Elsevier.

Examples

```
data(bfiN) # loading example data set
# calculating itemparameters for 5 neuroticism items with 6 answer categories (0-5).
neuro_itempar<-pair(daten = bfiN, m = 6)
summary(neuro_itempar)
summary(neuro_itempar, sortdif=TRUE) # ordered by difficulty
# plotting threshold profiles for 5 neuroticism items.
plot(neuro_itempar)
plot(neuro_itempar, sortdif=TRUE) # plotting ordered by difficulty
################ with unequal number of categories
data(sim200x3)
res<-pair(sim200x3)
summary(res)
plot(res)
```

pairSE *Item Parameter Calculation with Standard Errors*

Description

Calculates Rasch/PCM item parameters and resampling standard errors.

Usage

```
pairSE(
  daten,
  m = NULL,
  w = NULL,
  nsample = 30,
  size = 0.5,
  seed = "no",
  pot = TRUE,
  zerocor = TRUE,
  verbose = TRUE,
  likelihood = NULL,
  pot2 = 2,
  delta = TRUE,
  conv = 1e-04,
  maxiter = 3000,
  progress = TRUE,
  init = NULL,
  zerosum = TRUE,
  ...,
  method = c("subsampling", "bootstrap", "jackknife")
)
```

Arguments

daten	Response matrix or data.frame, one independent person per row.
m	Category counts; see pair. If NULL, inferred once from full data.
w	Finite non-negative case weights, one per row, with a finite positive sum.
nsample	Integer number of random replicates, at least two. Ignored for jackknife, which uses N replicates.
size	Fraction strictly between zero and one for subsampling. The rounded sample size must be between 2 and N-1. The alias "jack" selects jackknife when method is omitted or "jackknife". Ignored for bootstrap and explicit jackknife.
seed	Numeric seed for random resampling, or "no" to use the current RNG state.
pot, zerocor, likelihood, pot2, delta, conv, maxiter, progress, init, zerosum	See pair.

verbose	Print resampling progress?
...	Additional arguments passed to pair.
method	"subsampling" (default), "bootstrap", or "jackknife". Must be named; previous positional arguments retain their positions.

Details

For subsampling, rows are sampled without replacement and the replicate standard deviation is multiplied by sqrt(m/(N-m)), using the actual rounded sample size m. This is exact for the sample mean and an asymptotic approximation for regular estimators of independent, identically distributed rows. It is not a guarantee for sparse or nonregular pairwise fits. At m=N/2 the factor is one. Bootstrap draws N rows with replacement and uses the replicate standard deviation. Jackknife omits each row exactly once and uses sqrt((N-1)/N * sum((estimate_i - mean(estimate_i))^2)). No parameter bias correction is applied. Point estimates always come from the complete data. More random replicates reduce Monte Carlo variability, but do not fix bias. Category counts are held fixed from the full-data fit. Failed or non-finite replicate fits cause an error; unused threshold cells remain NA. Weights travel with their rows; rows are sampled uniformly, not proportional to weights. This treats observed row/weight pairs as the sampling units and does not implement design-based variance estimation for survey, cluster or stratified samples, nor expand integer frequency weights into independent rows. The jackknife is deterministic and does not set or consume the random seed.

Value

A pairSE list with threshold, sigma, SE, SEsigma, weights and resp, in their previous order. Attribute resampling records method, N, actual sample size, number of replicates and the multiplier applied to R's replicate standard deviation.

Examples

```
data(sim200x3)
pairSE(sim200x3, nsample=30, seed=42, verbose=FALSE)
pairSE(sim200x3, method="bootstrap", nsample=30, seed=42, verbose=FALSE)
## Not run: pairSE(sim200x3, method="jackknife", verbose=FALSE)
```

---

pairwise.item.fit *Item Fit Indices*

Description

function for calculating item fit indices. The procedures for calculating the fit indices are based on the formulas given in Wright & Masters, (1982, P. 100), with further clarification given in http://www.rasch.org/rmt/rmt34e.htm.

Usage

```
pairwise.item.fit(pers_obj, na_treat = NA)
```

Arguments

pers_obj	an object of class "pers" as a result from function pers
na_treat	value to be assigned to residual cells which have missing data in the original response matrix. default is set to na_treat=NA to ignore these cells in further calculations. An option is to set these residuals to 0 using na_treat=0, which implys that they are imputed as 'fitting data', i.e., zero residuals. This can attenuate contrasts (see. http://www.rasch.org/rmt/rmt142m.htm).

Details

contrary to many IRT software using Ml based item parameter estimation, pairwise will not exclude persons, showing perfect response vectors (e.g. c(0,0,0) for dataset with three variables), prior to the scaling. Therefor the fit statistics computed with pairwise may deviate somewhat from the fit statistics produced by IRT software using Ml based item parameter estimation (e.g. R-package eRm), depending on the amount of persons with perfect response vectors in the data.

Value

an object of class c("pifit", "data.frame") containing item fit indices.

References

Wright, B. D., & Masters, G. N. (1982). *Rating Scale Analysis.* Chicago: MESA Press.

Wright, B. D., & Masters, G. N. (1990). Computation of OUTFIT and INFIT Statistics. *Rasch Measurement Transactions, 3*(4), 84–85.

Examples

```
########
data(sim200x3)
result <- pers(pair(sim200x3))
pairwise.item.fit(pers_obj=result) # item fit statistic
```

---

pairwise.person.fit *Person Fit Indices*

Description

function for calculating person fit indices. The procedures for calculating the fit indices are based on the formulas given in Wright & Masters, (1982, P. 100), with further clarification given in http://www.rasch.org/rmt/rmt34e.htm.

Usage

```
pairwise.person.fit(pers_obj, na_treat = NA)
```

Arguments

pers_obj	an object of class "pers" as a result from function pers.
na_treat	value to be assigned to residual cells which have missing data in the original response matrix. default is set to na_treat=NA to ignore these cells in further calculations. An option is to set these residuals to 0 using na_treat=0, which implys that they are imputed as 'fitting data', i.e., zero residuals. This can attenuate contrasts (see. http://www.rasch.org/rmt/rmt142m.htm).

Details

contrary to many IRT software using ML based item parameter estimation, pairwise will not exclude persons, showing perfect response vectors (e.g. c(0,0,0) for dataset with three variables), prior to scaling. Therefor the fit statistics computed with pairwise may deviate somewhat from the fit statistics produced by IRT software using ML based item parameter estimation (e.g. R-package eRm), depending on the amount of persons with perfect response vectors in the data.

Value

an object of class c("ppfit", "data.frame") containing person fit indices

References

Wright, B. D., & Masters, G. N. (1982). *Rating Scale Analysis.* Chicago: MESA Press.

Wright, B. D., & Masters, G. N. (1990). Computation of OUTFIT and INFIT Statistics. *Rasch Measurement Transactions, 3*(4), 84–85.

Examples

```
########
data(sim200x3)
result <- pers(pair(sim200x3))
pairwise.person.fit(pers_obj=result) # item fit statistic
```

---

pairwise.S *The Fischer-Scheiblechner Statistic S on item level (Wald like Test)*

Description

This function calculates the S-statistic on item level proposed by Fischer and Scheiblechner (1970) on item level for dicho- or polytomous item response formats by splitting the data into two subsamples. For polytomous Items the test is performed on item category level. Several splitting options are available (see arguments). The S-statistic is also mentioned in van den Wollenberg, (1982) – an article in Psychometrika, which might be available more easily (see details).

Usage

```
pairwise.S(
  daten,
  m = NULL,
  split = "random",
  splitseed = "no",
  verbose = FALSE,
  ...
)
```

Arguments

daten	a data.frame or matrix with optionaly named colums (names of items), potentially with missing values, comprising polytomous or dichotomous (or mixed category numbers) responses of n respondents (rows) on k items (colums) coded starting with 0 for lowest category to *m*-1 for highest category, with *m* beeing a vector (with length k) with the number of categories for the respective item.
m	an integer (will be recycled to a vector of length k) or a vector giving the number of response categories for all items - by default `m = NULL`, `m` is calculated from data, assuming that every response category is at least once present in data. For sparse data it is strongly recomended to explicitly define the number of categories by defining this argument.
split	Specifies the splitting criterion. Basically there are three different options available - each with several modes - which are controlled by passing the corresponding character expression to the argument. 1) Using the rawscore for splitting into subsamples with the following modes: `split = "median"` median raw score split - high score group and low score group; `split = "mean"` mean raw score split - high score group and low score group. 2) Dividing the persons in `daten` into subsamples with equal size by random allocation with the following modes: `split = "random"` (which is equivalent to `split = "random.2"`) divides persons into two subsamples with equal size. In general the number of desired subsamples must be expressed after the dot in the character expression - e.g. `split = "random.6"` divides persons into 6 subsamples (with equal size) by random allocation etc. 3) The third option is using a manifest variable as a splitting criterion. In this case a vector with the same length as number of cases in `daten` must be passed to the argument grouping the data into subsamples. This vector should be coded as `"factor"` or a `"numeric"` integer vector with min = 1.
splitseed	numeric, used for `set.seed(splitseed)` for random splitting - see argument `split`.
verbose	logical, if `verbose = TRUE` (default) a message about subsampling is sent to console when calculating standard errors.
...	additional arguments `nsample`, `size`, `seed`, `pot` for caling `pairSE` are passed through - see description for `pairSE`.

Details

The data is splitted in two subsamples and then item thresholds, the parameter (Sigma) and their standard errors (SE) for the items according the PCM (or RM in case of dichotonimies) are calculated for each subsample. This function internaly calls the function `pairSE`. Additional arguments (see description of function `pairSE`) for parameter calculation are passed through. This item fit statistic is also (perhaps misleadingly) namend as 'Wald test' in other R-packages. The S-statistic, as implemented in `pairwise`, is defined according to Fischer and Scheiblechner (1970); see also equation (3) in van den Wollenberg, (1982), p. 124 in the following equation:

$$S_i = \frac{\hat{\sigma}_i^{(1)} - \hat{\sigma}_i^{(2)}}{\sqrt{\left(S_{\hat{\sigma}_i}^{(1)}\right)^2 + \left(S_{\hat{\sigma}_i}^{(2)}\right)^2}}$$

where $\hat{\sigma}_i^{(1)}$ is the estimate of the item parameter of subsample 1, $\hat{\sigma}_i^{(2)}$ is the estimate of the item parameter of subsample 2 and $S_{\hat{\sigma}_i}^{(1)}$ and $S_{\hat{\sigma}_i}^{(2)}$ are the respective standard errors. In Fischer (1974), p. 297, the resulting test statistic (as defined above) is labeled with Z_i, as it is asymtotically normally distributed. Contrary to the 'Wald-type' test statistic W_i, which was drived by Glas and Verhelst (2005) from the (general) χ^2 distributed test of statistical hypotheses concerning several parameters, which was introduced by Wald (1943).

Value

A (list) object of class `"pairS"` containing the test statistic and item difficulty parameter sigma and their standard errors for the two or more subsamples.

A note on standard errors

Estimation of standard errors is done by repeated calculation of item parameters for subsamples of the given data. This procedure is mainly controlled by the arguments `nsample` and `size` (see arguments in `pairSE`). With regard to calculation time, the argument `nsample` is the 'time killer'. On the other hand, things (estimation of standard errors) will not necessarily get better when choosing large values for `nsample`. For example choosing `nsample=400` will only result in minimal change for standard error estimation in comparison to (`nsample=30`) which is the default setting (see examples).

References

description of function `pairSE{pairwise}`.

Fischer, G. H., & Scheiblechner, H. (1970). Algorithmen und Programme fuer das probabilistische Testmodell von Rasch. *Psychologische Beitrage*, (12), 23–51.

van den Wollenberg, A. (1982). Two new test statistics for the rasch model. *Psychometrika, 47*(2), 123–140. https://doi.org/10.1007/BF02296270

Glas, C. A. W., & Verhelst, N. D. (1995). *Testing the Rasch Model*. In G. Fischer & I. Molenaar (Eds.), Rasch models: Foundations, recent developments, and applications. New York: Springer.

Wald, A. (1943). Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical Society, 54*(3), 426–482. https://doi.org/10.1090/S0002-9947-1943-0012401-3

Fischer, G. H. (1974). *Einführung in die Theorie psychologischer Tests*. Bern: Huber.

Examples

```
##########
data("kft5")
S_ran_kft <- pairwise.S(daten = kft5,m = 2,split = "random")
summary(S_ran_kft)
summary(S_ran_kft,thres = FALSE)
#### polytomous examples
data(bfiN) # loading example data set
data(bfi_cov) # loading covariates to bfiN data set

# calculating itemparameters and SE for two subsamples by gender
S_gen <- pairwise.S(daten=bfiN, split = bfi_cov$gender)
summary(S_gen)
summary(S_gen,thres = FALSE)

# other splitting criteria
## Not run:
S_med <- pairwise.S(daten=bfiN, split = "median")
summary(S_med)

S_ran<-pairwise.S(daten=bfiN, split = "random")
summary(S_ran)

S_ran.4<-pairwise.S(daten=bfiN, split = "random.4")
summary(S_ran.4) # currently not displayed

###### example from details section 'Some Notes on Standard Errors' ########
S_def<-pairwise.S(daten=bfiN, split = "random",splitseed=13)
summary(S_def)
######
S_400<-pairwise.S(daten=bfiN, split = "random", splitseed=13 ,nsample=400)
summary(S_400)

## End(Not run)
```

`pairwise.SepRel` *Person Separation Reliability*

Description

This function calculates an Index of Person Separation, that is the proportion of person variance that is not due to error.

Usage

```
pairwise.SepRel(pers_obj, na.rm = TRUE)
```

Arguments

pers_obj	an object of class "pers" as a result from function pers.
na.rm	a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds.

Details

none

Value

An object of class c("pairwiseSepRel","list").

References

Andrich, D. (1982). An index of person separation in latent trait theory, the traditional KR.20 index, and the Guttman scale response pattern. *Education Research and Perspectives, 9*(1), 95–104.

Examples

```
######################
########
data(bfiN) # loading reponse data
pers_obj <- pers(pair(bfiN))
result <- pairwise.SepRel(pers_obj)
result
str(result) # to see whats in ;-)
####
```

---

pairwise_PACKAGE *Rasch Model Parameters with pairwise*

Description

Performs the explicit calculation – not estimation! – of the Rasch item parameters for dichotomous and polytomous response formats using a pairwise comparison approach (see Heine & Tarnai, 2015) a procedure that is based on the principle for item calibration introduced by Choppin (1968, 1985). On the basis of the item parameters, person parameters (WLE) are calculated according to Warm's weighted likelihood approach (Warm, 1989). Item- and person fit statistics and several functions for plotting are available.

Details

In case of dichotomous answer formats the item parameter calculation for the Rasch Model (Rasch, 1960), is based on the construction of a pairwise comparison matrix M*nij* with entries f*ij* representing the number of respondents who got item *i* right and item *j* wrong according to Choppin's (1968, 1985) conditional pairwise algorithm.

For the calculation of the item thresholds and difficulty in case of polytomous answer formats, according to the Partial Credit Model (Masters, 1982), a generalization of the pairwise comparison algorithm is used. The construction of the pairwise comparison matrix is therefore extended to the comparison of answer frequencies for each category of each item. In this case, the pairwise comparison matrix M*nicjc* with entries f*icjc* represents the number of respondents who answered to item *i* in category *c* and to item *j* in category *c-1* widening Choppin's (1968, 1985) conditional pairwise algorithm to polytomous item response formats. Within R this algorithm is simply realized by matrix multiplication.

In general, for both polytomous and dichotomous response formats, the benefit in applying this algorithm lies in it's capability to return stable item parameter 'estimates' even when using data with a relative high amount of missing values, as long as the items are still proper linked together.

The recent version of the package 'pairwise' computes item parameters for dichotomous and polytomous item responses – and a mixture of both – according the partial credit model using the function `pair`.

Based on the explicit calculated item parameters for a dataset, the person parameters may thereupon be estimated using any estimation approach. The function `pers` implemented in the package uses Warm's weighted likelihood approach (WLE) for estimation of the person parameters (Warm, 1989). When assessing person characteristics (abilities) using (rotated) booklet designs an 'incidence' matrix should be used, giving the information if the respective item was in the booklet (coded 1) given to the person or not (coded 0). Such a matrix can be constructed (out of a booklet allocation table) using the function `make.incidenz`.

Item- and person fit statistics, see functions `pairwise.item.fit` and `pairwise.person.fit` respectively, are calculated based on the squared and standardized residuals of observed and the expected person-item matrix. The implemented procedures for calculating the fit indices are based on the formulas given in Wright & Masters, (1982, p. 100), with further clarification given at `http://www.rasch.org/rmt/rmt34e.htm`.

Further investigation of item fit can be done by using the function `ptbis` for point biserial correlations. For a graphical representation of the item fit, the function `gif` for plotting empirical and model derived category probability curves, or the function `esc` for plotting expected (and empirical) score curves, can be used.

The function `iff` plots or returns values of the item information function and the function`tff`plots or returns values of the test information function.

To detect multidimensionality within a set of Items a rasch residual factor analysis proposed by Wright (1996) and further discussed by Linacre (1998) can be performed using the function `rfa`.

For a 'heuristic' model check the function `grm` makes the basic calculations for the graphical model check for dicho- or polytomous item response formats. The corresponding S3 plotting method is `plot.grm`.

Author(s)

Joerg-Henrik Heine <jhheine@googlemail.com>

References

Choppin, B. (1968). Item Bank using Samplefree Calibration. *Nature, 219*(5156), 870-872.

Choppin, B. (1985). A fully conditional estimation procedure for Rasch model parameters. *Evaluation in Education, 9*(1), 29-42.

Heine, J. H. & Tarnai, Ch. (2015). Pairwise Rasch model item parameter recovery under sparse data conditions. *Psychological Test and Assessment Modeling, 57*(1), 3–36.

Heine, J. H. & Tarnai, Ch. (2011). Item-Parameter Bestimmung im Rasch-Modell bei unterschiedlichen Datenausfallmechanismen. *Referat im 17. Workshop 'Angewandte Klassifikationsanalyse'* [Item parameter determination in the Rasch model for different missing data mechanisms. Talk at 17. workshop 'Applied classification analysis'], Landhaus Rothenberge, Muenster, Germany 09.-11.11.2011

Heine, J. H., Tarnai, Ch. & Hartmann, F. G. (2011). Eine Methode zur Parameterbestimmung im Rasch-Modell bei fehlenden Werten. *Vortrag auf der 10. Tagung der Fachgruppe Methoden & Evaluation der DGPs.* [A method for parameter estimation in the Rasch model for missing values. Paper presented at the 10th Meeting of the Section Methods & Evaluation of DGPs.] Bamberg, Germany, 21.09.2011 - 23.09. 2011.

Heine, J. H., & Tarnai, Ch. (2013). Die Pairwise-Methode zur Parameterschätzung im ordinalen Rasch-Modell. *Vortrag auf der 11. Tagung der Fachgruppe Methoden & Evaluation der DGPs.* [The pairwise method for parameter estimation in the ordinal Rasch model. Paper presented at the 11th Meeting of the Section Methods & Evaluation of DGPs.] Klagenfurt, Austria, 19.09.2013 - 21.09. 2013.

Linacre, J. M. (1998). Detecting multidimensionality: which residual data-type works best? *Journal of outcome measurement, 2*, 266–283.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174.

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danmarks pædagogiske Institut.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427–450.

Wright, B. D., & Masters, G. N. (1982). *Rating Scale Analysis.* Chicago: MESA Press.

Wright, B. D. (1996). Comparing Rasch measurement and factor analysis. *Structural Equation Modeling: A Multidisciplinary Journal, 3*(1), 3–24.

`pers` *WLE - Rasch Person Parameter*

Description

This is the (new) main function for calculation of person estimates based on answering dichotomous or polytomous items according theRasch Model (Rasch, 1960) and Partial Credit Model (Masters, 1982), given the item parameters (object of class `"pair"` - as a result of `pair()`) and and the datamatrix (argument `daten`) containing the person respose vectors (rows), using an WL approach, introduced by Warm (1989).

Usage

```
pers(
  itempar,
  daten = NULL,
  incidenz = NULL,
  na_treat = NULL,
  limit = 1e-05,
  iter = 50,
  Nrel = FALSE,
  tecout = FALSE
)
```

Arguments

itempar	The item parameter prior calculated or estimated. A list object of class "pair" as a result of applying the function pair() to the data. Or an 'ordinary' "matrix" with nrow = k (number of items) and ncol = m (maximum number of thresholds), holding the 'thurstonian' thresholds of the respective item. Some matrix entries may be NA, depending on the number of categories of the respective item.
daten	A "matrix" (or "data.frame") optionaly with named colums (names of items) and named rows (person IDs). This argument can be left empty when the argument itempar (above) is of class "pair". daten holds polytomous or dichotomous (or mixted category numbers) responses of n respondents (rows) on k items (colums) coded starting with 0 for lowest category to m-1 for highest category, with m beeing a vector (with length k) with the number of categories for the respective item. Responses in daten must be stored as "integers" (not "factors" !) and may have missing values.
incidenz	This argument is only relevant when items are assigned to different booklets. For such a booklet-design a "matrix" should be assigned to this argument, with the same dimensions like daten, containig 0 and 1 integer codes, giving the information (for every person) if the respective item was in the respective booklet (coded 1) given to the person or not (coded 0).
na_treat	optionaly an integer (vector) defining the type of treatment to missing responses in the argument daten. If set to na_treat=NULL (default) missing responses are treated as missings and the respective person is assigned to an corresponding missing group for estimation. An option is to set na_treat to any integer value between 0 (lowest category) and the numeric code for the maximum ctaegory of the respective item.
limit	numeric giving the limit at which accuracy the WL-algorithm stops.
iter	numeric giving the maximum numer of iteration to perform.
Nrel	logical with default set to Nrel=FALSE to include persons with perfect response vectors for calculating WLE reliability. If set to Nrel=TRUE persons with perfect response vectors are excluded for calculating WLE reliability.
tecout	logical default set to FALSE. If set to TRUE the result will be a (very) long list with estimation details for every case in daten. In case of a booklet-design the list entries will be divided by "booklet".

Details

no detail in the moment.

Value

An object of class `c("pers", "data.frame")` or a (very long) `"list"` (when setting on `techout=TRUE`) containing the person parameters.

References

Masters, G. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149–174.

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: Danmarks pædagogiske Institut.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427–450.

Examples

```
############
data(sim200x3)
result <- pers(itempar=pair(sim200x3))
summary(result)
plot(result)
logLik(result) # Log-Likelihood for 'estimated' model
logLik(result, sat=TRUE) # Log-Likelihood for saturated model
AIC(logLik(result)) # AIC for 'estimated' model
AIC(logLik(result, sat=TRUE)) # AIC for saturated model
BIC(logLik(result)) # BIC for 'estimated' model
BIC(logLik(result, sat=TRUE)) # BIC for saturated model
###### following example requires package eRm ######
# require(eRm)
# # itemparameter with eRm:
# itempar_eRm <- thresholds(PCM(sim200x3))$ threshtable[[1]][,2:3]
# # pairwise personparameter with eRm-itemparameter and data:
# summary(pers(itempar=itempar_eRm,daten=sim200x3))
# # eRm personparameter:
# person.parameter(PCM(sim200x3))
# # personparameter with pairwise:
# summary(pers(pair(sim200x3)))
```

---

plot.grm *S3 Plotting Graphical Model Check*

Description

S3 plotting method for object of class`c("grm","list")`

Usage

```
## S3 method for class 'grm'
plot(
  x,
  xymin = NULL,
  xymax = NULL,
  ci = 2,
  main = NULL,
  col.error = "blue",
  col.diag = "red",
  itemNames = TRUE,
  cex.names = 0.8,
  type = "b",
  xlab = NULL,
  ylab = NULL,
  pch = 43,
  las = 3,
  cex.axis = 0.5,
  ...
)
```

Arguments

x	object of classc("grm","list")
xymin	optional lower limit for xy-axis
xymax	optional upper limit for xy-axis
ci	numeric defining confidence intervall for point estimator
main	see plot
col.error	vector of colors for error bars
col.diag	color for the diagonal of the plot
itemNames	logical wether to plot itemnames
cex.names	magnification factor for itemnames
type	see plot
xlab	see plot
ylab	see plot
pch	see plot
las	see plot
cex.axis	see plot
...	other parameters passed to plot

plot.pair *S3 Plotting Thurstonian Thresholds*

Description

S3 plotting method for object of class"pair"

Usage

```
## S3 method for class 'pair'
plot(
  x,
  sortdif = FALSE,
  ra = "auto",
  main = NULL,
  col.lines = (1:dim(x$threshold)[2]),
  type = "b",
  xlab = "items",
  ylab = "logits",
  pch = (1:dim(x$threshold)[2]),
  las = 3,
  cex.axis = 0.8,
  ...
)
```

Arguments

x	object of class"pair"
sortdif	logical wether to order items by difficulty
ra	either the character "auto" (default) or an numeric, defining the (logit) range for y-axis
main	see plot
col.lines	vector of colors for threshold profile lines
type	see plot
xlab	see plot
ylab	see plot
pch	see plot
las	see plot
cex.axis	see plot
...	other parameters passed to plot

plot.pairSE *S3 Plotting Thustonian Thresholds with SE*

Description

S3 plotting method for object of classc("pairSE","list")

Usage

```
## S3 method for class 'pairSE'
plot(
  x,
  ci = 2,
  sortdif = FALSE,
  ra = "auto",
  main = NULL,
  col.lines = 1:(dim(x$threshold)[2]),
  col.error = 1:(dim(x$threshold)[2]),
  type = "b",
  xlab = "items",
  ylab = "logits",
  pch = 20,
  las = 3,
  cex.axis = 0.8,
  ...
)
```

Arguments

x	object of classc("pairSE","list")
ci	numeric defining confidence intervall for point estimator
sortdif	logical wether to order items by difficulty
ra	either the character "auto" (default) or an numeric, defining the (logit) range for y-axis
main	see plot
col.lines	vector of colors for threshold profile lines
col.error	vector of colors for error bars
type	see plot
xlab	see plot
ylab	see plot
pch	see plot
las	see plot
cex.axis	see plot
...	other parameters passed to plot

plot.pers *S3 Plotting Person - Item Map*

Description

S3 plotting method for object of class"pers"

Usage

```
## S3 method for class 'pers'
plot(
  x,
  ra = NULL,
  sortdif = FALSE,
  main = NULL,
  ylab = "Logits",
  itemNames = TRUE,
  fillCol = "grey60",
  lineCol = "grey40",
  cex = 0.7,
  pos = 4,
  breaks = "Sturges",
  pch = 1,
  ...
)
```

Arguments

x	object of class"pers"
ra	an integer, defining the (logit) range for y-axis
sortdif	logical wether to order items by difficulty
main	see plot
ylab	see plot
itemNames	logical wether to use itemnames in the resulting plot
fillCol	color for bar filling of the ability histogram
lineCol	color for bar lines of the ability histogram
cex	see text
pos	see text
breaks	see hist
pch	see points
...	other parameters passed to hist and text.

plot.rfa *S3 Plotting Rasch Residual Factor Analysis*

Description

S3 plotting Method for object of class"rfa"

Usage

```
## S3 method for class 'rfa'
plot(
  x,
  com = 1,
  ra = "auto",
  main = NULL,
  labels = NULL,
  xlab = "logits",
  ylab = "loadings",
  srt = 0,
  cex.axis = 0.8,
  cex.text = 0.8,
  col.text = NULL,
  ...
)
```

Arguments

x	object of class"rfa"
com	an integer giving the number of the principal component used for plotting
ra	either the character "auto" (default) or an numeric, defining the (logit) range for x-axis
main	see plot
labels	a character vector specifying the plotting pattern to use. see text. At default the itemnames are used.
xlab	see plot
ylab	see plot
srt	see text or par
cex.axis	see plot
cex.text	see argument cex in function text
col.text	see argument col in function text
...	other parameters passed through.

ptbis *Point Biserial Correlations*

Description

Calculation of the point biserial correlations for dicho- or polytomous item categories with total scale (person parameter).

Usage

```
ptbis(y, daten = NULL)
```

Arguments

y	either an object of class "pers", or an numeric vector as an result of any scaling approach (WLE, MLE, Rawscore, etc.) relating to the Items (columns) in daten.
daten	if argument y is not an object of class "pers", a "data.frame", potentially with missing values, comprising dicho- or polytomous items (columns).

Details

no details in the moment.

Value

An object of class c("data.frame", "ptbis") containing item statistics.

Examples

```
######################
########
data(sim200x3) # loading reponse data
y <- rowSums(sim200x3)
ptbis(y=y, daten=sim200x3)
####
result <- pers(pair(sim200x3))
ptbis(y= result)
```

Q *Person Fit Index Q*

Description

function for calculating the person fit index Q, which was proposed by Tarnai and Rost (1990).

Usage

```
Q(obj = NULL, data = NULL, threshold = NULL, ...)
```

Arguments

obj	an object of class "pers" or class "pair"as a result from function pers or pair respectively.
data	optional response data when object of class "pers" or class "pair" is not provided.
threshold	optional in case that object of class "pers" or class "pair" is not provided. Threshold values as matrix with row and columnnames !! – items as rows and thresholds as columns. Thresholds should be ordered from left to right, some items may have less thresholds than the others, in this case the respective row/column is filled with an NA value - see examples.
...	not used so far.

Details

The person Q-index proposed by Tarnai and Rost, (1990) is solely based on the empirical responses and the item parameters. Thus the computation of person parameters using the function pers is not required - see examples. But for convenience return objects of both functions are accepted in function Q.

Value

a vector holding the Q-index for every person.

References

Tarnai, C., & Rost, J. (1990). *Identifying aberrant response patterns in the Rasch model: the Q index*. Münster: ISF.

Examples

```
######################
data(bfiN) # get some data
ip <- pair(daten = bfiN,m = 6) # item parameters according the partial credit model
Q(ip)
```

```
### with data an thresholds as external objects #####
threshold <- matrix(seq(-3,3,length.out = 9),ncol = 3)
dimnames(threshold) <- list(c("I1","I2","I3"),c("1","2","2"))
threshold
resp_vec <- c(3,0,2,1,2,2,2,2,1,3,0,NA,NA,0,2,3,NA,2,NA,2,1,2,NA,1,2,2,NA)
resp_emp <- matrix(resp_vec,ncol = 3,byrow = TRUE)
colnames(resp_emp) <- c("I1","I2","I3")
resp_emp
Qindex <- Q(data = resp_emp,threshold = threshold)
cbind(resp_emp,Qindex)

#### unequal number of thresholds ###################
threshold <- matrix(seq(-3,3,length.out = 9),ncol = 3)
dimnames(threshold) <- list(c("I1","I2","I3"),c("1","2","2"))
threshold[2,3] <- NA

resp_vec <- c(3,0,2,1,2,2,2,2,1,3,0,NA,NA,0,2,3,NA,2,NA,2,1,2,NA,1,2,2,NA)
resp_emp <- matrix(resp_vec,ncol = 3,byrow = TRUE)
colnames(resp_emp) <- c("I1","I2","I3")
resp_emp
Qindex <- Q(data = resp_emp,threshold = threshold)
cbind(resp_emp,Qindex)
```

---

q3 *Q3 Fit Statistic*

Description

Calculation of Q3 fit statistic for the rasch model based on the residuals, which was proposed by Yen (1984).

Usage

```
q3(
  pers_obj,
  na_treat = 0,
  use = "complete.obs",
  res = "stdr",
  method = "pearson"
)
```

Arguments

pers_obj	an object of class "pers" as a result from function pers.
na_treat	value to be assigned to residual cells which have missing data in the original response matrix. default is set to na_treat=0 to set the residuals to 0, which implys that they are imputed as 'fitting data', i.e., zero residuals. This can attenuate contrasts (see. http://www.rasch.org/rmt/rmt142m.htm). An option is to set it to na_treat=NA.

use — a character string as used in function cor or cov, giving a method for computing covariances or correlations in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". The default is set to use="complete.obs" which will exclude cases by listwise deletion to keep the correlation matrix positive definit.

res — a character string defining which type of (rasch–) residual to analyze when computing the correlations. This must be (exactly) one of the strings "sr" for score residuals , "stdr" for standardised residuals, "srsq" for score residuals squared, or "stdrsq" for standardised residuals squared. The default is set to res="stdr" refering to Linacre (1998).

method — a character string as used in function cor, indicating which correlation coefficient is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated. The default is set to method="pearson".

Details

The lower level letter 'q' was used (intead of 'Q') for naming the function because the name 'Q3' was already used in another IRT package – namly TAM. As perhaps some users like to use both packages simultaniously, an alternative naming convention was choosen for 'pairwise'. No other details in the moment.

Value

An object of class c("Q3","list").

References

Yen, W. M. (1984). Effects of Local Item Dependence on the Fit and Equating Performance of the Three-Parameter Logistic Model. *Applied Psychological Measurement, 8*(2), 125–145. https://doi.org/10.1177/01466216840

Examples

```
######################
########
data(bfiN) # loading reponse data
pers_obj <- pers(pair(bfiN))
result <- q3(pers_obj)
str(result) # to see whats in ;-)
####
```

Qtest *Model Fit based on Person Fit Index Q*

Description

function for performing a model fit test based on calculating the person fit index Q for sub samples. Persons are ranked according their fit and in turn a Kruskal-Wallis rank sum test (see kruskal.test) is performed to test if the parameters of the distribution of Q are the same in each group (sample).

Usage

```
Qtest(
  obj = NULL,
  data = NULL,
  threshold = NULL,
  split = "median",
  splitseed = "no",
  ...
)
```

Arguments

obj	an object of class "pers" or class "pair"as a result from function pers or pair respectively.
data	optional response data when object of class "pers" or class "pair" is not provided.
threshold	optional in case that object of class "pers" or class "pair" is not provided. Threshold values as matrix with row and columnnames !! – items as rows and thresholds as columns. Thresholds should be ordered from left to right, some items may have less thresholds than the others, in this case the respective row/column is filled with an NA value - see examples.
split	Specifies the splitting criterion. Basically there are three different options available - each with several modes - which are controlled by passing the corresponding character expression to the argument. 1) Using the rawscore for splitting into subsamples with the following modes: split = "median" median raw score split - high score group and low score group; split = "mean" mean raw score split - high score group and low score group. Finaly split = "score" that is splitting daten into as many subsamples as there are raw score groups - discarding min and max (theoretical) score group - which matches the concept proposed by Andersen (1973). 2) Dividing the persons in daten into subsamples with equal size by random allocation with the following modes: split = "random" (which is equivalent to split = "random.2") divides persons into two subsamples with equal size. In general the number of desired subsamples must be expressed after the dot in the character expression - e.g. split = "random.6" divides persons into 6 subsamples (with equal size) by random allocation etc. 3) The third option is using a manifest variable as a splitting criterion. In this case a vector with the same length as number of cases in daten must be passed to the argument grouping the data into subsamples. This vector should be coded as "factor" or a "numeric" integer vector with min = 1.
splitseed	numeric, used for set.seed(splitseed) for random splitting - see argument split.
...	not used so far.

Details

The person Q-index proposed by Tarnai and Rost, (1990) is solely based on the empirical responses and the item parameters. Thus the computation of person parameters using the function pers is not required - see examples. But for convenience return objects of both functions are accepted in function Q. This offers the opportunity to get some inference on model fit only based on item parameters.

Value

A list with class "htest" containing the following components as returned by kruskal.test.

References

Tarnai, C., & Rost, J. (1990). *Identifying aberrant response patterns in the Rasch model: the Q index*. Münster: ISF.

Andersen, E. B. (1973). A goodness of fit test for the rasch model. *Psychometrika, 38*(1), 123–140.

Examples

```
#######################
data(bfiN) # get some data
ip <- pair(daten = bfiN,m = 6) # item parameters according the partial credit model
Q(ip)

Qtest(ip)

### with data an thresholds as external objects #####
threshold <- matrix(seq(-3,3,length.out = 9),ncol = 3)
dimnames(threshold) <- list(c("I1","I2","I3"),c("1","2","2"))
threshold
resp_vec <- c(3,0,2,1,2,2,2,2,1,3,0,NA,NA,0,2,3,NA,2,NA,2,1,2,NA,1,2,2,NA)
resp_emp <- matrix(resp_vec,ncol = 3,byrow = TRUE)
colnames(resp_emp) <- c("I1","I2","I3")
resp_emp
Qindex <- Q(data = resp_emp,threshold = threshold)
cbind(resp_emp,Qindex)

Qtest_res <- Qtest(data = resp_emp,threshold = threshold)
Qtest_res

#### unequal number of thresholds ###################
threshold <- matrix(seq(-3,3,length.out = 9),ncol = 3)
dimnames(threshold) <- list(c("I1","I2","I3"),c("1","2","2"))
threshold[2,3] <- NA

resp_vec <- c(3,0,2,1,2,2,2,2,1,3,0,NA,NA,0,2,3,NA,2,NA,2,1,2,NA,1,2,2,NA)
resp_emp <- matrix(resp_vec,ncol = 3,byrow = TRUE)
colnames(resp_emp) <- c("I1","I2","I3")
resp_emp
Qindex <- Q(data = resp_emp,threshold = threshold)
cbind(resp_emp,Qindex)
```

```
Qtest_res <- Qtest(data = resp_emp,threshold = threshold)
Qtest_res
```

`residuals.pers` *S3 residuals for Object of class "pers"*

Description

S3 residuals method to extract the (Rasch) residuals for object of class`"pers"`

Usage

```
## S3 method for class 'pers'
residuals(object, res = "sr", na_treat = 0, ...)
```

Arguments

`object`	object of class`"pers"`
`res`	a character string defining which type of (rasch–) residual to return. This must be (exactly) one of the strings "exp" for expected scores "sr" for score residuals (default), "stdr" for standardised residuals, "srsq" for score residuals squared, or "stdrsq" for standardised residuals squared. The default is set to res="sr".
`na_treat`	value to be assigned to residual cells which have missing data in the original response matrix. Default is set to na_treat=0 to set the residuals to 0, which implys that they are imputed as 'fitting data', i.e., zero residuals. This can attenuate contrasts (see. http://www.rasch.org/rmt/rmt142m.htm). An option is to set it to na_treat=NA.
`...`	not used jet.

`rfa` *Rasch Residual Factor Analysis*

Description

Calculation of the rasch residual factor analysis proposed by Wright (1996) and further discussed by Linacre (1998) to detect multidimensionality.

Usage

```
rfa(
  pers_obj,
  na_treat = 0,
  tr = FALSE,
  use = "complete.obs",
  res = "stdr",
  method = "pearson",
  cor = TRUE
)
```

Arguments

pers_obj	an object of class "pers" as a result from function pers.
na_treat	value to be assigned to residual cells which have missing data in the original response matrix. default is set to na_treat=0 to set the residuals to 0, which implys that they are imputed as 'fitting data', i.e., zero residuals. This can attenuate contrasts (see. http://www.rasch.org/rmt/rmt142m.htm). An option is to set it to na_treat=NA.
tr	a logical value indicating whether the data (the residual matrix) is transposed prior to calculation. This would perform a person analysis rather than a item analysis. The default is set to item analysis.
use	a character string as used in function cor or cov, giving a method for computing covariances or correlations in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". The default is set to use="complete.obs" which will exclude cases by listwise deletion to keep the correlation matrix positive definit.
res	a character string defining which type of (rasch–) residual to analyze when computing covariances or correlations. This must be (exactly) one of the strings "sr" for score residuals , "stdr" for standardised residuals, "srsq" for score residuals squared, or "stdrsq" for standardised residuals squared. The default is set to res="stdr" refering to Linacre (1998).
method	a character string as used in function cor or cov, indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated. The default is set to method="pearson".
cor	a logical value indicating whether the calculation should use the correlation matrix or the covariance matrix.The default is set to cor=TRUE to use the correlation matrix.

Details

no details in the moment.

Value

An object of class c("rfa","list").

References

Wright, B. D. (1996). Comparing Rasch measurement and factor analysis. *Structural Equation Modeling: A Multidisciplinary Journal, 3*(1), 3–24.

Linacre, J. M. (1998). Detecting multidimensionality: which residual data-type works best? *Journal of outcome measurement, 2*, 266–283.

Examples

```
######################
########
data(bfiN) # loading reponse data
pers_obj <- pers(pair(bfiN))
result <- rfa(pers_obj)
summary(result)
plot(result)
####
```

---

sim200x3 *Simulated Data*

Description

Simulated data for 200 'subjects' 'answering' to 3 items with unequal number of categories – one dichotomous and two polytoumous items.

Usage

```
data(sim200x3)
```

Format

A data.frame containing 3 variables and 200 obsevations.

Details

This simulated data is used as an example in the rasch module of the 'ALMO - Statistiksystem'.

Source

https://www.almo-statistik.de/

References

Holm, K. (2014). ALMO Statistik-System. *P14.8 Das allgemeine ordinale Rasch-Modell* https://www.almo-statistik.de/download/Ordinales_Rasch_Modell.pdf

Examples

```
data(sim200x3)
dim(sim200x3)
############################################################
apply(sim200x3,2,table)
```

`simra` *Simulate Response Pattern under Dichotomous and Polytomous Rasch Model*

Description

function for simulation of response patterns following the dichotomous and/or polytomous Rasch model based on the category probabilities given the model parameters. At default, when just calling `simra()` 1 replication of responses to 5 items with difficulties -2, -1, 0, 1, 2 from 100 persons with ability drawn from N(0|1) are sampled.

Usage

```
simra(
  itempar = matrix(seq(-2, 2, length = 5)),
  theta = 100,
  pers_obj = NULL,
  replicate = 1,
  seed = seq(1, replicate, 1),
  ...
)
```

Arguments

`itempar`	a "matrix" with `nrow` = `k` (number of items) and ncol = m (maximum number of thresholds), holding the 'thurstonian' thresholds of the respective item. Some of the rightmost matrix entries may be NA, depending on the number of categories of the respective item.
`theta`	either one of the following (1) a numeric vector of length `n` providing the values of the person parameter (ability) for `n` persons to be used in the simulation. (2) a integer defining the number of values `n` to draw from N(0|1).
`pers_obj`	an object of class `"pers"` as a result from function `pers`. If an object of class `"pers"` is assigned to this argument the model parameters in it are taken to simulate the responses. At default (`pers_obj` = `NULL`) simulation is done by considering the model parameters given in the other arguments (see above).
`replicate`	an integer defining how many replicates (data matrices) `r` to draw based on the model parameters.
`seed`	a numeric vector with legnth of number of replications used for `set.seed` prior to each replicate to keep the result repeatable. If `seed` = `NULL` no seed is set.
`...`	arguments passed through.

Details

no details in the moment.

Value

an array with `dim(n,k,r)` response patterns (`k` items in colums `n` persons in rows and `r` replications in the third dimension).

Examples

```
########
simra() # 100 dichotomous probabilistic response pattern
### 100 polytomous response pattern (4 items; each 4 answer categories)
v <- c(-1.0,-0.5,0.0,0.5,-0.75,-0.25,0.25,0.75,-0.5,0.0,0.5,1.0)
itempar <- matrix(v,nrow = 4,ncol = 3)
simra(itempar = itempar)
simra(itempar = itempar,replicate = 10) # draw 10 replications
```

---

summary.grm *S3 Summary for graphical Model Check*

Description

S3 summary method for object of classc(`"grm","list"`)

Usage

```
## S3 method for class 'grm'
summary(object, ci = 2, ...)
```

Arguments

`object`	object of classc(`"grm","list"`)
`ci`	numeric with default `ci=2` to return cinfidence intervalls for point estimator.
`...`	other parameters passed trough

summary.pair *S3 Summary for Item Parameter*

Description

S3 summary method for object of class"pair"

Usage

```
## S3 method for class 'pair'
summary(object, sortdif = FALSE, ...)
```

Arguments

object	object of class"pair"
sortdif	logical with default sortdif=FALSE whether to order items by difficulty.
...	other parameters passed trough

summary.pairS *S3 Summary for S-Statistic Test (Wald Test)*

Description

S3 summary method for object of class"pairS"

Usage

```
## S3 method for class 'pairS'
summary(object, thres = TRUE, ...)
```

Arguments

object	object of class"pairS"
thres	logical whether to output results based on the thresholds
...	other parameters passed trough

summary.pairSE *S3 Summary for Item Parameter with Standard Errors*

Description

S3 summary method for object of classc("pairSE","list")

Usage

```
## S3 method for class 'pairSE'
summary(object, sortdif = FALSE, ...)
```

Arguments

object	object of classc("pairSE","list")
sortdif	logical with default sortdif=FALSE wether to order items by difficulty.
...	other parameters passed trough

summary.pairwiseSepRel *S3 Summary for Person Separation Reliability*

Description

S3 summary method for object of class"pairwiseSepRel"

Usage

```
## S3 method for class 'pairwiseSepRel'
summary(object, ...)
```

Arguments

object	object of class"pairwiseSepRel"
...	other parameters passed trough

summary.pers *S3 Summary for Thetas*

Description

S3 summary method for object of class"pers"

Usage

```
## S3 method for class 'pers'
summary(object, short = TRUE, sortwle = FALSE, ...)
```

Arguments

object	object of class"pers"
short	logical with default short=TRUE - if set to short=FALSE a "data.frame" with WLE estimates (and their respective standard errors) for every row (person) in the original dataset will be returned.
sortwle	logical wether to order persons by ability - ignored when short=TRUE
...	other parameters passed trough

summary.pifit *S3 Summary for Item-Fit-Statistics*

Description

S3 summary method for object of classc("pifit", "data.frame")

Usage

```
## S3 method for class 'pifit'
summary(
  object,
  sort = FALSE,
  by = "INFIT.ZSTD",
  decreasing = FALSE,
  relative = FALSE,
  ...
)
```

Arguments

object	object of class"pifit", "data.frame"
sort	logical with default sort=FALSE - if set to sort=TRUE items are ordered by absolute FIT.
by	character passing the type of Fit-Statistic to sort by - ignored when sort=FALSE. valid options are: "INFIT.ZSTD" (default), "OUTFIT.MSQ", "OUTFIT.ZSTD" and "INFIT.MSQ".
decreasing	see order
relative	logical with default relative=FALSE to return the fit statistics as proposed by Wright & Masters, (1982, P. 100) with no further modifications. If relative=TRUE the sample adjusted fit statistics are returned in a way that their mean (for the present sample) equals 1 using formula: fit_i+ = 1 – mean(fit).
...	other parameters passed trough - see order

References

Wright, B. D., & Masters, G. N. (1982). *Rating Scale Analysis.* Chicago: MESA Press.

Wright, B. D., & Masters, G. N. (1990). Computation of OUTFIT and INFIT Statistics. *Rasch Measurement Transactions, 3*(4), 84–85.

---

summary.ppfit *S3 Summary for Person-Fit-Statistics*

Description

S3 summary method for object of classc("ppfit", "data.frame")

Usage

```
## S3 method for class 'ppfit'
summary(
  object,
  sort = FALSE,
  by = "INFIT.ZSTD",
  decreasing = FALSE,
  relative = FALSE,
  ...
)
```

Arguments

object	object of class"ppfit", "data.frame"
sort	logical with default sort=FALSE - if set to sort=TRUE persons are ordered by absolute FIT.

by	character passing the type of Fit-Statistic to sort by - ignored when sort=FALSE. valid options are: "INFIT.ZSTD" (default), "OUTFIT.MSQ", "OUTFIT.ZSTD" and "INFIT.MSQ".
decreasing	see order
relative	logical with default relative=FALSE to return the fit statistics as proposed by Wright & Masters, (1982, P. 100) with no further modifications. If relative=TRUE the sample adjusted fit statistics are returned in a way that their mean (for the present sample) equals 1 using formula: fit_i+ = 1 – mean(fit).
...	other parameters passed trough - see order

References

Wright, B. D., & Masters, G. N. (1982). *Rating Scale Analysis.* Chicago: MESA Press.

Wright, B. D., & Masters, G. N. (1990). Computation of OUTFIT and INFIT Statistics. *Rasch Measurement Transactions, 3*(4), 84–85.

---

summary.q3 *S3 Summary for Q3 Fit Statistic*

Description

S3 summary method for object of class"q3"

Usage

```
## S3 method for class 'q3'
summary(object, maxrc = 3, ...)
```

Arguments

object	object of class"q3"
maxrc	numerical with default maxrc=3 to specify the output of the maximum number of highest residual correlations in terms of absolute value.
...	other parameters passed trough

summary.rfa *S3 Summary for Rasch Factor Analysis*

Description

S3 summary method for object of class"pair"

Usage

```
## S3 method for class 'rfa'
summary(object, sortdif = FALSE, ...)
```

Arguments

object	object of class"pair"
sortdif	logical with default sortdif=FALSE wether to order items / persons by difficulty / ability.
...	other parameters passed trough

tff *Test information function*

Description

plotting function for plotting the test information function (TIF) with optional item-information overlays. Computes and plots the test information function (TIF) for a Rasch or Partial Credit Model fitted by pairwise::pair(). Selected item information functions (IIFs) can be added to the same plot. #'

Usage

```
tff(
  pair_obj,
  items = NULL,
  item.info = FALSE,
  x = NULL,
  ra = 4,
  nodes = 161,
  main = "Test Information Function",
  plot = TRUE,
  cat = FALSE,
  lwd = 2.5,
  col = 1,
  test.lty = 1,
  item.lwd = 1.25,
```

```
  item.col = NULL,
 item.lty = c("dashed", "dotted", "dotdash", "longdash", "twodash", "44", "13", "1343",
    "73", "2262", "12223242", "solid"),
  item.points = FALSE,
 item.pch = c(1, 2, 0, 5, 6, 3, 4, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
    22, 23, 24, 25),
  point.every = 12,
  point.cex = 0.7,
  category.lty = c("dotted", "dotdash", "longdash", "twodash"),
  category.lwd = 0.7,
  category.col = NULL,
  show.legend = TRUE,
  legend.position = "topright",
  legend.cex = 0.8,
  legend.ncol = NULL,
  xlab = "Theta (ability logits)",
  ylab = "Information",
  ylim = NULL,
  bty = "n",
  verbose = FALSE,
  ...
)
```

Arguments

pair_obj	An object of class "pair", returned by pairwise::pair().
items	NULL, numeric item positions, or character item names defining which items contribute to the test information. NULL selects all items.
item.info	Controls the item-information overlays. FALSE or NULL plots only the TIF; TRUE or "all" plots the IIFs of all items selected through ‘items‘; a numeric or character vector plots only the requested IIFs.
x	Numeric theta values. If NULL, ’nodes’ values from -’ra’ to ’ra’ are used. A single value suppresses plotting, as in the earlier versions of the function.
ra	Positive theta range used when x is NULL.
nodes	Number of theta values used when x is NULL.
main	Plot title.
plot	Logical; draw the plot?
cat	Logical; additionally draw category-information contributions p_ik(theta) * (k - E[X_i | theta])^2 for the test items. This option is mainly retained for compatibility and may produce many curves.
lwd	Line width of the test information function.
col	Colour of the test information function. With col = "black" (or col = 1), item curves default to black and are distinguished by line types and, optionally, plotting symbols.
test.lty	Line type of the test information function.

item.lwd	Line width of item information functions.
item.col	Optional colours for item information functions. If NULL, black is used in a black-and-white display; otherwise colours are created automatically.
item.lty	Line types used to distinguish item information functions.
item.points	Logical; add plotting symbols to item curves?
item.pch	Plotting symbols used to distinguish items.
point.every	Put an item symbol at every nth theta value.
point.cex	Size of item plotting symbols.
category.lty	Line types for category-information contributions.
category.lwd	Line width for category-information contributions.
category.col	Optional colour for category-information contributions. If NULL, the corresponding item colour is used.
show.legend	Logical; draw a legend for the TIF and displayed IIFs?
legend.position	Position of the item legend.
legend.cex	Character expansion of the legend.
legend.ncol	Number of legend columns. NULL chooses this automatically.
xlab, ylab	Axis labels.
ylim	Optional y-axis limits.
bty	Box type passed to plot().
verbose	Logical; print which items contribute to the TIF and which IIFs are displayed?
...	Further arguments passed to the initial plot().

Details

The Item information is calculated as the conditional variance of the item score, `I_i(theta) = sum_k p_ik(theta) * (k - E[X_i | theta])^2`. For a dichotomous Rasch item this reduces to `p(theta) * (1 - p(theta))`. Test information is the sum of the item information functions of the items selected through 'items'.

Value

When plot = TRUE, invisibly returns a list containing theta values, test information, item information, category contributions, and item selections. When plot = FALSE and cat = FALSE, returns a data frame (or a named numeric vector for one theta value) containing the TIF and requested IIFs. With cat = TRUE, the complete result list is returned.

Examples

```
# result <- pairwise::pair(sim200x3)
# tff(result)                                  # TIF only
# tff(result, item.info = TRUE)                # TIF + all IIFs
# tff(result, item.info = c(1, 3))             # selected IIFs
# tff(result, item.info = c("V1", "V3"))
```

```
# tff(result, items = 1:4, item.info = TRUE)   # TIF from items 1:4
# tff(result, item.info = 1:5, col = "black",
#     item.points = TRUE)
```

Index

∗ **datasets**
bfi_cov, 4
bfiN, 5
bfiN_miss, 7
cog, 9
cogBOOKLET, 9
DEU_PISA2012, 11
KCT, 22
kft5, 23
Neoffi5, 26
sim200x3, 55
∗ **fit**
andersentest.pers, 3
catprob, 8
esc, 12
gif, 15
grm, 17
iff, 20
lrtest.pers, 24
pairwise.item.fit, 31
pairwise.person.fit, 32
pairwise.S, 33
pairwise.SepRel, 36
Q, 48
q3, 49
Qtest, 50
rfa, 53
tff, 63
∗ **item**
catprob, 8
esc, 12
gif, 15
iff, 20
pair, 27
pairSE, 30
pairwise.item.fit, 31
pairwise.S, 33
∗ **main**
pair, 27
pairSE, 30
pers, 39
∗ **methods**
logLik.pers, 24
plot.grm, 41
plot.pair, 43
plot.pairSE, 44
plot.pers, 45
plot.rfa, 46
residuals.pers, 53
summary.grm, 57
summary.pair, 58
summary.pairS, 58
summary.pairSE, 59
summary.pairwiseSepRel, 59
summary.pers, 60
summary.pifit, 60
summary.ppfit, 61
summary.q3, 62
summary.rfa, 63
∗ **misc**
deltapar, 10
ftab, 14
make.incidenz, 25
ptbis, 47
simra, 56
∗ **model**
andersentest.pers, 3
grm, 17
lrtest.pers, 24
q3, 49
Qtest, 50
rfa, 53
tff, 63
∗ **package**
pairwise_PACKAGE, 37
∗ **person**
pairwise.person.fit, 32
pairwise.SepRel, 36

pers, 39
Q, 48
Qtest, 50

andersentest.pers, 3

bfi_cov, 4
bfiN, 5, *5*
bfiN_miss, *5*, 7

catprob, 8, *29*
cog, 9, *9*
cogBOOKLET, 9, *25*
cor, *50*, *54*
cov, *50*, *54*

deltapar, 10
DEU_PISA2012, 11

esc, 12, *38*

ftab, 14

gif, 15, *38*
grm, *3*, 17, *38*

hist, *45*

iff, 20, *38*

KCT, 22
kft5, 23
kruskal.test, *50*, *52*

logLik.pers, 24
lrtest.pers, 24

make.incidenz, 25, *38*

Neoffi5, 26

order, *61*, *62*

pair, *3*, *4*, *8*, *10*, *21*, 27, *30*, *31*, *38–40*, *48*, *51*
pairSE, *17–19*, 30, *34*, *35*
pairwise *(*pairwise_PACKAGE*)*, 37
pairwise-package *(*pairwise_PACKAGE*)*, 37
pairwise.item.fit, 31, *38*
pairwise.person.fit, 32, *38*
pairwise.S, 33
pairwise.SepRel, 36
pairwise_PACKAGE, 37
par, *46*
pers, *3*, *15*, *24*, *25*, *32*, *33*, *37*, *38*, 39, *48*, *49*, *51*, *52*, *54*, *56*
plot, *42–46*
plot.grm, *38*, 41
plot.pair, *29*, 43
plot.pairSE, 44
plot.pers, 45
plot.rfa, 46
points, *45*
ptbis, *38*, 47

Q, 48, *50*
q3, 49
Qtest, 50

residuals.pers, 53
rfa, *38*, 53

set.seed, *56*
sim200x3, 55
simra, 56
summary.grm, 57
summary.pair, 58
summary.pairS, 58
summary.pairSE, 59
summary.pairwiseSepRel, 59
summary.pers, 60
summary.pifit, 60
summary.ppfit, 61
summary.q3, 62
summary.rfa, 63

text, *45*, *46*
tff, *38*, 63