Package ‘pairwise’

November 9, 2013

Type Package

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

Date 2013-05-11

Author Joerg-Henrik Heine <`jhheine@googlemail.com`>

Version 0.1.5

License GPL-3

Depends R (>= 2.10.1)

Title Rasch Model Parameters by Pairwise Algorithm

Description The package pairwise offers functions for the explicit calculation, not estimation!, of the rasch item parameters for dichotomous and polytomous item responses, using a pairwise comparison approach.

Suggests PP

Collate
’grmSE.dicho.R’ ’grmSE.poly.R’ ’ipSE.dicho.R’ ’ipSE.poly.R’’itempar.dicho.R’ ’itempar.poly.R’ ’pairwise-package.r’’plot.grmSE.R’ ’plot.grmSEpo.R’ ’plot.ippw.R’ ’plot.ippwpo.R’’plot.ippwpoSE.R’ ’plot.ipSE.R’ ’thetaML.

R topics documented:

`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}`.

Usage

```
data(bfiN)
```

Format

A data frame containing 5 variables and 2800 obsevations.

Details

The other variables from the original bfi dataset were skipped to have a simple example data frame. For further Information on the original dataset see R-package `{psych}`.

Source

http://cran.r-project.org/web/packages/psych/index.html

References

Revelle, William (2012), psych: Procedures for Psychological, Psychometric, and Personality Research.*R package version 1.2.12*

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 to have a simple example data frame. For further Information on the original dataset see R-package `{psych}`.

Source

http://cran.r-project.org/web/packages/psych/index.html

References

Revelle, William (2012), psych: Procedures for Psychological, Psychometric, and Personality Research.*R package version 1.2.12*

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
```

`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 http://pisa2003.acer.edu.au/downloads.php

Source

http://pisa2003.acer.edu.au/downloads.php

References

Database - PISA 2003, *Downloadable Data*, http://pisa2003.acer.edu.au/downloads.php

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 packge, 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 http://pisa2003.acer.edu.au/downloads.php

Source

http://pisa2003.acer.edu.au/downloads.php

References

Database - PISA 2003, *Downloadable Data*, http://pisa2003.acer.edu.au/downloads.php

Examples

```
data(cogBOOKLET)
cogBOOKLET
```

`grmSE.dicho` *Graphical Model Check for dichotomous 1PL Model*

Description

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

Usage

```
grmSE.dicho(daten, teil = "no", splitseed = "no",
  sortdif = FALSE, ...)
```

Arguments

`daten`	The response data as a data.frame or a matrix, potentially with missing values, comprising dichotomous responses of respondents (rows) on some items (colums) coded in the 0 1 manner.
`teil`	Specifies the splitting criterion. Basicly 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: `teil = "median"` median raw score split - high score group and low score group; `teil = "mean"` mean raw score split - high score group and low score group; `teil = "score"` splitting `daten` into as many subsamples as there are raw score groups (discarding min and max score group) 2) Dividing the persons in `daten` into subsamples with equal size by random allocation with the following modes: `teil = "random"` (which is equivalent to `teil = "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. `teil = "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 numeric indicating the column number of the variable in `daten` must be passed to the argument - e.g. `teil = 1` indicates that the variable in the first column of `daten` will be used as splitting criterion - (this variable will of course be used only as splitting criterion). The variable in `daten` should be coded as `factor` or a numeric integer vector with min = 1 if `daten` is a matrix.
`splitseed`	numeric, used for `set.seed(splitseed)` for random splitting - see argument `teil`
`sortdif`	logical, In contrast to to the default setting in the function `ipSE.dicho` the argument `sortdif` here is set to `FALSE`, so the items will kept in original order – see description for `ipSE.dicho`.
`...`	additional arguments `nsample`, `size`, `seed`, `pot` for caling `ipSE.dicho` are passed through - see description for `ipSE.dicho`.

Details

The data is splitted in two or more subsamples and then item parameter (Sigma) and their standard error (SE) for dichotomous items according the 1PL Model are calculatetd for each subsample. Additional arguments (see description of function `ipSE.dicho`) 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 grmSE 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 `ipSE.dicho` `{pairwise}`.

Examples

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

# calculating itemparameters and SE for two random allocated subsamples
grmse<-grmSE.dicho(daten=cog[,4:34], teil = "random")
```

```
summary(grmse)

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

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

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

# plotting with item names
plot(grmse,ci=0, itemNames=TRUE,srt=90)

# Changing the size of the item names (too big to read)
plot(grmse,itemNames=TRUE, cex.names = 1.3)

# plotting with item names and rotate them about 90 degrees
plot(grmse,itemNames=TRUE,srt=90, cex.names = .5)

# Changing the color of the CI ellipses
plot(grmse,itemNames=TRUE, srt=90, cex.names = .5, col.error="green")

###### example from details section 'Some Notes on Standard Errors' ########
# grmse_400<-grmSE.dicho(daten=cog[,4:34], teil = "random",nsample=400)
######
# plot(grmse)
# plot(grmse_400)
```

`grmSE.poly` *Graphical Model Check for polytomous PCM*

Description

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

Usage

```
grmSE.poly(daten, m = max(daten, na.rm = TRUE) + 1,
  teil = "no", splitseed = "no", sortdif = FALSE,
  verbose = TRUE, ...)
```

Arguments

`daten`	daten a data matrix with optionaly named colums (names of items) or a data.frame, potentially with missing values, comprising polytomous responses of respondents (rows) on some items (colums) coded starting with 0 for lowest category to *m*-1 for highest category, with *m* beeing the number of categories for all items.
`m`	number of response categories for all items - by default *m* is defined as `m = max(daten,na.rm=TRUE)+1`.

`teil`	Specifies the splitting criterion. Basicly 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: `teil = "median"` median raw score split - high score group and low score group; `teil = "mean"` mean raw score split - high score group and low score group; `teil = "score"` splitting `daten` into as many subsamples as there are raw score groups (discarding min and max score group) 2) Dividing the persons in `daten` into subsamples with equal size by random allocation with the following modes: `teil = "random"` (which is equivalent to `teil = "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. `teil = "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 numeric indicating the column number of the variable in `daten` must be passed to the argument - e.g. `teil = 1` indicates that the variable in the first column of `daten` will be used as splitting criterion - (this variable will of course be used only as splitting criterion). The variable in `daten` should be coded as `factor` or a numeric integer vector with min = 1 if `daten` is a matrix.
`splitseed`	numeric, used for `set.seed(splitseed)` for random splitting - see argument `teil`
`sortdif`	logical, In contrast to to the default setting in the function `ipSE.poly` the argument `sortdif` here is set to `FALSE`, so the items will kept in original order - see description for `ipSE.poly`.
`verbose`	logical, if verbose = TRUE (default) a message about subsampling whe calculation standrderrors is sent to console
`...`	additional arguments `nsample`, `size`, `seed`, `pot` for caling `ipSE.poly` are passed through - see description for `ipSE.poly`.

Details

The data is splitted in two or more subsamples and then item thresholds, the parameter (Sigma) and their standard errors (SE) for polytomous items according the PCM are calculatetd for each subsample. Additional arguments (see description of function `ipSE.poly`) 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 grmSEpo 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 `ipSE.poly{pairwise}`.

Examples

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

# calculating itemparameters and SE for two random allocated subsamples
grmSEpoly<-grmSE.poly(daten=bfiN, teil = "random")

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

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

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

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

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

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

###### example from details section 'Some Notes on Standard Errors' ########
# grmSEpoly<-grmSE.poly(daten=bfiN, teil = "random",splitseed=13)
# plot(grmSEpoly)
######
# grmSEpoly_400<-grmSE.poly(daten=bfiN, teil = "random", splitseed=13 ,nsample=400)
# plot(grmSEpoly_400)
```

`ipSE.dicho` *Item Parameter calaculation with Standard Errors for dichotomous 1PL Model*

Description

Calculation of the item parameter (Sigma) and their standard error (SE) for dichotomous items according the 1PL Rasch Model using a pairwise comparison algorithm (Choppin, 1968, 1985). Missing values up to an high amount in data matrix are allowed, as long as items are proper linked together.

Usage

```
ipSE.dicho(daten, sortdif = TRUE, nsample = 30,
  size = 0.5, seed = "no", pot = TRUE, zerocor = TRUE,
  ...)
```

Arguments

`daten`	The response data as a data.frame or a matrix, potentially with missing values, comprising dichotomous responses of respondents (rows) on some items (colums) coded in the 0 1 manner.
`sortdif`	logical, if TRUE (default) items are sorted in an ascending order by difficulty for output.
`nsample`	numeric specifying the number of subsamples sampled from data, which is the number of replications of the parameter calculation. WARNING! specifying high values for `nsample` (> 100) may result in long computing time without leadig to "better" estimates for SE. This may also be the case when choosing argument `size="jack"` (see argument `size`) in combination with large datasets ($N > 500$).
`size`	numeric with valid range between 0 and 1 (but not exactly 0 or 1) specifying the size of the subsample of `data` when bootstraping for SE estimation. As an alternative, `size` can be set to the character `"jack"` (`size="jack"`). This will set the subsample size to N-1 and set `nsample=N` (see argument `nsample`), with N beeing the number of persons in `daten`.
`seed`	numeric used for `set.seed(seed)`.
`pot`	logical, if TRUE (default) a power of three of the pairwise comparison matrix is used for further calculations.
`zerocor`	logical, if TRUE (default) unobserved combinations (1-0, 0-1) in data for each pair of items are given a frequency of one conf. proposal by Alexandrowicz(2011, p.373).
`...`	additional parameters passed through.

Details

Item Parameter calculation is based on the construction of a paired 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. 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*nij* matrix, can be used (Choppin, 1968, 1985). Using powers *k* of M*nij* 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`.

Value

An object of class ipSE containing the item difficulty parameter Sigma and standard errors for item difficulties Sigma.

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

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.

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

Examples

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

# calculating itemparameters and their SE for 31 math items
se_sigma<-ipSE.dicho(daten=cog[,4:34], pot=TRUE)

summary(se_sigma) # summary for result

# plotting item difficulties with a CI = 95%
plot(se_sigma)

# use different color for CI
plot(se_sigma, col.error="green")

# without CI bars
plot(se_sigma,ci=0)

###### example from details section 'Some Notes on Standard Errors' ########
# se_sigma_400<-ipSE.dicho(daten=cog[,4:34], pot=TRUE,nsample=400)
# plot(se_sigma)
# plot(se_sigma_400)
```

`ipSE.poly` *Item Parameter calculation with Standard Errors for polytomous Partial Credit Model*

Description

Calculation of the item threshold, the difficulty and their standard errors (SE) respectively for polytomous items according to the Partial Credit Model. All parameters are calculatetd using a generalization of the pairwise comparison algorithm (Choppin, 1968, 1985). Missing values up to an high amount in data matrix are allowed, as long as items are proper linked together.

Usage

```
ipSE.poly(daten, m = max(daten, na.rm = TRUE) + 1,
  sortdif = TRUE, nsample = 30, size = 0.5, seed = "no",
  pot = TRUE, verbose = TRUE, ...)
```

Arguments

`daten`	a data matrix with optionaly named colums (names of items) or a data.frame, potentially with missing values, comprising polytomous responses of respondents (rows) on some items (colums) coded starting with 0 for lowest category to *m*-1 for highest category, with *m* beeing the number of categories for all items.
`m`	number of response categories for all items - by default *m* is defined as `m = max(daten,na.rm=TRUE)+1`.
`sortdif`	logical, if TRUE (default) items are sorted in an ascending order by difficulty for output.
`nsample`	numeric specifying the number of subsamples sampled from data, which is the number of replications of the parameter calculation. WARNING! specifying high values for `nsample` (> 100) may result in long computing time without leadig to "better" estimates for SE. This may also be the case when choosing argument `size="jack"` (see argument `size`) in combination with large datasets (*N* > 500).
`size`	numeric with valid range between 0 and 1 (but not exactly 0 or 1) specifying the size of the subsample of `data` when bootstraping for SE estimation. As an alternative, `size` can be set to the character `"jack"` (`size="jack"`). This will set the subsample size to *N*-1 and set `nsample=N` (see argument `nsample`), with *N* beeing the number of persons in `daten`.
`seed`	numeric used for `set.seed(seed)`.
`pot`	logical, if TRUE (default) a power of three of the pairwise comparison matrix is used for further calculations.
`verbose`	logical, if verbose = TRUE (default) a message about subsampling whe calculation standrderrors is sent to console.
`...`	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.

Estimation of standard errors is done by repeated calculation of item parameters for subsamples of the given data.

To avoid numerical problems with off diagonal zeros when constructing the pairwise comparison matrix M*nicjc*, 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`.

Value

A (list) object of class ippwpoSE containing the item category thresholds, difficulties sigma and their standrd errors.

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

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.

Examples

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

# calculating itemparameters and their SE for 5 neuroticism items with 6 answer categories (0-5).
neuro_itempar<-ipSE.poly(daten = bfiN, m = 6)

summary(neuro_itempar) # summary for result

# plotting item thresholds with with their CI = 95%
plot(neuro_itempar)

###### example from details section 'Some Notes on Standard Errors' ########
# neuro_itempar_400<-ipSE.poly(daten = bfiN, m = 6,nsample=400)
# plot(neuro_itempar)
# plot(neuro_itempar_400)
```

`itempar.dicho` *Item Parameter dichotomous 1PL*

Description

Calculation of the item parameter for dichotomous items according the 1PL Rasch Model using a pairwise comparison algorithm (Choppin, 1968, 1985). Missing values up to an high amount in data matrix are allowed, as long as items are proper linked together.

Usage

```
itempar.dicho(daten, sortdif = FALSE, pot = TRUE,
  zerocor = TRUE, ...)
```

Arguments

`daten`	a data matrix, potentially with missing values, comprising dichotomous responses of respondents (rows) on some items (colums) coded in the 0 1 manner.
`sortdif`	logical default is `sortdif=FALSE`. If argument is set to `sortdif=TRUE` the items are sorted in an ascending order by difficulty for output.
`pot`	logical, if TRUE (default) a power of three of the pairwise comparison matrix is used for further calculations.
`zerocor`	logical, if TRUE (default) unobserved combinations (1-0, 0-1) in data for each pair of items are given a frequency of one conf. proposal by Alexandrowicz(2011, p.373).
`...`	additional parameters passed through

Details

Parameter calculation is based on the construction of a paired 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. 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*nij* matrix, can be used (Choppin, 1968, 1985). Using powers *k* of M*nij* 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`.

Value

An object of class ippw containing item difficulties sigma.

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.

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

Examples

```
data(cog) # loading example data set
sigma<-itempar.dicho(daten=cog[,4:34], pot=TRUE) # calculating itemparameters for 31 math items
sigma
#######
plot(sigma) # plotting item difficulties
```

`itempar.poly` *Item Parameter for polytomous Partial Credit Model*

Description

Calculation of the item parameter for polytomous items according to the Partial Credit Model using a generalization of the pairwise comparison algorithm (Choppin, 1968, 1985). Missing values up to an high amount in data matrix are allowed, as long as items are proper linked together.

Usage

```
itempar.poly(daten, m = max(daten, na.rm = TRUE) + 1,
  sortdif = TRUE, pot = TRUE, zerocor = TRUE, ...)
```

Arguments

`daten`	a data matrix with optionaly named colums (names of items) or a data.frame, potentially with missing values, comprising polytomous responses of respondents (rows) on some items (colums) coded starting with 0 for lowest category to *m*-1 for highest category, with *m* beeing the number of categories for all items.
`m`	number of response categories for all items - by default *m* is defined as `m = max(daten,na.rm=TRUE)+1`.
`sortdif`	logical, if TRUE (default) items are sorted in an ascending order by difficulty for output.
`pot`	logical, if TRUE (default) a power of three of the pairwise comparison matrix is used for further calculations.
`zerocor`	logical, if TRUE (default) unobserved combinations (1-0, 0-1) in data for each pair of items are given a frequency of one conf. proposal by Alexandrowicz(2011, p.373).
`...`	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`.

Value

A (list) object of class ippwpo containing the item category thresholds and difficulties sigma.

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.

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

Examples

```
data(bfiN) # loading example data set
# calculating itemparameters for 5 neuroticism items with 6 answer categories (0-5).
neuro_itempar<-itempar.poly(daten = bfiN, m = 6)
neuro_itempar
#################
# plotting threshold profiles for 5 neuroticism items.
# 6 categories - 5 thresholds
plot(neuro_itempar)
```

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

Description

This function converts a booklet allocation table (like in `cogBOOKLET`) into a incidenz matrix used in the function `ppML.dicho`.

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 itmes 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 incidenz 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 al booklets of the entire set of booklets.

Value

an incidenz 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$BOOKID) # make just the incidenz matrix
inc
make.incidenz(tab=cogBOOKLET, bookid=d$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 incidenz matrix:
make.incidenz(tab=cogBOOKLET, bookid=d$BOOKID, item_order=names(cog)[4:34])
#######################
```

`pairwise` *Rasch Model Parameters with pairwise*

Description

The package `pairwise` performs the explicit calculation, not estimation!, of the Rasch item parameters for dichotomous an polytomous response formats using a pairwise comparison approach (Choppin, 1968, 1985).

Details

In case of dichotomous answer formats the parameter calculation 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 stabel 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.

Based on the explicit calculated item parameters for a dataset, the person parameters may thereupon be estimated using an mle or wle approach, for example implementetd in the R-package `{PP}` by Manuel Reif.

The recent version of the package 'pairwise' computes item parameters for dichotomous and polytomous item responses according the 1PL (when using the function `itempar.dicho`) and the partial credit model (when using the function `itempar.poly`).

The calculation of standard errors, when using functions `ipSE.dicho` or `ipSE.poly` is realized by bootstrap or jack-knife technique.

Splitting the data (e.g. by rawscore or by random) and calculating model parameter for each subsample can be done with the functions `grmSE.dicho` and `grmSE.poly`

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. (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.

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

Reif, M. (2012). PP: Person Parameter estimation. *R package version 0.2*, Abgerufen von http://CRAN.R-project.org/package=PP

`ppML.dicho` *ML Person Parameter dichotomous 1PL*

Description

Calculation of the person parameters for dichotomous items, given the item parameters and and the datamatrix (argument `daten`) containing the person respose vectors (rows), according the 1PL Rasch Model using an ML approach.

Usage

```
ppML.dicho(daten, SIGMA, incidenz = NULL, na_treat = 0,
  limit = 1e-05, iter = 20, tecout = FALSE)
```

Arguments

`daten`	a matrix (or data.frame) containing (numeric) responses of the persons (rows) coded in the (0,1) manner.
`SIGMA`	numeric vector with item (difficulty) parameters.

`incidenz`	This 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 codes giving the information (for every person) if the respective item was in the booklet given to the person.
`na_treat`	numeric defining the type of treatment to missing responses in the argument `daten`. If set to `na_treat=0` (default) missing responses are treated as wrong answers. An option is to set `na_treat` to any value between 0 and 1 e.g.: `na_treat=0.5`
`limit`	numeric giving the limit at which accuracy the ML-algorithm stops.
`iter`	numeric giving the maximum numer of iteration to perform.
`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 entrys will be divided by "booklet".

Details

no detail in the moment.

Value

An object of class "ppMLdicho" or "list" (depending on `techout`) containing the person parameters.

Examples

```
############
data(cog);data(cogBOOKLET) # loading reponse and 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 incidenz matrix
result<-ppML.dicho(daten=d[,4:34], SIGMA=itempar.dicho(d[,4:34]),incidenz = inc )
summary(result)
summary(result,FALSE) # return just the ML person estimates and their standard errors.
```

`ppWL.dicho` *WL Person Parameter dichotomous 1PL*

Description

Calculation of the person parameters for dichotomous items, given the item parameters and and the datamatrix (argument `daten`) containing the person respose vectors (rows), according the 1PL Rasch Model using an WL approach.

Usage

```
ppWL.dicho(daten, SIGMA, incidenz = NULL, na_treat = 0,
  limit = 1e-05, iter = 20, tecout = FALSE)
```

Arguments

`daten`	a matrix (or data.frame) containing (numeric) responses of the persons (rows) coded in the (0,1) manner.
`SIGMA`	numeric vector with item (difficulty) parameters.
`incidenz`	This 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 codes giving the information (for every person) if the respective item was in the booklet given to the person.
`na_treat`	numeric defining the type of treatment to missing responses in the argument `daten`. If set to `na_treat=0` (default) missing responses are treated as wrong answers. An option is to set `na_treat` to any value between 0 and 1 e.g.: `na_treat=0.5`
`limit`	numeric giving the limit at which accuracy the WL-algorithm stops.
`iter`	numeric giving the maximum numer of iteration to perform.
`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 entrys will be divided by "booklet".

Details

no detail in the moment.

Value

An object of class "ppWLdicho" or "list" (depending on `techout`) containing the person parameters.

Examples

```
############
data(cog);data(cogBOOKLET) # loading reponse and 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 incidenz matrix
result<-ppWL.dicho(daten=d[,4:34], SIGMA=itempar.dicho(d[,4:34]),incidenz = inc )
summary(result)
summary(result,FALSE) # return just the WL person estimates and their standard errors.
```

`ptbis.dicho` *Point Biserial Correlations dichotomous 1PL*

Description

Calculation of the point biserial correlations for dichotomous item categories with total scale parameter (person parameter).

Usage

```
ptbis.dicho(daten, scale)
```

Arguments

`daten`	a data matrix, potentially with missing values, comprising dichotomous items (columns).
`scale`	an integer vector as an result of any scaling approach (WLE, MLE, Rawscore, etc.) relating to the Items (columns) in `daten`.

Details

no details in the moment.

Value

An object of class "list" and "itanaldicho" containing item statistics.

Examples

```
######################
##### compute person ML estimates first ########
data(cog);data(cogBOOKLET) # loading reponse and 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 incidenz matrix
result<-ppML.dicho(daten=d[,4:34], SIGMA=itempar.dicho(d[,4:34]),incidenz = inc )
MLscale<-summary(result,FALSE)[,1] # return just the ML person estimates and their standard errors.
##### now compute item statistics ########
ptbis.dicho(daten=d[,4:34], scale=MLscale)
```

`thetaML.dicho` *ML Theta Parameter dichotomous 1PL*

Description

Calculation of the theta person parameter for dichotomous items, given the item parameters and one respose vector according the 1PL Rasch Model using an ML approach.

Usage

```
thetaML.dicho(response, sigma, na_treat = 0,
  theta_init = log(sum(response, na.rm = TRUE)),
  limit = 1e-05, iter = 20)
```

Arguments

`response`	numeric vector with responses of a person coded in the (0,1) manner.
`sigma`	numeric vector with item (difficulty) parameters.
`na_treat`	numeric defining the type of treatment to missing responses in the argument `response`. If set to `na_treat=0` (default) missing responses are treated as wrong answers. An option is to set `na_treat` to any value between 0 and 1 e.g.: `na_treat=0.5` which is guessing probability in dichotomous case.
`theta_init`	numeric giving the startvalue for estimation of theta. Default is `log()` of the persons rawscore (while NAs are removed for rawscore calculation).
`limit`	numeric giving the limit at which accuracy the ML-algorithm stops.
`iter`	numeric giving the maximum numer of iteration to perform.

Details

as this function is usualy not called directly by the user, there are no checks for plausibillity of the arguments !. Instead of calling this function directly, consider using the function `ppML.dicho`.

Value

An object of class thetaMLdicho containing the person parameter.

Examples

```
######## ... ####
```

`thetaWL.dicho` *WL Theta Parameter dichotomous 1PL*

Description

Calculation of the theta person parameter for dichotomous items, given the item parameters and one respose vector according the 1PL Rasch Model using an WL approach.

Usage

```
thetaWL.dicho(response, sigma, na_treat = 0,
  theta_init = log(sum(response, na.rm = TRUE)),
  limit = 1e-05, iter = 20)
```

Arguments

`response`	numeric vector with responses of a person coded in the (0,1) manner.
`sigma`	numeric vector with item (difficulty) parameters.
`na_treat`	numeric defining the type of treatment to missing responses in the argument `response`. If set to `na_treat=0` (default) missing responses are treated as wrong answers. An option is to set `na_treat` to any value between 0 and 1 e.g.: `na_treat=0.5` which is guessing probability in dichotomous case.
`theta_init`	numeric giving the startvalue for estimation of theta. Default is `log()` of the persons rawscore (while NAs are removed for rawscore calculation).
`limit`	numeric giving the limit at which accuracy the WL-algorithm stops.
`iter`	numeric giving the maximum numer of iteration to perform.

Details

as this function is usualy not called directly by the user, there are no checks for plausibillity of the arguments !. Instead of calling this function directly, consider using the function `ppWL.dicho`.

Value

An object of class thetaWLdicho containing the person parameter.

Examples

```
######## ... ####
```

Index