# Package ‘mokken’

May 1, 2020

**Version** 3.0.2  
**Date** 2020-04-27  
**Title** Conducts Mokken Scale Analysis  
**Author** L. Andries van der Ark [aut, cre],  
Letty Koopman [aut],  
J. Hendrik Straat [ctb],  
Don van den Bergh [ctb]  
**Maintainer** L. Andries van der Ark &lt;L.A.vanderArk@uva.nl&gt;  
**Depends** R (&gt;= 3.5.0), graphics, poLCA  
**Suggests** MASS  
**Description** Contains functions for performing Mokken scale analysis on test and questionnaire data (e.g., Sijsma and Van der Ark, 2017, &lt;doi:10.1111/bmsp.12078&gt;). It includes an automated item selection algorithm, and various checks of model assumptions.  
**License** GPL (&gt;= 2)  
**URL** https://sites.google.com/a/tilburguniversity.edu/avdrark/mokken  
**LinkingTo** Rcpp  
**Imports** Rcpp  
**NeedsCompilation** yes  
**Repository** CRAN  
**Date/Publication** 2020-04-30 23:10:02 UTC  

## R topics documented:

- mokken-package ........................................... 2  
- acl .................................................. 7  
- aisp .............................................. 8  
- autonomySupport ....................................... 11  
- cavalini ............................................ 12  
- check.bounds ......................................... 13  
- check.ca ........................................... 14
Mokken Scale Analysis

Description

Mokken scale analysis (Mokken, 1971; Sijtsma and Molenaar, 2002; Sijtsma and Van der Ark, 2017) is a scaling procedure for both dichotomous and polytomous items. It consists of an item selection algorithm to partition a set of items into Mokken scales and several methods to check the assumptions of two nonparametric item response theory models: the monotone homogeneity model and the double monotonicity model. The output of this R-package resembles the output of the stand-alone program MSP (Molenaar and Sijtsma, 2000).

Details

Package: mokken
Type: Package
Version: 3.0.2
Date: 2020-04-28
License: GPL Version 2 or later
The package contains principal functions for Mokken scale analysis.
The package contains the following data sets

- acl: Scores on a personality checklist.
- cavalini: Scores on an inventory on industrial malodor
- transreas: Scores on a transitive reasoning test
- DS14: Scores on a Type D test (bootstrap sample)

A guide for Mokken scale analysis in R for people who do not know R (Van der Ark, 2010) is available as a vignette from https://sites.google.com/a/tilburguniversity.edu/avdrark/mokken.

Thanks are due to Michael Allerhand, Don van den Berg, Geert H. van Kollenburg, Letty Koopman, Renske E. Kuipjers, Rudy Ligtsvoet, Hannah E. M. Oosterhuis, J. Hendrik Straat, and Daniel W. van der Palm for contributing R code; to Don van den Bergh, Geert H. van Kollenburg, Letty Koopman, Rudy Ligtsvoet, Patrick Mair, J. Hendrik Straat, and Don van Ravenswaaij for testing the software; to Wijbrandt van Schuur for comments on the vignette; to Michael Allerhand, Stephen Cubbellotti, Michael Dewey, Jasmin Durstín, Wilco H. M. Emons, Jue Huang, Michael Kubový, Ivo Molenaar, Jonathan Rose, Tobias Schlaffer, Klaas Sijtsma, Iris Smits, Jia Jia Su, Stefan Vermeent, Roger Watson, and Na Yang for reporting comments or bugs; to Diederick Stol (ProfitWise) for financial support, to Harrie C. M. Vorst, Pierre Cavalini, and Johan DeNollet for permission to use their data; to Robert J. Mokken for lending his last name.

Version 0 was introduced in Van der Ark (2007). It included the functions

- coefH: Scalability coefficients
- coefZ: Test statistics for scalability coefficients
- check.monotonicity: Investigate monotonicity assumptions
- check.restscore: Investigate nonintersection assumption using Method Restscore
- check.pmatrix: Investigate nonintersection assumption using Method Pmatrix
- search.normal: Mokken’s automated item selection algorithm

The following major modifications have been made.

- ais: More general automated item selection algorithm.
  Function search has become obsolete (Version 2.0)
- check.reliability: Compute reliability coefficients (Version 2.0)
- check.iio: Investigate invariant item orderings (Version 2.4)
- coefH: Standard errors for scalability coefficients included (Version 2.6)

All updates until version 2.7 are described in Van der Ark (2012). The following modifications have been made in Version 2.7 in comparison to previous versions.

- check.errors: Inclusion new function to compute weighted Guttman errors for each person.
  plot has been added.
- check.iio: Computation of number of active pairs for dichotomous items has been corrected.
The following modifications have been made in Version 2.7.1 in comparison to previous versions.

mokken Some legal issues

The following modifications have been made in Version 2.7.2 in comparison to previous versions.

check.iio Violations of IIO for dichotomous items are now tested using a z-test rather than a t-test.

The following modifications have been made in Version 2.7.3 in comparison to previous versions.

plot.iio.class Confidence envelopes around estimated response functions
plot.monotonicity.class Confidence envelopes around estimated response functions
plot.restscore.class Confidence envelopes around estimated response functions

The following modifications have been made in Version 2.8.1 in comparison to previous versions.

aisp Startsets have been added

The following modifications have been made in Version 2.8.2 in comparison to previous versions.

recode New
check.ca New
check.norms New
check.errors Outlier score O+ has been included

The following modifications have been made in Version 2.8.3 in comparison to previous versions.

twoway New
DS14 New data set
check.errors Outlier cutoff scores have been included

The following modifications have been made in Version 2.8.4 in comparison to previous versions.

check.iio New code for computing HT for large samples
The following modifications have been made in Version 2.8.5 in comparison to previous versions.

**MLcoefH**  
New code for computing two-level scalability coefficients and standard errors

**autonomySupport**  
New two-level data set.

The following modifications have been made in Version 2.8.9 and 2.8.10 in comparison to previous versions.

**coefH**  
Included possibility to include a fixed item-step order

**MLcoefH**  
Code updated

**check.errors**  
Code updated

The following modifications have been made in Version 2.8.11 in comparison to previous versions.

**plot**  
The level of transparency of the plotted confidence intervals can be adjusted manually

**MLcoefH**  
Code updated

The following modifications have been made in Version 2.8.12 in comparison to previous versions.

**check.monotonicity**  
Z statistic adjusted (Molenaar & Sijtsma, 2000. p. 72)

**check.norms**  
Z Output corrected for `nice.output = FALSE`

The following modifications have been made in Version 2.9.0 in comparison to previous versions.

**coefH**  
Z Solution of Koopman et al. (2017) implemented to solve the problem of equal item steps and code updated

**MLcoefH**  
Z Solution of Koopman et al. (2017) implemented to solve the problem of equal item steps and code updated

The following modifications have been made in Version 3.0.0 in comparison to previous versions.

**aisp**  
Genetic algorithm has been reprogrammed and is now much faster.

Argument `lowerbound` can now be a vector, enabling the investigation of several lower bounds simultaneously.

Extra argument to specify which standard errors should be used in Z-test (Koopman et al., 2020b).

Extra argument to indicate which null-hypothesis should be used to test $H_i$ (i.e., $H_i = c$ or $H_i = 0$)(Koopman et al., 2020b).

Extra argument to handle nested data (Koopman et al., 2020b).

**check.iio**  
Computation of Coefficient HT for large samples is now much faster.

**coeffH**  
New standard errors for nested data (Koopman et al. 2020a).

**coefZ**  
Extra argument to compute the Z-score using `lowerbound` as the null hypothesis (Koopman et al., 2020b).

Extra argument to compute Z-score using delta method standard error, but the original method remains available.

Extra argument to compute Z-score in nested data (Koopman et al., 2020b).

**ICC**  
New function for ICCs in two-level Mokken scale analysis (Koopman et al. 2020a)

**MLcoefH**  
Extra argument for weighted proportions. Reduces bias in two-level standard errors (Koopman et al. 2020a)

Extra argument for a fixed item-step order
mokken-package

SWMD

New data file (Koopman et al. 2020a)

Author(s)

L. Andries van der Ark & L. Koopman Maintainer: L. Andries van der Ark <L.A.vanderArk@uva.nl>.

References


Examples

# Personality test
data(acl)

# Select the items of the scale Communality
Communality <- acl[,1:10]

# Compute scalability coefficients
coeFH(Communality)

# Investigate the assumption of monotonicity
monotonicity.list <- check.monotonicity(Communality)
summary(monotonicity.list)
plot(monotonicity.list)
# Investigate the assumption of non-intersecting ISRFs using method restscore
restscore.list <- check.restscore(Communality)
summary(restscore.list)
plot(restscore.list)

# Investigate the assumption of non-intersecting ISRFs using method pmatrix
pmatrix.list <- check.pmatrix(Communality)
summary(pmatrix.list)
plot(pmatrix.list)

# Investigate the assumption of IIO using method MIIO
iio.list <- check.iio(Communality)
summary(iio.list)
plot(iio.list)

# Compute the reliability of the scale
check.reliability(Communality)

# Partition the scale into mokken scales
aisp(Communality)

## acl

### Adjective Checklist Data

＜Description＞
Scores of 433 students on 218 items from a Dutch version of the Adjective Checklist.

### Usage

data(acl)

### Format

A 433 by 218 matrix containing integers. dimnames(acl)[[2]] are adjectives.

### Details

Each item is an adjective with five ordered answer categories (0 = completely disagree, 1 = disagree, 2 = agree nor disagree, 3 = agree, 4 = completely agree). The respondents were instructed to consider whether an adjective described their personality, and mark the answer category that fits best to this description. The 218 items constitute 22 scales (see table); 77 items of the 218 items that constitute the ten scales were negatively worded. The negatively worded items are indicated by an asterisk in the dimnames and their item scores have been reversed. The Deference scale measures in fact the opposite of Deference.

<table>
<thead>
<tr>
<th>Communality</th>
<th>Items 1-10</th>
<th>Change</th>
<th>Items 111-119</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achievement</td>
<td>Items 11-20</td>
<td>Succorance</td>
<td>Items 120-129</td>
</tr>
<tr>
<td>Source</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data were kindly made available by H. C. M. Vorst from the University of Amsterdam. The original Adjective Checklist was developed by Gough and Heilbrun (1980).</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>References</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data(acl)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>aisp</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Automated Item Selection Procedure (AISP) for Mokken Scale Analysis</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a matrix with as many rows as there are items, indicating to which scale an item belongs for each lowerbound.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aisp(X, lowerbound=.3, search=&quot;normal&quot;, alpha=.05, StartSet=FALSE, popsize=20, maxgens=default.maxgens, pxover=0.5, pmutation=0.1, verbose=NULL, type.se = &quot;delta&quot;, test.Hi = FALSE, level.two.var = NULL)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>X</code> matrix or data frame of numeric data containing the responses of <code>nrow(X)</code> respondents to <code>ncol(X)</code> items. Missing values are not allowed.</td>
</tr>
</tbody>
</table>
Type of item selection procedure: "normal": Mokken's automated item selection procedure (Mokken, 1971; Molenaar and Sijtsma, 2000; Sijtsma and Molenaar, 2002); "ga": item selection using a genetic algorithm (Straat, van der Ark, and Sijtsma, 2013). The default is "normal".

Value or vector with numeric scaling criteria; 0 <= lowerbound < 1. The default is 0.3.

Type I error level. The default is 0.05.

Startset of items for the first scale. Vector of item numbers. If StartSet == FALSE no startset is provided (default).

Size of the population of items in genetic algorithm The default is 20.

Number of generations in genetic algorithm. The default is 10^((log2(ncol(X)/5)) * 1000).

Cross-over probability in genetic algorithm. The default is 0.5.

Mutation probability in genetic algorithm. The default is 0.1.

Logical, indicating whether should output to the screen the results of the model. If FALSE, no output is produced. The default is TRUE.

Indicates which type of standard errors is used in a Z-test whether coefficients meet the scaling criteria: "delta": uses standard errors approximated by the delta method (Kuijpers, Van der Ark, Kroon, 2013; Koopman, Zijlstra, Van der Ark, 2020); "Z": uses original Z-test (Mokken, 1971; Molenaar and Sijtsma, 2000; Sijtsma and Molenaar, 2002). The default is "delta".

If FALSE: tests if Hi is significantly larger than zero; If TRUE tests if Hi is significantly larger than lowerbound. The default is FALSE.

vector of length nrow(X) or matrix with number of rows equal to nrow(X) that indicates the level two variable for nested data (Koopman et al., 2020).

Each scale must consist of at least two items, hence the number of Mokken scales cannot exceed ncol(X)/2. Procedure may be slow for large data sets. Especially if the genetic algorithm is used. There is not yet an option search="extended". aisp replaces the function search.normal in earlier versions.

An matrix with J rows. Each entry refers to an item. Items with same integer belong to the same Mokken scale. A zero indicates an unscalable item. If n is the largest integer, then n Mokken scales were found.

L. A. van der Ark <l.a.vanderark@uva.nl>, D. van den Bergh L. Koopman J. H. Straat
References


See Also
c coefH, check.iio, check.monotonicity, check.pmatrix, check.reliability, check.restscore

Examples

data(acl)

# Select the scale Communality consisting of 10 items.
Communality <- acl[,1:10]

# Partition these 10 items into mokken scales using Mokken's automated item selection procedure.
scale <- aisp(Communality)
coefH(Communality[,scale==1])

# Same but using items 1 and 2 in the startset.
scale <- aisp(Communality, StartSet = c(1, 2), verbose = TRUE)
coefH(Communality[,scale==1])

# Perform aisp for increasing lowerbounds
scales <- aisp(Communality, lowerbound = seq(0, .55, .05))
scales

# Use a significant test for criteria Hi > c (rather than the point estimate)
scale <- aisp(Communality, test.Hi = TRUE, verbose = TRUE)
coefH(Communality[,scale==1])

# Partition these 10 items into mokken scales using a genetic algorithm.
```r
scale <- aisp(Communality, search="ga", maxgens=1000)
coefH(Communality[, scale==1])

# Perform aisp on two-level data
data(autonomySupport)
scores <- autonomySupport[, -1]
classes <- autonomySupport[, 1]
scale <- aisp(scores, level.two.var = classes)
coefH(scores[, scale==1], level.two.var = classes)
```

---

**autonomySupport**

*Autonomy Support Data*

**Description**

A two-level dataset with scores of 14 teachers who are rated by a group of pupils on 7 items from a Dutch Autonomy Support questionnaire (group size ranged between 5 and 39 pupils, total number of pupils is 259).

**Usage**

data(autonomySupport)

**Format**

A 259 by 8 data frame containing integers. The first column reflects a teacher indicator, the remaining columns the 7 items, see `colnames(autonomySupport)`.

**Details**

Each item has five ordered answer categories from *not at all/never* (score 1) to *certainly/always* (score 5). The items reflect several autonomy supportive behaviours from teachers.

<table>
<thead>
<tr>
<th>Item</th>
<th>Short</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item 1</td>
<td>Choose</td>
<td>The teacher lets me choose what I am going to do</td>
</tr>
<tr>
<td>Item 2</td>
<td>Decide</td>
<td>The teacher decides which task I will start with (inversely coded)</td>
</tr>
<tr>
<td>Item 3</td>
<td>Task</td>
<td>I get to choose which task I will start with</td>
</tr>
<tr>
<td>Item 4</td>
<td>Listen</td>
<td>The teacher listens to me when I disagree with something</td>
</tr>
<tr>
<td>Item 5</td>
<td>Help</td>
<td>The teacher helps me when I ask for it</td>
</tr>
<tr>
<td>Item 6</td>
<td>Accept</td>
<td>The teacher accepts me for who I am</td>
</tr>
<tr>
<td>Item 7</td>
<td>Understand</td>
<td>The teacher helps me when I do not understand a task</td>
</tr>
</tbody>
</table>

**Source**

The seven items are a subset from a self-constructed 27-item questionnaire on teacher’s autonomy support. Data were collected and made available by L. Koopman from the University of Amsterdam.
References


See Also

MLcoefH,

Examples

data(autonomySupport)

cavalini

### Coping Strategies

Data came from 17 polytomous items administered to 828 respondents (Cavalini, 1992) asking them how they coped actively with the bad smell from a factory in the neighborhood of their homes.

Usage

data(cavalini)

Format

A 828 by 17 matrix containing integers. attributes(cavalini) gives details on the items.

Details

Items have four ordered answer categories, *never* (score 0), *seldom* (1), *often* (2), and *always* (3). The 17 items constitute 4 scales (for detailed information, see Sijtsma and Molenaar, 2002, pp. 82-86).

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item1</td>
<td>Keep windows closed</td>
</tr>
<tr>
<td>Item2</td>
<td>No laundry outside</td>
</tr>
<tr>
<td>Item3</td>
<td>Search source of malodor</td>
</tr>
<tr>
<td>Item4</td>
<td>No blankets outside</td>
</tr>
<tr>
<td>Item5</td>
<td>Try to find solutions</td>
</tr>
<tr>
<td>Item6</td>
<td>Go elsewhere for fresh air</td>
</tr>
<tr>
<td>Item7</td>
<td>Call environmental agency</td>
</tr>
<tr>
<td>Item8</td>
<td>Think of something else</td>
</tr>
<tr>
<td>Item9</td>
<td>File complaint with producer</td>
</tr>
<tr>
<td>Item10</td>
<td>Acquiesce in odor annoyance</td>
</tr>
<tr>
<td>Item11</td>
<td>Do something to get rid of it</td>
</tr>
<tr>
<td>Item12</td>
<td>Say “it might have been worse”</td>
</tr>
<tr>
<td>Item13</td>
<td>Experience unrest</td>
</tr>
</tbody>
</table>
check.bounds

Item14  Talk to friends and family
Item15  Seek diversion
Item16  Avoid breathing through the nose
Item17  Try to adapt to situation

References


See Also

check.iio,

Examples

data(cavalini)
attributes(cavalini)$labels

---

check.bounds  

*Check the relative lower bound for scalability coefficients*

Description

Returns the relative bounds for Mokken’s scalability coefficients for dichotomous items as described by Ellis (2014).

Usage

check.bounds(X, quant = .90, lower = TRUE, upper = FALSE)

Arguments

X  

matrix or data frame of numeric data containing the responses of nrow(X) respondents to ncol(X) items. Missing values are not allowed

quant  

numerical value between 0 and 1 used for the computation of lower bound L2rij. The computation deviates somewhat from the proposal in Ellis (2014) because the stats function quantile is used.

lower  

Boolean: If TRUE, the lower bounds are given.

upper  

Boolean: If TRUE, the upper bounds are given.
Value
List containing two lists UpperBounds and LowerBounds, each containing a list of two J x J matrices (J = number of items): L1rij (overestimator of the lower bound for the correlation) and L2rij (underestimator of the lower bound for the correlation).

Author(s)
L. A. van der Ark <L.A.vanderArk@uva.nl>

References

Examples
```r
data(acl)
Communality <- acl[,1:10]
R <- cor(Communality)
res <- check.bounds(Communality, upper = TRUE)
L1rij <- res$LowerBounds$L1rij
L2rij <- res$LowerBounds$L2rij
U1rij <- res$UpperBounds$U1rij
U2rij <- res$UpperBounds$U2rij
# Correlations that meet L1rij (possibly an overestimation of the lower bound).
R >= L1rij
# Correlations that meet U1rij (possibly an overestimation of the upper bound).
R <= U1rij
# Correlations that meet L2rij (possibly an underestimation of the lower bound).
R >= L2rij
# Correlations that meet U2rij (possibly an underestimation of the upper bound).
R <= U2rij
```
Arguments

- **X**: Matrix of integers, missing values are not allowed.
- **Windex**: Boolean. Should output contain indices W1, W2, and W3?
- **MINSIZE**: Minimum sample size of a rest-score group.
- **NWEIGHTOPTION**: Weight of sample size on each conditional covariance. Options: "noweight" (each covariance has weight 1, default in Straat et al., 2016) and "sqrt" (each covariance has weight \( \sqrt{N_k(x)} \)), this option was used in an older, decrepit, version of Straat et al., 2016.
- **COVWEIGHTOPTION**: Weight of each conditional covariance on the computation of W1, W2, and W3. Options: "pnorm" (weight equals \( P[\text{cov} < 0] \), default in Straat et al., 2014) and "noweight" (if \( \text{cov} < 0 \), then weight = 1, and weight = 0, otherwise; this option was used in a previous version of Straat et al., 2014).
- **MINGROUP**: Minimum sample size of the conditioning variable to compute a covariance. Since the term N-3 is used in the computation of the standard error, N = 4 is the default.

Value

- list of three components:
  1. **InScale** (vector of booleans with length equal to the number of items): indicates whether an item is still in the scale.
  2. **Index** (list): Numerical values of indices W1, W2, and W3 (shown only if **Windex** = TRUE). Index has three subcomponents: W1, W2, and W3.
  3. **Flagged** (list): Boolean indicating whether a value of W1, W2, and W3 is flagged (1) or not (0) (shown only if **Windex** = TRUE) Index has three subcomponents: F1, F2, and F3, corresponding to the flagging of indices W1, W2, and W3, respectively.

Subcomponents correspond to the iteration. The first subcomponent refers to the situation with all items in the test, the second subcomponent refers to the situation with the worst item deleted, the third subcomponent refers to the situation with the two worst items deleted, etc.

Author(s)

L. A. van der Ark (<L.A.vanderArk@uva.nl>) and J. H. Straat

References


See Also

- DS14, recode, twoway
check.errors

Examples

data(DS14)

# Handle missing data and recode negatively worded items
X <- DS14[, 3 : 16]
X <- twoway(X)
X <- recode(X, c(1, 3))

# Negative affectivity
Na <- X[, c(1, 3, 6, 8, 10, 11, 14)]

# Social inhibition
Si <- X[, c(2, 4, 5, 7, 9, 12, 13)]

check.ca(Na, TRUE)

check.errors(X, returnGplus = TRUE, returnOplus = FALSE)

Description

Returns a list containing outlier scores Gplus (number of Guttman errors; Guttman, 1944) and Oplus for each respondent (Zijlstra, van der Ark and Sijtsma, 2007).

Usage

check.errors(X, returnGplus = TRUE, returnOplus = FALSE)

Arguments

X matrix or data frame of numeric data containing the responses of nrow(X) respondents to ncol(X) items. Missing values are not allowed

returnGplus Boolean. If TRUE the output contains outlier score Gplus

returnOplus Boolean. If TRUE the output contains outlier score Oplus

Value

List. Depending on the values of returnGplus and returnOplus, the output contains outlier score Gplus (the number of Guttman errors) and Oplus for each respondent

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>
check.iio

Check of Invariant Item Ordering

Description

Returns a list (of class iio.class) with results from the investigation of invariant item ordering. Three methods may be used for the investigation of invariant item ordering. (1) Method MIIO (manifest invariant item ordering: investigates the manifest item response functions for all pairs of items). For polytomous items, t-tests are used to test violations, for dichotomous items z-tests are used to test violations. (2) Method MS-CPM (manifest scale - cumulative probability model: investigates the manifest item step response functions for all pairs of items). Z-tests are used to test violations. (3) Method IT (increasingness in transposition: investigates all bivariate joint probabilities for all pairs of items). Chi-square tests are used to test violations.

For a complete description of Method MIIO, see Ligtvoet, Van der Ark, Te Marvelde, and Sijtsma (2010); for a complete description of the Method MS-CPM and Method IT with reference to Method MIIO, see Ligtvoet, Van der Ark, Bergsma, and Sijtsma (2011).

References


See Also

check.ca, check.iio, check.monotonicity, check.pmatrix, check.reliability coefH, plot.restscore.class, summary.restscore.class

Examples

data(acl)
Communality <- acl[,1:10]
Gplus <- check.errors(Communality, TRUE, FALSE)$Gplus
Oplus <- check.errors(Communality, FALSE, TRUE)$Oplus
hist(Gplus, breaks = 0:max(Gplus))
Usage

```
check.iio(X, method="MIIO", minvi = default.minvi, minsize = default.minsize,
alpha = .05, item.selection=TRUE, verbose=FALSE)
```

Arguments

- **X**: matrix or data frame of numeric data containing the responses of `nrow(X)` respondents to `ncol(X)` items. Missing values are not allowed.
- **method**: Either "MIIO" (default), "MSCPM", or "IT". Partial matching is allowed (e.g. `method="ms"` is equivalent to `method="MSCPM"`).
- **minvi**: minimum size of a violation that is reported. By default `minvi = .03` times the number of item step response functions (`m`) for Method MIIO; `minvi = .03` for Method IT and Method MSCPM.
- **minsize**: minimum size of a rest score group. By default `minsize = N/10` if `N ≥ 500`; `minsize = N/5` if `250 ≤ N < 500`; and `minsize = max(N/3, 50)` if `N < 250`.
- **alpha**: Nominal Type I error for t test (Method MIIO), z test (Method MSCPM), or McNemar test (Method IT). Default `alpha = .05`.
- **item.selection**: Conduct backward item selection procedure (see Ligtvoet et al., 2010). Default `item.selection=TRUE`.
- **verbose**: Show the results of the backward item selection algorithm on screen. Default `verbose=FALSE`.

Details

The output is of class `iio.class`, and is often numerous. Functions `plot` and `summary` can be used to summarize the output. See Van der Ark (2014) for an example.

Value

- **results**: A list with as many components as there are item pairs. Each component itself is also a list containing the results of the investigation of IIO.
- **violations**: A matrix: Summary of the backward item selection (Corresponds to Table 4 in Ligtvoet et al., 2010, and Table 1 in Ligtvoet et al., 2011). The first column gives, for each item, the number of violations of IIO. If the number of violations is nonzero, then the item with the largest number of violations is removed. If two or more items have the maximum number of violations, then from those items the item producing the lowest value of Loevinger’s H is removed. The second column shows the number of violations with one item removed, the third column shows the number of violations with two items removed, etc.
- **items.removed**: List of the items removed in chronological order.
- **HT**: Coefficient HT for the remaining items. For the use of coefficient HT see Ligtvoet et al. (2010). If the sample size is extremely large coefficient HT is estimated using a random subsample.
- **method**: The argument method.
- **item.mean**: The mean item scores.
Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References


See Also

check.errors, check.monotonicity, check.pmatrix, check.reliability, check.rests, coefH, plot.iio.class, summary.iio.class.

Examples

# Examples from Ligtvoet et al. (2010).

data(acl)

Order <- acl[,41:50]
summary(check.iio(Order))
plot(check.iio(Order))

Autonomy <- acl[,91:100]
summary(check.iio(Autonomy))
plot(check.iio(Autonomy))

# Examples from Ligtvoet et al. (2011).

data(cavalini)
X1 <- cavalini[,c(3,5,6,7,9,11,13,14)]

# Use Method MIIO and remove items violating MIIO
iio.list1 <- check.iio(X1)
summary(iio.list1)
X2 <- X1[,is.na(charmatch(dimnames(X1)[[2]],names(iio.list1$items.removed)))]
# Use Method MSCPM and remove items violating MSCPM
iio.list2 <- check.iio(X2, method="MSCPM")
summary(iio.list2)
X3 <- X2[, is.na(charmatch(dimnames(X2)[[2]], names(iio.list2$items.removed)))]

# Use Method IT
iio.list3 <- check.iio(X3, method="IT")
summary(iio.list3)

---

check.monotonicity  Check of Monotonicity

Description

Returns a list (of class monotonicity.class) with results from the investigation of monotonicity (Junker and Sijtsma, 2000; Mokken, 1971; Molenaar and Sijtsma, 2000; Sijtsma and Molenaar, 2002).

Usage

check.monotonicity(X, minvi = 0.03, minsize = default.minsize)

Arguments

X matrix or data frame of numeric data containing the responses of nrow(X) respondents to ncol(X) items. Missing values are not allowed

minvi minimum size of a violation that is reported

minsize minimum size of a rest score group. By default minsize = N/10 if N ≥ 500; minsize = N/5 if 250 ≤ N < 500; and minsize = max(N/3, 50) if N < 250

Details

The output is of class monotonicity.class, and is often numerous. Functions plot and summary can be used to summarize the output. See Van der Ark (2007) for an example.

Value

results A list with as many components as there are items. Each component itself is also a list containing the results of the check of monotonicity.

I.labels The item labels

Hi The item scalability coefficients Hi

m The number of answer categories.

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>
References


See Also

`check.errors, check.iio, check.restscore, check.pmatrix, check.reliability, coefH, plot.monotonicity.class, summary.monotonicity.class`

Examples

```r
data(acl)
Communality <- acl[,1:10]
monotonicity.list <- check.monotonicity(Communality)
plot(monotonicity.list)
summary(monotonicity.list)
```

---

**check.norms**

*Standard errors for norm statistics*

**Description**

The function presents standard errors for the mean, standard deviation, standard scores, stanine boundaries, and percentiles based on a vector of test scores (Oosterhuis, Van der Ark, and Sijtsma, in press).

**Usage**

`check.norms(y, nice.output = TRUE)`

**Arguments**

- `y` numerical vector. Typically a numerical vector of length N, representing the test scores of N respondents. Missing values are not allowed
- `nice.output` Logical: If TRUE, norm statistics and standard errors are combined in an a single object of class noquote
Value

list of five components:

(1) mean: Sample mean and its standard error (noquote).
(2) sd: Sample standard deviation and its standard error (noquote).
(3) z: For each unique testscore, the test score, its frequency, the corresponding estimated standard score and its standard error (noquote).
(4) sta9: The estimates of the 8 boundaries of the stanines and their standard error (noquote).
(5) z: For each unique testscore, the test score, its frequency, the corresponding estimated percentile rank and its standard error (noquote).

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl> and H. E. M. Oosterhuis

References


See Also

DS14, recode, twoway

Examples

data(DS14)

# Handle missing data and recode negatively worded items
X <- DS14[, 3 : 16]
X <- twoway(X)
X <- recode(X, c(1, 3))

# Negative affectivity
Na <- X[, c(1, 3, 6, 8, 10, 11, 14)]

# Social inhibition
Si <- X[, c(2, 4, 5, 7, 9, 12, 13)]

# Norms
check.norms(rowSums(Na))
check.norms(rowSums(Si))
check.pmatrix

Check of Nonintersection Using Method Pmatrix

Description

Returns a list (of class pmatrix.class) with results from the investigation of nonintersection using method pmatrix (Mokken, 1971; Molenaar and Sijtsma, 2000; Sijtsma and Molenaar, 2002).

Usage

check.pmatrix(X, minvi = 0.03)

Arguments

- **X**: matrix or data frame of numeric data containing the responses of nrow(X) respondents to ncol(X) items. Missing values are not allowed
- **minvi**: minimum size of a violation that is reported

Details

The output is often numerous. Functions plot and summary can be used to summarize the output. See Van der Ark (2007) for an example.

Value

- **results**: A list with as many components as there are item pairs. Each component itself is also a list containing the results of the check of nonintersection using Method pmatrix. The P(++) matrix and P(- -) (Molenaar and Sijtsma, 2000; Sijtsma and Molenaar, 2002) are also included.
- **I.item**: vector indicating to which items the rows and column the P(++) matrix belong
- **I.step**: the labels of the item steps in order of popularity
- **I.labels**: the item labels
- **Hi**: the item scalability coefficients Hi
- **minvi**: the value of minvi

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References

check.reliability

Description

Returns a list of reliability statistics: Molenaar Sijtsma (MS, 1984, 1988) statistic (a.k.a rho; also see, Sijtsma and Molenaar, 1987; Van der Ark, 2010), Cronbach’s (1951) alpha, Guttman’s (1945) lambda 2, and the latent class reliability coefficient (LCRC; Van der Ark, Van der Palm, and Sijtsma, 2011).

Usage

check.reliability(X, MS = TRUE, alpha = TRUE, lambda.2 = TRUE, LCRC = FALSE, nclass = nclass.default, irc = FALSE)

Arguments

X matrix or data frame of numeric data containing the responses of nrow(X) re-
spondents to ncol(X) items. Missing values are not allowed
MS Boolean. If TRUE, The MS statistic is computed.
alpha Boolean. If TRUE, Cronbach’s alpha is computed.
lambda.2 Boolean. If TRUE, Guttman’s Lambda 2 is computed.
LCRC Boolean. If TRUE, the LCRC is computed.
nclass Integer. Number of latent classes for the computation of LCRC. By default: the
number of items minus 1.
irc Boolean. If TRUE, the item-rest correlation (a.k.a. corrected item-total corre-
lation) is computed.
check.reliability

Details

The computation of LCRC depends on the package poLCA, which in its turn depends on the packages MASS and scatterplot3d. Computation of the LCRC may be time consuming if the number of latent classes is large. The optimal number of latent classes should be determined prior to the computation of the LCRC, using software for latent class analysis (e.g., the R-package poLCA).

Value

MS Molenaar Sijtsma statistic (a.k.a. rho).
alpha Cronbach’s alpha
lambda.2 Guttman’s Lambda 2
LCRC LCRC

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References


See Also

check.errors, check.iio, check.monotonicity, check.pmatrix check.restscore, coefH
check.restscore

Examples

data(acl)
Communality <- acl[,1:10]
check.reliability(Communality, LCRC = TRUE)

Description

Returns a list (of class restscore.class) with results from the investigation of nonintersection using method restscore (Mokken, 1971; Molenaar and Sijtsma, 2000; Sijtsma and Molenaar, 2002).

Usage

check.restscore(X, minvi = 0.03, minsize = default.minsize)

Arguments

X matrix or data frame of numeric data containing the responses of nrow(X) respondents to ncol(X) items. Missing values are not allowed
minvi minimum size of a violation that is reported
minsize minimum size of a rest score group. By default minsize = N/10 if N ≥ 500; minsize = N/5 if 250 ≤ N < 500; and minsize = max(N/3, 50) if N < 250

Details

The output is often numerous. Procedure may be slow for large data sets. Functions plot and summary can be used to summarize the output. See Van der Ark (2007) for an example.

Value

results A list with as many components as there are item pairs. Each component itself is also a list containing the results of the check of nonintersection using method restscore.
I.labels The item labels
Hi The item scalability coefficients
m The number of answer categories.

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>
References

See Also
check.errors, check.iio, check.monotonicity, check.pmatrix, check.reliability coefH, plot.restscore.class, summary.restscore.class

Examples
```r
data(acl)
Communality <- acl[,1:10]
restscore.list <- check.restscore(Communality)
plot(restscore.list)
summary(restscore.list)
```

```r
coefH

Scalability coefficients H
```

Description
Computes item-pair scalability coefficients Hij, item scalability coefficients Hi, and scale scalability coefficient H (Loevinger, 1948; Mokken, 1971, pp. 148-153; Molenaar & Sijtsma, 2000, pp. 11-13; Sijtsma & Molenaar, chap. 4; Van der Ark, 2007; 2010), as well as their standard errors (Kuijpers, Van der Ark, & Croon, 2013; also see Van der Ark, Croon, & Sijtsma, 2008). Mokken’s coefficients and standard errors can also be estimated in two-level data (Koopman, Zijlstra, & Van der Ark, 2020). It is also possible to compare scalability coefficients across groups using the item-step ordering of the entire sample (cf. CHECK=GROUPS option in MSP; Molenaar and Sijtsma, 2000).

Usage
```r
coefH(X, se = TRUE, nice.output = TRUE, level.two.var = NULL, group.var = NULL, fixed.itemstep.order = NULL)
```

Arguments
- **X**: matrix or data frame of numeric data containing the responses of nrow(X) respondents to ncol(X) items. Missing values are not allowed
- **se**: Logical: If TRUE, the standard errors of the scalability coefficients are given
nice.output Logical: If TRUE, scalability coefficients and standard errors are combined in an a single object of class noquote

level.two.var vector of length nrow(X) or matrix with number of rows equal to nrow(X) that indicates the level two variable for nested data to get appropriate standard errors (Koopman et al., 2020).

group.var vector of length nrow(X) or matrix with number of rows equal to nrow(X) to be used as grouping variable

fixed.itemstep.order matrix with number of rows equal to the number of item steps (m) and number of columns equal to the number of items (J). The matrix should consis the integers 1 : (m * J), indicating a predefined order of the items steps with respect to popularity. Value 1 indicates the easiest (most popular) item step, value (m * J) indicates the most difficult item step.

Details

May not work if any of the item variances equals zero. Such items should not be used in a test and removed from the data frame.

If nice.output = TRUE and se = TRUE, the result is a list of 3 objects of class noquote; if nice.output = FALSE and se = TRUE, the result is a list of 6 matrices (3 for the scalability coefficients and 3 for the standard errors); and if se = FALSE, the result is a list of 3 matrices (for the scalability coefficients). If level.two.var is not null the standard errors are adjusted to take the nesting into account. If group.var = Y with Y having K values, an additional element named Groups is added to the list. Element Groups shows the scalability coefficients per group ordered by means of sort (see Sys.getlocale for details). group.var returns coefficients for groups containing at least two case. Computation of standard errors can be slow for a combination of a large sample size and a large number of items.

Value

Hij scalability coefficients of the item pairs (possibly with standard errors; see details)

Hi vector containing scalability coefficients of the items (possibly with standard errors; see details)

H scalability coefficient of the entire scale (possibly with standard error; see details)

se.Hij standard errors of the scalability coefficients of the item pairs (only if nice.output = FALSE and se = TRUE; see details)

se.Hi standard errors of the scalability coefficients of the items (see details)

se.H standard error of the scalability coefficient of the entire scale (see details)

Groups Scalability coefficients for subgroups (see details)

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl> L. Koopman
References


See Also
coeffZ, search.normal

Examples

data(acl)
Communality <- acl[,1:10]

# Compute scalability coefficients and standard errors
coeffH(Communality)

# Scalability coefficients but no standard errors
coeffH(Communality, se=FALSE)

# Scalability coefficients for different groups:
subgroup <- ifelse(acl[,11] < 2,1,2)
coeffH(Communality, group.var = subgroup)

# Nested data:
data(autonomySupport)
scores <- autonomySupport[, -1]
classes <- autonomySupport[, 1]
coeffH(scores, level.two.var = classes)
Description

Computes Zij-values of item pairs, Zi-values of items, and Z-value of the entire scale, which are used to test whether Hij, Hi, and H, respectively, are significantly greater than some given lower-bound using the original method Z (Molenaar and Sijtsma, 2000, pp. 59-62; Sijtsma and Molenaar, p. 40; Van der Ark, 2007; 2010) or the delta method (Kuijpers, Van der Ark, & Croon, 2013; Koopman, Zijlstra, & Van der Ark, 2020). The delta method can also handle nested data.

Used in the function aisp

Usage

coefZ(X, lowerbound = 0, type.se = "delta", level.two.var = NULL)

Arguments

X matrix or data frame of numeric data containing the responses of nrow(X) respondents to ncol(X) items. Missing values are not allowed
lowerbound Value of the null hypothesis to which the scalability are compared to compute the Z-score (see details), 0 <= lowerbound < 1. The default is 0.
type.se Indicates which type of standard errors is used to compute the Z-score: "delta": uses standard errors approximated by the delta method (Kuijpers, Van der Ark, Kroon, 2013; Koopman, Zijlstra, Van der Ark, 2020); "Z": uses original Z-test (Mokken, 1971; Molenaar and Sijtsma, 2000; Sijtsma and Molenaar, 2002). The default is "delta".
level.two.var vector of length nrow(X) or matrix with number of rows equal to nrow(X) that indicates the level two variable for nested data (Koopman et al., 2020).

Details

The Z-score for the estimated item-pair coefficient Hij with standard error SE(Hij) is computed as

\[ Z = \frac{(Hij - lowerbound)}{SE(Hij)} \]

Unlike coefH, standard errors are not provided.

Value

Zij matrix containing the Z-values of the item-pairs
Zi vector containing Z-values of the items
Z Z-value of the entire scale

Author(s)

L. A. van der Ark <l.a.vanderark@uva.nl> L. Koopman
References


See Also

coeffH, aisp

Examples

data(acl)
Communality <- acl[,1:10]

# Compute the Z-score of each coefficient
coeffH(Communality)
coeffZ(Communality)

# Using lowerbound .3
coeffZ(Communality, lowerbound = .3)

# Z-scores for nested data
data(autonomySupport)
scores <- autonomySupport[, -1]
classes <- autonomySupport[, 1]
coeffH(scores, level.two.var = classes)
coeffZ(scores, level.two.var = classes)

Description

Gender, age, and item scores on the DS14 questionnaire of 541 coronary artery disease patients.
Usage

data(DS14)

Format

A 541 by 16 matrix containing gender, age, and item scores on the DS14 questionnaire.

Details

The DS14 (Denollet, 2005) is the most accepted and widely used diagnostic instrument for the assessment of the type-D pattern. Type D (distressed) is defined as the joint tendency towards negative affectivity (e.g., worry, irritability, gloom) and social inhibition (e.g., reticence and a lack of self-assurance). DS14 contains 14 items, each having five ordered response categories (0 = completely disagree, 1 = disagree, 2 = agree nor disagree, 3 = agree, 4 = completely agree). Items 2, 4, 5, 7, 9, 12, and 13 measure negative affectivity. Items 1, 3, 6, 8, 10, 11, and 14 measure social inhibition. Items 1 and 3 are negatively worded (indicated by an asterisk in the dimnames).

The data contain the gender (Male) of the patients (1 = male, 0 = female), the age (Age) of the patients in years, and the scores to DS14. Ten item scores are missing. Items 1 and 3 must be recoded before the data can be used meaningfully. The data have been used to investigate predictive value of social inhibition and negative affectivity for cardiovascular events and mortality in patients with coronary artery disease (Denollet et al., 2013), to investigate the relation between Type D and inflammation and endothelial dysfunction’ (van Dooren et al., 2016), and to investigate the relation between Type D and increased macrophage activity (Zuccarella-Hackl et al., 2016) These data have also been analyzed in papers on Mokken scale analysis (Sijsma and Van der Ark, 2016; Straat et al., 2016).

Source

Data were kindly made available by J. Denollet from Tilburg University.

References


See Also

recode, twoway

Examples

data(DS14)

# Handle missing data and recode negatively worded items
X <- DS14[, 3 : 16]
X <- twoway(X)
X <- recode(X, c(1, 3))
head(X)

Description

Computes the intraclass correlation for multilevel data (ICC; Snijders & Bosker, 1999, p. 17) for each item and the total scale of a questionnaire (Koopman, Van der Ark, & Zijlstra, 2020) and the F-test for the null hypothesis that the (total scale) ICC is zero (Snijders & Bosker, 2012, p. 22)

Usage

ICC(X)

Arguments

X matrix or data frame of numeric data containing a grouping column and the item scores of nrow(X) respondents to ncol(X) -1 items. Missing values are not allowed

Value

itemICC The ICC per item
scaleICC The ICC for the total scale and the corresponding results for the F-test

Author(s)

Letty Koopman <V.E.C.Koopman@uva.nl> L. Andries van der Ark <L.A.vanderArk@uva.nl>
References


See Also

SWMD, MLcoefH.

Examples

# Data example (Koopman et al., 2020)
data(SWMD)

# Compute ICC
ICC(SWMD)

MLcoefH | Two-level scalability coefficients H

Description

Computes all types of two-level scalability coefficients (Snijders, 2001; Crisan, Van de Pol, & Van der Ark, 2016), that is, between- and within-rater item-pair scalability coefficients and their ratio (HBij, HWij, and BWij, respectively), between- and within-rater item scalability coefficients and their ratio (Hbi, HWi, and BWi, respectively), and between- and within-rater total scale scalability coefficients and their ratio (HB, HW, and BW, respectively). In addition, standard errors are estimated (Koopman, Zijlstra, & Van der Ark, 2019). Note that this version is an adaptation of the estimation methods described in Snijders and in Koopman et al., as the group proportions are now by default weighted for group size (Koopman, Zijlstra, & Van der Ark, 2020). As a result, the estimates for the autonmysupport data differs from the printed table in Koopman et al. (2019).

Usage

MLcoefH(X, se = TRUE, nice.output = TRUE, subject = 1, fixed.itemstep.order = NULL, weigh.props = TRUE)

Arguments

X matrix or data frame of numeric data containing a subject indicator column and the responses of nrow(X) raters to ncol(X)-1 items. Missing values are not allowed

se Logical: If TRUE, the standard errors are printed alongside the scalability coefficients
nice.output Logical: If TRUE, scalability coefficients and standard errors are combined in an a single object of class noquote. Item-pair ratios BWij are only printed if FALSE

subject Represents the subject column. Default is column 1.

fixed.itemstep.order matrix with number of rows equal to the number of item steps (m) and number of columns equal to the number of items (J). The matrix should consis the integers 1 : (m * J), indicating a predefined order of the items steps with respect to popularity. Value 1 indicates the easiest (most popular) item step, value (m * J) indicates the most difficult item step.

weigh.props If TRUE: Use weighted proportions across groups to estimate coefficients and standard errors, if FALSE: Use averaged proportions across groups to estimate coefficients and standard errors.

Details
If se = TRUE and nice.output = TRUE, the result is a list of 3 objects of class noquote; if se = TRUE and nice.output = FALSE, the result is a list of 3 matrices, one per set of coefficients; and if se = FALSE, the result is a list of 3 matrices containing only the scalability coefficients. Computation of standard errors can be slow for a combination of many subjects and a large number of items.

Value

Hij Scalability coefficients of the item pairs, upper triangle are the between-rater coefficients, and the lower triangle the within-rater coefficients (possibly with standard errors; see details). If nice.output = FALSE this returns a matrix with the type of coefficients per column

Hij Within-rater scalability coefficients of the item pairs (possibly with standard errors; see details)

Hi Between- and within-rater scalability coefficients of the items and their ratio (possibly with standard errors; see details)

H between- and within-rater scalability coefficients of the entire scale and their ratio (possibly with standard error; see details)

Author(s)
Letty Koopman <V.E.C.Koopman@uva.nl> L. Andries van der Ark <L.A.vanderArk@uva.nl>

References


See Also

coeffH,

Examples

```r
# Small data example (Koopman et al., 2019)
smallData <- data.frame(Subs = c(1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3),
                        Xa = c(0, 0, 1, 0, 1, 1, 1, 2, 1, 0, 1, 2, 0, 0, 0),
                        Xb = c(0, 0, 1, 0, 2, 2, 2, 1, 2, 1, 2, 2, 1, 1, 0))
MLcoefH(smallData)

# Load real data example. Note that due to an estimation adaptation (Koopman et al., 2020)
# the results differ from the table in Koopman et al. (2019).
data(autonomySupport)

# Compute scalability coefficients with or without standard errors and nice output
H.se.nice <- MLcoefH(autonomySupport)
H.se.nice

H.se.not <- MLcoefH(autonomySupport, nice.output = FALSE)
H.se.not

H.nice <- MLcoefH(autonomySupport, se = FALSE)
H.nice

H.not <- MLcoefH(autonomySupport, se = FALSE, nice.output = FALSE)
H.not
```

MLweight  
*Weights for Guttman Errors in two-level test data*
**MLweight**

**Description**

Computes weights for Guttman errors in two-level test data (Koopman, Van der Ark, & Zijlstra, 2017)

**Usage**

```r
MLweight(X, maxx = NULL, minx = NULL, itemstep.order = NULL)
```

**Arguments**

- **X**: Data matrix with a subject column and two item columns. The subject column is assumed to be the first.
- **maxx**: The highest possible answer category. If not specified it is determined by using the highest item score.
- **minx**: The lowest possible answer category. If not specified it is determined by using the lowest item score.
- **itemstep.order**: The two columns pertaining the two items in question, from a (possibly larger) matrix with number of rows equal to the number of item steps (m) and number of columns equal to the number of items (J). The matrix should consist the integers 1 : (m * J), indicating a predefined order of the items steps with respect to popularity. Value 1 indicates the easiest (most popular) item step, value (m * J) indicates the most difficult item step.

**Value**

Returns a vector with the weights for each item-score pattern of a given item-pair. In case of ties in item popularities the average weights across possible item-orderings are returned.

**Author(s)**

L. Koopman <V.E.C.Koopman@uva.nl> L. A. van der Ark <L.A.vanderArk@uva.nl>

**References**


**See Also**

- MLcoefH
Examples

```r
smallData <- data.frame(Subs = c(1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 3),
Xa = c(2, 0, 1, 0, 2, 0, 2, 1, 2, 2, 1, 2, 2, 2),
Xb = c(1, 1, 0, 1, 2, 1, 2, 1, 0, 2, 2, 2, 2, 2),
Xc = c(0, 0, 0, 1, 0, 2, 2, 1, 2, 1, 0, 0, 1, 2))

# Compute the weights
I <- 3
for(i in 1:(I - 1)) for(j in (i + 1):I)
print(MLweight(smallData[, c(1, i + 1, j + 1)], minx = 0, maxx = 2))
```

plot.iio.class  
Plot iio.class objects

Description

S3 Method to plot objects of class iio.class. Graphic display of the checks of iio. One graph for each item plotting the estimated item response functions.

Usage

```r
## S3 method for class 'iio.class'
plot(x, item.pairs = all.pairs, ci = TRUE, alpha = .05,
     color = c("black", "blue"), transparency = 20, ask = TRUE, ...)
```

Arguments

- `x`: Object of class iio.class produced by `check.iio`.
- `item.pairs`: Vector containing the numbers of the item pairs for which the results are depicted graphically. For example, `item.pairs = 1` prints the results for items 1 and 2, `item.pairs = 2` prints the results for items 1 and 3, `item.pairs = 3` prints the results for items 2 and 3, and `item.pairs = 4` prints the results for items 2 and 4. Default the results for all item pairs are depicted.
- `ci`: Boolean. If TRUE (default), then confidence envelops are plotted around IRFs.
- `alpha`: Type of plotted (1 - alpha) confidence intervals. By default 95-percent confidence intervals are depicted.
- `color`: Color of the plotted curves and confidence envelops. Defaults are black for the first item and blue for the second item.
- `transparency`: Transparancy of the confidence intervals. Higher values result in more opaque colors for the confidence intervals.
- `ask`: Boolean. If TRUE (default), then `par("ask"=TRUE)`; i.e., a hard return between subsequent plots is required. If FALSE, then `par("ask"=FALSE)`.
- `...`: Optional graphical parameters will be ignored.
The plot function corresponds to method MIIO; each graph plots the estimated item response functions (item rest-score functions) for two items. For details of the method, see Ligtvoet et al. (2010, 2011); Sijtsma et al. (2012). For details of the confidence envelopes, see Van der Ark (2012b). For the implementation in R, see Van der Ark (2012a). For ask==FALSE, the default graphic device in R may only display the last graph.

Returns a graph.

L. A. van der Ark <L.A.vanderArk@uva.nl>


See Also

`check.iio`, `summary.iio.class`

Examples

data(acl)
Communality <- acl[,1:10]
iio.list <- check.iio(Communality)
summary(iio.list)
plot(iio.list)
plot.monotonicity.class

Plot monotonicity.class objects

Description

S3 Method to plot objects of class monotonicity.class. Graphic display of the checks of monotonicity. One graph for each item plotting the estimated item step response functions and/or item response function, plus confidence envelopes (Van der Ark, 2012).

Usage

## S3 method for class 'monotonicity.class'
plot(x, items = all.items, curves = "both", ci = TRUE, 
   alpha = .05, color = "black", transparancy = 20, ask = TRUE, ...)

Arguments

x Object of class monotonicity.class produced by check.monotonicity.
items vector containing the numbers of the items for which the results are depicted graphically. Default the results for all items are depicted.
curves "ISRF": Item step response functions (ISRFs) are depicted; "IRF": item response functions (IRFs) are depicted; "both": Both ISRFs and IRFs are depicted. Default is "both"
ci Boolean. If TRUE (default), then confidence envelops are plotted around IRFs and ISRFs.
alpha Type of plotted (1 - alpha) confidence intervals. By default 95-percent confidence intervals are depicted
color Color of the plotted curves and confidence envelops. Default is black.
transparancy Transparancy of the confidence intervals. Higher values result in more opaque colors for the confidence intervals.
ask Boolean. If TRUE (default), then par("ask"=TRUE); i.e., a hard return between subsequent plots is required. If FALSE, then par("ask"=FALSE).
...
Optional graphical parameters will be ignored

Details

For details of the method, see Molenaar and Sijtsma (2000) and Sijtsma and Molenaar (2002). For details of the confidence envelopes, see Van der Ark (2012) For the implementation in R, see Van der Ark (2007). For curves="both", both plots are plotted simultaneously using layout(matrix(c(1,2)1,2))
For ask="FALSE", the default graphic device in R may only display the last graph.

Value

Returns a graph.
plot.pmatrix.class

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References


See Also

check.monotonicity, summary.monotonicity.class

Examples

data(acl)
Communality <- acl[,1:10]
monotonicity.list <- check.monotonicity(Communality)
plot(monotonicity.list)
summary(monotonicity.list)

plot.pmatrix.class

Plot pmatrix.class objects

Description

S3 Method to plot objects of class pmatrix.class. Graphic display of the checks of pmatrix. One graph for each item plotting the rows of the P(++) matrix and rows of the P(−) matrix. If nonintersection holds the lines in the plots of the P(++) matrix must be nondecreasing and the lines in the plots of the P(−) matrix must be nonincreasing.

Usage

## S3 method for class 'pmatrix.class'
plot(x, items = all.items, pmatrix = "both", ci = TRUE,
   alpha = .05, color = "black", transparancy = 20, ask = TRUE, ...)

head(Communality, 4)
Communality[][]

> plot.pmatrix.class

plot.pmatrix.class

Plot pmatrix.class objects

Description

S3 Method to plot objects of class pmatrix.class. Graphic display of the checks of pmatrix. One graph for each item plotting the rows of the P(++) matrix and rows of the P(−) matrix. If nonintersection holds the lines in the plots of the P(++) matrix must be nondecreasing and the lines in the plots of the P(−) matrix must be nonincreasing.

Usage

## S3 method for class 'pmatrix.class'
plot(x, items = all.items, pmatrix = "both", ci = TRUE,
   alpha = .05, color = "black", transparancy = 20, ask = TRUE, ...)

head(Communality, 4)
Communality[][]

> plot.pmatrix.class

plot.pmatrix.class

Plot pmatrix.class objects

Description

S3 Method to plot objects of class pmatrix.class. Graphic display of the checks of pmatrix. One graph for each item plotting the rows of the P(++) matrix and rows of the P(−) matrix. If nonintersection holds the lines in the plots of the P(++) matrix must be nondecreasing and the lines in the plots of the P(−) matrix must be nonincreasing.

Usage

## S3 method for class 'pmatrix.class'
plot(x, items = all.items, pmatrix = "both", ci = TRUE,
   alpha = .05, color = "black", transparancy = 20, ask = TRUE, ...)

head(Communality, 4)
Communality[][]

> plot.pmatrix.class

plot.pmatrix.class

Plot pmatrix.class objects

Description

S3 Method to plot objects of class pmatrix.class. Graphic display of the checks of pmatrix. One graph for each item plotting the rows of the P(++) matrix and rows of the P(−) matrix. If nonintersection holds the lines in the plots of the P(++) matrix must be nondecreasing and the lines in the plots of the P(−) matrix must be nonincreasing.

Usage

## S3 method for class 'pmatrix.class'
plot(x, items = all.items, pmatrix = "both", ci = TRUE,
   alpha = .05, color = "black", transparancy = 20, ask = TRUE, ...)

head(Communality, 4)
Communality[][]

> plot.pmatrix.class

plot.pmatrix.class

Plot pmatrix.class objects

Description

S3 Method to plot objects of class pmatrix.class. Graphic display of the checks of pmatrix. One graph for each item plotting the rows of the P(++) matrix and rows of the P(−) matrix. If nonintersection holds the lines in the plots of the P(++) matrix must be nondecreasing and the lines in the plots of the P(−) matrix must be nonincreasing.

Usage

## S3 method for class 'pmatrix.class'
plot(x, items = all.items, pmatrix = "both", ci = TRUE,
   alpha = .05, color = "black", transparancy = 20, ask = TRUE, ...)
Arguments

x
Object of class pmatrix.class produced by check.pmatrix.

items
vector containing the numbers of the item pairs for which the results are depicted graphically. Default the results for all items are depicted.

pmatrix
Valid options are "ppp", "pmm", and "both". If pmatrix="ppp", then the P(++) matrix is plotted, if pmatrix="pmm", then the P(−) matrix is plotted, if pmatrix="both", then both the P(++) matrix and P(−) matrix are plotted.

ci
Boolean. If TRUE, then confidence envelopes are plotted around IRFs and ISRFs.

alpha
Type of plotted (1 - alpha) confidence intervals. By default 95-percent confidence intervals are depicted

color
Color of the plotted curves and confidence envelopes. Default is black.

color
Transparancy of the confidence intervals. Higher values result in more opaque colors for the confidence intervals.

ask
Boolean. If TRUE (default), then par("ask"=TRUE); i.e., a hard return between subsequent plots is required. If FALSE, then par("ask"=FALSE).

...
Optional graphical parameters will be ignored

Details

The default graphic device in R may only display the last graph.

In the plot of the P(++) matrix and the P(−) matrix, the x-axis contains the $k = (J - 1)m$ item steps not pertaining to item j in order of popularity (ascending). Let $Y_g = 1$ indicate that the g-th item step has been taken: i.e. $X_i \geq y$ and let $Y_g = 0$ indicate that the g-th item step has not been taken: i.e. $X_i < y$. The $m$ lines in the plot of the P(++) matrix connect $P(X_j \geq x, Y_1 = 1), \ldots, P(X_j \geq x, Y_k = 1)), x = 1, \ldots, m$. The $m$ lines in the plot of the P(−) matrix connect $P(X_j < x, Y_1 = 0), \ldots, P(X_j < x, Y_k = 0)), x = 1, \ldots, m$.

If the number of item steps on the x-axis is greater than 10, then the labels are not displayed.

Value

Returns a graph.

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References


plot.restscore.class

See Also

check.pmatrix, summary.pmatrix.class

Examples

data(acl)
Communality <- acl[,1:10]
pmatrix.list <- check.pmatrix(Communality)
plot(pmatrix.list)
summary(pmatrix.list)

plot.restscore.class  Plot restscore.class objects

Description

S3 Method to plot objects of class restscore.class. Graphic display of the checks of restscore. One graph for each item pair plotting the estimated item step response functions (ISRFs); confidence envelopes are optional. Intersections of the lines indicate violations of nonintersection.

Usage

## S3 method for class 'restscore.class'
plot(x, item.pairs = all.pairs, ci = TRUE, alpha = .05,
     color = c("black", "blue"), transparancy = 20, ask = TRUE, ...)

Arguments

x Object of class restscore.class produced by check.restscore.

item.pairs vector containing the numbers of the item pairs for which the results are depicted graphically. For example, item.pairs = 1 prints the results for items 1 and 2, item.pairs = 2 prints the results for items 1 and 3, item.pairs = J prints the results for items 1 and SJ$, and item.pairs = J+1 prints the results for items 2 and 3. Default the results for all item pairs are depicted.

ci Boolean. If TRUE (default), then confidence envelops are plotted around ISRFs.

alpha Type of plotted (1 - alpha) confidence intervals. By default 95-percent confidence intervals are depicted

color Color of the plotted lines and confidence envelops. Defaults are black for the first item and blue for the second item.

transparancy Transparancy of the confidence intervals. Higher values result in more opaque colors for the confidence intervals.

ask Boolean. If TRUE (default), then par("ask"=TRUE); i.e., a hard return between subsequent plots is required. If FALSE, then par("ask"=FALSE).

... Optional graphical parameters will be ignored
Details

For details of the method, see Molenaar and Sijtsma (2000) and Sijtsma and Molenaar (2002). For details of the confidence envelopes, see Van der Ark (2012) For the implementation in R, see Van der Ark (2007). For ask==FALSE, the default graphic device in R may only display the last graph. The default number of plots can increase rapidly for large numbers of items.

Value

Returns a graph.

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References


See Also

check.restscore, summary.restscore.class

Examples

data(acl)
Communality <- acl[,1:10]
restscore.list <- check.restscore(Communality)
plot(restscore.list)
summary(restscore.list)

recode

Recodes negatively worded items

Description

Returns a matrix or data.frame with the indicated items recoded.

Usage

type{recode(X, items = NULL, values = defaultValues)}
Arguments

- **X**: matrix or data frame of numeric data containing the responses of `nrow(X)` respondents to `ncol(X)` items. Missing values are allowed
- **items**: Vector of integers indicating the items to be recoded
- **values**: Vector of possible item scores. By default the range of the observed values is taken

Details

The result is `X` for which columns `items` have been recoded.

Value

The result is `X` for which columns `items` have been recoded.

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

See Also

- **DS14**, **twoway**

Examples

```r
data(DS14)

# Handle missing data and recode negatively worded items
X <- DS14[, 3 : 16]
X <- twoway(X)
X <- recode(X, c(1, 3))
head(X)
```

---

**summary.iio.class**  
**Summarize iio.class objects**

Description

S3 Method for summary of objects of class `iio.class`. Summarize checks of invariant item ordering.

Usage

```r
## S3 method for class 'iio.class'
summary(object, ...)
```
Arguments

object  list produced by `check.iio`
...  Optional parameters will be ignored

Value

method  String describing the method used for investigating invariant item ordering: Either "MIIO" (Method Manifest Invariant Item Ordering), "MSCPM" (Method Manifest Scale Cumulative Probability Model), or "IT" (Method Increasingness in Transposition)

item.summary  Matrix with ncol(X) rows and 10 columns, showing for each item a summary of the violations of an invariant item ordering: \( \text{itemH} \) = Item-scalability coefficient; \(#ac\) = number of active pairs that were investigated; \(#vi\) = number of violations in which the item is involved; \(#vi/#ac\) = propotion of active pairs that is involved in a violation; \(\text{maxvi}\) = maximum violation; \(\text{sum}\) = sum of all violations; \(\text{tmax}\) (for method MIIO), \(\text{zmax}\) (for method MSCP), or \(\text{xmax}\) (for method IT) = maximum t-value, z-value, and chi-square value, respectively; \(\text{tsig}\) (for method MIIO), \(\text{zsig}\) (for method MSCP), or \(\text{xsig}\) (for method IT) = number of significant t-values, z-values, and chi-square values, respectively; \(\text{crit}\) = Crit value (Molenaar and Sijtsma, 2000, pp. 49, 74).

backward.selection  Matrix showing the number of violations for each item (rows) at each step of the backward item selection process (columns). The first column shows the number of violations for each item. Then in an iterative procedure, the item whose removal results in the largest decrease of violations is removed and the number of violations is computed again. If the reduction in the number of violations is undecisive then, from the candidate items, the item having the smallest scalability coefficient is removed. The backward selection procedure stops when there are no more violations.

HT  Numeric: Coefficient HT for the selected items. Given an IIO, coefficient HT expresses the strength of the ordering (Ligtvoet et al., 2010).

Author(s)

L. A. van der Ark <l.a.vanderArk@uva.nl>

References


### summary.monotonicity.class

#### Summarize monotonicity.class objects

**Description**

S3 Method for `summary` of objects of class `monotonicity.class`. Summarizes checks of monotonicity

**Usage**

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

**Arguments**

- `object` list produced by `check.monotonicity`
- `...` Optional parameters will be ignored

### Examples

# Examples from Ligtvoet et al. (2011).

data(cavalini)
X1 <- cavalini[,c(3,5,6,7,9,11,13,14)]

# Use Method MIIO and remove items violating MIIO
iio.list1 <- check.iio(X1)
summary(iio.list1)
plot(iio.list1)
X2 <- X1[,is.na(charmatch(dimnames(X1)[[2]],names(iio.list1$items.removed)))]

# Use Method MSCPM and remove items violating MSCPM
iio.list2 <- check.iio(X2,method="MSCPM")
summary(iio.list2)
X3 <- X2[,is.na(charmatch(dimnames(X2)[[2]],names(iio.list2$items.removed)))]

# Use Method IT
iio.list3 <- check.iio(X3,method="IT")
summary(iio.list3)

---


**See Also**

`check.iio, plot.iio.class`
Value

Matrix with ncol(\(X\)) rows and 10 columns, showing for each item a summary of the violations of monotonicity: \(i\text{temH} = \) Item-scalability coefficient; \#ac = number of active pairs that were investigated; \#vi = number of violations in which the item is involved; \#vi/\#ac = proportion of active pairs that is involved in a violation; maxvi = maximum violation; sum = sum of all violations; zmax = maximum z-value; zsig = number of significant z-values; crit = Crit value (Molenaar and Sijtsma, 2000, pp. 49, 74).

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References


See Also

check.monotonicity, plot.monotonicity.class

Examples

data(acl)
Communality <- acl[,1:10]
monotonicity.list <- check.monotonicity(Communality)
plot(monotonicity.list)
summary(monotonicity.list)

---

**summary.pmatrix.class**  
**Summarize pmatrix.class objects**

Description

S3 Method for summary of objects of class pmatrix.class. Summarize checks of nonintersection using method pmatrix.

Usage

```r
## S3 method for class 'pmatrix.class'
summary(object, ...)
```
**Arguments**

- **object**
  - list produced by `check.pmatrix`
- ...
  - Optional parameters will be ignored

**Details**

For obtaining the P++ and P-- matrix, see examples.

**Value**

Matrix with `ncol(X)` rows and 10 columns, showing for each item a summary of the violations of nonintersection using method pmatrix: itemH = Item-scalability coefficient; #ac = number of active pairs that were investigated; #vi = number of violations in which the item is involved; #vi/#ac = proportion of active pairs that is involved in a violation; maxvi = maximum violation; sum = sum of all violations; zmax = maximum z-value; zsig = number of significant z-values; crit = Crit value (Molenaar and Sijtsma, 2000, pp. 49, 74).

**Author(s)**

L. A. van der Ark <L.A.vanderArk@uva.nl>

**References**


**See Also**

- `check.pmatrix`
- `plot.pmatrix.class`

**Examples**

```r
data(acl)
Communality <- acl[,1:10]
pmatrix.list <- check.pmatrix(Communality)
plot(pmatrix.list)
summary(pmatrix.list)

# Small example showing how to retrieve the P++ matrix and the P-- matrix
SmallExample <- acl[,1:4]
pmatrix.list <- check.pmatrix(SmallExample)
pmatrix.list$results$Ppp
pmatrix.list$results$Pmm
```
Summary.restscore.class

Summarize restscore.class objects

Description

S3 Method for summary of objects of class restscore.class. Summarize checks of nonintersection using method restscore.

Usage

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

Arguments

- `object`: list produced by `check.restscore`
- `...`: Optional parameters will be ignored

Value

Matrix with ncol(X) rows and 10 columns, showing for each item a summary of the violations of nonintersection using method restscore: `itemH` = Item-scalability coefficient; `#ac` = number of active pairs that were investigated; `#vi` = number of violations in which the item is involved; `#vi/#ac` = proportion of active pairs that is involved in a violation; `maxvi` = maximum violation; `sum` = sum of all violations; `zmax` = maximum z-value; `zsig` = number of significant z-values; `crit` = Crit value (Molenaar and Sijtsma, 2000, pp. 49, 74).

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References


See Also

- `check.restscore`
- `plot.restscore.class`
Examples

data(acl)
Communality <- acl[,1:10]
restscore.list <- check.restscore(Communality)
plot(restscore.list)
summary(restscore.list)

SWMD Data Subset

Description

A subset of the COOL5-18 data (see below) with scores of 651 pupils nested in 30 classes on the 7-item Schaal Welbevinden Met Docenten [Scale Well-Being With Teachers] (Peetsma, Wagenaar, & de Kat, 2001; Zijsling, Keuning, Keizer-Mittelhauser, Naaijer, & Timmermans, 2017). R-code to get this subsample is available as online supplement to Koopman, Zijlstra, & Van der Ark (2020).

Usage

data(SWMD)

Format

A 651 by 8 data frame containing integers. The first column reflects a class indicator, the remaining columns the 7 items, see colnames(SWMD).

Details

Each item has five ordered answer categories from not true at all (score 0) to completely true (score 4).

<table>
<thead>
<tr>
<th>Item</th>
<th>Short</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lv_wdo1</td>
<td>The teachers usually know how I feel</td>
</tr>
<tr>
<td>2</td>
<td>lv_wdo2</td>
<td>I can talk about problems with the teachers</td>
</tr>
<tr>
<td>3</td>
<td>lv_wdo3</td>
<td>If I feel unhappy, I can talk to the teachers about it</td>
</tr>
<tr>
<td>4</td>
<td>lv_wdo4</td>
<td>I feel at ease with the teachers</td>
</tr>
<tr>
<td>5</td>
<td>lv_wdo5</td>
<td>The teachers understand me</td>
</tr>
<tr>
<td>6</td>
<td>lv_wdo6</td>
<td>I have good contact with the teachers</td>
</tr>
<tr>
<td>7</td>
<td>lv_wdo7</td>
<td>I would prefer to have other teachers (inversely coded)</td>
</tr>
</tbody>
</table>

The items were translated from Dutch. For the original items, see p. 79 in Zijsling et al. (2017).

Source

transreas

is accessible after login.

References


See Also

MLcoefH, ICC,

Examples

# Data example (Koopman et al., 2020)
data(SWMD)

# Compute ICC
ICC(SWMD)

transreas Transitive Reasoning

Description

Data came from 12 dichotomous items administered to 425 children in grades 2 through 6 (Verweij, Sijtsma, and Koops, 1996). Each item is a transitive reasoning task.

Usage
data(transreas)

Format

A 425 by 13 (grade and scores on 12 items) matrix containing integers. attributes(transreas) gives details on the items

Details

Items have two ordered answer categories, incorrect (score 0), correct (1). (for detailed information, see Sijtsma and Molenaar, 2002, p. 33).
### References


### Examples

```r
# Construction of Table 3.1 in Sijtsma and Molenaar (2002, p. 33)
data(transreas)
grades <- transreas[,1]
item.scores <- transreas[,-1]
Total.group <- round(apply(item.scores,2,mean),2)
for (i in 2:6) assign(paste("Grade." ,i,"sep=""),
round(apply(item.scores[grades==i,],2,mean),2))
Task <- c(9,12,10,11,4,5,2,7,3,1,8,6)
Property <- attributes(transreas)$property
Format <- attributes(transreas)$format
Objects <- attributes(transreas)$objects
Measures <- attributes(transreas)$measures
Table.3.1 <- data.frame(Task,Property,Format,Objects,Measures,
Total.group,Grade.2,Grade.3,Grade.4,Grade.5,Grade.6)
Table.3.1
```

### Description

Returns a single or multiple completed data sets using two-way imputation with normally distributed errors.
Usage

\texttt{twoway(X, nCompletedDataSets = 1, minX = defaultMinX, maxX = defaultMaxX, seed = FALSE)}

Arguments

- **X**: matrix or data frame of integer data containing the score of \texttt{now(X)} respondents to \texttt{nicol(X)} items. Typically \texttt{X} contains missing values.
- **nCompletedDataSets**: Number of completed data sets.
- **minX**: Minimum item score. By default, the minimum item score is the lowest score found in the data.
- **maxX**: Maximum item score. By default, the maximum item score is the highest score found in the data.
- **seed**: Seed for random sampling. If \texttt{seed} = \texttt{FALSE} (default), no seed is given, otherwise \texttt{seed} must be a numeric value. Replications having the same seed result in exactly the same outcome value.

Details

For single imputation (\texttt{nCompletedDataSets} == 1, default) the function returns an object of the same class as \texttt{X}, for multiple imputation (\texttt{nCompletedDataSets} > 1) the function returns a list. References for two-way imputation include Bernaards and Sijtsma (2000), Sijtsma and Van der Ark (2003), and Van Ginkel, Van der Ark, and Sijtsma (2007).

Value

The result is \texttt{X} for which the missing values have been replaced by imputed values. For multiple imputations, the result is a list of matrices/data frames. For single imputations, the result is a matrix/data frame.

Author(s)

L. A. van der Ark <L.A.vanderArk@uva.nl>

References


See Also

\texttt{DS14, recode}
Examples

data(DS14)

# Handle missing data and recode negatively worded items
X <- DS14[, 3 : 16]
X <- twoway(X)
X <- recode(X, c(1, 3))
head(X)
Index

*Topic datasets
  acl, 7
  autonomySupport, 11
  cavalini, 12
  DS14, 31
  SWMD, 51
  transreas, 52

  acl, 3, 7
  aisp, 3–5, 8, 31
  autonomySupport, 5, 11

  cavalini, 3, 12
  check.bounds, 13
  check.ca, 4, 14, 17
  check.errors, 3–5, 16, 19, 21, 24, 25, 27
  check.iio, 3–5, 10, 13, 17, 19, 21, 24, 25, 27, 38, 39, 46, 47
  check.monotonicity, 3, 5, 10, 17, 19, 20, 24, 25, 27, 40, 41, 47, 48
  check.norms, 4, 5, 21
  check.pmatrix, 3, 4, 10, 17, 19, 21, 23, 25, 27, 42, 43, 49
  check.reliability, 3, 10, 17, 19, 21, 24, 24, 27
  check.relistscore, 3, 4, 10, 19, 21, 24, 25, 26, 43, 44, 50
  coefH, 3–5, 10, 17, 19, 21, 24, 25, 27, 27, 31, 36
  coefZ, 3, 5, 29, 30

  DS14, 3, 4, 15, 22, 31, 45, 54

  ICC, 5, 33, 52

  MLcoefH, 5, 12, 34, 34, 37, 52
  MLweight, 36
  mokken, 4
  mokken (mokken-package), 2
  mokken-package, 2

  plot, 5
  plot.iio.class, 4, 19, 38, 47
  plot.monotonicity.class, 4, 21, 40, 48
  plot.pmatrix.class, 24, 41, 49
  plot.relistscore.class, 4, 17, 27, 43, 50
  recode, 4, 15, 22, 33, 44, 54

  search (aisp), 8
  search.normal, 3, 29
  sort, 28
  summary.iio.class, 19, 39, 45
  summary.monotonicity.class, 21, 41, 47
  summary.pmatrix.class, 24, 43, 48
  summary.relistscore.class, 17, 27, 44, 50
  SWMD, 6, 34, 51
  Sys.getlocale, 28

  transreas, 3, 52
  twoway, 4, 15, 22, 33, 45, 53