Package ‘ThurMod’

March 5, 2023

Type Package

Title Thurstonian CFA and Thurstonian IRT modelling

Version 1.0.0

Date 2023-03-03

Description Fit Thurstonian forced-choice models (CFA (simple and factor) and IRT) in R. This package allows for the analysis of item response modeling (IRT) as well as confirmatory factor analysis (CFA) in the Thurstonian framework. Currently, estimation can be performed by Mplus and lavaan. References:
Brown, A, & Maydeu-Olivares, A. (2011). Item response modeling of forced-choice questionnaires.Educational and Psychological Measurement, 71(3), 460-502. https://doi.org/10.1177/0013164410375112;
Jansen, M. T., & Schulze, R. (2023). The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data. Manuscript submitted;
Maydeu-Olivares, A., & Böckenholt, U. (2005). Structural equation modeling of paired-comparison and ranking data. Psychological Methods, 10(3), 285-304. https://doi.org/10.1037/1082-989X.10.3.285.

VignetteBuilder knitr

License GPL (>= 3)

Depends R (>= 3.5.0)

Imports utils, MASS, matrixStats, lavaan

Encoding UTF-8

NeedsCompilation no

Roxygen list(old_usage = TRUE)

RoxygenNote 7.2.0

R topics documented:

`blocksort` *Sorts the blocks in ascending numbering*

Description

This function sorts all items in a block into ascending order.

Usage

```
blocksort(blocks)
```

Arguments

`blocks`	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns represent the number of items per block.

Value

Returns a matrix consisting of the blocks where all items per blocks are sorted in ascending order.

Examples

```
# Define 30 items divided by ten triplets as blocks
blocks <- matrix(c(1:30), ncol = 3)

# sort the blocks
blocksort(blocks)
```

`count.combn` *Count paired comparisons*

Description

This function calculates the number of paired comparisons needed to compare a set of N items.

Usage

```
count.combn(nitem)
```

Arguments

`nitem`	Number of items.

Details

This function is only useful, if the number of paired comparisons of a full design, that is all possible paired comparisons, is of interest. Then the number is

$$\frac{N \times (N-1)}{2}$$

Value

An integer corresponding to the number of paired comparisons.

Examples

```
# Number if paired comparisons for a set of 15 items = 105.
count.combn(15)
```

`count.xblocks` *Determine the number of extra blocks*

Description

This function determines the minimal number of extra blocks needed in order to link all blocks.

Usage

```
count.xblocks(blocks)
```

Arguments

`blocks`	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

Details

The matrix of blocks must be constructed so that the number of columns corresponds to the number of items per block. The number of rows corresponds to the number of blocks for the specific measure. If $p$ is the number of blocks, and $k$ is the number of items per block (e.g. $k=3$ for triplets), then the number of extra blocks can be determined by (see also Jansen & Schulze, 2023)

$$\lceil \frac{p-1}{k-1} \rceil$$

Value

An integer corresponding to the number of extra blocks needed.

References

Jansen, M. T., & Schulze, R. (2023). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Examples

```
# Define a matrix of blocks
blocks <- matrix(1:15,ncol=3, byrow=TRUE)

# Determine the number of extra blocks needed
count.xblocks(blocks)
```

designA *Create the Thurstonian design matrix for paired comparison and ranking data.*

Description

Creates the Thurstonian design matrix for paired comparison and ranking data, given by blocks or the number of items.

Usage

```
designA(blocks = NULL, nitems = max(unique(blocks)))
```

Arguments

`blocks`	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.
`nitems`	The number of items that are included in the design.

Details

Each Thurstonian design can be defined by blocks of at least two items. The function determines the fundamental design matrix A of the Thurstonian design, including all possible paired comparisons that can be derived by the blocks. For further information of the importance of the design matrix, see Jansen and Schulze (2023a,2023b).

Value

Returns a design matrix which includes all paired comparisons derivable from the blocks.

References

Jansen, M. T., & Schulze, R. (2023a). *Linear factor analytic Thurstonian forced-choice models: Current status and issues*. Manuscript submitted.

Jansen, M. T., & Schulze, R. (2023b). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Examples

```
# Define a matrix of blocks
blocks <- matrix(1:15,ncol=3, byrow=TRUE)

# Get the design matrix
loading_Matrix <- designA(blocks)
```

FC *Paired comparisons of $N=12$ items (Thurstonian modelling)*

Description

This data set contains synthetic data of 1000 participants on all binary indicators of 12 items. For each paired comparison, participants had to rank the two alternative items according to their preference. It is assumed that transitivity holds (that is, the data comes from a ranking task). More details can be found in Brown and Maydeu-Olivares (2011), Jansen and Schulze (2023) and Maydeu-Olivares and Böckenholt (2005).

Usage

```
data(FC)
```

Format

A data frame with 1000 observations on 66 variables. For a variable ixiy, the result is the response preferences between item x and item y. It is coded a 1, if item x is preferred over item y, and 0 otherwise.

References

Brown, A, & Maydeu-Olivares, A. (2011). Item response modeling of forced-choice questionnaires. *Educational and Psychological Measurement*, *71*(3), 460-502. https://doi.org/10.1177/0013164410375112

Jansen, M. T., & Schulze, R. (2023). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Maydeu-Olivares, A., & Böckenholt, U. (2005). Structural equation modeling of paired-comparison and ranking data. *Psychological Methods*, *10*(3), 285-304. https://doi.org/10.1037/1082-989X.10.3.285.

fit.correct *Correct degree of freedom and fit indices in Thurstonian block models*

Description

Correct fit indices (RMSEA and CFI) by correcting the degrees of freedom after estimation a Thurstonian model.

Usage

```
fit.correct(n, blocks, chi2_mod, df_mod, chi2_base, df_base)
```

Arguments

n	The number of respondents.
blocks	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.
chi2_mod	The χ^2 value of the estimated model.
df_mod	The degrees of freedom of the estimated model.
chi2_base	The χ^2 value of the baseline model.
df_base	The degrees of freedom of the baseline model.

Details

If a ranking design is used (variances of binary indicators is zero) there are redundancies among the thresholds and tetrachoric correlations to be estimated. This is the case, whenever the number of items per block is larger than two. In these cases the degrees of freedom must be corrected by subtracting the redundancies. For more details see Jansen and Schulze (2023) and Maydeu-Olivares (1999).

Value

Returns a vector containing corrected degrees of freedom, and the corrected RMSEA and CFI values.

References

Jansen, M. T., & Schulze, R. (2023). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Maydeu-Olivares, A. (1999). Thurstonian modeling of ranking data via mean and covariance structure analysis. *Psychometrika*, *64*(3), 325-340. https://doi.org/10.1007/BF02294299

Examples

```
# Define 30 items divided by ten triplets as blocks
blocks <- matrix(c(1:30), ncol = 3)

# Assume the model yield the following fit, with 426 respondents
# chi2_mod = 224.456, df_mod = 59, chi2_base = 1056.566, df_base = 90
```

```
fit.correct(426,blocks,224.456,59,1056.566,90)

# The corrected values are rmsea = 0.0917892; cfi =  0.8184749
```

fit.lavaan *Performs lavaan estimation of the given model.*

Description

This function writes a lavaan syntax given the specifications of the Thurstonian forced choice model. Additionally it runs the code (given lavaan is installed) and returns the results.

Usage

```
fit.lavaan(blocks, itf, model, data = NULL, estimator = "ULSMV",
  rename_list = NULL)
```

Arguments

`blocks` A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

`itf` A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.

`model` A descriptor for the model. Can be one of ‘’lmean’‘, ‘’uc’‘, ‘’irt’‘ or ‘’simple2’‘, ‘’simple3’‘ or ‘’simple5’‘. The Number behind the ‘’simple’‘ statement defines the Thurstone case.

`data` A matrix or data frame including the binary indicators as columns and respondents as rows.

`estimator` Which estimator should be used? All estimators that are available in ‘lavaan‘ can be used. Defaults to ‘’ULSMV’‘.

`rename_list` A list with two vectors to rename the objects in the syntax. Vector one is the original names, vector two the new names. Defaults to ‘NULL‘.

Details

The syntax currently is able to perform model analysis for the latent utility model (‘’simple’‘ and ‘’lmean’‘; Maydeu-Olivares & Böckenholt, 2005) the unconstrained factor model (‘’uc’‘; Maydeu-Olivares & Böckenholt, 2005) and the IRT model(‘’irt’‘; Maydeu-Olivares & Brown, 2010). Additionally, all model types can be performed with all types of forced choice designs (full, block, partially linked block, linked block). For an overview and review see Jansen and Schulze (2023a,2023b).

The standard naming procedure ixiy, for the comparison of items x and y, can be changed by specifying the ‘rename_list‘ argument. The first vector of the schould be the vector of original names, for example ‘c(’i1i2’,’i1i3’,’i2i3’,’Trait1’,’Trait2’,’Trait3’)‘ the second vector should contain the new names, for example ‘c(’A01E12’,’A01C13’,’E01C23’,’Agree’,’Extra’,’Consc’)‘.

Value

Returns a lavaan object containing the specified results, after model analysis.

References

Maydeu-Olivares, A., & Brown, A. (2010). Item response modeling of paired comparison and ranking data. *Multivariate Behavioural Research*, *45*(6), 935-974. https://doi.org/10.1080/00273171.2010.531231

Jansen, M. T., & Schulze, R. (2023a). *Linear factor analytic Thurstonian forced-choice models: Current status and issues*. Manuscript submitted.

Jansen, M. T., & Schulze, R. (2023b). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Maydeu-Olivares, A., & Böckenholt, U. (2005). Structural equation modeling of paired-comparison and ranking data. *Psychological Methods*, *10*(3), 285-304. https://doi.org/10.1037/1082-989X.10.3.285

Examples

```
# read data set FC
data(FC)

# set seed and define blocks
set.seed(1)
blocks <- matrix(sample(1:12,12), ncol = 3)

# define the item-to-factor relation
itf <- rep(1:4,3)

# perform analysis
fit.lavaan(blocksort(blocks),itf,'irt',data=FC)
```

---

fit.mplus *Performs Mplus estimation of the given model.*

Description

This function writes the Mplus syntax given the specifications of a Thurstonian forced choice design. Additionally it runs the code (given Mplus is installed) and returns the results.

Usage

```
fit.mplus(blocks, itf, model, input_path = "myFC_model.inp",
  output_path = "myFC_model.out", data_path = "myDataFile.dat",
  fscore_path = "myFactorScores.dat", title = "myFC_model", ID = FALSE,
  byblock = TRUE, estimator = "ULSMV", data_full = F, standardized = T,
  rename_list = NULL, ...)
```

Arguments

`blocks` A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

`itf` A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.

`model` A descriptor for the model. Can be one of ‘’lmean’‘, ‘’uc’‘, ‘’irt’‘ or ‘’simple2’‘, ‘’simple3’‘ or ‘’simple5’‘. The Number behind the ‘’simple’‘ statement defines the Thurstone case.

`input_path` Path to save the Mplus input file. Defaults to ‘’myFC_model.inp’‘.

`output_path` Path to the Mplus output file. Defaults to ‘’myFC_model.out’‘.

`data_path` Path of the data file for Mplus. Defaults to ‘’myDataFile.dat’‘.

`fscore_path` Path to save the file of factor scores. Defaults to ‘’myFactorScores.dat’‘.

`title` Title of the Mplus model. Defaults to ‘’myFC_model’‘.

`ID` Logical. Should a ID variable be included? The ID must be the first variable in the data set. Defaults to ‘FALSE‘.

`byblock` Logical. Should the order in Mplus variable statement be the same as in the blocks. Defaults to ‘TRUE‘.

`estimator` Which estimator should be used? All Estimators that are available in Mplus can be used. Defaults to ‘’ULSMV’‘.

`data_full` Logical. Are the data considered to be from a full design? Defaults to ‘FALSE‘.

`standardized` Logical. Should standardized values be computed? Defaults to ‘TRUE‘.

`rename_list` A list with two vectors to rename the objects in the syntax. Vector one is the original names, vector two the new names. Defaults to ‘NULL‘.

`...` Further arguments passed to function ‘read.mplus‘.

Details

The syntax currently is able to perform model analysis for the latent utility models (‘’simple’‘ and ‘’lmean’‘; Maydeu-Olivares & Böckenholt, 2005) the unconstrained factor model (‘’uc’‘; Maydeu-Olivares & Böckenholt, 2005) and the IRT model(‘’irt’‘; Maydeu-Olivares & Brown, 2010). Additionally, all model types can be performed with all types of forced choice designs (full, block, partially linked block, linked block). For an overview and review see Jansen and Schulze (2023a,2023b).

The function writes and saves the Mplus input files, keeps the output files and reads the results specified for the function ‘read.mplus‘.

The standard naming procedure ixiy, for the comparison of items x and y, can be changed by specifying the ‘rename_list‘ argument. The first vector of the schould be the vector of original names, for example ‘c(’i1i2’,’i1i3’,’i2i3’,’Trait1’,’Trait2’,’Trait3’)‘ the second vector should contain the new names, for example ‘c(’A01E12’,’A01C13’,’E01C23’,’Agree’,’Extra’,’Consc’)‘.

Value

Returns a list containing the specified results, after model analysis.

References

Maydeu-Olivares, A., & Brown, A. (2010). Item response modeling of paired comparison and ranking data. *Multivariate Behavioural Research*, *45*(6), 935-974. https://doi.org/10.1080/00273171.2010.531231

Jansen, M. T., & Schulze, R. (2023a). *Linear factor analytic Thurstonian forced-choice models: Current status and issues*. Manuscript submitted.

Jansen, M. T., & Schulze, R. (2023b). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Maydeu-Olivares, A., & Böckenholt, U. (2005). Structural equation modeling of paired-comparison and ranking data. *Psychological Methods*, *10*(3), 285-304. https://doi.org/10.1037/1082-989X.10.3.285

Examples

```
# read and save data set FC
data(FC)
write.table(FC,'my_data.dat',quote=FALSE, sep=" ", col.names = FALSE, row.names = FALSE)

# set seed and define blocks
set.seed(1)
blocks <- matrix(sample(1:12,12), ncol = 3)

# define the item-to-factor relation
itf <- rep(1:4,3)

# perform analysis
fit.mplus(blocksort(blocks),itf,'irt',data_path = 'mydata.dat', data_full = TRUE,
input_path = 'myFC_model')
```

---

get.xblocks *Get extra blocks in a Thurstonian design, that links all blocks.*

Description

The function creates extra blocks for a Thurstonian design, that links all initial blocks with as few extra blocks as possible. The number of extra blocks is determined by ‘count.xblocks‘ (see Jansen & Schulze, 2023).

Usage

```
get.xblocks(blocks, itf, multidim, item_not = NULL, min = FALSE,
  show.warnings = FALSE)
```

Arguments

`blocks` A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

`itf` A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.

`multidim` Logical. Should the items within each linking block be forced to be multidimensional?

`item_not` The items that are differently keyed compared to the majority of items.

`min` Logical. Should a minimal number of blocks contain mixed keyed items?

`show.warnings` Logical. Should warnings be shown?

Value

The result is a matrix where the rows correspond to the specific extra blocks.

References

Jansen, M. T., & Schulze, R. (2023). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data.* Manuscript submitted.

Examples

```
# Define a matrix of blocks
blocks <- matrix(1:15,ncol=3, byrow=TRUE)

# define the item-to-factor relation
itf <- rep(1:3,5)

# Get the extra blocks for a completely linked design
get.xblocks(blocks, itf, FALSE)
```

`get.xblocks.any` *Get extra blocks in a Thurstonian design, that links as few blocks as possible.*

Description

The function creates extra blocks in a Thurstonian design, that links as few initial blocks as possible, with the number of blocks determined by ‘count.xblocks‘. This is only useful for comparisons between linked and partially linked block designs (Jansen & Schulze, 2023).

Usage

```
get.xblocks.any(blocks, itf, multidim)
```

Arguments

`blocks` A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

`itf` A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.

`multidim` Logical. Should the items within each linking block be forced to be multidimensional?

Details

The main strategy of the function is to create extra blocks that link as few blocks as possible, with the number of blocks determined by 'count.xblocks'. Therefore, first all combinations of additional blocks with the first two blocks are created. If more extra blocks are needed the function uses block three, four, etc..

Value

The result is a matrix where the rows correspond to the specific extra blocks.

References

Jansen, M. T., & Schulze, R. (2023). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data.* Manuscript submitted.

Examples

```
# Define a matrix of blocks
blocks <- matrix(1:15,ncol=3, byrow=TRUE)

# define the item-to-factor relation
itf <- rep(1:3,5)

# Get the extra blocks for a partially linked design
get.xblocks.any(blocks, itf, FALSE)
```

`i.name` *Creates names for paired comparisons of a given design.*

Description

This function creates names for paired comparisons in the ixiy scheme. If items 1 and 2 are compared, this corresponds to i1i2.

Usage

```
i.name(blocks)
```

Arguments

`blocks` A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

Value

Returns a character vector, containing names for all binary indicators of a design.

Examples

```
# Define 30 items divided by ten triplets as blocks
blocks <- matrix(c(1:30), ncol = 3)

i.name(blocks)
```

---

metablock *Find all general blocks*

Description

This function creates meta interlinked blocks within a Thurstonian design.

Usage

```
metablock(blocks)
```

Arguments

blocks	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

Details

This function creates meta interlinked blocks of a block design. These are blocks, that have at least one link from each of its items to any other of its items. If there is not such a link between every item, there are at least two meta blocks.

Value

Returns a list of items that form meta interlinked blocks.

Examples

```
# Define 30 items divided by ten triplets as blocks
blocks <- matrix(c(1:30), ncol = 3)

# Add one block to link the first three blocks.
blocks <- rbind(blocks,c(1,2,3))

# Find mata blocks
metablock(blocks)
```

`mod.matrices` *Create model matrices for Thurstonian modeling*

Description

This function creates and returns model matrices of Thurstonian model equations.

Usage

```
mod.matrices(blocks, itf, model)
```

Arguments

`blocks` A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

`itf` A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.

`model` A descriptor for the model. Can be one of ‘’lmean’‘, ‘’uc’‘, ‘’irt’‘ or ‘’simple2’‘, ‘’simple3’‘ or ‘’simple5’‘. The Number behind the ‘’simple’‘ statement defines the Thurstone case.

Value

Returns a list of elements containing model matrix information.

Examples

```
# set seed and define blocks
set.seed(1)
blocks <- matrix(sample(1:12,12), ncol = 3)

# define the item-to-factor relation
itf <- rep(1:4,3)

mod.matrices(blocks,itf,'irt')
```

`pair.combn` *Determine all paired comparisons*

Description

This function returns a matrix containing all paired comparisons defined by a design.

Usage

```
pair.combn(blocks, unique = TRUE)
```

Arguments

`blocks`	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.
`unique`	Logical. Should only unique paired comparisons be returned?

Value

Returns a matrix with all paired comparisons defined by a design.

Examples

```
#' # Define 30 items divided by three triplets as blocks
blocks <- matrix(c(1:30), ncol = 3)

# Get all blocks
pair.combn(blocks)
```

---

rankA *Determine the rank of the design matrix defined by the blocks.*

Description

This function determines the rank of the fundamental design matrix defined by the blocks.

Usage

```
rankA(blocks)
```

Arguments

`blocks`	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

Value

Returns the rank of the design matrix as an integer.

Examples

```
# Define nine items divided by three triplets as blocks
blocks <- matrix(c(1:9), ncol = 3)

# Determine the rank of the design matrix
rankA(blocks)
```

read.mplus *Reads results from Mplus output file.*

Description

This function reads and returns results from an Mplus output file.

Usage

```
read.mplus(blocks, itf, model, output_path = "myFC_model.out",
  convergence = T, fit.stat = T, loading = T, cor = T, intercept = T,
  threshold = T, resvar = T, standardized = F)
```

Arguments

blocks	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.
itf	A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.
model	A descriptor for the model. Can be one of ‘’lmean’‘, ‘’uc’‘, ‘’irt’‘ or ‘’simple2’‘, ‘’simple3’‘ or ‘’simple5’‘. The Number behind the ‘’simple’‘ statement defines the Thurstone case.
output_path	Path to the Mplus output file. Defaults to ‘’myFC_model.out’‘.
convergence	Logical. Should a message for convergence be returned? Defaults to ‘TRUE‘.
fit.stat	Logical. Should fit statistics be returned? Defaults to ‘TRUE‘.
loading	Logical. Should loading estimates be returned? Defaults to ‘TRUE‘.
cor	Logical. Should latent correlation estimates be returned? Defaults to ‘TRUE‘.
intercept	Logical. Should intercepts be returned? Does only work for ‘model = ’lmean’‘. Defaults to ‘TRUE‘.
threshold	Logical. Should thresholds be returned? Does only work for ‘model = ’uc’‘ or ‘’irt’‘. Defaults to ‘TRUE‘.
resvar	Logical. Should residual variances be returned? Defaults to ‘TRUE‘.
standardized	Logical. Should standardized values be returned? Defaults to ‘FALSE‘.

Value

Returns a list containing the specified results, after model analysis.

Examples

```
# read and save data set FC
data(FC)
write.table(FC,'my_data.dat',quote=FALSE, sep=" ", col.names = FALSE, row.names = FALSE)

# set seed and define blocks
set.seed(1)
```

```
blocks <- matrix(sample(1:12,12), ncol = 3)

# define the item-to-factor relation
itf <- rep(1:4,3)

# perform analysis
fit.mplus(blocksort(blocks),itf,'irt',data_path = 'mydata.dat', data_full = TRUE,
input_path = 'myFC_model')

# After estimation
read.mplus(blocks,itf,'irt',output_path = 'myFC_model.out')
```

`redundancies` *Determine the number of redundancies*

Description

This function determines the number of redundancies among the tetrachoric correlations and thresholds.

Usage

```
redundancies(blocks, warn = T)
```

Arguments

`blocks`	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.
`warn`	Logical. Should warning messages be returned? Defaults to ‘TRUE‘.

Details

If a ranking design is used (variances of binary indicators is zero) there are redundancies among the thresholds and tetrachoric correlations to be estimated. This is the case, whenever the number of items per block is larger than two. In these cases the degrees of freedom must be corrected by subtracting the redundancies. For more details see Jansen and Schulze (2023) and Maydeu-Olivares (1999).

Value

Returns an integer of the number of redundancies.

References

Jansen, M. T., & Schulze, R. (2023). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Maydeu-Olivares, A. (1999). Thurstonian modeling of ranking data via mean and covariance structure analysis. *Psychometrika*, *64*(3), 325-340. https://doi.org/10.1007/BF02294299

Examples

```
# Define 30 items divided by ten triplets as blocks
blocks <- matrix(c(1:30), ncol = 3)

# define the item-to-factor relation
itf <- rep(1:3,10)

# Determine the redundancies
redundancies(blocks)
```

`sim.data` *Create data based on Thurstonian model equations*

Description

Simulates a data set of paired comparisons or ranking data based a Thurstonian latent utility model.

Usage

```
sim.data(nfactor, nitem, nperson, itf, variables = NULL, varcov, lmu,
  loadings, transitive = T, var = 0, fvalues = F, sim = T)
```

Arguments

`nfactor` The number of factors.

`nitem` The number of items.

`nperson` The number of data points (= respondents) to simulate.

`itf` A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.

`variables` A vector containing the names of paired comparison variables to return. If ‘NULL‘ (default), all variables are returned.

`varcov` A matrix defining the variance-covariance matrix of the factors.

`lmu` A vector defining the latent means of items.

`loadings` A vector defining the loadings of items.

`transitive` Logical. Should the data be transitive? If ‘TRUE‘, ranking data is simulated, else paired comparison data is simulated. Defaults to ‘TRUE‘.

`var` A vector containing the variances for each paired comparison. Defaults to 0.

`fvalues` Logical. Should simulated factor values be returned? Defaults to ‘FALSE‘.

`sim` Logical. Should the simulated data be returned? Defaults to ‘TRUE‘.

Value

Returns a list containing the true factor scores and the data, or a matrix containing the data.

Examples

```
nfactor <- 4
nitem <- 12
nperson <- 1000
itf <- rep(1:4,3)
varcov <- diag(1,4)

# latent utility means
set.seed(69)
lmu <- runif(nitem, -1, 1)
loadings <- runif(nitem, 0.30, 0.95)

FC <- sim.data(nfactor=nfactor, nitem=nitem, nperson=nperson, itf=itf,
varcov = varcov, lmu = lmu, loadings=loadings)
```

---

syntax.lavaan *Create lavaan syntax for Thurstonian forced choice analysis*

Description

This function writes a lavaan syntax given the specifications of the Thurstonian forced choice model.

Usage

```
syntax.lavaan(blocks, itf, model, rename_list = NULL)
```

Arguments

`blocks`	A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.
`itf`	A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.
`model`	A descriptor for the model. Can be one of ‘’lmean’‘, ‘’uc’‘, ‘’irt’‘ or ‘’simple2’‘, ‘’simple3’‘ or ‘’simple5’‘. The Number behind the ‘’simple’‘ statement defines the Thurstone case.
`rename_list`	A list with two vectors to rename the objects in the syntax. Vector one is the original names, vector two the new names. Defaults to ‘NULL‘.

Value

Returns a description of the user-specified model. Typically, the model is described using the lavaan model syntax. See ‘lavaan::model.syntax‘ for more information.

#’ @details The syntax currently is able to perform model analysis for the latent utility models (‘’simple’‘ and ‘’lmean’‘; Maydeu-Olivares & Böckenholt, 2005) the unconstrained factor model (‘’uc’‘; Maydeu-Olivares & Böckenholt, 2005) and the IRT model(‘’irt’‘; Maydeu-Olivares & Brown, 2010). Additionally, all model types can be performed with all types of forced choice designs (full, block, partially linked block, linked block). For an overview and review see Jansen and Schulze (2023a,2023b).

The standard naming procedure ixiy, for the comparison of items x and y, can be changed by specifying the ‘rename_list‘ argument. The first vector of the schould be the vector of original names, for example ‘c(’i1i2’,’i1i3’,’i2i3’,’Trait1’,’Trait2’,’Trait3’)‘ the second vector should contain the new names, for example ‘c(’A01E12’,’A01C13’,’E01C23’,’Agree’,’Extra’,’Consc’)‘.

References

Maydeu-Olivares, A., & Brown, A. (2010). Item response modeling of paired comparison and ranking data. *Multivariate Behavioural Research*, *45*(6), 935-974. https://doi.org/10.1080/00273171.2010.531231

Jansen, M. T., & Schulze, R. (2023a). *Linear factor analytic Thurstonian forced-choice models: Current status and issues*. Manuscript submitted.

Jansen, M. T., & Schulze, R. (2023b). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Maydeu-Olivares, A., & Böckenholt, U. (2005). Structural equation modeling of paired-comparison and ranking data. *Psychological Methods*, *10*(3), 285-304. https://doi.org/10.1037/1082-989X.10.3.285

Examples

```
# read data set FC
data(FC)

# set seed and define blocks
set.seed(1)
blocks <- matrix(sample(1:12,12), ncol = 3)

# define the item-to-factor relation
itf <- rep(1:4,3)

# Create lavaan model syntax
syntax.lavaan(blocks,itf,'irt')
```

`syntax.mplus` *Create Mplus syntax for Thurstonian forced choice designs.*

Description

This function writes and saves the Mplus syntax given the specifications of a Thurstonian forced choice design.

Usage

```
syntax.mplus(blocks, itf, model, input_path = "myFC_model.inp",
  data_path = "myDataFile.dat", fscore_path = "myFactorScores.dat",
  title = "myFC_model", ID = FALSE, byblock = TRUE,
  estimator = "ULSMV", data_full = F, standardized = T,
  rename_list = NULL)
```

Arguments

`blocks` A matrix defining the blocks of the model. The number of rows must be the number of blocks, each row represents a block and contains the item numbers. The number of columns present the number of items per block.

`itf` A vector defining the items-to-factor relation. For example ‘c(1,1,1,2,2,2)‘ defines six items, the first three correspond to factor 1, the second three correspond to factor 2.

`model` A descriptor for the model. Can be one of ‘’lmean’‘, ‘’uc’‘, ‘’irt’‘ or ‘’simple2’‘, ‘’simple3’‘ or ‘’simple5’‘. The Number behind the ‘’simple’‘ statement defines the Thurstone case.

`input_path` Path to save the Mplus input file. Defaults to ‘’myFC_model.inp’‘.

`data_path` Path of the data file for Mplus. Defaults to ‘’myDataFile.dat’‘.

`fscore_path` Path to save the file of factor scores. Defaults to ‘’myFactorScores.dat’‘.

`title` Title of the Mplus model. Defaults to ‘’myFC_model’‘.

`ID` Logical. Should a ID variable be included? The ID must be the first variable in the data set. Defaults to ‘FALSE‘.

`byblock` Logical. Should the order in Mplus variable statement be the same as in the blocks. Defaults to ‘TRUE‘.

`estimator` Which estimator should be used? All Estimators that are available in Mplus can be used. Defaults to ‘’ULSMV’‘.

`data_full` Logical. Are the data considered to be from a full design? Defaults to ‘FALSE‘.

`standardized` Logical. Should standardized values be computed? Defaults to ‘TRUE‘.

`rename_list` A list with two vectors to rename the objects in the syntax. Vector one is the original names, vector two the new names. Defaults to ‘NULL‘.

Details

The syntax currently is able to perform model analysis for the latent utility models (‘’simple’‘ and ‘’lmean’‘; Maydeu-Olivares & Böckenholt, 2005) the unconstrained factor model (‘’uc’‘; Maydeu-Olivares & Böckenholt, 2005) and the IRT model(‘’irt’‘; Maydeu-Olivares & Brown, 2010). Additionally, all model types can be performed with all types of forced choice designs (full, block, partially linked block, linked block). For an overview and review see Jansen and Schulze (2023a,2023b).

The function writes and saves the Mplus input files.

The standard naming procedure ixiy, for the comparison of items x and y, can be changed by specifying the ‘rename_list‘ argument. The first vector of the schould be the vector of original names, for example ‘c(’i1i2’,’i1i3’,’i2i3’,’Trait1’,’Trait2’,’Trait3’)‘ the second vector should contain the new names, for example ‘c(’A01E12’,’A01C13’,’E01C23’,’Agree’,’Extra’,’Consc’)‘.

References

Maydeu-Olivares, A., & Brown, A. (2010). Item response modeling of paired comparison and ranking data. *Multivariate Behavioural Research*, *45*(6), 935-974. https://doi.org/10.1080/00273171.2010.531231

Jansen, M. T., & Schulze, R. (2023a). *Linear factor analytic Thurstonian forced-choice models: Current status and issues*. Manuscript submitted.

Jansen, M. T., & Schulze, R. (2023b). *The Thurstonian linked block design: Improving Thurstonian modeling for paired comparison and ranking data*. Manuscript submitted.

Maydeu-Olivares, A., & Böckenholt, U. (2005). Structural equation modeling of paired-comparison and ranking data. *Psychological Methods*, *10*(3), 285-304. https://doi.org/10.1037/1082-989X.10.3.285

Examples

```
# read and save data set FC
data(FC)
write.table(FC,'my_data.dat',quote=FALSE, sep=" ", col.names = FALSE, row.names = FALSE)

# set seed and define blocks
set.seed(1)
blocks <- matrix(sample(1:12,12), ncol = 3)

# define the item-to-factor relation
itf <- rep(1:4,3)

# Create and save Mplus syntax
syntax.mplus(blocks,itf,'lmean',data_path = 'my_data.dat', data_full = TRUE,
input_path = 'myFC_model')
```

Index