

Package ‘random.polychor.pa’

July 26, 2016

Title A Parallel Analysis with Polychoric Correlation Matrices

Version 1.1.4-2

Date 2016-07-18

Description The Function performs a parallel analysis using simulated polychoric correlation matrices. The nth-percentile of the eigenvalues distribution obtained from both the randomly generated and the real data polychoric correlation matrices is returned. A plot comparing the two types of eigenvalues (real and simulated) will help determine the number of real eigenvalues that outperform random data. The function is based on the idea that if real data are non-normal and the polychoric correlation matrix is needed to perform a Factor Analysis, then the Parallel Analysis method used to choose a non-random number of factors should also be based on randomly generated polychoric correlation matrices and not on Pearson correlation matrices. Random data sets are simulated assuming or a uniform or a multinomial distribution or via the bootstrap method of resampling (i.e., random permutations of cases). Also Multigroup Parallel analysis is made available for random (uniform and multinomial distribution and with or without difficulty factor) and bootstrap methods. An option to choose between default or full output is also available as well as a parameter to print Fit Statistics (Chi-squared, TLI, RMSEA, RMR and BIC) for the factor solutions indicated by the Parallel Analysis.

Depends psych, nFactors, boot

Imports MASS, mvtnorm, sfsmisc

License GPL (>= 2)

LazyLoad yes

LazyData yes

NeedsCompilation no

Encoding UTF-8

Author Fabio Presaghi [aut, cre],
Marta Desimoni [ctb]

Maintainer Fabio Presaghi <fabio.presaghi@uniroma1.it>

Repository CRAN

Date/Publication 2016-07-26 12:43:04

R topics documented:

NEWS	2
Parallel-Analysis-of-Polychoric-Correlations	4
random.polychor.pa	6

Index	14
--------------	-----------

NEWS	<i>News for Package</i> random.polychor.pa
------	---

Description

The function performs a parallel analysis using simulated polychoric correlation matrices. The function will extract the eigenvalues from each random generated polychoric correlation matrix and from the polychoric correlation matrix of real data. A plot comparing eigenvalues extracted from the specified real data with simulated data will help determine which of real eigenvalue outperform random data. A series of matrices comparing MAP vs PA-Polychoric vs PA-Pearson correlations methods, FA vs PCA solutions are finally presented. Random data sets are simulated assuming or a uniform or a multinomial distribution or via the bootstrap method of resampling (i.e., random permutations of cases). Also Multigroup Parallel analysis is made available for random (uniform and multinomial distribution and with or without difficulty factor) and bootstrap methods. An option to choose between default or full output is also available as well as a parameter to print Fit Statistics (Chi-squared, TLI, RMSEA, RMR and BIC) for the factor solutions indicated by the Parallel Analysis.

Changes in version 1.1.4-2

Version 1.1.4-2 fixed minor bugs when running the example 1, and when displaying the time needed to complete the simulations.

Changes in version 1.1.4-1

Version 1.1.4-1 a problem with the psych dependency was fixed:

the option (`polycor=TRUE`) in the `polychoric` function was removed and consequently it is also no more possible to call the `polycor` function in running the `random.polychor.pa` function

Changes in version 1.1.4

- Version 1.1.4 added a number of cahnges:
 - a parameter `distr` allows to shift the simulation from uniform distribtuion to multinomial distribution
- Bootstrap method (with random permutations of cases) was added
- Multi-Group for random and bootstrap version of the Parallel Analysis is made available
- Fit statistics (Chi-squared, TLI, RMSEA, RMR, BIC) for all factor solution indicated by Parallel Analysis

- option to print a default output (number of factors indicated by Parallel Analysis) or a full output (adds: matrices of simulated and empirical eigenvalues for random, bootstrap, and multigroup)

Changes in version 1.1.3.6

- In version 1.1.3.6 a check for the range of quantile (between 0 and 1) was added.

Changes in version 1.1.3.5

- The search for zeroes within the provided datafile was removed, so data with zeroes are now accepted.

Changes in version 1.1.3.5

- In version 1.1.3.5 a parameter was added, `diff.fact`, in order to simulate random dataset with the same probability of observing each category for each variable as that observed in the provided (empirical) dataset.

Changes in version 1.1.3

Version 1.1.3 tackles two problems signalled by users:

the possibility to make available the results of simulation for plotting them in other software. Now the `random.polychor.pa()` will show, upon request, all the data used in the scree-plot.

- The function `polichoric()` of the `psych()` package does not handle data matrices that include 0 as possible category and will cause the function to stop with error. So a check for the detection of the 0 code within the provided `data.matrix` is now added and will cause the `random.polychor.pa` function to stop with a warning message.

Changes in version 1.1.2

- Version 1.1.2 simply has updated the function that calculates the polychoric correlation matrix due to changes in the `psych()` package.

Changes in version 1.1.1

- Version 1.1.1, fixed a minor bug in the regarding the estimated time needed to complete the simulation.
- Also in this version, the function is now able to manage supplied `data.matrix` in which variables representing factors (i.e., variables with ordered categories) are present and may cause an error when the Pearson correlation matrix is calculated.

Parallel-Analysis-of-Polychoric-Correlations*A Parallel Analysis with Random Polychoric Correlation Matrices*

Description

The function performs a parallel analysis using simulated polychoric correlation matrices. The function will extract the eigenvalues from each random generated polychoric correlation matrix and from the polychoric correlation matrix of real data. A plot comparing eigenvalues extracted from the specified real data with simulated data will help determine which of real eigenvalue outperform random data. A series of matrices comparing MAP vs PA-Polychoric vs PA-Pearson correlations methods, FA vs PCA solutions are finally presented.

Details

Package: random.polychor.pa
Type: Package
Version: 1.1.4-2
Date: 2016-07-18
License: GPL Version 2 or later
LazyLoad: yes

The function perform a parallel analysis (Horn, 1965) using randomly simulated polychoric correlations and generates `nrep` random samples of the same dimension of the empirical provided data.matrix(i.e, with the same number of participants and of variables). Items are allowed to have varying number of categories. The function will extract the eigenvalues from each randomly generated polychoric matrices and the declared quantile will be extracted. Eigenvalues from polychoric correlation matrix obtained from real data are also computed and compared, in a (scree) plot, with the eigenvalues extracted from the simulation (Polychoric matrices). Recently Cho, Li & Bandalos (2009), showed that in using PA method, it is important to match the type of the correlation matrix used to recover the eigenvalues from real data with the type of correlation matrix used to estimate random eigenvalues. Crossing the type of correlations (using Polychoric correlation matrix to estimate real eigenvalues and random simulated Pearson correlation matrices) may result in a wrong decision (i.e., retaining less non-random factors than the needed). A comparison with eigenvalues extracted from both randomly simulated Pearson correlation matrices and real data is also included. Finally, for both type of correlation matrix (Polychoric vs Pearson), the two versions (the classic squared coefficient and the 4th power coefficient) of Velicer's MAP criterion are calculated. In version 1.1.1, a minor bug in the regarding the estimated time needed to complete the simulation is fixed. Also in this version, the function is now able to manage supplied data.matrix in which variables representing factors (i.e., variables with ordered categories) are present and may cause an error when the Pearson correlation matrix is calculated. As the `poly.mat()` function used to calculate the polychoric correlation matrix is going to be deprecated in favor of `poly.mat(polychoric())` function, the `random.polychor.pa` was consequently updated (version 1.1.2) to account for changes in `psych()` package. Version 1.1.3 tackles two problems signaled by users: 1) the possibility to make available the results of simulation for plotting them in other software. Now the `random.polychor.pa`

will show, upon request, all the data used in the scree-plot. 2) The function `polichoric()` of the `psych()` package does not handle data matrices that include 0 as possible category and will cause the function to stop with error. So a check for the detection of the 0 code within the provided data.matrix is now added and will cause the `random.polychor.pa` function to stop with a warning message. In version 1.1.3.5 a parameter was added, `diff.fact` in order to simulate random data set with the same probability of observing each category for each variable as that observed in the provided (empirical) data set. This parameter allows to simulate data sets that have the same item difficulties distribution as well as the same difficulty factors of the real data (Ledesma and Valero-Mora, 2007). Finally the search for zeroes within the provided data file was removed, so data with zeroes are now accepted. In version 1.1.3.6 a check for the range of quantile (between 0 and 1) was added. In version 1.1.4 was added the possibility to choose between uniform and multinomial distribution, between random and bootstrap distribution and between single or multiple sample parallel analysis. Also two switches were added: one allows to print or a default output (including only the number of factors to retain following Parallel Analysis and the scree-plot) or a full output (including the eigenvalues distributions of simulated and empirical data sets); the other switch allows to print a further Fit Statistics for the factor solutions indicated by the Parallel Analysis.

The default value of the `comparison` is set to `random` and this means that random uniform samples (`runif()`) will be computed. Two types of distribution are available: the uniform distribution and the multinomial distribution (Liu and Rijmen, 2008). To switch between them just set accordingly the parameter `distr`. As method of resampling is now also available the bootstrap method that is based on random permutations of cases obtained via the `boot()` function.

Multigroup version of random samples may be obtained by setting the parameter `comparison` to `random-mg`. To perform multigroup Parallel Analysis simulating random samples with multinomial distributions the parameter `distr` should be set to `multinomial`. Multigroup Parallel Analysis is also available for the bootstrap method by setting the `comparison` parameter to `bootstrap-mg`. When Multigroup Parallel Analysis is used, the first variable (column) of the dataset should be reserved to the factor used to identify the different samples (i.e., gender, age groups, nationalities, etc.).

Fit statistics (Chi-squared, TLI, RMSEA, RMR, BIC) for all factor solutions indicated by Parallel Analysis are available when the parameter `fit.pa` is set to `TRUE`. Consider that in estimating the these Fit Statistic indexes not allways converge and this may break the computations for the parallel analysis. In this case switch the parameter off. Moreover the values of these indexes are not allways within the expected range.

The `print.all` parameter when set to `TRUE` will show the table of eigenvalue distributions for simulated (random or bootstrap) and empirical data. For bootstrap simulation, also the bias and standard error will be printed. If the `diff.fact` is set to `TRUE` when random simulations are requested, then also the weighting matrix used to reproduce the item frequencies in random data set will be printed.

Author(s)

Fabio Presaghi <fabio.presaghi@uniroma1.it> and Marta Desimoni <marta.desimoni@uniroma1.it>

References

- Cho, S.J., Li, F., & Bandalos, D., (2009). Accuracy of the Parallel Analysis Procedure With Polychoric Correlations. *Educational and Psychological Measurement*, 69, 748-759.
- Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 32, 179-185.

Ledesma RD, Valero-Mora P (2007) Determining the number of factors to retain in EFA: an easy to use computer program for carrying out parallel analysis. *Practical Assessment, Research & Evaluation* 12:1–11

O'Connor, B. P. (2000). SPSS and SAS programs for determining the number of components using parallel analysis and Velicer's MAP test. *Behavior Research Methods, Instrumentation, and Computers*, 32, 396-402.

Reckase, M.D. (2009). *Multidimensional Item Response Theory*. Springer.

Velicer, W. F. (1976). Determining the number of factors from the matrix of partial correlations. *Psychometrika*, 41, 321-327.

Velicer, W. F., Eaton, C. A., & Fava, J. L. (2000). Construct explication through factor or component analysis: A review and evaluation of alternative procedures for determining the number of factors or components. In R. D. Goffin & E. Helmes (Eds.), *Problems and solutions in human assessment: Honoring Douglas N. Jackson at seventy* (pp. 41-72). Norwell, MA: Kluwer Academic.

random.polychor.pa *A Parallel Analysis with Randomly Generated Polychoric Correlation Matrices*

Description

The function performs a parallel analysis using simulated polychoric correlation matrices. The eigenvalues (extracted following both FA and PCA methods) from each random generated polychoric correlation matrix and from the polychoric correlation matrix of real solutions from Polychoric vs Pearson correlations, FA vs PCA and PA vs MAP are presented. Random data sets are simulated assuming or a uniform or a multinomial distribution or via the bootstrap method of resampling (i.e., random permutations of cases). Also Multigroup Parallel analysis is made available for random (uniform and multinomial distribution and with or without difficulty factor) and bootstrap methods. An option to choose between default or full output is also available as well as a parameter to print Fit Statistics (Chi-squared, TLI, RMSEA, RMR and BIC) for the factor solutions indicated by the Parallel Analysis.

Usage

```
random.polychor.pa(nvar="NULL", n.ss="NULL", nrep, nstep="NULL",
                  data.matrix, q.eigen, r.seed = "NULL", diff.fact=FALSE, distr="NULL",
                  comparison = "random", fit.pa=FALSE, print.all=FALSE)
```

Arguments

nvar	Number of variables (items) in the raw data matrix. From version 1.1 of the function, it is no more needed to specify nvar as this information is derived from the number of columns of the data.matrix. Default value is set to "NULL" for compatibility with past version of the function
n.ss	Number of participants of the raw data matrix. From version 1.1 of the function, it is no more needed to specify n.ss as this information is derived from the number of rows of the data.matrix. Default value is set to "NULL" for compatibility with past version of the function.

nrep	Number of random samples that should be simulated
nstep	Number of ordered categories of the item (e.g., Likert-like 3 ordered category item). This information is no longer needed as the new version of the function (1.1) allows also for items with varying number of categories. The number of categories from each item is derived directly from the data.matrix. A table summarizing the different groups of item with different number of categories will be showed. Default value is set to "NULL" for compatibility with past version of the function.
data.matrix	the name of raw data matrix. The raw data.matrix should be numeric and none of the ordered category should be coded as 0 (zero). No automatic recode routine is provided within the function to deal with alphanumeric content of the ordered categories of manifest variables. So the user performs all these recodings before running the function.
q.eigen	a number comprised within the interval of 0 and 1 and indicating the quantile that is used to choose the number of non-random factors (e.g., .50 or .95 or .99)
r.seed	eventually, a preferred number that will be used to initialize the random generator. Default value: 1335031435.
diff.fact	default value is FALSE and in this case the function will estimate random datasets without trying to reproduce each observed category with the same probability as that observed in the empirical dataset provided. If the parameter is set on TRUE, the function will simulate random samples with the same proportion of each category for each item as that of empirical dataset. This parameter allows to simulate data sets that have the same item difficulties distribution as well as the same difficulty factors of the real data (Ledesma and Valero-Mora, 2007)
distr	this parameter allows to choose between uniform (that is also the default) or multinomial distribution for the simulation of random data sets.
comparison	four methods are now available: random, bootstrap, multigroup-random, multigroup-bootstrap. Allowed values are: random for random simulated dataset; bootstrap for bootstrap simulations with permutation as method of resampling; random-mg for random multi-group simulated datasets; bootstrap-mg for bootstrap multi-group simulated datasets. When bootstrap method is chosen, the bootstrapped eigenvalue distribution is compared with that obtained from empirical data and the PA algorithm will indicate the number of empirical eigenvalues that are greater than the corresponding bootstrapped eigenvalues.
fit.pa	when set to TRUE, the fit statistics (Chi-squared, TLI, RMSEA, RMR, BIC) for all factor solutions indicated by Parallel Analysis will be printed. This parameter allows a call to the fa() function of the psych package, and passes just the number of factor to retain with all remaining values set to the default (i.e., fm="minres", rotate="oblimin", etc.)
print.all	when set to TRUE, all the simulated (random or bootstrap) and empirical eigenvalue distributions will be printed for each samples. When comparison is equal to bootstrap or bootstrap-mg also the bootstrap bias and standard error estimates will be printed.

Details

The function perform a parallel analysis (Horn, 1976) using randomly simulated polychoric correlations and generates `nrep` random samples of simulated data with the same number of participants and of variables of the provided `data.matrix`. The function will read the entered `data.matrix` and extracts: the number of units (i.e., number of rows); the number of variables (i.e., number of columns); and the number of categories of each item. From version 1.1, the function accepts also variables with varying number of categories (e.g., three items with only two categories and two items with three categories, etc.). In version 1.1.1, the function is also able to manage supplied `data.matrix` in which variables represent factors (i.e., variables with ordered categories) may cause an error when the Pearson correlation matrix is calculated. The information in the supplied `data.matrix` are used to generate the `nrep` random raw datasets with the same characteristics of the original real data set. So only three information are needed for the problem to run: the number of replications (`nrep`), the data matrix (`data.matrix`) and the percentile to be used (`q.eigen`).

A check for missing values within the real dataset is present and if present will be treated LISTWISE. In this case a warning message will prompt the user signalling how NA were treated (LISTWISE is by now the only treatment considered) and the new sample size. No further checks are made on the raw data, so out-of-range values are not detected and it is on the behalf of the user to make a preliminary check on the reliability of data. A table summarizing the groups of items with different number of categories will be shown along with the main results of the PA.

The function will extract the eigenvalues from each randomly generated polychoric matrices and the requested percentile is returned. Eigenvalues from polychoric correlation matrix obtained from real data is also computed and compared with the eigenvalues extracted from the simulation (Polychoric matrices) in a (scree) plot. Results from classical Factor Analysis and Principal Component Analysis are made available. Separated random data sets are simulated for the two analysis.

Recently, Cho, Li & Bandalos (2009) showed that, in using PA method, it is important to match the type of the correlation matrix used to recover the eigenvalues from real data with the type of correlation matrix used to estimate random eigenvalues. Crossing the type of correlations (using Polychoric correlation matrix to estimate real eigenvalues and random simulated Pearson correlation matrices) may result in a wrong decision (i.e., retaining more non-random factors than the needed). A comparison with eigenvalues extracted from both randomly simulated Pearson correlation matrices and real data is also included.

Finally, for both type of correlation matrix (Polychoric vs Pearson), the two versions (the classic squared coefficient and the 4th power coefficient) of Velicer's MAP criterion are calculated (Velicer, 1976; Velicer, Eaton, & Fava, 2000) by implementing under R the code released by O'Connor (2000) for SPSS, SAS and MATLAB.

As the `poly.mat()` function used to calculate the polychoric correlation matrix is going to be deprecated in favour of `polychoric()` function, the `random.polychor.pa` was consequently updated (version 1.1.2) to account for changes in `psych()` package.

Version 1.1.3 tackles two problems signalled by users: 1) the possibility to make available the results of simulation for plotting them in other softwares. Now the `random.polychor.pa` will show, upon request, all the data used in the scree-plot. 2) The function `polichoric()` of the `psych()` package does not handle data matrices that include 0 as possible category and will cause the function to stop with error. So a check for the detection of the 0 code within the provided `data.matrix` is now added and will cause the `random.polychor.pa` function to stop with a warning message.

In version 1.1.3.5 a parameter was added, `diff.fact`, in order to simulate random dataset with the same probability of observing each category for each variable as that observed in the provided

(empirical) dataset. This parameter was added for those researchers that want to replicate random datasets with the same distribution of item difficulties as the real data (Ledesma, Valero-Mora, 2007; Reckase, 2009, pp.216). Finally the search for zeroes within the provided datafile was removed, so data with zeroes are now accepted.

In version 1.1.3.6 a check for the range of quantile (between 0 and 1) was added.

In version 1.1.4 was added the possibility to choose between uniform and multinomial distribution, between random and bootstrap distribution and between single or multiple sample parallel analysis. Also two switches were added: one allows to print or a default output (including only the number of factors to retain following Parallel Analysis and the scree-plot) or a full output (including the eigenvalues distributions of simulated and empirical data sets); the other switch allows to print a further Fit Statistics for the factor solutions indicated by the Parallel Analysis.

The default value of the `comparison` is set to `random` and this means that random uniform samples (`runif()`) will be computed. Two types of distribution are available: the uniform distribution and the multinomial distribution (Liu and Rijmen, 2008). To switch between them just set accordingly the parameter `distr`. As method of resampling is now also available the bootstrap method that is based on random permutations of cases obtained via the `boot()` function.

Multigroup version of random uniform samples maybe obtained by setting the parameter `comparison` to `random-mg`. To perform multigroup Parallel Analysis simulating random samples with multinomial distributions the parameter `distr` should be set to `multinomial`. Multigroup Parallel Analysis is also available for the bootstrap method by setting the `comparison` parameter to `bootstrap-mg`. When Multigroup Parallel Analysis is used, the first variable (column) of the dataset should be reserved to the factor used to identify the different samples (i.e., gender, age groups, nationalities, etc.).

Fit statistics (Chi-squared, TLI, RMSEA, RMR, BIC) for all factor solutions indicated by Parallel Analysis are available when the parameter `fit.pa` is set to `TRUE`. Consider that in estimating these Fit Statistic indexes not always converge and this may break the computations for the parallel analysis. In this case switch the parameter off. Moreover the values of these indexes are not always within the expected range.

The `print.all` parameter when set to `TRUE` will show the tables of eigenvalue distributions for simulated (random or bootstrap) and empirical data. For bootstrap simulation, also the bias and standard error will be printed. If the `diff.fact` is set to `TRUE` when random simulations are requested, then also the weighting matrix used to reproduce the item frequencies in random data set will be printed.

Value

The default output (`print.all=FALSE`) prints the number of factors for Polychoric and Pearson Correlation PA methods for Factor Analysis and Principal Components Analysis (PCA) methods along with the number of factors chosen by the two Velicer's MAP criteria (original and 4th power) for both Polychoric and Pearson correlation matrices. Furthermore, the function will return the (scree) plot of the eigenvalues for real (Polychoric vs Pearson correlation matrices) and simulated data (Polychoric vs Pearson correlation matrices). If the parameter `print.all` is set to `TRUE` then other than the default output, it will also be printed the tables of eigenvalue distributions for simulated (random (uniform or multinomial) or bootstrap) and empirical data. For bootstrap simulation, also the bias and standard error will be printed. If the `diff.fact` is set to `TRUE` when random simulations are requested, then also the weighting matrix used to reproduce the item frequencies in random data set will be printed.

Note

In running the random.polychor.pa function it should be reminded that it may take a lot of time to complete the simulation. This is due in part to the fact that the estimation of the polychoric correlation matrix is cumbersome and in part to the fact that the code is not optimized, but simply it does the work.

Occasionally, in calculating the polychoric correlation matrix it may occur an error when the matrix is non-positive definite. In this case you have to re-run the simulation.

A note should be made concerning the method used (from version 1.1) to read the raw data.matrix supplied by the user and used to retrieve the three basic information needed to build the random matrices (number of rows, number of columns and the number of categories for each manifest variable). The number of categories for each variable is derived from the raw data.matrix, so if the possible number of categories for a specific item is for example 5, but subjects endorse only three out of the five categories then the random.polychor.pa function will simulate a variable with only three categories. This means that the function guarantees that the empirical and the simulated data matrix are similar, but this also means that by changing the sample of participants the simulated data will change (even if slightly).

In computing the polychoric correlation matrix for both simulated and empirical data sets, the parameter global of the polychoric() function is set to FALSE and the text message "The items do not have an equal number of response alternatives, global set to FALSE" is suppressed with the suppressMessages() function.

Author(s)

Fabio Presaghi <fabio.presaghi@uniroma1.it> and Marta Desimoni <marta.desimoni@uniroma1.it>

References

- Cho, S.J., Li, F., & Bandalos, D., (2009). Accuracy of the Parallel Analysis Procedure With Polychoric Correlations. *Educational and Psychological Measurement*, 69, 748-759.
- Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 32, 179-185.
- Ledesma RD, Valero-Mora P (2007) Determining the number of factors to retain in EFA: an easy to use computer program for carrying out parallel analysis. *Practical Assessment, Research & Evaluation* 12:1-11
- Liu, O.L., and Rijmen, F., (2008) A modified procedure for parallel analysis of ordered categorical data. *Behavior Research Methods*, 40 (2), 556-562. doi: 10.3758/BRM.40.2.556
- O'Connor, B. P. (2000). SPSS and SAS programs for determining the number of components using parallel analysis and Velicer's MAP test. *Behavior Research Methods, Instrumentation, and Computers*, 32, 396-402.
- Reckase, M.D. (2009). *Multidimensional Item Response Theory*. Springer.
- Velicer, W. F. (1976). Determining the number of factors from the matrix of partial correlations. *Psychometrika*, 41, 321-327.
- Velicer, W. F., Eaton, C. A., & Fava, J. L. (2000). Construct explication through factor or component analysis: A review and evaluation of alternative procedures for determining the number of factors or components. In R. D. Goffin & E. Helmes (Eds.), *Problems and solutions in human assessment: Honoring Douglas N. Jackson at seventy* (pp. 41-72). Norwell, MA: Kluwer Academic.

See Also

nFactors, psych, paran

Examples

```

### EXAMPLE 1:
### this example is particularly instructive on how the solution may
### change by changing the type of correlation, method of extraction and
### method of selection.
require(psych)
data(bfi)
names(bfi)
bfi.data<-na.exclude(as.matrix(bfi[1:200, 1:5]))
head(bfi.data)
random.polychor.pa(nrep=3, data.matrix=bfi.data, q.eigen=.99)

### EXAMPLE 2:
### in this example one of the categories of item1 is recoded: 2=1
### so this item has 5 categories: 1 (2) 3 4 5 6
### category 1 is within brackets as it has frequency=0
### so this is a case where empirical data (0 2 3 4 5 6) diverge from
### theoretical data (0 1 2 3 4 5 6)
#require(psych)
#data(bfi)
#raw.data.1<-as.matrix(bfi)
#raw.data.1 <- (raw.data.1[1:200,1:25])
#for(i in 1:nrow(raw.data.1)) { if(raw.data.1[i,1]==2) raw.data.1[i,1]<-1}
#test.2<-random.polychor.pa(nrep=100, data.matrix=raw.data.1, q.eigen=.99)
#test.2

### EXAMPLE 3:
##### for SPSS users #####
### the following instructions can used to load a SPSS data file (.sav).
### 1) load the library to read external datafile (e.g., SPSS datafile)
### 2) choose the SPSS datafile by pointing directly in the folder
#   on your hard-disk
### 3) select only the variables (i.e., the items) needed to for
#   Parallel Analysis
#> library(foreign) ### load the needed library
#> raw.data <- read.spss(choose.files(), use.value.labels=TRUE,
#   max.value.labels=Inf, to.data.frame=TRUE)
#> raw.spss.item <- na.exclude(raw.data[,2:4])
#> summary (raw.spss.item)
#> random.polychor.pa(nrep=5, data.matrix=raw.spss.item, q.eigen=.99)

### EXAMPLE 4a:
### in this case the parameter diff.fact is set to TRUE, so the function
### will simulate random dataset with the same probability of occurrence
### of each category for each item in the observed dataset.
### Dichotomous variables are used in this example.
#require(psych)
#data(bock)

```

```

### DICHTOMOUS
#random.polychor.pa(nrep=3, data.matrix=lsat6, q.eigen=.99, diff.fact=TRUE)

### EXAMPLE 4b:
### in this case the parameter diff.fact is set to TRUE, so the function
### will simulate random dataset with the same probability of occurrence
### of each category for each item in the observed dataset.
### Polytomous variables are used in this example.
#require(psych)
#data(bfi)
#raw.data.4a<-as.matrix(bfi)
#raw.data.4a <- (raw.data.4a[1:200,1:25])
### POLYTHOMOUS
#random.polychor.pa(nrep=100, data.matrix=raw.data.4a, q.eigen=.99, diff.fact=TRUE)

### EXAMPLE 5:
### RANDOM VS BOOSTRAP SIMULATED DATA
### UNIFORM VS MULTINOMIAL DISTRIBUTION
#require(psych)
#data(bfi)
#names(bfi)
#bfi.data<-na.exclude(as.matrix(bfi[1:200, 1:5]))
#head(bfi.data)
### RANDOM samples and UNIFORM distribution
#random.polychor.pa(nrep=100, data.matrix=bfi.data, q.eigen=.95, comparison=c("random"),
# distr="uniform", fit.pa=T, print.all=T)
### RANDOM samples and MULTINOMIAL distribution
#random.polychor.pa(nrep=100, data.matrix=bfi.data, q.eigen=.95, comparison=c("random"),
# distr="multinomial", fit.pa=T, print.all=T)

### BOOSTRAP and UNIFORM distribution
#random.polychor.pa(nrep=100, data.matrix=bfi.data, q.eigen=.95, comparison=c("bootstrap"),
# distr="uniform", fit.pa=T, print.all=T)
### BOOSTRAP and MULTINOMIAL distribution
#random.polychor.pa(nrep=100, data.matrix=bfi.data, q.eigen=.95, comparison=c("bootstrap"),
# distr="multinomial", fit.pa=T, print.all=T)

### EXAMPLE 6:
### MULTIGROUP PARALLEL ANALYSIS: 2 samples
#require(psych)
#data(bfi)
#names(bfi)
#table(bfi$gender)
#table(bfi$education)
#table(bfi$age)
#bfi.data.gender<-cbind(bfi[1:200, 26], bfi[1:200, 1:5])
#head(bfi.data.gender)

# MULTIGROUP PARALLEL ANALYSIS: RANDOM UNIFORM
#random.polychor.pa(nrep=100, data.matrix=bfi.data.gender, q.eigen=.95, comparison=c("random-mg"),
# distr="uniform", fit.pa=T, print.all=T)
# MULTIGROUP PARALLEL ANALYSIS: RANDOM MULTINOMIAL

```

```
#random.polychor.pa(nrep=100, data.matrix=bfi.data.gender, q.eigen=.95, comparison=c("random-mg"),
# distr="multinomial", fit.pa=T, print.all=T)

# MULTIGROUP PARALLEL ANALYSIS: BOOTSTRAP UNIFORM
#random.polychor.pa(nrep=100, data.matrix=bfi.data.gender, q.eigen=.95,
# comparison=c("bootstrap-mg"), distr="uniform", fit.pa=T, print.all=T)
# MULTIGROUP PARALLEL ANALYSIS: BOOTSTRAP MULTINOMIAL
#random.polychor.pa(nrep=100, data.matrix=bfi.data.gender, q.eigen=.95,
# comparison=c("bootstrap-mg"), distr="multinomial", fit.pa=T, print.all=T)

### EXAMPLE 7:
### MULTIGROUP PARALLEL ANALYSIS: 11 samples
#require(psych)
#data(bfi)
#names(bfi)
#table(bfi$age)
#raw.data<- subset(bfi, bfi$age>=17 & bfi$age<=27)
#table(raw.data$age)
#bfi.data.age<-cbind(raw.data[, 28], raw.data[, 1:10])
#head(bfi.data.age)

#random.polychor.pa(nrep=100, data.matrix=bfi.data.age, q.eigen=.95, comparison=c("random-mg"))
```

Index

- *Topic **BOOTSTRAP**
 - Parallel-Analysis-of-Polychoric-Correlations, [4](#)
 - random.polychor.pa, [6](#)
 - random.polychor.pa, [6](#)
 - random.polychor.pa-package (Parallel-Analysis-of-Polychoric-Correlations), [4](#)
 - *Topic **EXPLORATORY FACTOR ANALYSIS**
 - Parallel-Analysis-of-Polychoric-Correlations, [4](#)
 - random.polychor.pa, [6](#)
 - *Topic **FIT INDEXES**
 - Parallel-Analysis-of-Polychoric-Correlations, [4](#)
 - random.polychor.pa, [6](#)
 - *Topic **MULTIGROUP**
 - Parallel-Analysis-of-Polychoric-Correlations, [4](#)
 - random.polychor.pa, [6](#)
 - *Topic **MULTINOMIAL DISTRIBUTION**
 - Parallel-Analysis-of-Polychoric-Correlations, [4](#)
 - random.polychor.pa, [6](#)
 - *Topic **PARALLEL ANALYSIS**
 - Parallel-Analysis-of-Polychoric-Correlations, [4](#)
 - random.polychor.pa, [6](#)
 - *Topic **POLYCHORIC CORRELATION MATRIX**
 - Parallel-Analysis-of-Polychoric-Correlations, [4](#)
 - random.polychor.pa, [6](#)
 - *Topic **UNIFORM DISTRIBUTION**
 - Parallel-Analysis-of-Polychoric-Correlations, [4](#)
 - random.polychor.pa, [6](#)
- NEWS, [2](#)
- Parallel-Analysis-of-Polychoric-Correlations, [4](#)