Package ‘MonteCarloSEM’

January 20, 2025

Type Package

Title Monte Carlo Data Simulation Package

Version 0.0.8

Description Monte Carlo simulation allows testing different conditions given to the correct structural equation models. This package runs Monte Carlo simulations under different conditions (such as sample size or normality of data). Within the package data sets can be simulated and run based on the given model.
First, continuous and normal data sets are generated based on the given model. Later Fleishman's power method (1978) <DOI:10.1007/BF02293811> is used to add non-normality if exists.
When data generation is completed (or when generated data sets are given) model test can also be run.
Please cite as ``Orçan, F. (2021). MonteCarloSEM: An R Package to Simulate Data for SEM. International Journal of Assessment Tools in Education, 8 (3), 704-713.''

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Imports Matrix, stats, utils, lavaan

Copyright Fatih Orcan, Kahramanmaras Sutcu Imam University, Turkey

NeedsCompilation no

Author Fatih Orcan [aut, cre] (<`https://orcid.org/0000-0003-1727-0456`>)

Maintainer Fatih Orcan <`fatihorcan84@gmail.com`>

Repository CRAN

Date/Publication 2024-04-05 09:52:59 UTC

Contents

`categorize` *This function uses pre-given data sets to create categorical data sets for given thresholds.*

Description

Previously simulated data sets are utilized to create categorical data sets by the given thresholds.

Usage

```
categorize(f.loc, threshold, dataList = "Data_List.dat")
```

Arguments

`f.loc`	File location. Generated data sets will be saved at the user-defined location.
`threshold`	The threshold values.
`dataList`	List of the names of data sets generated earlier either with the package functions or any other software.

Author(s)

Fatih Orçan

Examples

```
tres<-c(-Inf, -1.645, -.643, .643, 1.645, Inf) # five categories
categorize(f.loc=tempdir(), threshold = tres)
```

`fcors.value` *This function specifies the correlation matrix between the factors.*

Description

The user specifies the correlation matrix between the factors. The values entered should be between -1 and +1. The values can be given by column or row but should be given in order. Please see the example for a correlation among three factors. In case there is only one factor following line should be entered "cors.value(nf=1, cors=c(1,1,1))"

Usage

```
fcors.value(nf, cors)
```

Arguments

`nf`	the number of factor/s.
`cors`	vector of the correlations.

Value

The function returns the factor correlation matrix. This is a symmetric matrix, which shows the correlation values among the factors in the model.

Author(s)

Fatih Orcan

Examples

```
# This example represents a three-factor CFA model
#
fcors.value(nf=3, cors=c(1,.5,.6,.5,1,.4,.6,.4,1))
```

`fit.simulation` *This function runs a model for simulated data by using the lavaan package.*

Description

Generated data sets (Generated by sim.skewed() or sim.normal() functions) will be fitted to pre-specified model. The lavaan package is used to fit the model. After running the model, fit indices and parameters estimated with their standard errors will be printed to a Comma Separated Values (CSV) file. The name of the output file is "All_Results.csv". Each line in the file represents the results of a simulation. The columns are self-explanatory but the second column (named Notes) needs a more detailed explanation. This column shows if the model convergency. If the model is converged without any problem the value will be "CONVERGE" If it is not converged the value will be "NONCONVERGE" and all the values in the line will be "NA" If there are some kind of warnings (such as negative variance) during the model run the value will be "WARNING" and based on the warning type some of the values might be "NA". To run the simulation, previously generated (either with the package functions or any other software) data sets and the list of the data sets (i.e., "Data_List.dat") should be located in the same folder in the working directory.

Usage

```
fit.simulation(
  model,
  PEmethod = "ML",
  Ordered = FALSE,
  dataList = "Data_List.dat",
  f.loc,
  missing = NULL
)
```

Arguments

`model`	Lavaan model
`PEmethod`	Parameter estimation method. The default is ML.
`Ordered`	Logical, TRUE means that the data will be defined as ordered.
`dataList`	List of the names of data sets generated earlier either with the package functions or any other software.
`f.loc`	File location. It indicates where the simulated data sets and "dataList" are located.
`missing`	as in the lavaan package (See lavOptions)

Author(s)

Fatih Orcan

Examples

```
#   Data needed to be generated at the first step.
fc<-fcors.value(nf=3, cors=c(1,.5,.6,.5,1,.4,.6,.4,1))
fl<-loading.value(nf=3, fl.loads=c(.5,.5,.5,0,0,0,0,0,0,0,0,.6,.6,.6,0,0,0,0,0,0,0,0,.4,.4))
sim.normal(nd=10, ss=100, fcors=fc, loading<-fl,  f.loc=tempdir())

#  Then simulation should be run at the second step.
```

```
lavaanM<-'
#CFA Model
f1 =~ NA*x1 + x2 + x3
f2 =~ NA*x4 + x5 + x6
f3 =~ NA*x7 + x8
#Factor Correlations
f1 ~~ f2
f1 ~~ f3
f2 ~~ f3
#Factor variance
f1 ~~ 1*f1
f2 ~~ 1*f2
f3 ~~ 1*f3
'
dl<-"Data_List.dat"  # should be located in the working directory.

# Note that this function uses data sets and the list files which were generated previously.
# If there are no such a data sets and the list file, it will print an error message
#  saying "cannot open the connection"

fit.simulation(model=lavaanM, PEmethod="MLR", Ordered=FALSE, dataList=dl, f.loc=tempdir())
```

---

loading.value *This function specifies the factor loading values.*

Description

The user specifies the factor loadings as a matrix. The values should be given by column for each factor. Columns represent factors and rows represent items. The values entered should be larger than 0 and smaller than 1. Please see the example for a loading matrix for a three-factor model.

Usage

```
loading.value(nf, fl.loads)
```

Arguments

`nf`	the number of factor/s.
`fl.loads`	vector of factor loadings

Value

The function returns the factor loading matrix. The number of columns shows the number of factors in the model. The rows show the number of items

Author(s)

Fatih Orçan

Examples

```
# This example represents a three-factor CFA model
#  where the factors are indicated by 3, 3, and 2 items respectively.
#
loading.value(nf=3, fl.loads=c(.6,.6,.6,0,0,0,0,0,0,0,0,.7,.7,.7,0,0,0,0,0,0,0,0,.8,.8))
```

`MAR.data` *This function inserts missingness (Missing at Random - MAR) into the given data sets.*

Description

Missing values (MAR) will be added to the generated data sets (Generated by sim.skewed() or sim.normal() functions). Under MAR, the missingness was associated with the values of the variable in the data set except itself. If baseV parameter was not given, two different and random variables in the data set are selected, and the missing values are assigned based on the mean of the two variables on the selected item. For example, if the data has 8 items and the second item will be assigned MAR values, two items among the item 1, 3, 4, 5, 6, 7, and 8 were selected randomly, let's say items 5 and 7. The mean of the items was then calculated and the values were sorted. Then, based on the given percent of missingness, 90 percent of the missing values were selected from the top. The remaining 10 percent of missing values were assigned from the rest of the variable. For example, let's say the sample size was 300, and 20 percent of missingness was wanted (missing count: 300x20 The missing values are shown as "NA" in the data files. The new data sets which have missing values will be saved as a different data file. In each data file, the first column shows sample numbers. The second and the other columns show actual data sets for each item. There will be a file named "MAR_List.dat". The file includes the names of the data sets which has missing values in it. Besides, a file named "Model_MAR_relations.dat" shows which item was associated with which random items that were used for the MAR calculation.

Usage

```
MAR.data(
  misg = NULL,
  baseV = NULL,
  perct = 10,
  dataList = "Data_List.dat",
  f.loc
)
```

Arguments

`misg`	A vector of 0s and 1s for each item. 0 indicates non-missing and 1 indicates items which have missing values. If misg is not indicated all items are considered as missing.
`baseV`	A list of items that MAR will be calculated based on. It has to be match with the misg parameter. If it is not given, two random items (except the variable itself) will be selected and used to get MAR values for the given items.

`perct`	The percent of missingness. The default is 10 percent.
`dataList`	List of the names of data sets generated earlier either with the package functions or any other software.
`f.loc`	File location. It indicates where the simulated data sets and "dataList" are located.

Author(s)

Fatih Orcan

Examples

```
#   Data needed to be generated at the first step.

fc<-fcors.value(nf=3, cors=c(1,.5,.6,.5,1,.4,.6,.4,1))
fl<-loading.value(nf=3, fl.loads=c(.5,.5,.5,0,0,0,0,0,0,0,0,.6,.6,.6,0,0,0,0,0,0,0,0,.4,.4))
floc<-tempdir()
sim.normal(nd=10, ss=100, fcors=fc, loading<-fl,  f.loc=floc)

 #  Missing values were added at the second step.

mis.items<-c(1,0,1,1,0,0,0,0)
bV<-list(c(0,0,0,0,0,0,1,1),NA,c(0,0,0,0,0,1,1,0),c(0,0,0,0,0,1,1,1), NA,NA,NA,NA)
dl<-"Data_List.dat"  # should be located in the working directory.
MAR.data(misg = mis.items, baseV=bV, perct = 20, dataList = dl, f.loc=floc )
```

MCAR.data — *This function inserts missingness (Missing Completely at Random - MCAR) into the given data sets.*

Description

Missing values (MCAR) will be added to the Generated data sets (Generated by sim.skewed() or sim.normal() functions). Missing values are assigned at random and are shown as "NA" in the data files. The new data sets which have missing values will be saved as a different data file. In each data file, the first column shows sample numbers. The second and the other columns show actual data sets for each item. There also be a file named "MCAR_List.dat". The file includes the names of the data sets which has missing values in it.

Usage

```
MCAR.data(misg = NULL, perct = 10, dataList = "Data_List.dat", f.loc)
```

Arguments

`misg`	vector of 0s and 1s for each item. 0 indicates non-missing and 1 indicates items that have missing values. If misg is not indicated all items are considered as missing.
`perct`	Percent of missingness. The default is 10 percent.
`dataList`	List of the names of data sets generated earlier either with the package functions or any other software.
`f.loc`	File location. It indicates where the simulated data sets and "dataList" are located.

Author(s)

Fatih Orcan

Examples

```
#   Data needed to be generated at the first step.

fc<-fcors.value(nf=3, cors=c(1,.5,.6,.5,1,.4,.6,.4,1))
fl<-loading.value(nf=3, fl.loads=c(.5,.5,.5,0,0,0,0,0,0,0,0,.6,.6,.6,0,0,0,0,0,0,0,0,.4,.4))
floc<-tempdir()
sim.normal(nd=10, ss=100, fcors=fc, loading<-fl,  f.loc=floc)

 #  Missing values were added at the second step.

mis.items<-c(1,1,1,0,0,0,0,0)
dl<-"Data_List.dat"  # should be located in the working directory.
MCAR.data(misg = mis.items, perct = 20, dataList = dl, f.loc=floc)
```

`MNAR.data` *This function inserts missingness (Missing Not at Random - MNAR) into the given data sets.*

Description

Missing values (MNAR) will be added to the Generated data sets (Generated by sim.skewed() or sim.normal() functions). Under MNAR, the missingness was associated with the values of the variable itself. In order to create MNAR, the variable was sorted first. Then, based on the given percent of missingness, 90 percent of the missing values were selected from the top. The remaining 10 percent of missing values were assigned from the rest of the variable. For example, let's say the sample size was 300, and 20 percent of missingness was wanted (missing count: 300x20 The missing values are shown as "NA" in the data files. The new data sets which have missing values will be saved as a different data file. In each data file, the first column shows sample numbers. The second and the other columns show actual data sets for each item. There also be a file named "MNAR_List.dat". The file includes the names of the data sets which has missing values in it.

Usage

```
MNAR.data(misg = NULL, perct = 10, dataList = "Data_List.dat", f.loc)
```

Arguments

`misg`	vector of 0s and 1s for each item. 0 indicates non-missing and 1 indicates items which have missing values. If misg is not indicated all items are considered as missing.
`perct`	Percent of missingness. The default is 10 percent.
`dataList`	List of the names of data sets generated earlier either with the package functions or any other software.
`f.loc`	File location. It indicates where the simulated data sets and "dataList" are located.

Author(s)

Fatih Orcan

Examples

```
#   Data needed to be generated at the first step.

fc<-fcors.value(nf=3, cors=c(1,.5,.6,.5,1,.4,.6,.4,1))
fl<-loading.value(nf=3, fl.loads=c(.5,.5,.5,0,0,0,0,0,0,0,0,.6,.6,.6,0,0,0,0,0,0,0,0,.4,.4))
floc<-tempdir()
sim.normal(nd=10, ss=100, fcors=fc, loading<-fl,  f.loc=floc)

 #  Missing values were added at the second step.

mis.items<-c(1,1,1,0,0,0,0,0)
dl<-"Data_List.dat"  # should be located in the working directory.
MNAR.data(misg = mis.items, perct = 20, dataList = dl, f.loc=floc)
```

`sim.categoric` *This function simulates (generates) categorical data sets by a given Confirmatory Factor Analysis model.*

Description

Based on a given Confirmatory Factor Analysis model, this function simulates data sets. In each data file, the first column shows sample numbers. The second and other columns show actual simulated data sets for each item. If the model has 2 factors and each factor has 3 items, for example, column names will be something like "ID, F1_x1, F1_x2, F1_x3, F2_x1, F2_x2, F2_x3". On the other hand, the number of rows shows the sample number of the data. Besides, there will be two more files saved in the folder. First of them is "Model_Info.dat". This file includes factor correlation and factor loading matrices. The second is "Data_List.dat". The file contains the names of the data sets which were generated.

Usage

```
sim.categoric(
  nd = 10,
  ss = 100,
  fcors,
  loading,
  f.loc,
  threshold,
  cont = "FALSE"
)
```

Arguments

`nd`	Number of the data set, an integer.
`ss`	Sample Size, an integer and larger than 10.
`fcors`	The factor correlation matrix, a symmetric matrix. If one-factor model is used this should be matrix(1,1,1).
`loading`	The factor loading matrix. The column represents factors and non-zero rows represent the number of items under each factor.
`f.loc`	File location. Generated data sets will be saved at the user-defined location.
`threshold`	The threshold values.
`cont`	TRUE or FALSE: Indicating whether original continuous data will be saved or not.

Author(s)

Fatih Orçan

Examples

```
fc<-fcors.value(nf=3, cors=c(1,.5,.6,.5,1,.4,.6,.4,1))
fl<-loading.value(nf=3, fl.loads=c(.5,.5,.5,0,0,0,0,0,0,0,0,.6,.6,.6,0,0,0,0,0,0,0,0,.4,.4))
tres<-c(-Inf, -1.645, -.643, .643, 1.645, Inf) # five categories

sim.categoric(nd=100,ss=100, fcors=fc,loading=fl, f.loc=tempdir(), threshold = tres)
```

`sim.normal` *This function simulates (generates) data sets by a given Confirmatory Factor Analysis model.*

Description

Based on a given Confirmatory Factor Analysis model, this function simulates data sets. In each data file, the first column shows sample numbers. The second and other columns show actual simulated data sets for each item. If the model has 2 factors and each factor has 3 items, for example, column names will be something like "ID, F1_x1, F1_x2, F1_x3, F2_x1, F2_x2, F2_x3". On the other hand, the number of rows shows the sample number of the data. Besides, there will be two more files saved in the folder. First of them is "Model_Info.dat". This file includes factor correlation and factor loading matrices. The second is "Data_List.dat". The file contains the names of the data sets which were generated.

Usage

```
sim.normal(nd = 10, ss = 100, fcors, loading, f.loc)
```

Arguments

`nd`	Number of the data set, an integer.
`ss`	The sample Size, an integer and larger than 10.
`fcors`	The factor correlation matrix, a symmetric matrix. If one-factor model is used this should be matrix(1,1,1).
`loading`	The factor loading matrix. The column represents factors and non-zero rows represent the number of items under each factor.
`f.loc`	File location. Generated data sets will be saved at the user-defined location.

Author(s)

Fatih Orçan

Examples

```
fc<-fcors.value(nf=3, cors=c(1,.5,.6,.5,1,.4,.6,.4,1))
fl<-loading.value(nf=3, fl.loads=c(.5,.5,.5,0,0,0,0,0,0,0,0,0,.6,.6,.6,0,0,0,0,0,0,0,0,0,.4,.4))

sim.normal(nd=10, ss=1000, fcors=fc, loading<-fl,  f.loc=tempdir())
```

`sim.skewed` *Simulates Data sets by a given Confirmatory Factor Analysis model.*

Description

Based on a given Confirmatory Factor Analysis model, this function simulates data sets. In each data file, the first column shows sample numbers. The second and other columns show actual simulated data sets for each item. If the model has 2 factors and each factor has 3 items, for example, column names will be something like "ID, F1_x1, F1_x2, F1_x3, F2_x1, F2_x2, F2_x3". On the other hand, the number of rows shows the sample number of the data. Besides, there will be two more files saved in the folder. First of them is "Model_Info.dat". This file includes factor correlation and factor loading matrices, a vector showing non-normal items and values of B, C, and D for Fleishman's power method. The second is "Data_List.dat". The file contains the names of the data sets which were generated.

Usage

```
sim.skewed(
  nd = 10,
  ss = 100,
  fcors,
  loading,
  nonnormal = NULL,
  Fleishman = NULL,
  f.loc
)
```

Arguments

`nd`	Number of the data set, an integer.
`ss`	The sample Size, an integer and larger than 10.
`fcors`	The factor correlation matrix, a symmetric matrix. If one-factor model is used this should be matrix(1,1,1).
`loading`	The factor loading matrix. The column represents number of factors and non-zero rows represent the number of items under each factor.
`nonnormal`	A vector of 0 and 1s. 0 indicates normal, and 1 indicates non-normal data generation. If nonnormal is not indicated normal data will be generated.
`Fleishman`	The B, C, and D values from Fleishman's power method. A = -C.
`f.loc`	File location. Generated data sets will be saved at the user-defined location.

Author(s)

Fatih Orçan

Examples

```
fc<-fcors.value(nf=3, cors=c(1,.5,.6,.5,1,.4,.6,.4,1))
fl<-loading.value(nf=3, fl.loads=c(.5,.5,.5,0,0,0,0,0,0,0,0,.3,.3,.3,0,0,0,0,0,0,0,0,.4,.4))
ifN<-c(1,1,1,0,0,0,0,0)
fleis<-c(1.0174852, .190995, -.018577) # The values for skewness=1 and kurtosis=1

sim.skewed(nd=10, ss=100, fcors=fc,loading=fl, nonnormal = ifN, Fleishman = fleis, f.loc=tempdir())
```

Index