Package 'MonteCarloSEM'

May 7, 2026

Type Package

Title Monte Carlo Simulation for Structural Equation Modeling

Version 2.0.0

Description Provides tools to conduct Monte Carlo simulations under different conditions (e.g., varying sample size, data normality) for structural equation models (SEMs).
Data can be simulated based on user-defined factor loadings and correlations, with optional non-normality added via Fleishman's power method (1978) <doi:10.1007/BF02293811>.
Once generated, models can be estimated using 'lavaan'. This package facilitates testing model performance across multiple simulation scenarios.
When data generation is completed (or when generated data sets are given) model tests 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

NeedsCompilation no

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

Maintainer Fatih Orçan <fatihorcan84@gmail.com>

Repository CRAN

Date/Publication 2026-01-28 13:40:07 UTC

Contents

`categorize` *Generates Categorical Data Sets from Continuous Data.*

Description

This function transforms previously simulated continuous data sets into categorical variables based on user-specified threshold values. The function reads in existing data sets, applies the thresholding procedure to discretize the observed scores, and saves the resulting categorical data sets into the designated file location. Additionally, it produces an updated list of the newly created categorical data sets for future reference.

Usage

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

Arguments

`f.loc`	A character string indicating the file path where the generated categorical data sets will be saved.
`threshold`	A numeric vector specifying the threshold values used to discretize the continuous data into ordered categories.
`dataList`	A character string giving the name of the file that contains the list of previously generated data sets.

Author(s)

Fatih Orçan

Examples

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

cov.mtx *Simulates Correlation matrix by a given SEM model.*

Description

This function generates the model-implied covariance and correlation matrices based on a specified structural equation model (SEM). The function returns the implied covariance and correlation matrices.

Usage

```
cov.mtx(Model, nobs)
```

Arguments

Model	A lavaan model object specifying the measurement and structural components of the SEM.
nobs	An integer indicating the number of observed indicators (xs) in the model.

Value

Returns model implied covariance and correlation matrices.

Author(s)

Fatih Orçan

Examples

```
LavaanM <- '
# Measurement model (fixed factor loading)
F1 =~ 0.7*x1 + 0.7*x2 + 0.7*x3
F2 =~ 0.7*x4 + 0.7*x5 + 0.7*x6
F3 =~ 0.7*x7 + 0.7*x8 + 0.7*x9
# Structural regressions
F2 ~ 0.4*F1
F3 ~ 0.6*F1
# Fix latent variances
F1 ~~ 1*F1
# Residual variances
F2 ~~ 1*F2
F3 ~~ 1*F3
# Correlated residuals
F2 ~~ 0.5*F3
'

cov.mtx(Model=LavaanM, nobs=9)
```

fcors.value *Specifies the Factor Correlation Matrix for a Model*

Description

This function generates a symmetric factor correlation matrix for a Structural Equation Model. The correlations must be provided as a vector of values between -1 and +1. The vector can be entered in either row-wise or column-wise order, but the correlations must be supplied in the correct sequence. If the model includes only a single factor, the function should be called as: "fcors.value(nf = 1, cors = c(1))"

Usage

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

Arguments

nf	An integer specifying the number of factors.
cors	A numeric vector of correlations among the factors. Values must be between -1 and +1.

Value

The function returns the factor correlation matrix.

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 *Fits Structural Equation Models to Simulated Data Using lavaan.*

Description

This function applies a pre-specified SEM model to previously generated data sets (e.g., from sim.skewed() or sim.normal()) by utilizing the lavaan package. After model estimation, fit indices and parameter estimates with their standard errors are exported to a Comma-Separated Values (CSV) file named All_Results.csv. Each row in this file corresponds to the results of a single simulation. Most columns are self-explanatory; however, the second column (Notes) requires further clarification. This column indicates the convergence status of the model: CONVERGE – The model converged without any issues. NONCONVERGE – The model failed to converge; in this case, all values in the row are recorded as NA. WARNING – The model converged but produced warnings (e.g., negative variance estimates). Depending on the warning type, some values may be recorded as NA. To run the simulation, previously generated data sets (either via the package functions or other software) must be stored in the same folder as the dataset list file (Data_List.dat) within the working directory.

Usage

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

Arguments

model	A Lavaan model
PEmethod	The parameter estimation method. The default is ML.
Ordered	Logical, If TRUE, variables are treated as ordered categorical; otherwise, as continuous.
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	A specification for handling missing data. As in the lavaan package (See lavOptions)

Author(s)

Fatih Orcan

Examples

```
# Step 1: Generate data
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())
```

```
# Step 2: Specify the model
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"  # must be available in the working directory

# Step 3: Fit the model across simulated data

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

`loading.value` *Specifies Factor Loading Values for a Model.*

Description

This function creates a factor loading matrix for a given Structural Equation Model (SEM). The loadings must be provided as a vector and are assigned to the matrix column by column, where each column corresponds to a latent factor and each row corresponds to an observed item. All factor loadings must be specified as values strictly greater than 0 and less than 1. The resulting matrix has dimensions equal to the number of items by the number of factors.

Usage

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

Arguments

`nf`	An integer specifying the number of factors.
`fl.loads`	A numeric vector of factor loadings. Values should be provided in column-wise order, corresponding to the items loading on each factor.

Value

The function returns the factor loading matrix.

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 *Introduces Missing at Random (MAR) Values into Data Sets.*

Description

This function introduces missing values under the Missing at Random (MAR) mechanism into previously generated data sets (e.g., those produced by sim.skewed() or sim.normal()). Under MAR, the probability of missingness is associated with other variables in the data set, but not with the variable itself. If the baseV argument is not provided, two random variables (excluding the target variable itself) are selected. Their mean is then used to determine missingness in the target variable. For example, assume a data set with 8 items where missing values are to be introduced for item 2. Two items are randomly selected from items 1, 3, 4, 5, 6, 7, and 8 (e.g., items 5 and 7). Their mean is calculated, sorted, and used as the basis for assigning missingness to the item 2. Following the MAR rule, 90 percents of the missing values are drawn from the highest scores, and the remaining 10 percents are drawn randomly from the rest. For instance, with a sample size of 300 and 20 percents missingness (60 cases), the mean of the selected auxiliary variables is sorted in decreasing order. Missing values are then introduced in 54 cases (90 percents of 60) from the top portion, while 6 cases (10 percents of 60) are drawn randomly from the lower 240 observations. The missing values are represented by NA in the output files. New data sets containing missing values are saved as separate files, preserving the originals. Additionally, a file named "MAR_List.dat" is created, which contains the names of all data sets with MAR missingness.

Usage

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

Arguments

misg	A numeric vector of 0s and 1s specifying which items will contain missing values. A value of 0 indicates the item will not include missingness, while 1 indicates missing values will be introduced. If omitted, all items are treated as eligible for missingness.

baseV	A list specifying the auxiliary variables on which MAR missingness will be based. This must match to the structure of misg. If not provided, two random variables (excluding the variable itself) are chosen automatically.
perct	The percentage of missingness to be applied (default = 10 percents).
dataList	The file name containing the list of previously generated data sets (e.g., "Data_List.dat"), either created by this package or by external software.
f.loc	The directory path where both the original data sets and the "dataList" file are located.

Author(s)

Fatih Orcan

Examples

```
# Step 1: Generate data sets

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)

# Step 2: Introduce MAR missing values

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"  # must be located in the working directory
MAR.data(misg = mis.items, baseV=bV, perct = 20, dataList = dl, f.loc=floc )
```

MCAR.data *Introduces Missing Completely at Random (MCAR) Values into Data Sets.*

Description

This function introduces missing values under the Missing Completely at Random (MCAR) mechanism into previously generated data sets (e.g., those produced by sim.skewed() or sim.normal()). Missing values are inserted at random locations according to user specifications and are denoted as "NA" in the resulting files. The modified data sets are saved as new files, preserving the original data sets. In each data file, the first column contains the sample identifiers, while the subsequent columns show actual data with some entries replaced by NA. Additionally, a file named "MCAR_List.dat" is created, listing the names of all data sets to which missing values were introduced.

Usage

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

Arguments

misg	A numeric vector of 0s and 1s specifying which items will contain missing values. A value of 0 indicates the item will not include missingness, while 1 indicates missing values will be introduced.If omitted, all items are treated as eligible for missingness.
perct	The percentage of missingness to be applied (default = 10 percents).
dataList	The file name containing the list of previously generated data sets (e.g., "Data_List.dat"), either created by this package or by external software.
f.loc	The directory path where both the original data sets and the "dataList" file are located.

Author(s)

Fatih Orcan

Examples

```
# Step 1: Generate data sets
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)

 # Step 2: Introduce missing values

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

MNAR.data *Introduces Missing Not at Random (MNAR) Values into Data Sets*

Description

This function introduces missing values under the Missing Not at Random (MNAR) mechanism into previously generated data sets (e.g., those produced by sim.skewed() or sim.normal()). Under the MNAR mechanism, the probability of missingness depends on the observed values of the variable itself. Specifically, the target variable is first sorted in decreasing order. Based on the specified percentage of missingness, 90 percents of missing values are assigned randomly among the highest values, while the remaining 10 percents are assigned randomly among the rest of the sample. For example, with a sample size of 300 and a target of 20 percents missingness (60 cases), the variable is sorted in descending order. Missing values are then introduced in 54 cases (90 percents of 60) from the top of the distribution, while the remaining 6 cases (10 percents of 60) are randomly chosen from the lower 240 observations. The missing values are represented by NA in the output files. New data sets containing missing values are saved as separate files, preserving the originals. Additionally, a file named "MNAR_List.dat" is created, which contains the names of all data sets with MNAR missingness.

Usage

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

Arguments

misg	A numeric vector of 0s and 1s specifying which items will contain missing values. A value of 0 indicates the item will not include missingness, while 1 indicates missing values will be introduced. If omitted, all items are treated as eligible for missingness.
perct	The percentage of missingness to be applied (default = 10 percents).
dataList	The file name containing the list of previously generated data sets (e.g., "Data_List.dat"), either created by this package or by external software.
f.loc	The directory path where both the original data sets and the "dataList" file are located.

Author(s)

Fatih Orcan

Examples

```
# Step 1: Generate data sets

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)

 # Step 2: Introduce MNAR missing values

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

---

sim.categoric *Simulates Categorical Data Sets Based on a Structural Equation Model (SEM).*

Description

This function generates categorical data sets from a specified SEM. The simulated data are organized such that the first column represents case identifiers, while the subsequent columns contain the simulated item responses. For example, in a model with two factors and three items per factor, the column labels will follow the format: "ID, F1_x1, F1_x2, F1_x3, F2_x1, F2_x2, F2_x3". The number of rows corresponds to the sample number of the data. In addition to the generated data sets, two supplementary files are also saved: (1) "Model_Info.dat" — containing the factor correlation and factor loading matrices (2) "Data_List.dat" — listing the names of all generated data files.

Usage

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

Arguments

nd	An integer, the number of data sets to be generated.
ss	An integer, the sample size per data set (must be greater than 10).
fcors	The factor correlation matrix, which must be symmetric. For one-factor models, this should be "matrix(1,1,1)".
loading	The factor loading matrix. Columns correspond to factors, while non-zero rows specify the number of items associated with each factor.
f.loc	File path indicating the directory where the generated data sets will be saved.
threshold	Threshold values used to categorize continuous simulated data.
cont	Logical: If TRUE, the original continuous data sets are also saved in addition to the categorical versions.

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 response categories

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

---

sim.normal *Simulates Data Sets Based on a Structural Equation Model (SEM).*

Description

This function generates data sets based on a specified SEM. The simulated data are organized such that the first column represents case identifiers, while the subsequent columns contain the simulated item responses. For example, in a model with two factors and three items per factor, the column labels will follow the format: "ID, F1_x1, F1_x2, F1_x3, F2_x1, F2_x2, F2_x3". The number of rows corresponds to the sample number of the data. In addition to the generated data sets, two supplementary files are also saved: (1) "Model_Info.dat" — containing the factor correlation and factor loading matrices (2) "Data_List.dat" — listing the names of all generated data files.

Usage

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

Arguments

nd	An integer, the number of data sets to be generated.
ss	An integer, the sample size per data set (must be greater than 10).
fcors	The factor correlation matrix, which must be symmetric. For one-factor models, this should be "matrix(1,1,1)".
loading	The factor loading matrix. Columns correspond to factors, while non-zero rows specify the number of items associated with each factor.
f.loc	File path indicating the directory where the generated data sets will be saved.

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))

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

---

sim.skewed *Simulates Data Sets from a Structural Equation Model (SEM) with Normal or Non-Normal Distributions*

Description

This function generates data sets based on a specified SEM. The simulated data are organized such that the first column represents case identifiers, while the subsequent columns contain the simulated item responses. For example, in a model with two factors and three items per factor, the column labels will follow the format: "ID, F1_x1, F1_x2, F1_x3, F2_x1, F2_x2, F2_x3". The number of rows corresponds to the sample number of the data. In addition to the generated data sets, two supplementary files are also saved: (1) "Model_Info.dat" — containing the factor correlation matrix, factor loading matrix, a vector indicating non-normal items, and the coefficients B, C, and D from Fleishman's power method (where A = –C). (2) "Data_List.dat" — listing the names of all generated data files.

Usage

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

Arguments

nd	An integer, the number of data sets to be generated.
ss	An integer, the sample size per data set (must be greater than 10).
fcors	The factor correlation matrix, which must be symmetric. For one-factor models, this should be "matrix(1,1,1)".
loading	The factor loading matrix. Columns correspond to factors, while non-zero rows specify the number of items associated with each factor.
nonnormal	A numeric vector of 0s and 1s indicating whether each variable should be generated as normal (0) or non-normal (1). If not specified, all variables are generated as normal.
Fleishman	A numeric vector containing the coefficients B, C, and D from Fleishman's power method. Note that A = –C.
f.loc	File path indicating the directory where the generated data sets and auxiliary files will be saved.

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 coefficients for skewness = 1, kurtosis = 1

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

Index