Package ‘BinNonNor’

April 15, 2020

Type Package
Title Data Generation with Binary and Continuous Non-Normal Components
Version 1.5.2
Date 2020-04-14
Author Gul Inan, Hakan Demirtas, Ran Gao
Maintainer Ran Gao <rgao8@uic.edu>
Description Generation of multiple binary and continuous non-normal variables simultaneously given the marginal characteristics and association structure based on the methodology proposed by Demirtas et al. (2012) <DOI:10.1002/sim.5362>.
License GPL-2 | GPL-3
Depends BB, corpcor, mvtnorm, Matrix
NeedsCompilation no
Repository CRAN
Date/Publication 2020-04-15 05:50:24 UTC

R topics documented:

BinNonNor-package ........................................... 2
Biserial.Corr.BN ............................................. 3
correlation.bound.check .................................. 4
correlation.limits .......................................... 6
fleishman.coef ............................................. 7
gen.Bin.NonNor ............................................ 8
Int.Corr.NN ................................................... 12
overall.corr.mat ........................................... 13
Tetra.Corr.BB ............................................... 14
validation.bin .............................................. 15
validation.corr ............................................. 16
validation.skewness.kurtosis ............................ 18
Index ......................................................... 20
BinNonNor-package

Data Generation with Binary and Continuous Non-Normal Components

Description

Provides R functions for generation of multiple binary and continuous non-normal variables simultaneously given the marginal characteristics and association structure based on the methodology proposed by Demirtas et al. (2012).

Details

Package: BinNonNor
Type: Package
Version: 1.5.2
Date: 2020-04-14
License: GPL-2 | GPL-3

This package consists of eleven functions. The functions validation.bin, validation.corr, and validation.skewness.kurtosis validate the specified quantities to avoid obvious specification errors. The function fleishman.coef computes the coefficients of the third order Fleishman polynomials that are used to simulate the continuous non-normal variables. correlation.limits returns the lower and upper bounds of the pairwise correlation of binary and binary and binary and continuous non-normal, and continuous non-normal and continuous non-normal pairs given their marginal distributions, i.e. returns the range of feasible pairwise correlations. The function correlation.bound.check checks the validity of the values of pairwise correlations. The functions Int.Corr.NN, Tetra.Corr.BB, and Biserial.Corr.BN computes intermediate correlation matrix for continuous non-normal and continuous non-normal combinations, tetrachoric correlations for binary and binary combinations, and biserial correlations for binary and continuous non-normal combinations, respectively. The function overall.corr.mat assembles the final correlation matrix. The engine function gen.Bin.NonNor generates mixed data in accordance with the specified marginal and correlational quantities. Throughout the package, variables are supposed to be inputted in a certain order, namely, first binary variables, and then continuous variables should be placed.

Author(s)

Gul Inan, Hakan Demirtas, Ran Gao
Maintainer: Ran Gao <rgao8@uic.edu>

References

Biserial.Corr.BN

**Computes the biserial correlation matrix for binary and continuous non-normal variables given the specified correlation matrix**

**Description**

This function computes the biserial correlation matrix for binary-continuous non-normal combinations as formulated in Demirtas et al. (2012).

**Usage**

```r
Biserial.Corr.BN(n.BB, n.NN, prop.vec, corr.vec = NULL, corr.mat = NULL, coef.mat)
```

**Arguments**

- `n.BB`: Number of binary variables.
- `n.NN`: Number of continuous non-normal variables.
- `prop.vec`: Probability vector for binary variables.
- `corr.vec`: Vector of elements below the diagonal of correlation matrix ordered column-wise.
- `corr.mat`: Specified correlation matrix.
- `coef.mat`: Matrix of coefficients produced from `fleishman.coef`.

**Value**

A matrix of size n.BB*n.NN.

**References**


**See Also**


**Examples**

```r
n.BB=2
n.NN=4
prop.vec=c(0.4,0.7)
corr.vec=NULL
corr.mat=matrix(c(1.0,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,1.0,0.4,0.5,0.6,-0.3,0.4,1.0,0.7,0.8,-0.3,0.5,0.7,1.0,0.9),
```
correlation.bound.check

Checks if the pairwise correlation among variables are within the feasible range

Description

This function checks if there are range violations among correlation of binary-binary, binary-continuous, and continuous-continuous combinations.

Usage

correlation.bound.check(n.BB, n.NN, prop.vec = NULL, corr.vec = NULL, corr.mat = NULL, coef.mat = NULL)

Arguments

n.BB Number of binary variables.
n.NN Number of continuous non-normal variables.
prop.vec Probability vector for binary variables.
corr.vec Specified correlation vector.
corr.mat Specified correlation matrix.
coef.mat Matrix of coefficients produced from fleishman.coef.

Value

The function returns TRUE if no specification problem is encountered. Otherwise, it returns an error message.
correlation.bound.check

References


See Also

defishman.coef, correlation.limits, validation.corr

Examples

prop.vec=c(0.4,0.7)
n.BB=2
n.NN=4
corr.vec=NULL
corr.mat=matrix(c(1.0,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,1.0,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,1.0,0.4,0.5,0.6,-0.3,-0.3,0.4,1.0,0.7,0.8,-0.3,-0.3,0.5,0.7,1.0,0.9,-0.3,-0.3,0.6,0.8,0.9,1.0),6,byrow=TRUE)

coef.mat=matrix(c(-0.31375, 0.00000, 0.10045, -0.10448,0.82632, 1.08574, 1.10502, 0.98085,0.31375, 0.00000, -0.10045, 0.10448,0.02271, -0.02945, -0.04001, 0.00272),4,byrow=TRUE)

correlation.bound.check(n.BB,n.NN,prop.vec,corr.vec=NULL,corr.mat,coef.mat)

cor.mat.BB=corr.mat[1:2,1:2]
correlation.bound.check(n.BB,n.NN=0,prop.vec,corr.vec=NULL,corr.mat=cor.mat.BB,coef.mat=NULL)

correlation.bound.check(n.BB=0,n.NN,prop.vec=NULL,corr.vec=NULL,corr.mat=cor.mat.NN,coef.mat)

n.BB=1
prop.vec=0.5
corr.mat=diag(n.BB)
correlation.bound.check(n.BB,n.NN=0,prop.vec,corr.vec=NULL,corr.mat=corr.mat,coef.mat=NULL)

## Not run:
cor.mat.NNnew=cor.mat.NN
cor.mat.NNnew[1,2]=0.92
cor.mat.NNnew[2,1]=0.92
correlation.bound.check(n.BB=0,n.NN,prop.vec=NULL,corr.vec=NULL,corr.mat=cor.mat.NNnew,coef.mat)
correlation.limits

Computes lower and upper correlation bounds for each pair of variables

Description

This function computes lower and upper limits for pairwise correlation of binary-binary, binary-continuous non-normal, and continuous non-normal-continuous non-normal combinations.

Usage

correlation.limits(n.BB, n.NN, prop.vec = NULL, coef.mat = NULL)

Arguments

n.BB  Number of binar variables.
n.NN  Number of continuous non-normal variables.
prop.vec  Probability vector for binary variables.
coef.mat  Matrix of coefficients produced from fleishman.coef.

Details

While the function computes the exact lower and upper bounds for pairwise correlations among binary-binary variables as formulated in Demirtas et al. (2012), it computes approximate lower and upper bounds for pairwise correlations among binary-continuous non-normal and continuous non-normal-continuous non-normal variables through the method suggested by Demirtas and Hedeker (2011).

Value

The function returns a matrix of size (n.BB + n.NN)*(n.BB + n.NN), where the lower triangular part of the matrix contains the lower bounds and the upper triangular part of the matrix contains the upper bounds of the feasible correlations.

References


See Also

fleishman.coef, correlation.bound.check
fleishman.coef

Examples

n.BB=2
n.NN=4
prop.vec=c(0.4,0.7)
coef.mat=matrix(c(-0.31375, 0.00000, 0.10045, -0.10448,
                  0.82632, 1.08574, 1.10502, 0.98085,
                  0.31375, 0.00000, -0.10045, 0.10448,
                  0.02271, -0.02945, -0.04001, 0.00272),4,byrow=TRUE)

limits=correlation.limits(n.BB,n.NN,prop.vec,coef.mat)
limits.bin=correlation.limits(n.BB,n.NN=0,prop.vec,coef.mat=NULL)
limits.nonnor=correlation.limits(n.BB=0,n.NN,prop.vec=NULL,coef.mat)

## Not run:
n.BB=1
prop.vec=0.5
limits=correlation.limits(n.BB,n.NN,prop.vec,coef.mat=NULL)

## End(Not run)

fleishman.coef

Computes the coefficients of Fleishman third order polynomials

Description

Computes the coefficients of Fleishman third order polynomials given the marginal skewness and kurtosis parameters of continuous variables.

Usage

fleishman.coef(n.NN, skewness.vec = NULL, kurtosis.vec = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n.NN</td>
<td>Number of continuous non-normal variables.</td>
</tr>
<tr>
<td>skewness.vec</td>
<td>Skewness vector for continuous non-normal variables.</td>
</tr>
<tr>
<td>kurtosis.vec</td>
<td>Kurtosis vector for continuous non-normal variables.</td>
</tr>
</tbody>
</table>

Details

The execution of the function may take some time since it uses multiple starting points to solve the system of nonlinear equations based on the third order Fleishman polynomials. However, since users need to run it only once for a given set of specifications, it does not constitute a problem.
Value

A matrix of coefficients. The columns represent the variables and rows represent the corresponding a, b, c, and d coefficients.

References


See Also

validation.skewness.kurtosis

Examples

```r
## Not run:
#Consider four nonnormal continuous variables, which come from
#Exp(1), Beta(4,4), Beta(4,2) and Gamma(10,10), respectively.
#Skewness and kurtosis values of these variables are as follows:
#n.NN=4
skewness.vec=c(2,0,-0.4677,0.6325)
kurtosis.vec=c(6,-0.5455,-0.3750,0.6)
coef.mat=fleishman.coef(n.NN,skewness.vec,kurtosis.vec)

n.NN=1
skewness.vec=c(0)
kurtosis.vec=c(-1.2)
coef.mat=fleishman.coef(n.NN,skewness.vec,kurtosis.vec)

n.NN=1
skewness.vec1=c(3)
kurtosis.vec1=c(5)
coef.mat=fleishman.coef(n.NN,skewness.vec1,kurtosis.vec1)
## End(Not run)
```

gen.Bin.NonNor

Simulates a sample of size n from a set of multivariate binary and continuous non-normal variables
Description

This function simulates a multivariate data set with binary and continuous components with pre-specified marginals and a correlation matrix. Setting n.NN=0 and quantities that are pertinent to the continuous part to NULL results in simulation of a sample of size n from a set of multivariate binary variables. Similarly, setting n.BB=0 and prop.vec=NULL results in simulation of a sample of size n from a set of multivariate continuous non-normal variables.

Usage

```r
gen.Bin.NonNor(n, n.BB, n.NN, prop.vec = NULL, mean.vec = NULL, variance.vec = NULL, skewness.vec = NULL, kurtosis.vec = NULL, final.corr.mat, coef.mat = NULL)
```

Arguments

- **n**: Number of variates.
- **n.BB**: Number of binary variables.
- **n.NN**: Number of continuous non-normal variables.
- **prop.vec**: Probability vector for binary variables.
- **mean.vec**: Mean vector for continuous non-normal variables.
- **variance.vec**: Variance vector for continuous non-normal variables.
- **skewness.vec**: Skewness vector for continuous non-normal variables.
- **kurtosis.vec**: Kurtosis vector for continuous non-normal variables.
- **final.corr.mat**: Final correlation matrix produced from `overall.corr.mat`.
- **coef.mat**: Matrix of coefficients produced from `fleishman.coef`.

Value

A matrix of size n*(n.BB + n.NN) of which the first n.BB columns are binary variables and the last n.NN columns are continuous variables.

References


See Also

`validation.bin`, `validation.skewness.kurtosis`, `overall.corr.mat`, `fleishman.coef`
Examples

## Not run:

```r
n = 1
n.BB = 2
n.NN = 4
prop.vec = c(0.4, 0.7)
mean.vec = c(1, 0.5, 4/6, 100)
variance.vec = c(1, 0.02777778, 0.03174603, 1000)
skewness.vec = c(2, 0, -0.4677, 0.6325)
kurtosis.vec = c(6, -0.5455, -0.3750, 0.6)
corr.mat = matrix(c(1, -0.3, -0.3, -0.3, -0.3, -0.3,
                    -0.3, 1, 0.4, 0.5, 0.6,
                    -0.3, 0.4, 1, 0.7, 0.8,
                    -0.3, 0.5, 0.7, 1, 0.9,
                    -0.3, 0.6, 0.8, 0.9, 1), 6, byrow = TRUE)
```

```r
coef.mat = fleishman.coef(n.NN, skewness.vec, kurtosis.vec)
```

```r
coef.mat = matrix(c(-0.31375, 0.00000, 0.10045, -0.10448,
                    0.82632, 1.08574, 1.10502, 0.98085,
                    0.31375, 0.00000, -0.10045, 0.10448,
                    0.02271, -0.02945, -0.04001, 0.00272), 4, byrow = TRUE)
```

```r
intcor.mat = Int.Corr.NN(n.NN, corr.vec = NULL, corr.mat, coef.mat)
```

```r
intcor.mat = matrix(c(1.0000000, 0.4487800, 0.5940672, 0.6471184,
                       0.4487800, 1.0000000, 0.709443, 0.8112701,
                       0.5940672, 0.709443, 1.0000000, 0.9436195,
                       0.6471184, 0.8112701, 0.9436195, 1.0000000), 4, byrow = TRUE)
```

```r
tetcor.mat = Tetra.Corr.BB(n.BB, prop.vec, corr.vec = NULL, corr.mat)
tetcor.mat = matrix(c(1.0000000, -0.4713861,
                      -0.4713861, 1.0000000), 2, byrow = TRUE)
```

```r
bicor.mat = Biserial.Corr.BN(n.BB, n.NN, prop.vec, corr.vec = NULL, corr.mat, coef.mat)
bicor.mat = matrix(c(-0.4253059, -0.3814058, -0.3862068, -0.3846430,
                     -0.4420613, -0.3964317, -0.4014219, -0.3997964), 2, byrow = TRUE)
```

```r
final.corr.mat = overall.corr.mat(n.BB, n.NN, prop.vec, corr.vec = NULL, corr.mat, coef.mat)
```

```r
final.corr.mat = matrix(c(1.0000000, -0.4713861, -0.4253059, -0.3814058, -0.3862068, -0.3846430,
                          -0.4713861, 1.0000000, -0.4420613, -0.3964317, -0.4014219, -0.3997964,
                          -0.4253059, -0.4420613, 1.0000000, 0.4487800, 0.5940672, 0.6471184,
                          -0.3814058, -0.3964317, 0.4487800, 1.0000000, 0.709443, 0.8112701,
                          -0.3862068, -0.4014219, 0.5940672, 0.709443, 1.0000000, 0.9436195,
                          -0.3846430, -0.3997964, 0.6471184, 0.8112701, 0.9436195, 1.0000000), 6, byrow = TRUE)
```
data = gen.Bin.NonNor(n,n.BB,n.NN,prop.vec,mean.vec,variance.vec,skewness.vec,
kurtosis.vec,final.corr.mat,coef.mat)

amat = final.corr.mat[1:2, 1:2]
multibin = gen.Bin.NonNor(n=1000,n.BB,n.NN=0,prop.vec,mean.vec=NULL,variance.vec=NULL,
  skewness.vec=NULL,kurtosis.vec=NULL,final.corr.mat=amat,coef.mat=NULL)

apply(multibin,2,mean)

multinonnor = gen.Bin.NonNor(n=100,n.BB=0,n.NN,prop.vec=NULL,mean.vec,variance.vec,
  skewness.vec,kurtosis.vec,final.corr.mat=bmat,coef.mat)
apply(multinonnor,2,mean)
apply(multinonnor,2,var)

n=1000
n.BB=1
n.NN=1
prop.vec=0.6
mean.vec=1
variance.vec=1
skewness.vec=2
kurtosis.vec=6
corr.vec=NULL
corr.mat=matrix(c(1,-0.3,-0.3,1),2,2)
coef.mat=matrix(c(-0.31375,0.82632,0.31375,0.02271),4,1)
final.corr.mat=overall.corr.mat(n.BB,n.NN,prop.vec,corr.vec=NULL,corr.mat,coef.mat)
data = gen.Bin.NonNor(n,n.BB,n.NN,prop.vec,mean.vec,variance.vec,skewness.vec,
kurtosis.vec,final.corr.mat,coef.mat)

n=1000
n.BB=1
n.NN=0
prop.vec=0.6
mean.vec=1
variance.vec=NULL
skewness.vec=NULL
kurtosis.vec=NULL
corr.vec=NULL
corr.mat=diag(1)
coef.mat=NULL

final.corr.mat=overall.corr.mat(n.BB,n.NN,prop.vec,corr.vec=NULL,corr.mat,coef.mat)
data = gen.Bin.NonNor(n,n.BB,n.NN,prop.vec,mean.vec,variance.vec,skewness.vec,
kurtosis.vec,final.corr.mat,coef.mat)

## End(Not run)
**Int.Corr.NN**

Computes an intermediate correlation matrix for continuous non-normal variables given the specified correlation matrix.

**Description**

This function computes the intermediate correlation matrix for continuous non-normal-continuous non-normal combinations as formulated in Demirtas et al. (2012).

**Usage**

Int.Corr.NN(n.NN, corr.vec = NULL, corr.mat = NULL, coef.mat)

**Arguments**

- **n.NN**: Number of continuous non-normal variables.
- **corr.vec**: Vector of elements below the diagonal of correlation matrix ordered column-wise.
- **corr.mat**: Specified correlation matrix.
- **coef.mat**: Matrix of coefficients produced from `fleishman.coef`.

**Value**

A correlation matrix of size n.NN\*n.NN.

**References**


**See Also**


**Examples**

```r
n.NN=4
corr.vec=NULL
corr.mat=matrix(c(1.0,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,1.0,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,1.0,0.4,0.5,0.6,-0.3,-0.3,0.4,1.0,0.7,0.8,-0.3,-0.3,0.5,0.7,1.0,0.9,-0.3,-0.3,0.6,0.8,0.9,1.0),6,byrow=TRUE)
coef.mat=matrix(c(-0.31375, 0.00000, 0.10045, -0.10448, 0.82632, 1.08574, 1.10502, 0.98085, -0.31375, 0.00000, 0.10045, -0.10448, 0.82632, 1.08574, 1.10502, 0.98085),4,byrow=TRUE)
```
**overall.corr.mat**

0.31375, 0.00000, -0.10045, 0.10448, 
0.02271, -0.02945, -0.04001, 0.00272, 4, byrow=TRUE)

intcor.mat=Int.Corr.NN(n.NN,corr.vec=NULL,corr.mat,coef.mat)

---

**Description**

This function computes the final correlation matrix by combining tetrachoric correlation for binary-binary combinations, biserial correlations for binary-continuous combinations, and intermediate correlation matrix for continuous-continuous combinations.

**Usage**

```r
overall.corr.mat(n.BB, n.NN, prop.vec = NULL, corr.vec = NULL, corr.mat = NULL, 
coef.mat = NULL)
```

**Arguments**

- `n.BB`: Number of binary variables.
- `n.NN`: Number of continuous non-normal variables.
- `prop.vec`: Probability vector for binary variables.
- `corr.vec`: Vector of elements below the diagonal of correlation matrix ordered column-wise.
- `corr.mat`: Specified correlation matrix.
- `coef.mat`: Matrix of coefficients produced from `fleishman.coef`.

**Value**

A matrix of size `(n.BB+n.NN)*(n.BB+n.NN)`.

**References**


**See Also**

Examples

```r
n.BB=2
n.NN=4
prop.vec=c(0.4,0.7)
corr.vec=NULL
corr.mat=matrix(c(1.0,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,1.0,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,0.4,0.5,0.6,-0.3,-0.3,0.4,1.0,0.7,0.8,-0.3,-0.3,0.5,0.7,1.0,0.9,-0.3,-0.3,0.6,0.8,0.9,1.0),6,byrow=TRUE)

coef.mat=matrix(c(-0.31375, 0.00000, 0.10045, -0.10448,
                   0.82632, 1.08574, 1.10502, 0.98085,
                   0.31375, 0.00000, -0.10045, 0.10448,
                   0.02271, -0.02945, -0.04001, 0.00272),4,byrow=TRUE)

final.corr.mat=overall.corr.mat(n.BB,n.NN,prop.vec,corr.vec=NULL,corr.mat,coef.mat)

corr.mat.BB=corr.mat[1:2,1:2]
final.corr.mat=overall.corr.mat(n.BB,n.NN=0,prop.vec,corr.vec=NULL,corr.mat=corr.mat.BB,coef.mat=NULL)

final.corr.mat=overall.corr.mat(n.BB=0,n.NN,prop.vec=NULL,corr.vec=NULL,corr.mat=corr.mat.NN,coef.mat)
```

Tetra.Corr.BB

Computes the tetrachoric correlation matrix for binary variables given the specified correlation matrix

Description

This function computes the tetrachoric correlation matrix for binary-binary combinations as formulated in Demirtas et al. (2012).

Usage

Tetra.Corr.BB(n.BB, prop.vec, corr.vec = NULL, corr.mat = NULL)
validation.bin

Arguments

- **n.BB**: Number of binary variables.
- **prop.vec**: Probability vector for binary variables.
- **corr.vec**: Vector of elements below the diagonal of correlation matrix ordered column-wise.
- **corr.mat**: Specified correlation matrix.

Value

A correlation matrix of size n.BB*n.BB.

References


See Also


Examples

```r
n.BB=2
prop.vec=c(0.4,0.7)
corr.vec=NULL
corr.mat=matrix(c(1.0,-0.3,-0.3,-0.3,-0.3,-0.3,
                 -0.3,1.0,-0.3,-0.3,-0.3,-0.3,
                 -0.3,-0.3,1.0,0.4,0.5,0.6,
                 -0.3,-0.3,0.4,1.0,0.7,0.8,
                 -0.3,-0.3,0.5,0.7,1.0,0.9,
                 -0.3,-0.3,0.6,0.8,0.9,1.0),6,by=TRUE)
tetcor.mat=Tetra.Corr.BB(n.BB,prop.vec,corr.vec=NULL,corr.mat)
```

validation.bin: Validates the marginal specification of the binary variables

Description

Checks whether the marginal specification of the binary part is valid and consistent.

Usage

validation.bin(n.BB, prop.vec = NULL)
validation.corr

Validates the specified correlation matrix

Description

This function validates the specified correlation vector and/or matrix for appropriate dimension, symmetry, range, and positive definiteness. If both correlation matrix and correlation vector are supplied, it checks whether the matrix and vector are conformable.
validation.corr

Usage

validation.corr(n.BB, n.NN, corr.vec = NULL, corr.mat = NULL)

Arguments

n.BB Number of binary variables.
n.NN Number of continuous non-normal variables.
corr.vec Vector of elements below the diagonal of correlation matrix ordered column-wise.
corr.mat Specified correlation matrix.

Value

The function returns TRUE if no specification problem is encountered. Otherwise, it returns an error message.

See Also

correlation.limits, correlation.bound.check

Examples

n.BB=2
n.NN=4
corr.vec=NULL
corr.mat=matrix(c(1.0,-0.3,-0.3,-0.3,-0.3,-0.3,
                   -0.3,1.0,-0.3,-0.3,-0.3,-0.3,
                   -0.3,-0.3,1.0,0.4,0.5,0.6,
                   -0.3,-0.3,0.4,1.0,0.7,0.8,
                   -0.3,-0.3,0.5,0.7,1.0,0.9,
                   -0.3,-0.3,0.6,0.8,0.9,1.0),6,byrow=TRUE)

validation.corr(n.BB,n.NN,corr.vec=NULL,corr.mat)

n.BB=2
n.NN=4
corr.vec=c(-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,-0.3,0.4,0.5,0.6,0.7,0.8,0.9)
validation.corr(n.BB,n.NN,corr.vec,corr.mat=NULL)

## Not run:

n.BB=0
n.NN=4
validation.corr(n.BB,n.NN,corr.vec=NULL,corr.mat)

n.BB=2
n.NN=0
validation.corr(n.BB,n.NN=0,corr.vec=NULL,corr.mat)

validation.corr(n.BB=0,n.NN=4,corr.vec=NULL,corr.mat=corr.matc)
validation.skewness.kurtosis

Validates the marginal specification of the continuous non-normal variables

Description

Checks whether the marginal specification of the continuous non-normal part is valid and consistent.

Usage

validation.skewness.kurtosis(n.NN, skewness.vec = NULL, kurtosis.vec = NULL)

Arguments

- **n.NN**: Number of continuous non-normal variables.
- **skewness.vec**: Skewness vector for continuous non-normal variables.
- **kurtosis.vec**: Kurtosis vector for continuous non-normal variables.

Value

The function returns TRUE if no specification problem is encountered. Otherwise, it returns an error message.
validation.skewness.kurtosis

References

Examples

```r
n.NN<-3
skewness.vec=c(0,2,3)
kurtosis.vec=c(-1.2,6,8)
validation.skewness.kurtosis(n.NN,skewness.vec,kurtosis.vec)

## Not run:
n.NN<-1
skewness.vec=c(0)
kurtosis.vec=c(-1.2)
validation.skewness.kurtosis(n.NN,skewness.vec,kurtosis.vec)

n.NN<-3
skewness.vec=c(0,2,3)
kurtosis.vec=c(-1.2,6,5)
validation.skewness.kurtosis(n.NN,skewness.vec)
validation.skewness.kurtosis(n.NN,kurtosis.vec)

n.NN<-3
skewness.vec=c(0,2,3)
kurtosis.vec=c(-1.2,6,5)
validation.skewness.kurtosis(n.NN,skewness.vec)
validation.skewness.kurtosis(n.NN,kurtosis.vec)

n.NN<0
skewness.vec=c(0,2,3)
kurtosis.vec=c(-1.2,6,8)
validation.skewness.kurtosis(n.NN,skewness.vec)
validation.skewness.kurtosis(n.NN,kurtosis.vec)

n.NN<-2
skewness.vec=c(0,2,3)
kurtosis.vec=c(-1.2,6,8)
validation.skewness.kurtosis(n.NN,skewness.vec)
validation.skewness.kurtosis(n.NN,kurtosis.vec)

n.NN<-2
skewness.vec=c(0,2,3)
kurtosis.vec=c(-1.2,6)
validation.skewness.kurtosis(n.NN,skewness.vec)
validation.skewness.kurtosis(n.NN,kurtosis.vec)

skewness.vec=c(2,3)
kurtosis.vec=c(1,5)
validation.skewness.kurtosis(n.NN,skewness.vec)

## End(Not run)
```
Index

*Topic concurrent generation of binary and continuous non-normal variables
BiNonNor-package, 2

*Topic generating multivariate binary variables
BiNonNor-package, 2

*Topic generating multivariate continuous non-normal variables
BiNonNor-package, 2

BiNonNor (BiNonNor-package), 2
BiNonNor-package, 2
Biserial.Corr.BN, 2, 3, 12, 13, 15

correlation.bound.check, 2, 4, 6, 17
correlation.limits, 2, 5, 6, 17
fleishman.coef, 2–6, 7, 9, 12, 13

gen.Bin.NonNor, 2, 8

Int.Corr.NN, 2, 3, 12, 13

overall.corr.mat, 2, 3, 9, 12, 13, 15

Tetra.Corr.BB, 2, 3, 12, 13, 14, 15

validation.bin, 2, 9, 15
validation.corr, 2, 5, 16
validation.skewness.kurtosis, 2, 8, 9, 18