Package ‘rbmn’

February 20, 2015

Version 0.9-2
Date 2013-08-01
Type Package
Title Handling Linear Gaussian Bayesian Networks
Author Jean-Baptiste Denis <Jean-Baptiste.Denis@jouy.inra.fr>
Maintainer Jean-Baptiste Denis <Jean-Baptiste.Denis@jouy.inra.fr>
Description Creation, manipulation, simulation of linear Gaussian Bayesian networks from text files and more...
License GPL (>= 2)
Depends MASS
Suggests bnlearn, igraph
NeedsCompilation no
Repository CRAN
Date/Publication 2013-08-01 15:29:29

R topics documented:

rbmn-package ................................................................. 3
adja2arcs ................................................................. 5
adja2crossed .............................................................. 6
adja2nbn ................................................................. 6
adja4nbn ................................................................. 7
adja4three ............................................................... 7
arc7nb4nbn ............................................................... 8
arc4nbn1nbn ............................................................. 9
bn2nbn ................................................................. 9
bnsfit2nbn .............................................................. 10
body composition ........................................................ 10
chain2correlation ......................................................... 11
chain2gema ............................................................. 12
chain2mn ............................................................... 13
chain2nbn ............................................................... 13
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>chain2pre</td>
</tr>
<tr>
<td>chain4chain</td>
</tr>
<tr>
<td>check8chain</td>
</tr>
<tr>
<td>check8gema</td>
</tr>
<tr>
<td>check8nbn</td>
</tr>
<tr>
<td>condi4joint</td>
</tr>
<tr>
<td>cor4var</td>
</tr>
<tr>
<td>crossed4nbn1nbn</td>
</tr>
<tr>
<td>dev4mn</td>
</tr>
<tr>
<td>diff8nbn</td>
</tr>
<tr>
<td>estimate8constrainednbn</td>
</tr>
<tr>
<td>estimate8nbn</td>
</tr>
<tr>
<td>gema2mn</td>
</tr>
<tr>
<td>gema2nbn</td>
</tr>
<tr>
<td>generate8chain</td>
</tr>
<tr>
<td>generate8nbn</td>
</tr>
<tr>
<td>inout4chain</td>
</tr>
<tr>
<td>is8nbn8chain</td>
</tr>
<tr>
<td>marginal4chain</td>
</tr>
<tr>
<td>mn2gema</td>
</tr>
<tr>
<td>mn4joint1condi</td>
</tr>
<tr>
<td>nb8bn</td>
</tr>
<tr>
<td>nbn2bnfit</td>
</tr>
<tr>
<td>nbn2chain</td>
</tr>
<tr>
<td>nbn2gema</td>
</tr>
<tr>
<td>nbn2mn</td>
</tr>
<tr>
<td>nbn2nbn</td>
</tr>
<tr>
<td>nbn2rr</td>
</tr>
<tr>
<td>nbn4nbn</td>
</tr>
<tr>
<td>nbn4rmatrix</td>
</tr>
<tr>
<td>normalize8nbn</td>
</tr>
<tr>
<td>order4chain</td>
</tr>
<tr>
<td>order4gema</td>
</tr>
<tr>
<td>order4nbn</td>
</tr>
<tr>
<td>print8chain</td>
</tr>
<tr>
<td>print8gema</td>
</tr>
<tr>
<td>print8mn</td>
</tr>
<tr>
<td>print8nbn</td>
</tr>
<tr>
<td>provided objects</td>
</tr>
<tr>
<td>reverse8chain</td>
</tr>
<tr>
<td>rm8nd4adja</td>
</tr>
<tr>
<td>rm8nd4nbn</td>
</tr>
<tr>
<td>rmatrix4nbn</td>
</tr>
<tr>
<td>simulate8gema</td>
</tr>
<tr>
<td>simulate8gmn</td>
</tr>
<tr>
<td>simulate8mn</td>
</tr>
<tr>
<td>simulate8nbn</td>
</tr>
<tr>
<td>state4chain</td>
</tr>
</tbody>
</table>
rbmn-package

**Description**

General functions to generate, transform, display general and particular linear Gaussian Bayesian networks \[/nbn/\] are provided.

Specific \[/nbn/\] are chain and crossed \[/nbn/\]. Focus is given in getting joint and conditional probability distributions of the set of nodes.

*rbmn* stands for *R’eseau Bay’esien MultiNormal*.

**Details**

Some basic concepts:

- **chain /nbn/s** are /nbn/s where all nodes are connected with two other nodes, except the two ending nodes of the chain having only one connection. (This is not the usual terminology in graphical models but I didn’t find a more appropriate word: suggestions are welcome.)
- **crossed /nbn/s** are /nbn/s having the node set defined as a Cartesian product of two series of items, and a DAG based on this structure. See the *crossed4nbn1nbn* function and/or Tian (2013) for details.
- An **adjacency matrix** is a matrix equivalent to the DAG associated to a /nbn/. Its rows as well as its columns are associated to the set of nodes. The \((i, j)\) cell is one when there is an arc going from node \(i\) to \(j\) and zero otherwise.

Three equivalent ways can be used to represent the joint probability distribution of a set of nodes respectively associated to the structures /mn/, /nbn/ and /gema/:

- **/mn/** (for multivariate normal) is just the list of the expectation \((\mu)\) and the variance matrix \((\Sigma)\).
- **/nbn/** (for normal Bayesian network) is a simple list, a component a node described with a list. The names are node names and each list associated to a node provides the conditional expectation and variance, the parent (if any) and the associated regression coefficients.
- **/gema/** (for generating matrices) is a list of a vector \((\mu)\) and a matrix \((\Lambda)\) such that the vector of the nodes can be defined by \(X = \mu + \Lambda E\) where \(E\) is a normal random vector with expectation zero and variance matrix unity.
- It is planned to add a fourth one under the name of /gbn/.

To relieve the memory effort, most names of the functions have been given a two (or more) components structure separated with a figure. This idea will be explained and exploited in a package to come named *documair*. The approximate meaning of the figures are:

- 0 (similar to ‘o’) *rbmn0chain.01* to indicate an object example provided by *rbmn*. 

**Index**

| rbmn-package | Linear Gaussian Bayesian network manipulations | 49 |
A number of ancillary functions have not been exported to give a better access to the main function of \texttt{rbmn}. Nevertheless they are available in the \texttt{NN/rbmn/r} directory, and with all their comments (equivalent to Rd files into \texttt{NN/rbmn/inst/original/} directory). Some of them are visible when defining the default arguments of some functions.

**Projected evolution of /mn/**

- Generalize the /mn/ object with a regression part like the output of function \texttt{condiTjoint} when argument \texttt{pour} is not of length zero and argument \texttt{x2} is not null. With such a structure, every node of a \texttt{nbn/} could be described with a /mn/ comprising a unique variable... Also the two arguments of function \texttt{mnTjoint1condi} would be just two /mn/ objects... This is also the generalized /mn/ proposed in function \texttt{simulateXgmn} under the argument of \texttt{loi}... Of course almost all functions dealing with \texttt{nbn/} objects will be to rewrite!
- Introduce a new object \texttt{gbn} for Gaussian Bayesian network similar to the list provided by function \texttt{nbnRrr}.

**TO DO list**

- Systemize the existence of \texttt{check8object} functions
- Introduce their systematic use conditioned with a \texttt{rbmn0check} variable.
- Follow the main checking of every functions
- Give (and use) class attributes to the main objects.
- Introduce the main objects in this short presentation.
- Make a true small example in this short presentation.
- Make the function \texttt{nbn4string7dag}.
- Add the computation made with \texttt{bnlearn/} in the example of \texttt{estimateXnbn}.
- Check the topological order within \texttt{nbn2nbn} depending on \texttt{rbmn0check} value.
- Make a super transformation function from an object associated to a Bayesian network to any other type, including itself.
- Correct the \texttt{ord} option in \texttt{order4chain}.
- Check the topological order in \texttt{rm8nd4adja}.
- Think about removing all \texttt{rmatrix} transformations to the benefit of the to-come \texttt{gbn} object.
- Introduce a check of non-negativity of \texttt{ma} into \texttt{cor4var}.
- Add examples to all functions without any.
**adja2arcs**

**Author(s)**

Jean-Baptiste Denis  
MIAj - Inra - Jouy-en-Josas  
F-78532 Jouy-en-Josas  
Maintainer: Jean-Baptiste Denis <Jean-Baptiste.Denis@Jouy.Inra.Fr>

**References**

(A technical report presenting the concepts used in `rbmn` is under redaction; it can be obtained as it is if asked.)


Tian S, Scutari M & Denis J-B (2013, submitted to JSFdS). "Predicting with Crossed Linear Gaussian Bayesian Networks".

**Examples**

```r
library(rbmnn)

## getting the data set
data(boco)
print(head(boco));
```

---

**adja2arcs**  
*Arc matrix from an adjacency matrix*

**Description**

returns the arc matrix from an adjacency matrix.

**Usage**

```r
adja2arcs(adj)
```

**Arguments**

- `adj`  
The adjacency matrix.

**Value**

a matrix with two columns ("from","to")

**Examples**

```r
adja2arcs(rbmnn@adja.02)
```
adja2crossed creates a crossed-adjacency matrix from two ones

Description
Like crossed4nbn1nbn but at the level of adjacency matrices. Must be much efficient when regression coefficients are not needed.

Usage
adja2crossed(adj1, adj2, nona=as.vector(outer(dimnames(adj1)[[1]], dimnames(adj2)[[1]], paste, sep="_")))

Arguments
adj1 The first adjacency matrix.
adj2 The second adjacency matrix.
nona The node names to give to the crossed /nbn/, the nodes of the nbn1 varying first.

Details
Just two Kronecker products of matrices.

Value
The resulting crossed adjacency matrix.

Examples
print(adja2crossed(rbnm0adja.01, rbnm0adja.01));

adja2nbn standardized /nbn/ from an adjacency matrix

Description
returns a nbn object with 0/1 regression coefficients having adja as adjacency matrix.

Usage
adja2nbn(adja)

Arguments
adja The initial adjacency matrix.
Value

The corresponding standardized nbn object.

Examples

```r
print8nbn(adjja2nbn(adjja4nbn(rbmn0nbn.03)));
```

---

```
adjja4nbn          adjacency matrix of a /nbn/
```

---

Description

returns a dimnamed matrix indicating with 1 an arc from row to column nodes (0 everywhere else);

i.e. the adjacency matrix.

Usage

```r
adjja4nbn(nbn)
```

Arguments

- `nbn` The initial nbn object.

Value

A dimnamed matrix

Examples

```r
adjja4nbn(rbmn0nbn.04);
```

---

```
adjja4three         Adjacency matrices of DAGs having three nodes
```

---

Description

Returns the list of the 25 adjacency matrices associated to DAGs comprising three nodes. The first

character of the name components gives the number of arcs in the DAG.

Usage

```r
adjja4three(nona=LETTERS[1:3])
```

Arguments

- `nona` The three node names.
Details

Poor filling...

Value

a named list having 25 components, each being a 3x3 matrix.

\[
\text{arc\textunderscore nbn4nbn} \quad \text{return the number(s) of arcs of a /nbn/}
\]

Description

returns the arc numbers of the node of /nbn/ object.

Usage

\[\text{arc\textunderscore nbn4nbn(nbn, each=FALSE)}\]

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nbn</td>
<td>The nbn object to consider.</td>
</tr>
<tr>
<td>each</td>
<td>When TRUE, returns a named vector of the number of parents of each node. If not the total number of arcs.</td>
</tr>
</tbody>
</table>

Details

Parents associated with a zero regression coefficient are not excluded in the counting.

Value

Either a number or a named vector of numbers (names being the node names).

Examples

\[\text{arc\textunderscore nbn4nbn(rbmn\textunderscore nbn.05)};\]
arcs4nbn1nbn

returns the list of 'parallel' arcs of a crossed-nbn

Description

Returns a list of matrices with two columns (as needed by estimate8constrainednbn) indicating corresponding arcs for each arcs/nodes of nbn1 (or nbn2) of the crossed /nbn/ obtained when crossing /nbn1/ and /nbn2/ with node names given by nona.

Usage

arcs4nbn1nbn(nbn1, nbn2, type="a1", nona=as.vector(outer(names(nbn1), names(nbn2), paste, sep="_")))

Arguments

- nbn1: The first generating /nbn/.
- nbn2: The second generating /nbn/.
- type: Must be "a1" to indicate that the parallelism must be done for each arc of nbn1. "a2" for each arc of nbn2. Or "n1" for each node of nbn1. Or "n2" for each node of nbn2.
- nona: The node names to give to the crossed /nbn/, the nodes of the nbn1 varying first.

Value

The resulting named (after node names) list of matrices.

Examples

print(arcs4nbn1nbn(rbmn0nbn.01, rbmn0nbn.04));

bn2nbn

transforms a /bn/ of /bnlearn/ package to a /nbn/

Description

returns a nbn object from a DAG (bn object) of /bnlearn/ package. O and 1 coefficients are introduced...

Usage

bn2nbn(bn)

Arguments

- bn: The object to be transformed.
Value

A list following the nbn specification

bnfit2nbn

transforms a bn.fit of bnlearn package to a nbn/

Description

returns a nbn object from a Gaussian bn.fit object of bnlearn package.

Usage

bnfit2nbn(bn.fit)

Arguments

bn.fit The object to be transformed.

Details

If bn.fit is not pertinent, a fatal error is issued.

Value

A list following the nbn specification

body composition

Body Composition Variables and Covariables

Description

Real-world data set extracted from the Nhanes data base comprising nine variables describing the body composition and five easy measurable covariables.

Usage

data(boco)
Format

The boco data set stored in variable boco comprises 100 individuals with the following variables:

- A the age in years
- H the height in cm
- W the weight in kg
- C the waist circumference in cm
- TF the trunk fat in kg
- LF the leg fat in kg
- AF the arm fat in kg
- TL the trunk lean in kg
- LL the leg lean in kg
- AL the arm lean in kg
- TB the trunk bone in kg
- LB the leg bone in kg
- AB the arm bone in kg

Source


Examples

```r
# load the data and build the correct network from the model string.
data(boco);
print(head(boco));
boco7dag <- "[H][W|H][TF|W;H];"
# to be finished
```

chain2correlation

*computes the correlation matrix of a chain*

Description

returns the correlation matrix of a `chain` object.

Usage

chain2correlation(chain)
Arguments

chain The chain object to consider.

Value

The correlation matrix. It is not sorted to respect a topological order contrary to chain2mn function.

Examples

chain2correlation(rbmn0chain.03); 

chain2gema transforms a /chain/ to a /gema/

Description

From a chain object returns the gema using a direct formulae. Much precised than to use the /nbn/ way.

Usage

chain2gema(chain)

Arguments

chain the chain object to be transformed.

Value

The corresponding gema object.

Examples

identical(chain2gema(rbmn0chain.02)$mu, rbmn0gema.02$mu);
print(chain2gema(rbmn0chain.02)$l1-rbmn0gema.02$l1);
chain2mn

computes the distribution of a chain

Description
returns the /mn/ object associated to a /chain/ object. Much better to use this function that the general function nbn2mn since exact formulae are applied.

Usage
chain2mn(chain, order=TRUE)

Arguments
chain The chain object to consider.
order Must a topological order be imposed?

Value
The resulting /mn/ object. Following the convention of mn objects, a topological order is given to it. This is necessary to retrieve the associate /nbn/.

Examples
print8mn(chain2mn(rbmn0chain.01));

chain2nbn transforms a /chain/ to a /nbn/

Description
From a chain object returns the nbn translation.

Usage
chain2nbn(chain)

Arguments
chain the chain object to be transformed.

Value
The corresponding nbn object.

Examples
print8nbn(chain2nbn(rbmn0chain.02), ordering=names(rbmn0nbn.02));
chain2pre

*computes the precision of a chain*

**Description**

returns the precision matrix of a chain, that is the inverse of its variance (correlation) matrix. Much better to use this function that `solve(chain2mn(chain)$gamma)` since exact formulae are applied.

**Usage**

`chain2pre(chain, corre=FALSE)`

**Arguments**

- **chain**
  The chain object to consider.
- **corre**
  To get the inverse of the correlation matrix instead of.

**Value**

A dimnamed matrix

**Examples**

```r
chain2pre(rbmn0chain.02);
```

---

chain4chain

*extracts a chain from a chain*

**Description**

returns the chain obtained from chain retaining only nodes indicated by nodes and conditioned with nodes indicated in condi.

**Usage**

`chain4chain(chain, nodes, condi=numeric(0), value=rep(0, length(condi)))`

**Arguments**

- **chain**
  The chain object to consider.
- **nodes**
  numeric (or character) vector giving the numbers (or names) of the nodes to be retained in the extracted chain.
- **condi**
  numeric (or character) vector giving the numbers (or names) of the conditioning nodes for the extracted chain.
- **value**
  Numerical values associated to condi.
Details
Integration is done for nodes not belonging to the extracted chain nor being in the conditioning subset. Then the distribution of the retained nodes is left identical to this in the initial chain.

Value
The resulting chain

Examples
```
chain4chain(rbmn0chain.02, c("a", "d"), c("b"), 12);
```

check8chain checks a /chain/ object

Description
checks the consistency of chain as a /chain/ object issues a fatal error with some clues if inconsistent.

Usage
```
check8chain(chain)
```

Arguments
chain The chain object to check.

Details
Looking a the code of this function provides a way to know which are the requirements of a /chain/ object.

Value
TRUE or a character containing some clue about the discovered inconsistency.

Examples
```
check8chain(rbmn0chain.01);
res <- check8chain(rbmn0adja.01);
if (is.na(as.logical(res))) { print(res);}
```
check8gema  checks a /gema/ object

Description
checks the consistency of gema as a /gema/ object issues a fatal error with some clues if inconsistent.

Usage
check8gema(gema)

Arguments
  gema  The gema object to check.

Details
Looking a the code of this function provides a way to know which are the requirements of a /chain/ object.

Value
TRUE or a character containing some clue about the discovered inconsistency.

Examples
check8gema(rbmn0gema.01);
res <- check8gema(rbmn0adja.01);
if (is.na(as.logical(res))) { print(res);}

check8nbn  checks a /nbn/ object

Description
checks the consistency of nbn as a /nbn/ object issues a fatal error with some clues if inconsistent.

Usage
check8nbn(nbn)

Arguments
  nbn  The nbn object to check.
Details

Looking at the code of this function provides a way to know which are the requirements of a /chain/ object.

Value

TRUE or a character containing some clue about the discovered inconsistency.

Examples

```r
check8nbv(rbmn@nbv.01);
res <- check8nbv(rbmn@adjv.01);
if (is.na(as.logical(res))) { print(res); }
```

condi4joint computes some conditional distribution of a multinormal vector

Description

returns the expectation and variance of a sub-vector conditionned with another (non overlapping) sub-vector from an initial random vector described by mn.

Usage

```r
condi4joint(mn, par, pour, x2=rep(0, length(pour)))
```

Arguments

- **mn**: list defining the distribution of the initial vector with $\mu$, its expectation, and $\gamma$, its variance matrix.
- **par**: names (or indices) of the sub-vector to give the distribution.
- **pour**: names (or indices) of the conditionning sub-vector (can be NULL when non conditioning).
- **x2**: values to consider for the conditioning sub-vector. When NULL the general form is supplied, not a /mn/ object.

Details

when no names are given to mn$\mu$, par and pour are supposed containing indices and default sequential names are provided.
Value

A list:
when $x_2$ provides the values taken by the conditioning part, it is a /mn/ object with its two components: $\mu$ for the expectation vector and $\gamma$ for the variance matrix.
when $x_2$ is NULL the list has got three components: $a$ for the fixed part of the expectation vector, $b$ for the regression coefficients to be associated to the non precised $x_2$ values, varying part of the expectation and $\gamma$ for the variance matrix.

Examples

```
print(mn(condijoint(rbmn@mn.04, c("1.1", "2.2", "1.2", "2.1"), NULL));
print(mn(condijoint(rbmn@mn.04, c("1.1", "2.2", "1.2", "2.1"), c", 0));
print(condijoint(rbmn@mn.04, c("1.1", "2.2", "1.2", "2.1"), "c", NULL));
```

---

cor4var returns the correlation matrix from the variance

Description

returns the correlation matrix from the variance preserving possible variable names

Usage

cor4var(ma)

Arguments

ma The variance matrix.

Details

Zero variances are detected and accepted (all associated correlation coefficients are forced to be zero.>>

Value

The correlation matrix

Examples

cor4var(rbmn@mn.04$gamma);
**crossed4nbn1nbn**

creates a crossed-nbn from two /nbn/s

---

**Description**

A crossed /nbn/ is a /nbn/ obtained when replacing each node of the first /nbn/ by the second /nbn/ and vice-versa. Let \( nn1/nnR \) and \( na1/naR \) be the node and arc numbers of the two nbns, the node number of the crossed nbn is \( nn1*nn2 \) and its arc number is \( nn1*na2+nn2*na1 \). The regression coefficients attributed to the crossed nbn are the products of the weights (\( we1/we2 \)) and the regression coefficients of the initial nbn.

**Usage**

```r
crossed4nbn1nbn(nbn1, nbn2, we1=rep(1, length(nbn1)), we2=rep(1, length(nbn2)), nona=as.vector(outer(names(nbn1), names(nbn2), paste, sep=" ")), )
```

**Arguments**

- **nbn1** The first generating /nbn/.
- **nbn2** The second generating /nbn/.
- **we1** The weight to apply to the nodes of the first generating /nbn/.
- **we2** The weight to apply to the nodes of the second generating /nbn/.
- **nona** The node names to give to the crossed /nbn/, the nodes of the nbn1 varying first.

**Details**

The \( \mu \) coefficient is the sum of the two corresponding \( \mu \)s of the generating nbn. The \( \sigma \) coefficient is the product of the two corresponding \( \sigma \)s of the generating nbn.

**Value**

The resulting crossed nbn object.

**Examples**

```r
print8nbn(crossed4nbn1nbn(rbmn0nbn.01, rbmn0nbn.04));
```
**dev4mn**

*Computes the deviance for a sample of multinormal vector*

**Description**

From the \( n \) observed values of a vector of size \( p \) (\( Y \)), their expectations (\( EY \)) and the variance matrix (\( VY \)) supposed identical for all vectors, returns the deviance, i.e. \(-2 \log(P(Y))\).

**Usage**

\[
\text{dev4mn}(Y, EY, VY)
\]

**Arguments**

- **Y**: Matrix \( nxp \) of the \( n \) observed values of length \( p \).
- **EY**: Expectation of \( Y \) (matrix \( nxp \) or vector \( p \)).
- **VY**: Matrix of the variance of each row of \( Y \) (matrix \( pxp \)).

**Details**

- When \( EY \) is a vector with length \( ncol(Y) \) this supposes that all observations have the same expectation.

**Value**

A scalar

**Examples**

\[
\text{dev4mn(matrix(runif(3), 1), t(rbmn0mn.01$mu), rbmn0mn.01$gamma));}
\]

---

**diff8nbn**

*returns a score of the difference between two /nbn/s*

**Description**

Returns a positive scalar value measuring, in some way, the difference existing within two /nbn/s sharing the same structure.

**Usage**

\[
\text{diff8nbn(nbn1, nbn2, type=1, scalar=TRUE)}
\]
**Arguments**

- **nbn1**: First nbn object.
- **nbn2**: Second nbn object.
- **type**: When 1, the score includes the difference between the sigmas. When -1, sigmas are not taken into account.
- **scalar**: When TRUE the squared norm is returned, if not the vector of difference.

**Details**

For type==1 it is the canonical euclidian difference between all parameters, including the sigma. The score to use to measure the differences between two successive estimations is not well established (see the code).

**Value**

Either a scalar or a named vector (according to scalar).

**Examples**

```r
diff8nbn(rbmn@nbn.01, rbmn@nbn.01);
diff8nbn(rbmn@nbn.01, rbmn@nbn.01, scalar=FALSE);
```

---

**Description**

Estimations of the parameters of a nbn is done when there are some equality constraints onto the regression coefficients.

Constant terms (mu) and conditional standard deviations (sigma) are supposed independent (that is not constrained with equalities).

Equality constraints are given by sarc, a list of matrices with two columns, indicating each the series of arcs having the same regression coefficient.

**Usage**

```r
estimate8constrainednbn(nbn, sarc, data, imp=0, nite=10, eps=10^-5)
```

**Arguments**

- **nbn**: nbn object.
- **sarc**: List of Matrices with two columns indicating the tails (1st column) and the heads (2d column) of the arcs having a common parameter. It is checked that these arcs are indeed included in nbn. Nodes must be indicated by their names (not their number).
estimate8nbn

data Data frame to be used for the estimation. It must comprise all necessary nodes (not only those involved in sarc but also the remaining parents of sarc[,2]). Usually, all used variables are centred but this is not required.

imp When 0 nothing displayed. When 1 the number of iterations is displayed. When 2 the successive values of the criterion are also displayed.

nite Maximum number of iterations.

eps relative difference in successive scores needed to stop the iterations.

Details

Not linked regression coefficients doesn’t require to be included in sarc, the function do it by itself. The score to use to measure the differences between two successive estimations is not well established (see the code).

Value

The resulting /nbn/ object with the estimated parameters.

Examples

data(boco);
print8nbn(rbn0nbn.05);
print8nbn(estimate8nbn(rbn0nbn.05, boco));
print8nbn(estimate8constrainednbn(rbn0nbn.05, rbmn0crarc.05, boco));

estimate8nbn estimating the /nbn/ parameters

Description

From a /nbn/ to describe the DAG, and a data.frame containing the necessary observations, returns the /nbn/ with all its parameters newly estimated.

Usage

estimate8nbn(nbn, data)

Arguments

nbn The initial /nbn/.

data The data frame comprising all /nbn/ nodes.

Details

No constraints are put on the parameters.
Value

The resulting /nbn/ with the estimated parameters.

Examples

data(boco);
print8nbn(rbmn0nbn.05);
print8nbn(estimate8nbn(rbmn0nbn.05, boco));

Description

gema2mn

from a /gema/ object defining a normal Bayesian network, computes the expectation and variance matrix.

Usage

gema2mn(gema)

Arguments

gema Initial gema object.

Value

a list with the following components: mu and gamma.

Examples

print8mn(gema2mn(rbmn0gema.04));

gema2nbn

computes a /nbn/ from a /gema/

Description

gema2nbn

from a /gema/ object defining a normal Bayesian network, computes more standard /nbn/ where each node is defined from its parents.

Usage

gema2nbn(gema)
Arguments

gema Initial gema object.

Details

using general formulae rather a sequential algorithm as done in the original gema2nbn implementa-
tion.

Value

the corresponding /nbn/.

Examples

print8nbn(gema2nbn(rbmn0gema.02));

generate8chain generation of a /chain/ /nbn/

Description

[randomly] generates a /chain/ /nbn/.

Usage

generate8chain(rnn=c(3, 7), proo=0.5, rcor=c(-1, 1), rmu=c(0, 0), rsig=c(0, 1),
nona=r.form3names(max(rnn)))

Arguments

rnn Range of the number of nodes.
proo Probabilit[y|ies] that the successive and acceptable nodes be colliders. Can be a
vector.
rcor Range of the correlations between neighbour nodes.
rmu Range of the expectations.
rsig Range of the standard deviations.
nona Proposed names for the maximum number of nodes, only the necessary first
ones will be used.

Details

Proposed ranges can be a unique value, implying no randomness in the value.
Roots are placed according to proo probabilities, then collider are placed in between with uniform
probability on the possibles nodes.
**generate8nbn**

**Value**

A /chain/ coding list is returned.

**Examples**

```r
set.seed(1234);
print8chain(generate8chain());
print8chain(generate8chain());
print8chain(generate8chain(rnn=10, rcor=0.5));
print8chain(generate8chain(rnn=10, rcor=0.5));
```

**Description**

To obtain systematic results, you have to call `set.seed` before hands.

**Usage**

```r
generate8nbn(rnn=c(3, 7), ppar=0.5, rreg=c(-1, 1), rmu=c(0, 0), rsig=c(0, 1),
        none=r.form3names(max(rnn)))
```

**Arguments**

- **rnn**
  - Range of the number of nodes.
- **ppar**
  - Probabilities (not a range) of the parent occurrence for each ancestor of every node. Can be a vector, cycled as necessary.
- **rreg**
  - Range of regression coefficients.
- **rmu**
  - Range of the conditional expectations.
- **rsig**
  - Range of the conditional standard deviations.
- **nona**
  - Proposed names for the maximum number of nodes, only the necessary first ones will be used.

**Details**

Node numbers are uniformly drawn. Parent numbers are independently drawn from all ancestors with the probability associated to the considered node. Regression coefficient are uniformly drawn. Conditional expectations and standard deviations are uniformly drawn. All range arguments can be given one value instead of two, to precise the unique value to use.

**Value**

a /nbn/ object, with nodes in topological order.
Examples

```r
set.seed(1234)
print8nbn(generate8nbn());
print8nbn(generate8nbn());
```

---

### `inout4chain`  
*reduces a chain to its inputs and outputs*

#### Description

From a chain returns the reduced chain comprising only inputs (that is root nodes) and outputs (that is colliders and ends which are not roots).

#### Usage

```r
inout4chain(chain)
```

#### Arguments

- **chain**: The chain object to consider.

#### Value

The resulting chain

#### Examples

```r
print8chain(inout4chain(rbmn@chain.02));
```

---

### `is8nbn8chain`  
*Checks if a given /nbn/ is a /chain/*

#### Description

returns TRUE [the order] or FALSE [NULL] according that nbn is a chain of not [according to order].

#### Usage

```r
is8nbn8chain(nbn, order=FALSE)
```

#### Arguments

- **nbn**: The nbn object to consider.
- **order**: When FALSE the answer to the question is returned with TRUE or FALSE. When TRUE the chain order of the nodes is returned if it is a /chain/ else NULL.
Value

A logical(1) when order si TRUE if not the resulting chain order versus NULL.

Examples

```r
is8nbn8chain(rbmn0nbn.01);
is8nbn8chain(rbmn0nbn.04);
```

---

**marginal4chain**

returns marginal expectations and standard deviations of a chain

---

**Description**

From a chain object returns a list with two components: $\mu$ and $\sigma$ vectors of marginal expectations and standard deviations.

**Usage**

```r
marginal4chain(chain)
```

**Arguments**

- `chain` the chain object to be considered.

**Value**

a list with the two components $\mu$ and $\sigma$.

**Examples**

```r
marginal4chain(rbmn0chain.02);
```

---

**mn2gema**

computes a /gema/ from a /mn/

---

**Description**

proposes generating matrices of a Bayesian network from a /mn/ object defining a multinormal distribution by expectation and variance, under the assumption that the nodes are in topological order.

**Usage**

```r
mn2gema(mn)
```
Arguments

\texttt{mn} \hspace{1cm} \text{Initial \texttt{mn} object.}

Value

a list with the \texttt{/gema/} components $\mu$ and $\Gamma$.

Examples

\texttt{print8gema(mn2gema(rbbmn0mn.04));}

\begin{verbatim}
\texttt{mn4joint1condi} \hspace{0.5cm} \textit{computes a joint distribution from a marginal and a conditional one for multinormal distributions}
\end{verbatim}

Description

returns the expectation and variance of the multinormal normal distribution defined through a marginal subcomponent and a conditional distribution.

Usage

\texttt{mn4joint1condi(1mar, 1con)}

Arguments

\texttt{1mar} \hspace{1cm} \text{list defining the distribution of the marginal part with $\mu$, its expectation, and $\Gamma$, its variance matrix (in fact a \texttt{/mn/} object).}

\texttt{1con} \hspace{1cm} \text{list defining the distribution of the conditional part (see the Details section).}

Details

The conditional distribution is defined with a list having $a$ for the constant part of the expectation; $b$ for the regression coefficient part of the expectation; and $S$ for the residual variance matrix.

Value

A list:

\begin{verbatim}
$\mu$ The expectation vector.
$\Gamma$ The joint variance matrix.
\end{verbatim}

that is a \texttt{/mn/} object.
Examples

```r
icon <- list(a=c(D=2, E=4),
             b=matrix(c(1:6, 2, dimnames=list(LETTERS[4:5], LETTERS[1:3])),
             S=matrix(c(1, 1, 1, 2), 2));

print8mn(nn4joint1condi(rbmn@nn.01, icon));
```

---

**nb8bn**  
*number of Bayesian networks*

**Description**

returns the number of different Bayesian networks having \( n \) labelled or not nodes. Non labelled nodes means that nodes are exchangeable: \( A \rightarrow B \) is identical to \( A \leftarrow B \).

**Usage**

```r
nb8bn(n, label=FALSE)
```

**Arguments**

- \( n \)  
  number of nodes. Must be less or equal to 18.

- \( label \)  
  Indicates if the nodes must be considered as labelled or not.

**Details**

When not labelled nodes, the results were proposed by Sloane in 'the on line encyclopedia of integer sequences' (http://oeis.org/A003087). For labelled nodes, just the application of the recursive formula of Robinson.

**Value**

Number of Bayesian networks

**Examples**

```r
nb8bn(5)
nb8bn(5, TRUE);
```
### nbn2bnfit

transforms a /nbn/ to a /bn.fit/ of /bnlearn/ package

**Description**

returns a bn.fit object from a Gaussian nbn object of /rbmn/ package.

**Usage**

```r
nbn2bnfit(nbn, onlydag=FALSE)
```

**Arguments**

- `nbn` The object to be transformed.
- `onlydag` Indicates if only the DAG must be computed. In that case a /bn/ object of /bnlearn/.

**Value**

The resulting bn.fit (or bn) object.

### nbn2chain

transforms a /nbn/ into a /chain/

**Description**

returns the chain obtained from nbn which is supposed to a chain. If it is not a chain, an error is issued.

**Usage**

```r
nbn2chain(nbn)
```

**Arguments**

- `nbn` The /nbn/ object to consider.

**Details**

It is advised to use is8nbn8chain before calling this function.

**Value**

The resulting chain

**Examples**

```r
print8chain(nbn2chain(rbmn0nbn.02));
```
nbn2gema

\textit{computes a /gema/ from a /nbn/}

\textbf{Description}

from a /nbn/ object defining a normal Bayesian network, computes the vector \( \mu \) and the matrix \( \mathbf{\Lambda} \) such that if the vector \( \mathbf{E} \) is a vector of i.i.d. centred and standardized normal, then \( \mu + \mathbf{\Lambda} \times \mathbf{E} \) has the same distribution as the input /nbn/.

\textbf{Usage}

\begin{verbatim}
nbn2gema(nbn)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{nbn} \hspace{1cm} nbn object for which the generating matrices.
\end{itemize}

\textbf{Value}

a list with the two following components: \( \mu \) and \( \mathbf{\Lambda} \).

\textbf{Examples}

\begin{verbatim}
identical(nbn2gema(rbmn0nbn.02), rbmn0gema.02);
\end{verbatim}

\textbf{nbn2mn}

\textit{computes the joint distribution of a /nbn/}

\textbf{Description}

Computes the joint distribution of a /nbn/ with three possible algorithms according to \texttt{algo}.

\textbf{Usage}

\begin{verbatim}
nbn2mn(nbn, algo=3)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{nbn} \hspace{1cm} The nbn object to be converted.
  \item \texttt{algo} \hspace{1cm} either 1: transforming the nbn into a gema first before getting the mn form; or 2: one variable after another is added to the joint distribution following a topological order; or 3: variances are computed through the different paths o
\end{itemize}

\textbf{Details}

To be explained if it works
Value

the resulting /mn/ object

Examples

print8mn(nbn2mn(rbmn0nbn.05));

bn2nbn computes the /nbn/ changing its topological order

Description

returns the proposed /nbn/ with a new topological order without modifying the joint distribution of all variables.
This allows to directly find regression formulae within the Gaussian Bayesian networks.

Usage

bn2nbn(nbn, norder)

Arguments

nbn The /nbn/ to transform.
norder The topological order to follow. It can be indicated by names or numbers. When not all nodes are included, the resulting /nbn/ is restricted to these nodes after marginalization.

Details

BE aware that for the moment, no check is made about the topological order and if it is not, the result is FALSE!

Value

The resulting /nbn/.

Examples

print8mn(nbn2mn(rbmn0nbn.01, algo=1));
print8mn(nbn2mn(rbmn0nbn.01, algo=2));
print8mn(nbn2mn(rbmn0nbn.01, algo=3));
print8mn(nbn2mn(nbn2nbn(rbmn0nbn.02, c(1, 2, 4, 5, 3))));
print8mn(nbn2mn(nbn2nbn(rbmn0nbn.02, c(4, 1, 2, 3, 5))));
nbn2rr computes standard matrices from a /nbn/

---

**Description**

from a /nbn/ object defining a normal Bayesian network, returns a list comprising (i) mm the vector of the mean of the different nodes when the parents are nought, (ii) ss the vector of the conditional standard deviations and (iii) rr the matrix of the regression coefficients of the direct parents (rr[i,j] contains the regression coefficient of the node j for its parents i or zero when i is not a parent of j.

**Usage**

```r
nbn2rr(nbn)
```

**Arguments**

- `nbn` nbn object.

**Value**

the resulting list with the three components: mm, ss and rr.

**Examples**

```r
nbn2rr(rbmn@nbn.01);
```

---

nbn4nbn From a /nbn/ computes the associated nbn1

---

**Description**

returns a /nbn/ object with the same structure as nbn but all $\mu$s are put to zero, all $\sigma$s to one as well as $\text{regcoef}$.

**Usage**

```r
nbn4nbn(nbn)
```

**Arguments**

- `nbn` The nbn object to transform.

**Details**

These coefficient values allows the easy study of the /nbn/ structure.
Value

The resulting nbn.

Examples

```r
print8nbn(nbn4nbn(rbmn0nbn.04));
```

Description

reverse of `rmatrix4nbn` but the standard deviations must be included.

Usage

```r
nbn4rmatrix(rmatrix)
```

Arguments

- `rmatrix` The regression coefficient matrix with the standard deviations in the diagonal.

Details

mus are put to nought

Value

A `nbn` object

Examples

```r
print8nbn(nbn4rmatrix(rmatrix4nbn(rbmn0nbn.02)));
```
normalize8nbn

normalize a /nbn/

Description

returns a nbn with a given expectation and variance through an transformation leaving the correlation unchanged.

Usage

normalize8nbn(nbn, mu=0, sigma=1)

Arguments

nbn The nbn object to transform.
mu Imposed expectations. When NULL nothing is changed. When of length one, this value is given to all the node expectations. If not the complete vector of expect
sigma The same as mu but for the standard deviations.

Value

The transformed nbn.

Examples

print8nbn(normalize8nbn(rbmn8nbn.01));

order4chain returns a topological order of a /chain/ or checks a proposed order.

Description

From a chain object returns one of the possible topological orders, through a permutation when is.null(ord). If not ord must be a proposed order to be checked given as a permutation if is.numeric(ord) or a vector of ordered names if is.character(ord).

Usage

order4chain(chain, ord=NULL)

Arguments

chain the chain object to be considered.
ord Indicates what must be done. NULL to get a topological order associated to the chain otherwise a permutation to be checked as one of the possible topological orders of the chain.
Details

For the moment the ord option is bad and an error message is returned when used.

Value

a permutation vector of the nodes of the /nbn/ or a named character with the nodes not having their parents before them; when it is of length zero this means that the check was successful.

Examples

```r
order4chain(rbmn@chain.02);
order4chain(rbmn@chain.02, order4chain(rbmn@chain.02));
```

---

**order4gema**

topological order of a /gema/

Description

returns one of the orders of the nodes such as the parents of any node are less ranked than it when is.null(ord). If not check that the proposed order is either a right permutation (is.numeric(ord)) or a vector of node names providing a topological order (is.character(ord)).

Usage

```r
order4gema(gema, ord=NULL)
```

Arguments

gema gema object for which the order must be computed.

ord NULL or an order to test as a permutation or a vector of names.

Details

When !is.null(ord) the order must be an order, if not an error is issued.

Value

a permutation vector of the nodes of the /gema/ or a named list with the nodes not having their parents before them. That is a topological order.

Examples

```r
names(rbmn@gema.04$mu)[order4gema(rbmn@gema.04)];
```
**Description**

returns one of the orders of the nodes such as the parents of any node are less ranked than it when `is.null(ord)`. If not check that the proposed order is either a right permutation (`is.numeric(ord)`) or a vector of node names providing a topological order (`is.character(ord)`).

**Usage**

`order4nbn(nbn, ord=NULL)`

**Arguments**

- `nbn` nbn object for which the order must be computed.
- `ord` NULL or an order to test as a permutation or a vector of names.

**Details**

When `!is.null(ord)` the order must be an order, if not an error is issued.

**Value**

a permutation vector of the nodes of the `nbn/` or a named list with the nodes not having their parents before them.

**Examples**

```r
names(rbmn0nbn.04)[order4nbn(rbmn0nbn.04)];
```

---

**print8chain**

*prints a /chain/ object*

**Description**

prints a /chain/ object.

**Usage**

`print8chain(chain, digits=3)`

**Arguments**

- `chain` The chain object to print.
- `digits` when not null, the number of digits for rounding the numerical values.
Details

See nbn2chain code for some details about the definition of a /chain/.

Value

nothing but something is printed

Examples

```r
c # print8chain(rbm0chain.01)
c # print8chain(rbm0chain.02)
c # print8chain(rbm0chain.03)
```

Description

prints a /gema/ object completely or a part of it according to what specification.

Usage

```r
c # print8gema(gema, what="mL", ordering=NULL, digits=3, printed=TRUE)
```

Arguments

gema gema object to be printed.
what a character(1): when comprising "m" the expectations are printed, "l" the linear combinations are printed.
ordering Nodes are given following the indices of "ordering" if numeric or the names if it is character. NULL means the identity permutation. Repetitions or missing nodes are accepted.
digits when not null, the number of digits for rounding.
printed TRUE to issue a printing, if not the prepared matrix is returned.

Value

The gema is printed or a matrix having n x ? is returned binding which elements are precised in the argument what.

Examples

```r
print8gema(rbm0gema.01);
print8gema(rbm0gema.02, "m");
print8gema(rbm0gema.03, "l", digit=1);
print8gema(rbm0gema.04, printed=FALSE);
```
print8mn

standard print function for a /mn/ object.

**Description**

prints a /mn/ object completely or a part of it.

**Usage**

`print8mn(mn, what="msC", ordering=NULL, digits=3, printed=TRUE)`

**Arguments**

- **mn**: mn object to be printed.
- **what**: a character(1); when comprising "m" the expectations are printed, "s" the standard deviations are printed, "C" the correlation matrix is printed, "S" the variance matrix is printed, "P" the precision matrix is printed, "p" the normalized precision matrix is printed.
- **ordering**: Nodes are given following the indices of "ordering" if numeric or the names if it is character. NULL means the identity permutation. Repetitions or missing nodes are accepted.
- **digits**: when not null, the number of digits for rounding the parameter values.
- **printed**: TRUE to issue a printing, if not the prepared matrix is returned.

**Value**

The `mn` is printed or a matrix having `mn` x `?` is returned binding which elements precised in the argument `what`.

**Examples**

`print8mn(rbmn0mn.01);`

print8nbn

print function for a /nbn/ object.

**Description**

prints a /nbn/ object.

**Usage**

`print8nbn(nbn, what="pr", digits=3, ordering=NULL)`
Arguments

- **nbn**: nbn object to be printed.
- **what**: a character(1); when comprising "p" the name of each node with its parents are given, when comprising "r" the formula regression of each node is given with the node, when comprising "m" the model is given.
- **digits**: when not null, the number of digits for rounding.
- **ordering**: Nodes are given following the indices of "ordering" if numeric or the names if it is character. NULL means the identity permutation. Repetitions or missing nodes are accepted.

Value

Nothing but but nbn is printed.

Examples

```r
print(nbn(rbmn@nbn.01));
print(nbn(rbmn@nbn.03, "pm", order=1:2))
```

---

**provided objects**

**Some examplifying structures**

---

Description

Small examples of adjacency matrices, /nbn/, /chain/, /gema/ and /mn/ objects.

Usage

```r
rbm@chain.01
rbmn@chain.02
rbmn@chain.03
rbmn@nbn.01
rbmn@nbn.02
rbmn@nbn.03
rbmn@nbn.04
rbmn@adja.01
rbmn@adja.02
rbmn@adja.03
rbmn@adja.04
rbmn@mn.01
rbmn@mn.02
rbmn@mn.03
rbmn@mn.04
rbmn@gema.01
rbmn@gema.02
rbmn@gema.03
rbmn@gema.04
```
reverse8chain

Details

- `rbmn0chain` objects are chain /nbn/ objects
- `rbmn0nbn` objects are general /nbn/ objects
- `rbmn0adja` objects are adjacency matrices
- `rbmn0mn` objects are /mn/ distributions
- `rbmn0gema` objects are /gema/ generating matrices

Every last numbers (#) refer to the same Gaussian Bayesian networks.

Author(s)

Jean-Baptiste Denis

---

**reverse8chain** reverses the nodes of a chain

---

Description

returns the chain obtained after reversing its node order

Usage

reverse8chain(chain)

Arguments

chain The chain object to consider.

Value

The resulting chain

Examples

print8chain(rbmn0chain.02);
print8chain(reverse8chain(rbmn0chain.02));
### rm8nd4adj

**Description**

Eliminates from the adjacency matrix (adj) all nodes not breaking the existing links. Important: the node order in adj must be topological.

**Usage**

```
rm8nd4adj(adj, nodes)
```

**Arguments**

- `adj` The relation matrix to consider (same format as those provided by the function adja4nbn. Must be in topological order, roots first.
- `nodes` Numeric or character vector providing the node numbers to use for the generation of the subset.

**Details**

When a node is removed, all its parents become parent of its children.

**Value**

The reduced adjacency matrix.

**Examples**

```
rm8nd4adj(rbmn@adj.a, 0.1);  
```

### rm8nd4nbn

**Description**

returns an /nbn/ object deduced from an original /nbn/ by integrating on a given subset of nodes.

**Usage**

```
rm8nd4nbn(nbn, nodes)
```

**Arguments**

- `nbn` The nbn object to reduce.
- `nodes` character or numeric vector giving the subset of nodes to remove.
Details

The transformation is made through the associated joint distributions for the probabilities and with the help of the function `rm8nd4adja` for the relationships.

Value

The resulting nbn.

Examples

```r
rm8nd4nbn(rbmn0nbn.04, "1.1");
```

Description

returns a dimnamed matrix indicating with rho an arc from row to column nodes (0 everywhere else) where rho is the regression coefficient. Also conditional standard deviations can be introduced as diagonal elements but mu coefficient are lost... It is advisable to normalize the nbn first.

Usage

```r
rmatrix4nbn(nbn, stdev=TRUE)
```

Arguments

- **nbn** The initial nbn object.
- **stdev** Indicates if the standard deviations must placed in the diagonal positions.

Value

A dimnamed matrix

Examples

```r
rmatrix4nbn(rbmn0nbn.02);
(rmatrix4nbn(rbmn0nbn.02, FALSE)>0)*1;
```
**simulate8gema**

simulates from a /gema/ object

**Description**

returns a matrix of simulated values with the variable in columns and the simulations in rows.

**Usage**

simulate8gema(gema, nbs)

**Arguments**

- **gema**
  - The gema object.
- **nbs**
  - number of simulations to return.

**Details**

Just the application of the standard formula to a white noise. Variables names are taken from those of gema$\mu$, when these does not exist, standard ones are provided.

**Value**

A matrix of size: nbs x length(gema$\mu$)

**Examples**

simulate8gema(rbmn@gema.01, 10);

**simulate8gmn**

simulates a multinormal vector with varying expectation

**Description**

returns a matrix of simulated values with the variable in columns and the simulations in rows.

**Usage**

simulate8gmn(loi, cova, nbs, tol=1e-7)
simulate8mn

Arguments

loi list defining the distribution of the initial vector with $\mu$, its expectation, $\gamma$, its variance matrix and $\rho$ a matrix of regression coefficients for the covariates modifying the expectation.

cova Values to give to the covariables. Must be a matrix with nbs rows and ncol(loi$\rho$) columns or a vector with ncol(loi$\rho$) values to be used for all simulations (i.e. to replace a matrix with identical rows).

nbs number of simulations to return.

tol tolerance value to be transmitted to mvrnorm.

Details

Just a call to the function simulate8mn, adding the terms to the expectation due to the regression...

Value

A matrix of size: nbs x length(loi$\mu$)

Examples

loi <- list(mu=c(2, E=4),
            rho=matrix(1:6, 2, dimnames=list(LETTERS[4:5],
                                  LETTERS[1:3])),
            gamma=matrix(c(1, 1, 2), 2));
cova <- matrix(runif(36), 12, dimnames=list(NULL, LETTERS[1:3]));
print(simulate8mn(loi, cova, 12));

simulate8mn simulates a multinormal vector

Description

returns a matrix of simulated values with the variable in columns and the simulations in rows.

Usage

simulate8mn(mn, nbs, tol=1e-7)

Arguments

mn list defining the distribution of the initial vector with $\mu$, its expectation, and $\gamma$, its variance matrix.

nbs number of simulations to return.

tol tolerance value to be transmitted to mvrnorm.
Details

Just a call to the basic function \texttt{mvrnorm}. Names of the variables are taken from those of \texttt{mn$\mu$}, when these does not exist, standard ones are provided.

Value

A matrix/data frame of size : \texttt{nbs} x \texttt{length(mn$\mu$)}

Examples

\begin{verbatim}
print(simulate8mn(rbmn0mn.01, 12));
\end{verbatim}

\begin{verbatim}
simulate8nbn simulates from a /nbn/ object
\end{verbatim}

Description

returns a matrix of simulated values with the variable in columns and the simulations in rows.

Usage

\texttt{simulate8nbn(nbn, nbs)}

Arguments

\begin{verbatim}
nbn The nbn object.
nbs number of simulations to return.
\end{verbatim}

Details

Just the sequential simulations of the nodes

Value

A matrix of size : \texttt{nbs} x \texttt{length(nbn)}

Examples

\begin{verbatim}
simulate8nbn(rbmn0nbn.01, 10);
\end{verbatim}
state4chain

returns the states of each node of a chain

Description

From a chain object returns a named character precising the role of each node: "r" for root, "c" for collider, "t" for transmitter and "l" for leaf.

Usage

state4chain(chain)

Arguments

chain the chain object to be considered.

Value

a character of the states named with node names.

Examples

state4chain(rbmn@chain.01);
state4chain(rbmn@chain.03);

string7dag4nbn

provides so-called string model of a nbn/

Description

returns a character(1) describing the dag of the nbn under the string form.

Usage

string7dag4nbn(nbn, sep=";")

Arguments

nbn The nbn.
sep Separation sign between parents after the conditioning sign (|).

Value

A character(1).
Examples

```r
string7dag4nbn(rbn0nbn.01);
string7dag4nbn(rbn0nbn.04, sep=", ");
```

| var2pre | returns the precision matrix from the variance |

Description

returns the precision matrix from the variance preserving possible variable names

Usage

```r
var2pre(ma)
```

Arguments

- `ma` The variance matrix.

Details

Non full rank matrices are accepted, a generalized inverse is returned and a warning is issued.

Value

The precision matrix

Examples

```r
var2pre(rbn0mn.04$gamma);
```
Index

*Topic PKEYWORDS
  rm8nd4adja, 42
*Topic datasets
  body composition, 10
*Topic examples
  provided objects, 40
*Topic package
  rbmn-package, 3
*Topic utilities
  rm8nd4adja, 42

adja2arcs, 5
adja2crossed, 6
adja2nbn, 6
adja4nbn, 7
adja4three, 7
arc7nb4nbn, 8
arc8nbn1nbn, 9
bn2nbn, 9
bnfit2nbn, 10
boco (body composition), 10
body composition, 10
chain2correlation, 11
chain2gema, 12
chain2mn, 13
chain2nbn, 13
chain2pre, 14
chain4chain, 14
check8chain, 15
check8gema, 16
check8nbn, 16
condi4joint, 17
cor4var, 18
crossed4nbn1nbn, 19
dev4mn, 20
diff8nbn, 20
estimate8constrainednbn, 21
estimate8nbn, 22
gema2mn, 23
gema2nbn, 23
generate8chain, 24
generate8nbn, 25
inout4chain, 26
is8nbn8chain, 26
marginal4chain, 27
mn2gema, 27
mn4joint1condi, 28
nb8nbn, 29
nbn2bnfit, 30
nbn2chain, 30
nbn2gema, 31
nbn2mn, 31
nbn2nbn, 32
nbn2rr, 33
nbn4nbn, 33
nbn4rmatrix, 34
normalize8nbn, 35
order4chain, 35
order4gema, 36
order4nbn, 37
print8chain, 37
print8gema, 38
print8mn, 39
print8nbn, 39
provided objects, 40
rbmn (rbmn-package), 3
rbmn-package, 3
rbmn0adja.01 (provided objects), 40
rbmn0adja.02 (provided objects), 40
rbmn0adja.03 (provided objects), 40
rbmn0adja.04 (provided objects), 40
rbmn0adja.05 (provided objects), 40
rbmn0chain.01 (provided objects), 40
rbmn0chain.02 (provided objects), 40
rbmn0chain.03 (provided objects), 40
rbmn0crarc.05 (provided objects), 40
rbmn0gema.01 (provided objects), 40
rbmn0gema.02 (provided objects), 40
rbmn0gema.03 (provided objects), 40
rbmn0gema.04 (provided objects), 40
rbmn0gema.05 (provided objects), 40
rbmn0mn.01 (provided objects), 40
rbmn0mn.02 (provided objects), 40
rbmn0mn.03 (provided objects), 40
rbmn0mn.04 (provided objects), 40
rbmn0mn.05 (provided objects), 40
rbmn0nb.01 (provided objects), 40
rbmn0nb.02 (provided objects), 40
rbmn0nb.03 (provided objects), 40
rbmn0nb.04 (provided objects), 40
rbmn0nb.05 (provided objects), 40
reverse8chain, 41
rm8nd4adjja, 42
rm8nd4nb, 42
rmatrix4nb, 43
simulate8gema, 44
simulate8gmn, 44
simulate8mn, 45
simulate8nb, 46
state4chain, 47
string7dag4nb, 47
var2pre, 48