Package ‘rbmn’

April 7, 2022

Version 0.9-5

Date 2022-04-07

Type Package

Title Handling Linear Gaussian Bayesian Networks

Author Jean-Baptiste Denis [aut, cre], Marco Scutari [ctb]

Maintainer Marco Scutari <scutari@bnlearn.com>

Description Creation, manipulation, simulation of linear Gaussian Bayesian networks from text files and more...

License GPL (>= 2)

Imports MASS

Suggests bnlearn, igraph

NeedsCompilation no

Repository CRAN

Date/Publication 2022-04-07 07:40:02 UTC

R topics documented:

`rbmn-package` *Linear Gaussian Bayesian network manipulations*

Description

General functions to generate, transform, display general and particular linear Gaussian Bayesian networks [/nbn/] are provided.
Specific /nbn/ are chain and crossed /nbn/s. 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 (`$mu`) and the variance matrix (`$gamma`).
- /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 (`li`) such that the vector of the nodes can be defined by `X = mu + li%*%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**.

- 1 (similar to an ~ and) ??? to link different objects or actions `train1car` for train and car.
- 2 (as usual but only one-to-one) `nbn2gema` means \"transforming a /nbn/ into a /gema/ objects\".
- 3 (remind the 'belong to' sign) `form3repeat` could be interpreted as "repeat action from the series of 'form' functions".
- 4 (associated to 'from') `adja4nbn` means "get the adjacency matrix from a /nbn/ object".
- 7 (upper bar of '7' similar to the hyphen) `arc7nb4nbn` means "get the arc-numbers from a /nbn/".
- 8 (similar to 'a') `generate8nbn` or `print8nbn` for \"generating or printing a /nbn/ object\".

A number of ancillary functions have not been exported to give a better access to the main function of /rbmn/. Nevertheless they are available in the `../rbmn/R/` directory, and with all their comments (equivalent to `Rd` files into `../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 `condi4joint` when argument `pour` is not of length zero and argument `x2` is not null. With such a structure, every node of a /nbn/ could be described with a /mn/ comprising a unique variable... Also the two arguments of function `mn4joint1condi` would be just two /mn/ objects... This is also the generalized /mn/ proposed in function `simulate8gmn` under the argument of `loi`... Of course almost all functions dealing with /nbn/ objects will be to rewrite!
- Introduce a new object `gbn` for Gaussian Bayesian network similar to the list provided by function `nbn2rr`.

TO DO list

- Systemize the existence of `check8object` functions
- Introduce their systematic use conditionned with a `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 `nbn4string7dag`.
- Add the computation made with /bnlearn/ in the example of `estimate8nbn`.
- Check the topological order within `nbn2nbn` depending on `rbmn0check` value.
- Make a super transformation function from an object associated to a Bayesian network to any other type, including itself.
- Correct the `ord` option in `order4chain`.
- Check the topological order in `rm8nd4adja`.
- Think about removing all `rmatrix` transformations to the benefit of the to-come `gbn` object.
- Introduce a check of non-negativity of `ma` into `cor4var`.
- Add examples to all functions without any.

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.)

Scutari M (2010). "Learning Bayesian Networks with the bnlearn R Package". Journal of Statistical Software, **35**(3), 1-22. URL http://www.jstatsoft.org/v35/i03/.

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

Examples

```
library(rbmn)

## 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

```
adja2arcs(adj)
```

Arguments

`adj`	The adjacency matrix.

Value

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

Examples

```
adja2arcs(rbmn0adja.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(rbmn0adja.01, rbmn0adja.01));
```

`adja2nbn` *standardized /nbn/ from an adjacency matrix*

Description

returns a `nbn` object with O/1 regression coefficients having `adja` as adjacency matrix.

Usage

```
adja2nbn(adja)
```

Arguments

`adja`	The initial adjacency matrix.

Value

The corresponding standardized nbn object.

Examples

```
print8nbn(adja2nbn(adja4nbn(rbmn0nbn.03)));
```

---

adja4nbn *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

```
adja4nbn(nbn)
```

Arguments

nbn	The initial nbn object.

Value

A dimnamed matrix

Examples

```
adja4nbn(rbmn0nbn.04);
```

---

adja4three *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

```
adja4three(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.

`arc7nb4nbn` *returns the number(s) of arcs of a /nbn/*

Description

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

Usage

```
arc7nb4nbn(nbn, each=FALSE)
```

Arguments

`nbn`	The `nbn` object to consider.
`each`	When `TRUE`, returns a named vector of the number of parents of each node. If not the total number of arcs.

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

```
arc7nb4nbn(rbmn0nbn.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

Centers for Disease Control and Prevention. The 1999-2004 dual energy X-ray absorptiometry (DXA) multiple imputation data files and technical documentation.
Available from: http://www.cdc.gov/nchs/about/major/nhanes/dxx/dxa.html (accessed on 13_07_03).

Examples

```
# 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)$li-rbmn0gema.02$li);
```

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

```
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 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

```
check8nbn(rbmn0nbn.01);
res <- check8nbn(rbmn0adja.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

```
condi4joint(mn, par, pour, x2=NULL)
```

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 for non conditionning.
`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 x2 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 x2 is NULL the list has got three components: $mu for the fixed part of the expectation vector, $b for the regression coefficients to be associated to the non precised x2 values, varying part of the expectation and $gamma for the variance matrix.

Examples

```
print8mn(condi4joint(rbmn0mn.04, c("1.1", "2.2", "1.2", "2.1"), NULL));
print8mn(condi4joint(rbmn0mn.04, c("1.1", "2.2", "1.2", "2.1"), "C", 0));
print(condi4joint(rbmn0mn.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(rbmn0mn.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/nn2 and na1/na2 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

```
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 mus of the generating nbn.
The sigma coefficient is the product of the two corresponding sigmas of the generating nbn.
The regression coefficient are directed inherited from the nbn which is duplicated with this arc.

Value

The resulting crossed nbn object.

Examples

```
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

```
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

```
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

```
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

```
diff8nbn(rbmn0nbn.01, rbmn0nbn.01);
diff8nbn(rbmn0nbn.01, rbmn0nbn.01, scalar=FALSE);
```

---

estimate8constrainednbn

estimates the parameters of a nbn with equality constraints

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

```
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 (1rst 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).

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(rbmn0nbn.05);
print8nbn(estimate8nbn(rbmn0nbn.05, boco));
print8nbn(estimate8constrainednbn(rbmn0nbn.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));
```

---

gema2mn *computes a /mn/ from a /gema/*

Description

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

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 implementation.

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.

Value

A /chain/ coding list is returned.

Examples

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

---

generate8nbn — *returns a randomly built /nbn/ object.*

Description

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

Usage

```
generate8nbn(rnn=c(3, 7), ppar=0.5, rreg=c(-1, 1), rmu=c(0, 0), rsig=c(0, 1),
  nona=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

```
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

```
inout4chain(chain)
```

Arguments

chain	The chain object to consider.

Value

The resulting chain

Examples

```
print8chain(inout4chain(rbmn0chain.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

```
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

```
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

```
marginal4chain(chain)
```

Arguments

`chain`	the `chain` object to be considered.

Value

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

Examples

```
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

```
mn2gema(mn)
```

Arguments

`mn`	Initial `mn` object.

Value

a list with the /gema/ components `$mu` and `$li`.

Examples

```
print8gema(mn2gema(rbmn0mn.04));
```

`mn4joint1condi` *computes a joint distribution from a marginal and a conditional one for multinormal distributions*

Description

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

Usage

```
mn4joint1condi(lmar, lcon)
```

Arguments

`lmar`	list defining the distribution of the marginal part with `$mu`, its expectation, and `$gamma`, its variance matrix (in fact a /mn/ object).
`lcon`	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:

`\$mu`	The expectation vector.
`\$gamma`	The joint variance matrix.

that is a /mn/ object.

Examples

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

print8mn(mn4joint1condi(rbmn0mn.01, lcon));
```

---

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 -> B` is identical to `A <-B`.

Usage

```
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 encyclopedy 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

```
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

```
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

```
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

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

nbn2gema	*computes a /gema/ from a /nbn/*

Description

from a /nbn/ object defining a normal Bayesian network, computes the vector mu and the matrix li such that if the vector E is a vector of i.i.d. centred and standardized normal, then mu + li * E has the same distribution as the input /nbn/.

Usage

```
nbn2gema(nbn)
```

Arguments

nbn	nbn object for which the generating matrices.

Value

a list with the two following components: mu and li.

Examples

```
identical(nbn2gema(rbmn0nbn.02), rbmn0gema.02);
```

nbn2mn	*computes the joint distribution of a /nbn/*

Description

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

Usage

```
nbn2mn(nbn, algo=3)
```

Arguments

nbn	The nbn object to be converted.
algo	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 differents paths o

Details

To be explained if it works

Value

the resulting /mn/ object

Examples

```
print8mn(nbn2mn(rbmn0nbn.05));
```

---

nbn2nbn *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

```
nbn2nbn(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

```
nbn2rr(nbn)
```

Arguments

nbn	nbn object.

Value

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

Examples

```
nbn2rr(rbmn0nbn.01);
```

nbn4nbn *From a /nbn/ computes the associated nbn1*

Description

returns a /nbn/ object with the same structure as nbn but all $mu are put to zero, all $sigma to one as well as $regcof.

Usage

```
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

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

---

nbn4rmatrix *a /nbn/ from a regression matrix*

Description

reverse of rmatrix4nbn but the standard deviations must be included.

Usage

```
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

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

`normalize8nbn` *normalizes 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(rbmn0nbn.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

```
order4chain(rbmn0chain.02);
order4chain(rbmn0chain.02, order4chain(rbmn0chain.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

```
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

```
names(rbmn0gema.04$mu)[order4gema(rbmn0gema.04)];
```

order4nbn *topological order of a /nbn/*

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

```
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

```
print8chain(rbmn0chain.01);
print8chain(rbmn0chain.02);
print8chain(rbmn0chain.03);
```

`print8gema` *standard print function for a /gema/ object.*

Description

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

Usage

```
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 `nn x ?` is returned binding which elements are precised in the argument `what`.

Examples

```
print8gema(rbmn0gema.01);
print8gema(rbmn0gema.02, "m");
print8gema(rbmn0gema.03, "l", digit=1);
print8gema(rbmn0gema.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 `nn` 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

```
print8nbn(rbmn0nbn.01);
print8nbn(rbmn0nbn.03, "pm", order=1:2)
```

---

provided objects *Some examplifying structures*

Description

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

Usage

```
rbmn0chain.01
rbmn0chain.02
rbmn0chain.03
rbmn0nbn.01
rbmn0nbn.02
rbmn0nbn.03
rbmn0nbn.04
rbmn0adja.01
rbmn0adja.02
rbmn0adja.03
rbmn0adja.04
rbmn0mn.01
rbmn0mn.02
rbmn0mn.03
rbmn0mn.04
rbmn0gema.01
rbmn0gema.02
rbmn0gema.03
rbmn0gema.04
```

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));
```

rm8nd4adja — *removes somes nodes from an adjacency matrix*

Description

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

Usage

```
rm8nd4adja(adja, nodes)
```

Arguments

`adja`	The relation matrix to be 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

```
rm8nd4adja(rbmn0adja.04, "1.1");
```

rm8nd4nbn — *removes some nodes from a /nbn/*

Description

returns a /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

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

`rmatrix4nbn` *regression matrix of a /nbn/*

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

```
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

```
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(rbmn0gema.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)
```

Arguments

`loi`	list defining the distribution of the initial vector with `$mu`, its expectation, `$gamma`, its variance matrix and `$rho` a matrix of regression coefficients for the covariables modifying the expectation.
`cova`	Values to give to the covariables. Must be a matrix with `nbs` rows and `ncol(loi$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(D=2, E=4),
rho=matrix(1:6, 2, dimnames=list(LETTERS[4:5],
LETTERS[1:3])),
gamma=matrix(c(1, 1, 1, 2), 2));
cova <- matrix(runif(36), 12, dimnames=list(NULL, LETTERS[1:3]));
print(simulate8gmn(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 mvrnorm. Names of the variables are taken from those of mn$mu, when these does not exist, standard ones are provided.

Value

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

Examples

```
print(simulate8mn(rbmn0mn.01, 12));
```

---

simulate8nbn *simulates from a /nbn/ object*

Description

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

Usage

```
simulate8nbn(nbn, nbs)
```

Arguments

nbn	The nbn object.
nbs	number of simulations to return.

Details

Just the sequential simulations of the nodes

Value

A matrix of size : nbs x length(nbn)

Examples

```
simulate8nbn(rbmn0nbn.01, 10);
```

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(rbmn0chain.01);
state4chain(rbmn0chain.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

```
string7dag4nbn(rbmn0nbn.01);
string7dag4nbn(rbmn0nbn.04, sep=", ");
```

---

var2pre *returns the precision matrix from the variance*

Description

returns the precision matrix from the variance preserving possible variable names

Usage

```
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

```
var2pre(rbmn0mn.04$gamma);
```

Index