Package 'BiDAG'

July 14, 2020

Type Package

Title Bayesian Inference for Directed Acyclic Graphs

Version 1.4.1

Date 2020-06-18

Author Polina Suter [aut, cre], Jack Kuipers [aut]

Maintainer Polina Suter <polina.minkina@bsse.ethz.ch>

Description Implementation of a collection of MCMC methods for Bayesian structure learning of directed acyclic graphs (DAGs), both from continuous and discrete data. For efficient inference on larger DAGs, the space of DAGs is pruned according to the data. To filter the search space, the algorithm employs a hybrid approach, combining constraint-based learning with search and score. A reduced search space is initially defined on the basis of a skeleton obtained by means of the PC-algorithm, and then iteratively improved with search and score. Search and score is then performed following two approaches: Order MCMC, or Partition MCMC.
The BGe score is implemented for continuous data and the BDe score is implemented for binary data or categorical data. The algorithms may provide the maximum a posteriori (MAP) graph or a sample (a collection of DAGs) from the posterior distribution given the data. All algorithms are also applicable for structure learning and sampling for dynamic Bayesian networks.
References:
J. Kuipers, P. Suter and G. Moffa (2018) <arXiv:1803.07859v2>,
N. Friedman and D. Koller (2003) <doi:10.1023/A:1020249912095>,
D. Geiger and D. Heckerman (2002) <doi:10.1214/aos/1035844981>,
J. Kuipers and G. Moffa (2017) <doi:10.1080/01621459.2015.1133426>,
M. Kalisch et al.(2012) <doi:10.18637/jss.v047.i11>.

Acknowledgments We would like to thank Giusi Moffa for discussion and comments on the package and its manual.

License GPL (>= 2)

Imports Rcpp (>= 0.12.7), methods, pcalg, graph, Rgraphviz

Depends R (>= 3.5.0), graphics

LinkingTo Rcpp

RoxygenNote 7.1.1

Encoding UTF-8

LazyData TRUE

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-07-14 11:40:02 UTC

R topics documented:

`Asia` *Asia dataset*

Description

A synthetic dataset from Lauritzen and Spiegelhalter (1988) about lung diseases (tuberculosis, lung cancer or bronchitis) and visits to Asia.

Usage

```
Asia
```

Format

A data frame with 5000 rows and 8 binary variables:

- D (dyspnoea), binary 1/0 corresponding to "yes" and "no"
- T (tuberculosis), binary 1/0 corresponding to "yes" and "no"
- L (lung cancer), binary 1/0 corresponding to "yes" and "no"
- B (bronchitis), binary 1/0 corresponding to "yes" and "no"
- A (visit to Asia), binary 1/0 corresponding to "yes" and "no"
- S (smoking), binary 1/0 corresponding to "yes" and "no"
- X (chest X-ray), binary 1/0 corresponding to "yes" and "no"
- E (tuberculosis versus lung cancer/bronchitis), binary 1/0 corresponding to "yes" and "no"

Source

http://www.bnlearn.com/bnrepository/

References

Lauritzen S, Spiegelhalter D (1988). 'Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion)'. Journal of the Royal Statistical Society: Series B 50, 157-224.

`Asiamat` *Asiamat*

Description

An adjacency matrix representing the ground truth DAG used to generate a synthetic dataset from Lauritzen and Spiegelhalter (1988) about lung diseases (tuberculosis, lung cancer or bronchitis) and visits to Asia.

Usage

```
Asiamat
```

Format

A binary matrix with 8 rows and 8 columns representing an adjacency matrix of a DAG with 8 nodes:

- D (dyspnoea), binary 1/0 corresponding to "yes" and "no"
- T (tuberculosis), binary 1/0 corresponding to "yes" and "no"
- L (lung cancer), binary 1/0 corresponding to "yes" and "no"
- B (bronchitis), binary 1/0 corresponding to "yes" and "no"
- A (visit to Asia), binary 1/0 corresponding to "yes" and "no"
- S (smoking), binary 1/0 corresponding to "yes" and "no"
- X (chest X-ray), binary 1/0 corresponding to "yes" and "no"
- E (tuberculosis versus lung cancer/bronchitis), binary 1/0 corresponding to "yes" and "no"

Source

http://www.bnlearn.com/bnrepository/

References

Lauritzen S, Spiegelhalter D (1988). 'Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion)'. Journal of the Royal Statistical Society: Series B 50, 157-224.

`Boston` *Boston housing data*

Description

A dataset containing information collected by the U.S Census Service concerning housing in the area of Boston, originally published by Harrison and Rubinfeld (1978).

Usage

```
Boston
```

Format

A data frame with 506 rows and 14 variables:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- TAX - full-value property-tax rate per $10,000
- RAD - index of accessibility to radial highways
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - percentage lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

Source

http://lib.stat.cmu.edu/datasets/boston

References

Harrison, D and Rubinfeld, DL (1978) 'Hedonic prices and the demand for clean air', Journal of Environmental Economics and Management 5, 81-102.

compact2full *Deriving an adjecency matrix of a full DBN*

Description

This function transforms a compact 2-slice adjacency matrix of DBN into full T-slice adjacency matrix

Usage

```
compact2full(DBNmat, n.dynamic, n.slices, n.static = 0)
```

Arguments

DBNmat	a square matrix, representing initial and transitional structure of a DBN; the size of matrix is 2*n.dynamic+n.static
n.dynamic	integer, number of dynamic variables in one time slice
n.slices	integer, number of slices in an unrolled DBN
n.static	integer, number of static variables

Value

an adjacency matrix of an unrolled DBN

Examples

```
compact2full(DBNmat, n.dynamic=12, n.slices=5, n.static=3)
```

compareDAGs *Comparing two DAGs*

Description

This function compares one (estimated) DAG to another DAG (true DAG), returning a vector of 3 values: structural Hamming distance, number of true positive edges and number of false positive edges.

Usage

```
compareDAGs(eDAG, trueDAG)
```

Arguments

eDAG	an object of class graphNEL (package 'graph'), representing the DAG which should be compared to a ground truth DAG or an ajacency matrix corresponding to the DAG
trueDAG	an object of class graphNEL (package 'graph'), representing the ground truth DAG or an ajacency matrix corresponding to the DAG

Value

a vector of 5: SHD, number of true positive edges, number of false positive edges, number of false negative edges and true positive rate

Examples

```
Asiascore<-scoreparameters(8,"bde",Asia)
## Not run:
eDAG<-orderMCMC(Asiascore)
compareDAGs(eDAG$max$DAG,Asiamat)

## End(Not run)
```

---

compareDBNs *Comparing two DBNs*

Description

This function compares one (estimated) DBN structure to another DBN (true DBN). Comparisons for initial and transitional structures are returned separately if equalstruct equals TRUE.

Usage

```
compareDBNs(eDBN, trueDBN, struct = c("init", "trans"), n.dynamic, n.static)
```

Arguments

eDBN	an object of class graphNEL (or an ajacency matrix corresponding to this DBN), representing the DBN which should be compared to a ground truth DBN
trueDBN	an object of class graphNEL (or an ajacency matrix corresponding to this DBN), representing the ground truth DBN
struct	option used to determine if the initial or the transitional structure should be compared; accaptable values are init or trans
n.dynamic	number of dynamic variables in one time slice of a DBN
n.static	number of static variables in one time slice of a DBN; note that for function to work correctly all static variables have to be in the first n.static columns of the matrix

Value

a vector of 5: SHD, number of true positive edges, number of false positive edges, number of false negative edges and true positive rate

Examples

```
testscore<-scoreparameters(15, "bge", DBNdata, bgnodes=c(1,2,3), DBN=TRUE,
                           dbnpar=list(slices=5, stationary=TRUE))
## Not run:
DBNfit<-iterativeMCMC(testscore,moveprobs=c(0.11,0.84,0.04,0.01))
compareDBNs(DBNfit$max$DAG,DBNmat,struct="trans",n.dynamic=12,n.static=3)

## End(Not run)
```

---

dag.threshold *Estimating a graph corresponding to a posterior probability threshold*

Description

This function constructs a directed graph (not necessarily acyclic) including all edges with a posterior probability above a certain threshold. The posterior probability is evaluated as the Monte Carlo estimate from a sample of DAGs obtained via an MCMC scheme.

Usage

```
dag.threshold(MCMCchain, pbarrier, pdag = FALSE, burnin = 0.2)
```

Arguments

MCMCchain	list of adjacency matrices with dimensions equal to n and elements in {0,1}, representing a sample of DAGs from an MCMC scheme (objects of classes 'MCMCtrace' or 'MCMCres' are also valid data types)
pbarrier	threshold such that only edges with a higher posterior probability will be retained in the directed graph summarising the sample of DAGs
pdag	logical, if TRUE (FALSE by default) all DAGs in the MCMCchain are first converted to equivalence class (CPDAG) before the averaging
burnin	(optional) number between 0 and 1, indicates the percentage of the samples which will be the discarded as 'burn-in' of the MCMC chain; the rest of the samples will be used to calculate the posterior probabilities; 0.2 by default

Value

a square matrix with dimensions equal to the number of variables representing the adjacency matrix of the directed graph summarising the sample of DAGs

Examples

```
Bostonscore<-scoreparameters(14, "bge", Boston)
## Not run:
orderfit<-orderMCMC(Bostonscore, MAP=FALSE, iterations=25000, chainout=TRUE)
hdag<-dag.threshold(orderfit, pbarrier=0.9)

## End(Not run)
```

DAGscore *Calculating the BGe/BDe score of a single DAG*

Description

This function calculates the score of a DAG defined by its adjacency matrix. Acceptable data matrices are homogeneous with all variables of the same type: continuous, binary or categorical. The BGe score is evaluated in the case of continuous data and the BDe score is evaluated for binary and categorical variables.

Usage

```
DAGscore(scorepar, incidence)
```

Arguments

`scorepar`	an object of class `scoreparameters`, containing the data and scoring parameters; see constructor function `scoreparameters`
`incidence`	a square matrix of dimensions equal to the number of nodes, representing the adjacency matrix of a DAG; the matrix entries are in `{0,1}` such that `incidence[i,j]` equals 1 if there is a directed edge from node `i` to node `j` in the DAG and `incidence[i,j]` equals 0 otherwise

Value

the log of the BGe or BDe score of the DAG

References

Geiger D and Heckerman D (2002). Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. The Annals of Statistics 30, 1412-1440.

Heckerman D and Geiger D (1995). Learning Bayesian networks: A unification for discrete and Gaussian domains. In Eleventh Conference on Uncertainty in Artificial Intelligence, pages 274-284.

Kuipers J, Moffa G and Heckerman D (2014). Addendum on the scoring of Gaussian directed acyclic graphical models. The Annals of Statistics 42, 1689-1691.

Examples

```
myScore<-scoreparameters(8, "bde", Asia)
DAGscore(myScore, Asiamat)
```

DBNdata *A simulated data set from a 2-step dynamic Bayesian network A synthetic dataset containing 100 observations generated from a random dynamic Bayesian network with 12 continuous dynamic nodes and 3 static discrete nodes. The DBN imcludes observations from 5 time slices.*

Description

A simulated data set from a 2-step dynamic Bayesian network

A synthetic dataset containing 100 observations generated from a random dynamic Bayesian network with 12 continuous dynamic nodes and 3 static discrete nodes. The DBN imcludes observations from 5 time slices.

Usage

```
DBNdata
```

Format

A data frame with 100 rows and 63 (3+12*5) columns representing observations of 15 variables: 3 static variables (first 3 columns) which do not change over time and 12 dynamic variables observed in 5 conseecutive time slices.

DBNmat *An adjacency matrix of a dynamic Bayesian network*

Description

An adjacency matrix representing the ground truth DBN used to generate a synthetic dataset DBNdata. The matrix is a compact representation of a 2-step DBN, such that initial structure is stored in the first 15 columns of the matrix and transitional structure is stored in the last 12 columns of the matrix.

Usage

```
DBNmat
```

Format

A binary matrix with 27 rows and 27 columns representing an adjacency matrix of a DBN. Rows and columns of the matrix correspond to 15 variables of a DBN across 2 time slices.

DBNscore — *Calculating the BGe/BDe score of a single DBN*

Description

This function calculates the score of a DBN defined by its compact adjacency matrix. Acceptable data matrices are homogeneous with all variables of the same type: continuous, binary or categorical. The BGe score is evaluated in the case of continuous data and the BDe score is evaluated for binary and categorical variables.

Usage

```
DBNscore(scorepar, incidence)
```

Arguments

`scorepar`	an object of class `scoreparameters`, containing the data and scoring parameters; see constructor function `scoreparameters`
`incidence`	a square matrix, representing initial and transitional structure of a DBN; the size of matrix is 2*nsmall+bgn, where nsmall is the number of variables per time slice excluding static nodes and bgn is the number of static variables the matrix entries are in `{0,1}` such that `incidence[i,j]` equals 1 if there is a directed edge from node `i` to node `j` in the DAG and `incidence[i,j]` equals 0 otherwise

Value

the log of the BGe or BDe score of the DBN

Examples

```
testscore<-scoreparameters(15, "bge", DBNdata, bgnodes=c(1,2,3), DBN=TRUE,
                        dbnpar=list(slices=5, stationary=TRUE))
DBNscore(testscore, DBNmat)
```

DBNunrolled — *An unrolled adjacency matrix of a dynamic Bayesian network*

Description

An adjacency matrix representing the ground truth DBN used to generate a synthetic dataset `DBNdata`. The matrix is an unrolled representation of a 2-step DBN, such that the static variables are represented in the first 3 columns/rows of the matrix.

Usage

```
DBNunrolled
```

Format

A binary matrix with 63 rows and 63 columns representing an adjacency matrix of a DBN. Rows and columns of the matrix correspond to 15 variables (s1, s2, s3, v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12) of a DBN across 5 time slices.

---

edges.posterior *Estimating posterior probabilities of single edges*

Description

This function estimates the posterior probabilities of edges by averaging over a sample of DAGs obtained via an MCMC scheme.

Usage

```
edges.posterior(MCMCchain, pdag = FALSE, burnin = 0.2, endstep = 1)
```

Arguments

MCMCchain	list of square matrices with elements in {0,1} and representing adjacency matrices of a sample of DAGs obtained via an MCMC scheme (objects of classes 'MCMCtrace' or 'MCMCres' are also valid data types)
pdag	logical, if TRUE (FALSE by default) all DAGs in the MCMCchain are first converted to equivalence class (CPDAG) before the averaging
burnin	(optional) number between 0 and 1, indicates the percentage of the samples which will be discarded as 'burn-in' of the MCMC chain; the rest of the samples will be used to calculate the posterior probabilities; 0.2 by default
endstep	(optional) number between 0 and 1; 1 by default

Value

a square matrix with dimensions equal to the number of variables; each entry [i,j] is an estimate of the posterior probability of the edge from node i to node j

Examples

```
Bostonscore<-scoreparameters(14, "bge", Boston)
## Not run:
samplefit<-orderMCMC(Bostonscore, iterations=25000,chainout=TRUE)
edgesposterior<-edges.posterior(samplefit, pdag=TRUE, burnin=0.2)

## End(Not run)
```

full2compact *Deriving a compact adjacency matrix of a DBN*

Description

This function transforms an unrolled adjacency matrix of DBN into a compact representation

Usage

```
full2compact(DBNmat, n.dynamic, n.static = 0)
```

Arguments

DBNmat	a square matrix, representing the structure of an unrolled DBN; the size of matrix is n.slices*n.dynamic+n.static; all static variables are assumed to be in the first n.static rows and columns of the matrix
n.dynamic	integer, number of dynamic variables in each time slice
n.static	integer, number of static variables

Examples

```
full2compact(DBNunrolled,n.dynamic=12,n.static=3)
```

graph2m *Deriving an adjacency matrix of a graph*

Description

This function derives the adjacency matrix corresponding to a graph object

Usage

```
graph2m(g)
```

Arguments

g	graph, object of class graphNEL (package 'graph')

Value

a square matrix whose dimensions are the number of nodes in the graph g, where element [i,j] equals 1 if there is a directed edge from node i to node j in the graph g, and 0 otherwise

Examples

```
Asiagraph<-m2graph(Asiamat)
Asia.adj<-graph2m(Asiagraph)
```

gsim — *A simulated data set from a Gaussian continuous Bayesian network*

Description

A synthetic dataset containing 1000 observations generated from a random DAG with 100 continuous nodes.

Usage

```
gsim
```

Format

A data frame with 1000 rows representing observations of 100 continuous variables: V1, ..., V100

gsim100 — *A simulated data set from a Gaussian continuous Bayesian network*

Description

A synthetic dataset containing 100 observations generated from a random DAG with 100 continuous nodes.

Usage

```
gsim100
```

Format

A data frame with 100 rows representing observations of 100 continuous variables: V1, ..., V100

gsimmat — *An adjacency matrix of a simulated dataset*

Description

An adjacency matrix representing the ground truth DAG used to generate a synthetic dataset with observations of 100 continuous variables.

Usage

```
gsimmat
```

Format

A binary matrix with 100 rows and 100 columns representing an adjacency matrix of a DAG with 100 nodes: V1, ..., V100

iterations.check — *Performance assessment of iterative MCMC scheme against a known Bayesian network*

Description

This function calculates the number of true and false positives, the true positive rate, the structural Hamming distance and score for each iteration in the search procedure implemented in the function `iterativeMCMC`.

Usage

```
iterations.check(MCMCmult, truedag, cpdag = TRUE, pbarrier = 0.5, trans = TRUE)
```

Arguments

`MCMCmult`	an object which of class `MCMCmult` (output of the function `iterativeMCMC`)
`truedag`	ground truth DAG which generated the data used in the search procedure; represented by an object of class `graphNEL`
`cpdag`	logical, if TRUE (FALSE by default) all DAGs in the `MCMCmult` are first converted to their respective equivalence class (CPDAG) before the averaging if parameter `sample` set to TRUE
`pbarrier`	threshold such that only edges with a higher posterior probability will be retained in the directed graph summarising the sample of DAGs at each iteration from `MCMCmult` if parameter `sample` set to TRUE
`trans`	logical, for DBNs indicates if model comparions are performed for transition structure; when `trans` equals FALSE the comparison is performed for initial structures of estimated models and the ground truth DBN; for usual BNs the parameter is disregarded

Value

A matrix with the number of rows equal to the number of elements in `MCMCmult`, and 5 columns reporting for the maximally scoring DAG uncovered at each iteration (or for a summary over the sample of DAGs if `sample` parameter set to TRUE) the number of true positive edges ('TP'), the number of false positive edges ('FP'), the true positive rate ('TPR'), the structural Hamming distance ('SHD') and the score of the DAG ('score'). Note that the maximum estimated DAG as well as the true DAG are first converted to the corresponding equivalence class (CPDAG) when calculating the SHD.

Examples

```
gsim.score<-scoreparameters(100, "bge", gsim)
## Not run:
MAPestimate<-iterativeMCMC(gsim.score)
iterations.check(MAPestimate, gsimmat)

## End(Not run)
```

`iterativeMCMC` *Structure learning with an iterative order MCMC algorithm on an expanded search space*

Description

This function implements an iterative search for the maximum a posteriori (MAP) DAG, by means of order MCMC. At each iteration, the current search space is expanded by allowing each node to have up to one additional parent not already included in the search space. By default the initial search space is obtained through the PC-algorithm (using the functions `skeleton` and `pc` from the 'pcalg' package [Kalisch et al, 2012]). At each iteration order MCMC is employed to search for the MAP DAG. The edges in the MAP DAG are added to the initial search space to provide the search space for the next iteration. The algorithm iterates until no further score improvements can be achieved by expanding the search space. The final search space may be used for the sampling versions of `orderMCMC` and `partitionMCMC`.

Usage

```
iterativeMCMC(
  scorepar,
  plus1it = NULL,
  moveprobs = NULL,
  MAP = TRUE,
  posterior = 0.5,
  iterations = NULL,
  stepsave = NULL,
  softlimit = 9,
  hardlimit = 12,
  alpha = 0.05,
```

```
  gamma = 1,
  startspace = NULL,
  blacklist = NULL,
  verbose = TRUE,
  chainout = FALSE,
  scoreout = FALSE,
  cpdag = FALSE,
  mergetype = "skeleton",
  addspace = NULL,
  scoretable = NULL,
  startorder = NULL,
  accum = FALSE
)
```

Arguments

scorepar	an object of class scoreparameters, containing the data and scoring parameters; see constructor function scoreparameters
plus1it	(optional) integer, a number of iterations of search space expansion; by default the algorithm iterates until no score improvement can be achieved by further expanding the search space
moveprobs	(optional) a numerical vector of 4 values in {0,1} corresponding to the probabilities of the following MCMC moves in the order space: • exchanging 2 random nodes in the order • exchanging 2 adjacent nodes in the order • placing a single node elsewhere in the order • staying still
MAP	logical, if TRUE (default) the search targets the MAP DAG (a DAG with maximum score), if FALSE at each MCMC step a DAG is sampled from the order proportionally to its score; when expanding a search space when MAP=TRUE all edges from the maximum scoring DAG are added to the new space, when MAP=FALSE only edges with posterior probability higher than defined by parameter posterior are added to the search space
posterior	logical, when MAP set to FALSE defines posterior probability threshold for adding the edges to the search space
iterations	(optional) integer, the number of MCMC steps, the default value is $3.5n^2 \log n$
stepsave	(optional) integer, thinning interval for the MCMC chain, indicating the number of steps between two output iterations, the default is iterations/1000
softlimit	(optional) integer, limit on the size of parent sets beyond which adding undirected edges is restricted; below this limit edges are added to expand the parent sets based on the undirected skeleton of the MAP DAG (or from its CPDAG, depending on the parameter mergecp), above the limit only the directed edges are added from the MAP DAG; the limit is 9 by default
hardlimit	(optional) integer, limit on the size of parent sets beyond which the search space is not further expanded to prevent long runtimes; the limit is 12 by default

`alpha` (optional) numerical significance value in `{0,1}` for the conditional independence tests in the PC-stage (by default 0.4 for $n < 50$, $20/n$ for $n > 50$)

`gamma` (optional) tuning parameter which transforms the score by raising it to this power, 1 by default

`startspace` (optional) a square matrix, of dimensions equal to the number of nodes, which defines the search space for the order MCMC in the form of an adjacency matrix; if NULL, the skeleton obtained from the PC-algorithm will be used; if `startspace[i,j]` equals to 1 (0) it means that the edge from node `i` to node `j` is included (excluded) from the search space; to include an edge in both directions, both `startspace[i,j]` and `startspace[j,i]` should be 1

`blacklist` (optional) a square matrix, of dimensions equal to the number of nodes, which defines edges to exclude from the search space; if `blacklist[i,j]` equals to 1 it means that the edge from node `i` to node `j` is excluded from the search space

`verbose` logical, if TRUE (default) prints messages on the progress of execution

`chainout` logical, if TRUE the saved MCMC steps are returned, FALSE by default

`scoreout` logical, if TRUE the search space from the last plus1 iterations and the corresponding score tables are returned, FALSE by default

`cpdag` logical, if set to TRUE the equivalence class (CPDAG) found by the PC algorithm is used as a search space, when FALSE (default) the undirected skeleton used as a search space

`mergetype` defines which edges are added to the search space at each expansion iteration; if set to

`addspace` (optional) a square matrix, of dimensions equal to the number of nodes, which defines the edges, which are added at to the search space only at the first iteration of iterative seach and do not necessarily stay afterwards; defined in the form of an adjacency matrix; if `addspace[i,j]` equals to 1 (0) it means that the edge from node `i` to node `j` is included (excluded) from the search space; to include an edge in both directions, both `addspace[i,j]` and `addspace[j,i]` should be 1

`scoretable` (optional) list of score tables which has to match startspace and addspace

`startorder` (optional) integer vector of length n, which will be used as the starting order in the MCMC algorithm, the default order is `c(1:n)`

`accum` logical, when TRUE at each search step expansion new edges are added to the current search space; when FALSE (default) the new edges are added to the starting space

- "dag", then edges from maximum scoring DAG are added;
- "cpdag", then the maximum scoring DAG is first converted to the CPDAG, from which all edges are added to the search space;
- "skeleton", then the maximum scoring DAG is first converted to the skeleton, from which all edges are added to the search space

Value

Depends on the logical parameters `chainout` and `scoreout`. If both are FALSE (default), an oject of class `MCMCmax`, containing a list of 4 elements:

- DAG - the adjacency matrix of the DAG with maximal score
- order - an order it belongs to
- score - the score of the reported DAG
- it - the iteration at which maximum was reached

If `chainout` is TRUE an object of class `MCMCtrace` is additionally returned, contains 4 lists (each of the 4 lists has length `iterations/stepsave`, i.e. the number of saved MCMC steps):

- incidence - contains a list of adjacency matrices of DAGs sampled at each step of MCMC
- DAGscores - contains a list of scores of DAGs sampled at each step of MCMC
- orderscores - contains a list of scores of orders of DAGs sampled at each step of MCMC
- order - contains a list of permutations of the nodes of DAGs sampled at each step of MCMC

If `scoreout` is TRUE an object of class `MCMCspace` is additionally returned, contains a list of 2 elements:

- adjacency - the adjacency matrix representing the search space
- scoretable - the list of score tables corresponding to this search space

References

Friedman N and Koller D (2003). A Bayesian approach to structure discovery in bayesian networks. Machine Learning 50, 95-125.

Kalisch M, Maechler M, Colombo D, Maathuis M and Buehlmann P (2012). Causal inference using graphical models with the R package pcalg. Journal of Statistical Software 47, 1-26.

Geiger D and Heckerman D (2002). Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. The Annals of Statistics 30, 1412-1440.

Kuipers J, Moffa G and Heckerman D (2014). Addendum on the scoring of Gaussian directed acyclic graphical models. The Annals of Statistics 42, 1689-1691.

Spirtes P, Glymour C and Scheines R (2000). Causation, Prediction, and Search, 2nd edition. The MIT Press.

Examples

```
## Not run:
Bostonpar<-scoreparameters(14,"bge",Boston)
itfit<-iterativeMCMC(Bostonpar, chainout=TRUE, scoreout=TRUE)
plot(itfit)

## End(Not run)
```

m2graph *Deriving a graph from an adjacancy matrix*

Description

This function derives a graph object corresponding to an adjacency matrix

Usage

```
m2graph(adj, nodes = NULL)
```

Arguments

adj	square adjacency matrix with elements in {0,1}, representing a graph
nodes	(optional) labels of the nodes, c(1:n) are used by default

Value

object of class graphNEL (package 'graph'); if element adj[i,j] equals 1, then there is a directed edge from node i to node j in the graph, and no edge otherwise

Examples

```
m2graph(Asiamat)
```

orderMCMC *Structure learning with the order MCMC algorithm*

Description

This function implements the order MCMC algorithm for the structure learning of Bayesian networks. This function can be used for MAP discovery and for sampling from the posterior distribution of DAGs given the data. Due to the superexponential size of the search space as the number of nodes increases, the MCMC search is performed on a reduced search space. By default the search space is limited to the skeleton found through the PC algorithm by means of conditional independence tests (using the functions skeleton and pc from the 'pcalg' package [Kalisch et al, 2012]). It is also possible to define an arbitrary search space by inputting an adjacency matrix, for example estimated by partial correlations or other network algorithms. Also implemented is the possibility to expand the default or input search space, by allowing each node in the network to have one additional parent. This offers improvements in the learning and sampling of Bayesian networks.

Usage

```
orderMCMC(
  scorepar,
  MAP = TRUE,
  plus1 = TRUE,
  startspace = NULL,
  blacklist = NULL,
  startorder = NULL,
  scoretable = NULL,
  moveprobs = NULL,
  iterations = NULL,
  stepsave = NULL,
  alpha = 0.05,
  cpdag = FALSE,
  gamma = 1,
  hardlimit = ifelse(plus1, 15, 22),
  chainout = TRUE,
  scoreout = FALSE,
  verbose = FALSE
)
```

Arguments

scorepar	an object of class scoreparameters, containing the data and score parameters, see constructor function scoreparameters
MAP	logical, if TRUE (default) the search targets the MAP DAG (a DAG with maximum score), if FALSE at each MCMC step a DAG is sampled from the order proportionally to its score
plus1	logical, if TRUE (default) the search is performed on the extended search space
startspace	(optional) a square matrix, of dimensions equal to the number of nodes, which defines the search space for the order MCMC in the form of an adjacency matrix. If NULL, the skeleton obtained from the PC-algorithm will be used. If startspace[i,j] equals to 1 (0) it means that the edge from node i to node j is included (excluded) from the search space. To include an edge in both directions, both startspace[i,j] and startspace[j,i] should be 1.
blacklist	(optional) a square matrix, of dimensions equal to the number of nodes, which defines edges to exclude from the search space. If blacklist[i,j] equals to 1 it means that the edge from node i to node j is excluded from the search space.
startorder	(optional) integer vector of length n, which will be used as the starting order in the MCMC algorithm, the default order is c(1:n)
scoretable	(optional) list of score tables calculated for example by the last iteration of the function iterativeMCMC, to avoid their recomputation The score tables must match the permissible parents in the search space defined by the startspace parameter.
moveprobs	(optional) a numerical vector of 4 values in {0,1} corresponding to the probabilities of the following MCMC moves in the order space

- exchanging 2 random nodes in the order
- exchanging 2 adjacent nodes in the order
- placing a single node elsewhere in the order
- staying still

`iterations` (optional) integer, the number of MCMC steps, the default value is $5n^2 \log n$

`stepsave` (optional) integer, thinning interval for the MCMC chain, indicating the number of steps between two output iterations, the default is `iterations/1000`

`alpha` (optional) numerical significance value in `{0,1}` for the conditional independence tests at the PC algorithm stage (by default 0.4 for $n < 50$, $20/n$ for $n > 50$)

`cpdag` (optional) logical, if TRUE the CPDAG returned by the PC algorithm will be used as the search space, if FALSE (default) the full undirected skeleton will be used as the search space

`gamma` (optional) tuning parameter which transforms the score by raising it to this power, 1 by default

`hardlimit` (optional) integer, limit on the size of parent sets in the search space

`chainout` logical, if TRUE the saved MCMC steps are returned, TRUE by default

`scoreout` logical, if TRUE the search space and score tables are returned, FALSE by default

`verbose` logical, if TRUE messages about the algorithm's progress will be printed, FALSE by default

Value

Depends on the logical parameters `chainout` and `scoreout`. If both are FALSE (default), an object of class `MCMCmax`, containing a list of 3 elements:

- DAG - the adjacency matrix of the DAG with maximal score
- order - an order it belongs to
- score - the score of the reported DAG

If `chainout` is TRUE an object of class `MCMCtrace` is additionally returned, contains 4 lists (each of the 4 lists has length `iterations/stepsave`, i.e. the number of saved MCMC steps):

- incidence - contains a list of adjacency matrices of DAGs sampled at each step of MCMC
- DAGscores - contains a list of scores of DAGs sampled at each step of MCMC
- orderscores - contains a list of scores of orders of DAGs sampled at each step of MCMC
- order - contains a list of permutations of the nodes of DAGs sampled at each step of MCMC

If `scoreout` is TRUE an object of class `MCMCspace` is additionally returned, contains a list of 2 elements:

- adjacency - the adjacency matrix representing the search space
- scoretable - the list of score tables corresponding to this search space

References

Friedman N and Koller D (2003). A Bayesian approach to structure discovery in bayesian networks. Machine Learning 50, 95-125.

Kalisch M, Maechler M, Colombo D, Maathuis M and Buehlmann P (2012). Causal inference using graphical models with the R package pcalg. Journal of Statistical Software 47, 1-26.

Geiger D and Heckerman D (2002). Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. The Annals of Statistics 30, 1412-1440.

Kuipers J, Moffa G and Heckerman D (2014). Addendum on the scoring of Gaussian acyclic graphical models. The Annals of Statistics 42, 1689-1691.

Spirtes P, Glymour C and Scheines R (2000). Causation, Prediction, and Search, 2nd edition. The MIT Press.

Examples

```
## Not run:
#find a MAP DAG with search space defined by PC and plus1 neighbourhood
Bostonscore<-scoreparameters(14,"bge",Boston)
#estimate MAP DAG
orderMAPfit<-orderMCMC(Bostonscore)
summary(orderMAPfit)
#sample DAGs from the posterior distribution
ordersamplefit<-orderMCMC(Bostonscore,MAP=FALSE,chainout=TRUE)
plot(ordersamplefit)

## End(Not run)
```

`partitionMCMC` *DAG structure sampling with partition MCMC*

Description

This function implements the partition MCMC algorithm for the structure learning of Bayesian networks. This procedure provides an unbiased sample from the posterior distribution of DAGs given the data. The search space can be defined either by a preliminary run of the function `iterativeMCMC` or by a given adjacency matrix (which can be the full matrix with zero on the diagonal, to consider the entire space of DAGs, feasible only for a limited number of nodes).

Usage

```
partitionMCMC(
  scorepar,
  startspace = NULL,
  blacklist = NULL,
  scoretable = NULL,
  startDAG = NULL,
```

```
  moveprobs = NULL,
  iterations = NULL,
  stepsave = NULL,
  gamma = 1,
  verbose = TRUE
)
```

Arguments

scorepar	an object of class scoreparameters, containing the data and scoring parameters; see constructor function scoreparameters.
startspace	(optional) a square matrix, of dimensions equal to the number of nodes, which defines the search space for the order MCMC in the form of an adjacency matrix; if NULL, the skeleton obtained from the PC-algorithm will be used. If startspace[i,j] equals to 1 (0) it means that the edge from node i to node j is included (excluded) from the search space. To include an edge in both directions, both startspace[i,j] and startspace[j,i] should be 1.
blacklist	(optional) a square matrix, of dimensions equal to the number of nodes, which defines edges to exclude from the search space; if blacklist[i,j]=1 it means that the edge from node i to node j is excluded from the search space
scoretable	(optional) list of score tables; for example calculated at the last iteration of the function iterativeMCMC, to avoid their recomputation; the score tables must match the permissible parents in the search space defined by the startspace parameter
startDAG	(optional) an adjacency matrix of dimensions equal to the number of nodes, representing a DAG in the search space defined by startspace. If startspace is defined but startDAG is not, an empty DAG will be used by default
moveprobs	(optional) a numerical vector of 5 values in {0,1} corresponding to the following MCMC move probabilities in the space of partitions: • swap any two elements from different partition elements • swap any two elements in adjacent partition elements • split a partition element or join one • move a single node into another partition element or into a new one • stay still
iterations	(optional) integer, the number of MCMC steps, the default value is $8n^2 \log n$
stepsave	(optional) integer, thinning interval for the MCMC chain, indicating the number of steps between two output iterations, the default is iterations/1000
gamma	(optional) tuning parameter which transforms the score by raising it to this power, 1 by default
verbose	logical, if set to TRUE (default) messages about progress will be printed

Value

an object of class MCMCtrace, which contains a list of 5 elements (each list contains iterations/stepsave elements):

- incidence - contains a list of adjacency matrices of DAGs sampled at each step of MCMC
- DAGscores - contains a list of scores of DAGs sampled at each step of MCMC
- partitionscores - contains a list of scores of partitions of DAGs sampled at each step of MCMC
- order - contains a list of permutations of the nodes in partitions of DAGs sampled at each step of MCMC
- partition - contains a list of partitions of DAGs sampled at each step of MCMC

References

Kuipers J and Moffa G (2017). Partition MCMC for inference on acyclic digraphs. Journal of the American Statistical Association 112, 282-299.

Geiger D and Heckerman D (2002). Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. The Annals of Statistics 30, 1412-1440.

Heckerman D and Geiger D (1995). Learning Bayesian networks: A unification for discrete and Gaussian domains. In Eleventh Conference on Uncertainty in Artificial Intelligence, pages 274-284.

Kalisch M, Maechler M, Colombo D, Maathuis M and Buehlmann P (2012). Causal inference using graphical models with the R package pcalg. Journal of Statistical Software 47, 1-26.

Kuipers J, Moffa G and Heckerman D (2014). Addendum on the scoring of Gaussian directed acyclic graphical models. The Annals of Statistics 42, 1689-1691.

Examples

```
## Not run:
myScore<-scoreparameters(14, "bge", Boston)
partfit<-partitionMCMC(myScore)
plot(partfit)

## End(Not run)
```

---

plotDBN *Plotting a DBN*

Description

This function can be used for plotting initial and transition structures of a dynamic Bayesian network.

Usage

```
plotDBN(DBN, struct = c("init", "trans"), n.dynamic, n.static)
```

Arguments

DBN	binary matrix (or a graph object) representing a 2-step DBN (compact or unrolled)
struct	option used to determine if the initial or the transition structure should be plotted; accaptable values are init or trans
n.dynamic	number of dynamic variables in one time slice of a DBN
n.static	number of static variables in one time slice of a DBN; note that for function to work correctly all static variables have to be in the first n.static columns of the matrix

Examples

```
## Not run:
plotDBN(DBNmat, "trans", n.dynamic=12,n.static=3)
plotDBN(DBNmat, "init", n.dynamic=12,n.static=3)

## End(Not run)
```

---

plotdiffs *Plotting difference between two graphs*

Description

This function plots an estimated graph such that the edges which are different to the ground truth graph are highlighted.

Usage

```
plotdiffs(edag, truedag, clusters = NULL)
```

Arguments

edag	object of class graphNEL (or its adjacency matrix), representing estimated structure (not necessarily acyclic) to be compared to the ground truth graph
truedag	object of class graphNEL (or its adjacency matrix), representing the ground truth structure (not necessarily acyclic)
clusters	(optional) a list of nodes to be represented on the graph as clusters

Value

plots the graph which includes edges from edag and truedag, however edges which are different in edag compared to truedag are coloured according to the type of a mistake: false positive with red, false negative with dashed grey, error in direction with magenta

Examples

```
Asiascore<-scoreparameters(8,"bde",Asia)
Asiamap<-orderMCMC(Asiascore)
plotdiffs(Asiamap$max$DAG,Asiamat)
Asiacp<-pcalg::dag2cpdag(m2graph(Asiamat))
mapcp<-pcalg::dag2cpdag(m2graph(Asiamap$max$DAG))
plotdiffs(mapcp,Asiacp)
```

---

plotdiffs.DBN *Plotting difference between two DBNs*

Description

This function plots an estimated DBN such that the edges which are different to the ground truth DBN are highlighted.

Usage

```
plotdiffs.DBN(
  eDBN,
  trueDBN,
  struct = c("init", "trans"),
  n.dynamic,
  n.static = 0
)
```

Arguments

eDBN	object of class graphNEL (or its adjacency matrix), representing estimated structure (not necessarily acyclic) to be compared to the ground truth graph
trueDBN	object of class graphNEL (or its adjacency matrix), representing the ground truth structure (not necessarily acyclic)
struct	option used to determine if the initial or the transition structure should be plotted; accaptable values are init or trans
n.dynamic	number of dynamic variables in one time slice of a DBN
n.static	number of static variables in one time slice of a DBN; note that for function to work correctly all static variables have to be in the first n.static columns of the matrix

Value

plots the graph which includes edges from edag and truedag, however edges which are different in edag compared to truedag are coloured according to the type of a mistake: false positive with red, false negative with dashed grey, error in direction with magenta

Examples

```
dbnscore<-scoreparameters(15,"bge",DBNdata,
dbnpar = list(samestruct=TRUE, slices=5, stationary=TRUE),
DBN=TRUE,bgnodes=c(1,2,3))
## Not run:
orderDBNfit<-iterativeMCMC(dbnscore,chainout = TRUE, mergetype = "skeleton",scoreout=TRUE,alpha=0.4)
plotdiffs.DBN(orderDBNfit$max$DAG,DBNmat,struct="trans",n.dynamic=12,n.static=3)
plotdiffs.DBN(orderDBNfit$max$DAG,DBNmat,struct="init",n.dynamic=12,n.static=3)

## End(Not run)
```

`plotpcor` *Comparing posterior probabilitites of single edges based on two samples*

Description

This function can be used to compare posterior probabilities of edges in a graph based on two samples of graphs

Usage

```
plotpcor(
  edgepmat1,
  edgepmat2,
  highlight = 0.3,
  cut = 0.05,
  main = "",
  xlab = "sample 1",
  ylab = "sample 2"
)
```

Arguments

`edgepmat1`	binary matrix, representing posterior probabilities of single edges in a Bayesian network
`edgepmat2`	binary matrix, representing posterior probabilities of single edges in a Bayesian network
`highlight`	numeric, defines maximum acceptable difference between posterior probabilities of an edge in two samples; points corresponding to higher differences are highlighted
`cut`	numeric value corresponding to a minimum posterior probabilitity which is included into calculation of squared correlation and MSE
`main`	character string, a title for the plot
`xlab`	character string, a title for the x-axis
`ylab`	character string, a title for the y-axis

Value

squared correlation and MSE of posterior probabilities higher than the value defined by the argument cut; also plots the posterior probabilitites from two samples against each other

Examples

```
Asiascore<-scoreparameters(8, "bde", Asia)
orderfit1<-orderMCMC(Asiascore,plus1=FALSE,iterations=10000)
orderfit2<-orderMCMC(Asiascore,plus1=FALSE,iterations=30000)
pedges1<-edges.posterior(orderfit1)
pedges2<-edges.posterior(orderfit2)
plotpcor(pedges1,pedges2)
```

---

plotpedges *Plotting posterior probabilities of single edges*

Description

This function plots posterior probabilities of all possible edges in the graph as a function of MCMC iterations. It can be used for convergence diagnostics of MCMC sampling algorithms order MCMC and partition MCMC.

Usage

```
plotpedges(MCMCtrace, cutoff = 0.2, pdag = FALSE, onlyedges = NULL)
```

Arguments

MCMCtrace	an object of class MCMCres
cutoff	number representing a threshold of posterior probability below which lines will not be plotted
pdag	logical, when true DAGs in a sample will be first coverted to CPDAGs
onlyedges	(optional) binary matrix, only edges corresponding to entries whuch equal 1 will be plotted

Value

plots the graph which includes edges from edag and truedag, however edges which are different in edag compared to truedag are coloured according to the type of a mistake: false positive with red, false negative with dashed grey, error in direction with magenta

Examples

```
score100<-scoreparameters(8, "bde", Asia[1:100,])
orderfit100<-orderMCMC(score100,plus1=TRUE)
score5000<-scoreparameters(8, "bde", Asia)
orderfit5000<-orderMCMC(score5000,plus1=TRUE)
## Not run:
plotpedges(orderfit100, pdag=TRUE)
plotpedges(orderfit5000, pdag=TRUE)

## End(Not run)
```

`sample.check` *Performance assessment of sampling algorithms against a known Bayesian network*

Description

This function calculates the number of true and false positives and the structural Hamming distance between a ground truth DAG and a directed graph summarising a sample of DAGs obtained from an MCMC scheme, as the posterior probability threshold is varied

Usage

```
sample.check(
  MCMCchain,
  truedag,
  pbarrier = c(0.99, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2),
  pdag = TRUE,
  burnin = 0.2,
  trans = TRUE
)
```

Arguments

`MCMCchain`	an object of class MCMCres, representing the output of structure sampling function `partitionMCMC` or `orderMCMC` (the latter when parameter `chainout`=TRUE)
`truedag`	ground truth DAG which generated the data used in the search procedure; represented by an object of class `graphNEL`
`pbarrier`	(optional) a vector of numeric values between 0 and 1, defining posterior probabilities according to which the edges of assessed structures are drawn, please note very low barriers can lead to very dense structures; by default $pbarrier = c(0.99, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2)$
`pdag`	logical, if TRUE (default) all DAGs in the MCMCchain are first converted to equivalence class (CPDAG) before the averaging
`burnin`	(optional) number between `0` and `1`, indicates the percentage of the samples which will be the discarded as 'burn-in' of the MCMC chain; the rest of the samples will be used to calculate the posterior probabilities; 0.2 by default

trans	logical, for DBNs indicates if model comparions are performed for transition structure; when trans equals FALSE the comparison is performed for initial structures of estimated models and the ground truth DBN; for usual BNs the parameter is disregarded

Value

A matrix with the number of rows equal to the number of posterior thresholds tested, and 4 columns reporting for each thresholded directed graphs the number of true positive edges ('TP'), the number of false positive edges ('FP'), the structural Hamming distance ('SHD') and the posterior threshold

Examples

```
gsim.score<-scoreparameters(100, "bge", gsim)
## Not run:
mapest<-iterativeMCMC(gsim.score)
ordersample<-orderMCMC(gsim.score, MAP=FALSE, startspace=mapest$endspace)
sample.check(ordersample, gsimmat)

## End(Not run)
```

---

scoreagainstDAG *Calculating the score of a sample against a DAG*

Description

This function calculates the score of a given sample against a DAG represented by its incidence matrix.

Usage

```
scoreagainstDAG(scorepar, incidence, datatoscore = NULL, marginalise = FALSE)
```

Arguments

scorepar	an object of class scoreparameters; see constructor function scoreparameters
incidence	a square matrix of dimensions equal to the number of variables with entries in {0,1}, representing the adjacency matrix of the DAG against which the score is calculated
datatoscore	(optional) a matrix (vector) containing binary (for BDe score) or continuous (for the BGe score) observations (or just one observation) to be scored; the number of columns should be equal to the number of variables in the Bayesian network, the number of rows should be equal to the number of observations; by default all data from scorepar parameter is used
marginalise	(optional for continuous data), whether to use the posterior mean for scoring (default) or to marginalise over the posterior distribution (more computationally costly)

Value

the log of the BDe/BGe score of given observations against a DAG

References

Heckerman D and Geiger D, (1995). Learning Bayesian networks: A unification for discrete and Gaussian domains. In Eleventh Conference on Uncertainty in Artificial Intelligence, pages 274-284, 1995.

Examples

```
Asiascore<-scoreparameters(8, "bde", Asia[1:100,]) #we wish to score only first 100 observations
scoreagainstDAG(Asiascore, Asiamat)
```

---

scoreparameters *Initialising score object*

Description

This function returns an object of class scoreparameters containing the data and parameters needed for calculation of the BDe/BGe score, or a user defined score.

Usage

```
scoreparameters(
  n,
  scoretype = c("bge", "bde", "bdecat"),
  data,
  weightvector = NULL,
  bgnodes = NULL,
  bgepar = list(am = 1, aw = NULL),
  bdepar = list(chi = 0.5, edgepf = 2),
  bdecatpar = list(chi = 0.5, edgepf = 2),
  dbnpar = list(samestruct = TRUE, slices = 2, stationary = TRUE),
  usrpar = list(pctesttype = c("bge", "bde", "bdecat")),
  edgepmat = NULL,
  nodeslabels = NULL,
  DBN = FALSE
)
```

Arguments

n	number of nodes (variables) in the Bayesian network; for DBN n represents the number of nodes per one time slice (including static variables)
scoretype	the score to be used to assess the DAG structure: "bge" for Gaussian data, "bde" for binary data, "bdecat" for categorical data, "dbn" for dynamic Bayesian networks, "usr" for a user defined score

`data` the data matrix with n columns (the number of variables) and a number of rows equal to the number of observations

`weightvector` (optional) a numerical vector of positive values representing the weight of each observation; should be NULL(default) for non-weighted data

`bgnodes` (optional) a numerical vector which contains numbers of columns in the data defining background nodes, background nodes are nodes which have no parents but can be parents of other nodes in the network; in case of DBNs bgnodes represent static variables which do not change over time

`bgepar` a list which contains parameters for BGe score:

- am (optional) a positive numerical value, 1 by default
- aw (optional) a positive numerical value should be more than `n+1`, `n+am+1` by default

`bdepar` a list which contains parameters for BDe score for binary data:

- chi (optional) a positive number of prior pseudo counts used by the BDe score, 0.5 by default
- edgepf (optional) a positive numerical value providing the edge penalization factor to be combined with the BDe score, 2 by default

`bdecatpar` a list which contains parameters for BDe score for categorical data:

- chi (optional) a positive number of prior pseudo counts used by the BDe score, 0.5 by default
- edgepf (optional) a positive numerical value providing the edge penalization factor to be combined with the BDe score, 2 by default

`dbnpar` which type of score to use for the slices

- samestruct logical, when TRUE the structure of the first time slice is assumed to be the same as internal structure of all other time slices
- slices integer representing the number of time slices in a DBN

`usrpar` a list which contains parameters for the user defined score

- pctesttype (optional) conditional independence test ("bde","bge","bdecat","usrCItest")
- suffStat (optional) a list containing sufficient statistics for the CI test
- otherpars (optional) a list containing other parameters needed for score evaluation

`edgepmat` (optional) a matrix of positive numerical values providing the per edge penalization factor to be added to the score, NULL by default

`nodeslabels` (optional) a vector of characters which denote the names of nodes in the Bayesian network; by default column names of the data will be taken

`DBN` logical, when TRUE the score is initialized for a dynamic Baysian network; FALSE by default

Value

an object of class `scoreparameters`, which includes all necessary information for calculating the BDe/BGe score

References

Geiger D and Heckerman D (2002). Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. The Annals of Statistics 30, 1412-1440.

Kuipers J, Moffa G and Heckerman D (2014). Addendum on the scoring of Gaussian acyclic graphical models. The Annals of Statistics 42, 1689-1691.

Heckerman D and Geiger D (1995). Learning Bayesian networks: A unification for discrete and Gaussian domains. In Eleventh Conference on Uncertainty in Artificial Intelligence, pages 274-284.

Scutari M (2016). An Empirical-Bayes Score for Discrete Bayesian Networks. Journal of Machine Learning Research 52, 438-448

Examples

```
myDAG<-pcalg::randomDAG(20, prob=0.15, lB = 0.4, uB = 2)
myData<-pcalg::rmvDAG(200, myDAG)
myScore<-scoreparameters(20, "bge", myData)
```

Index