Package 'BiDAG'

January 20, 2025

Type Package

Title Bayesian Inference for Directed Acyclic Graphs

Version 2.1.4

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

Maintainer Polina Suter <polina.suter@gmail.com>

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, G. Moffa (2022) <doi:10.1080/10618600.2021.2020127>,
N. Friedman and D. Koller (2003) <doi:10.1023/A:1020249912095>,
J. Kuipers and G. Moffa (2017) <doi:10.1080/01621459.2015.1133426>,
M. Kalisch et al. (2012) <doi:10.18637/jss.v047.i11>,
D. Geiger and D. Heckerman (2002) <doi:10.1214/aos/1035844981>,
P. Suter, J. Kuipers, G. Moffa, N.Beerenwinkel (2023) <doi:10.18637/jss.v105.i09>.

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

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports Rcpp (>= 0.12.7), methods, graph, Rgraphviz, RBGL, pcalg, graphics, Matrix, coda

LinkingTo Rcpp

RoxygenNote 7.2.0

Encoding UTF-8

LazyData TRUE

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-05-16 12:46:02 UTC

Contents

`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

https://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

https://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.

`bidag2coda` *Converting a single BiDAG chain to mcmc object*

Description

This function converts a single object of one of the BiDAG classes, namely 'orderMCMC' or 'partitionMCMC' to an object of class 'mcmc'. This object can be further used for convergence and mixing diagnostics implemented in the package coda

Usage

```
bidag2coda(
  MCMCtrace,
  edges = FALSE,
  pdag = TRUE,
  p = 0.1,
  burnin = 0.2,
  window = 100,
  cumulative = FALSE
)
```

Arguments

`MCMCtrace`	object of class `orderMCMC` or `partitionMCMC`
`edges`	logical, when FALSE (default), then only DAG score trace is extracted; when TRUE, a trace of posterior probabilities is extracted for every edge (based on the sampled DAGs defined by parameters 'window' and 'cumulative') resulting in up to n^2 trace vectors, where n is the number of nodes in the network
`pdag`	logical, when edges=TRUE, defines if the DAGs are converted to CPDAGs prior to computing posterior probabilities; ignored otherwise
`p`	numeric, between 0 and 1; defines the minimum probability for including posterior traces in the returned objects (for probabilities close to 0 PRSF diagnostics maybe too conservative)
`burnin`	numeric 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
`window`	integer, defines a number of DAG samples for averaging and computing edges' posterior probabilities; ignored when edges=FALSE
`cumulative`	logical, indicates if posterior probabilities should be calculated based on a cumulative sample of DAGs, where 25% of the first samples are discarded

Value

Object of class `mcmc` from the package **coda**

Author(s)

Polina Suter

Examples

```
## Not run:
library(coda)
myscore<-scoreparameters("bde",Asia)
ordersample<-sampleBN(myscore,"order")
order_mcmc<-bidag2coda(ordersample)
par(mfrow=c(1,2))
densplot(order_mcmc)
traceplot(order_mcmc)

## End(Not run)
```

`bidag2codalist` *Converting multiple BiDAG chains to mcmc.list*

Description

This function converts a list of objects of classes 'orderMCMC' or 'partitionMCMC' to an object of class 'mcmc.list'. This object can be further used for convergence and mixing diagnostics implemented in the R-package coda.

Usage

```
bidag2codalist(
  MCMClist,
  edges = FALSE,
  pdag = TRUE,
  p = 0.1,
  burnin = 0.2,
  window = 10,
  cumulative = FALSE
)
```

Arguments

`MCMClist`	a list of objects of classes `orderMCMC` or `partitionMCMC`
`edges`	logical, when FALSE (default), then only DAG score trace is extracted; when TRUE, a trace of posterior probabilities is extracted for every edge (based on the sampled DAGs defined by parameters 'window' and 'cumulative') resulting in up to n^2 trace vectors, where n is the number of nodes in the network
`pdag`	logical, when edges=TRUE, defines if the DAGs are converted to CPDAGs prior to computing posterior probabilities; ignored otherwise

p	numeric, between 0 and 1; defines the minimum probability for including posterior traces in the returned objects (for probabilities close to 0, PRSF diagnostics maybe too conservative; the threshold above 0 is recommended)
burnin	numeric 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
window	integer, defines a number of DAG samples for averaging and computing edges' posterior probabilities; ignored when edges=FALSE
cumulative	logical, indicates if posterior probabilities should be calculated based on a cumulative sample of DAGs, where 25% of the first samples are discarded

Value

Object of class mcmc.list from the package **coda**

Author(s)

Polina Suter

References

Robert J. B. Goudie and Sach Mukherjee (2016). A Gibbs Sampler for Learning DAGs. J Mach Learn Res. 2016 Apr; 17(30): 1–39.

Examples

```
## Not run:
library(coda)
scoreBoston<-scoreparameters("bge",Boston)
ordershort<-list()
#run very short chains -> convergence issues
ordershort[[1]] <- sampleBN(scoreBoston, algorithm = "order", iterations=2000)
ordershort[[2]] <- sampleBN(scoreBoston, algorithm = "order", iterations=2000)
codashort_edges<-bidag2codalist(ordershort,edges=TRUE,pdag=TRUE,p=0.05,burnin=0.2,window=10)
gd_short<-gelman.diag(codashort_edges, transform=FALSE, autoburnin=FALSE, multivariate=FALSE)
length(which(gd_short$psrf[,1]>1.1))/(length(gd_short$psrf[,1]))
#=>more MCMC iterations are needed, try 100000

## End(Not run)
```

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, slices, b = 0)
```

Arguments

DBNmat	a square matrix, representing initial and transitional structure of a DBN; the size of matrix is 2*dyn+b
slices	integer, number of slices in an unrolled DBN
b	integer, number of static variables

Value

an adjacency matrix of an unrolled DBN

Examples

```
compact2full(DBNmat, slices=5, b=3)
```

---

compareDAGs *Comparing two graphs*

Description

This function compares one (estimated) graph to another graph (true graph), returning a vector of 8 values:

- the number of true positive edges ('TP') is the number of edges in the skeleton of 'egraph' which are also present in the skeleton of 'truegraph'
- the number of false positive edges ('FP') is the number of edges in the skeleton of 'egraph' which are absent in the skeleton of 'truegraph'
- the number of fralse negative edges ('FN') is the number of edges in the skeleton of 'truegraph' which are absent in the skeleton of 'egraph'
- structural Hamming distance ('SHD') between 2 graphs is computed as TP+FP+the number of edges with an error in direction
- TPR equals TP/(TP+FN)
- FPR equals FP/(TN+FP) (TN stands for true negative edges)
- FPRn equals FP/(TP+FN)
- FDR equals FP/(TP+FP)

Usage

```
compareDAGs(egraph, truegraph, cpdag = FALSE, rnd = 2)
```

Arguments

egraph	an object of class graphNEL (package 'graph'), representing the graph which should be compared to a ground truth graph or an ajecency matrix corresponding to the graph
truegraph	an object of class graphNEL (package 'graph'), representing the ground truth graph or an ajecency matrix corresponding to this graph
cpdag	logical, if TRUE (FALSE by default) both graphs are first converted to their respective equivalence class (CPDAG); this affects SHD calculation
rnd	integer, rounding integer indicating the number of decimal places (round) when computing TPR, FPR, FPRn and FDR

Value

a named numeric vector 8 elements: SHD, number of true positive edges (TP), number of false positive edges (FP), number of false negative edges (FN), true positive rate (TPR), false positive rate (FPR), false positive rate normalized to the true number of edges (FPRn) and false discovery rate (FDR)

Examples

```
Asiascore<-scoreparameters("bde", Asia)
## Not run:
eDAG<-learnBN(Asiascore,algorithm="order")
compareDAGs(eDAG$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"), b = 0)
```

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
b	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 b 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("bge", DBNdata, DBN=TRUE,
dbnpar=list(samestruct=TRUE, slices=5, b=3))
## Not run:
DBNfit<-learnBN(testscore, algorithm="orderIter",moveprobs=c(0.11,0.84,0.04,0.01))
compareDBNs(DBNfit$DAG,DBNmat, struct="trans", b=3)

## End(Not run)
```

---

connectedSubGraph *Deriving connected subgraph*

Description

This function derives an adjacency matrix of a subgraph whose nodes are connected to at least one other node in a graph

Usage

```
connectedSubGraph(adj)
```

Arguments

adj	square adjacency matrix with elements in {0,1}, representing a graph

Value

adjacency matrix of a subgraph of graph represented by 'adj' whose nodes have at least one connection

Examples

```
dim(gsimmat) #full graph contains 100 nodes
gconn<-connectedSubGraph(gsimmat) #removing disconnected nodes
dim(gconn) #connected subgraph contains 93 nodes
```

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

Author(s)

Jack Kuipers, Polina Suter, the code partly derived from the order MCMC implementation from Kuipers J, Moffa G (2017) <doi:10.1080/01621459.2015.1133426>

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("bde", Asia)
DAGscore(myScore, Asiamat)
```

DBNdata *Simulated data set from a 2-step dynamic Bayesian network*

Description

A synthetic dataset containing 100 observations generated from a random dynamic Bayesian network with 12 continuous dynamic nodes and 3 static nodes. The DBN includes 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

Author(s)

Polina Suter, Jack Kuipers

Examples

```
testscore<-scoreparameters("bge", DBNdata, DBN=TRUE, dbnpar=list(slices=5, b=3))
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.

edgep *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

```
edgep(MCMCchain, pdag = FALSE, burnin = 0.2, endstep = 1)
```

Arguments

MCMCchain	an object of class partitionMCMC, orderMCMC or iterativeMCMC, representing the output of structure sampling function partitionMCMC or orderMCMC (the latter when parameter chainout=TRUE;
pdag	logical, if TRUE (FALSE by default) all DAGs in the MCMCchain are first converted to equivalence class (CPDAG) before the averaging
burnin	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	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`

Author(s)

Polina Suter

Examples

```
Bostonscore<-scoreparameters("bge", Boston)
## Not run:
samplefit<-sampleBN(Bostonscore, "order")
edgesposterior<-edgep(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, b = 0)
```

Arguments

`DBNmat`	a square matrix, representing the structure of an unrolled DBN; the size of matrix is slices*dyn+b; all static variables are assumed to be in the first b rows and columns of the matrix
`b`	integer, number of static variables; 0 by default

Examples

```
full2compact(DBNunrolled,b=3)
```

getDAG *Extracting adjacency matrix (DAG) from MCMC object*

Description

This function extracts an adjacency matrix of a maximum scoring DAG from the result of the MCMC run.

Usage

```
getDAG(x, amat = TRUE, cp = FALSE)
```

Arguments

x	object of class 'orderMCMC','partitionMCMC' or 'iterativeMCMC'
amat	logical, when TRUE adjacency matrix is returned and object of class 'graphNEL' otherwise
cp	logical, when TRUE the CPDAG (equivalence class) is returned and DAG otherwise; FALSE by default

Value

adjacency matrix of a maximum scoring DAG (or CPDAG) discovered/sampled in one MCMC run

Examples

```
myscore<-scoreparameters("bge", Boston)
## Not run:
itfit<-learnBN(myscore,algorithm="orderIter")
maxEC<-getDAG(itfit,cp=TRUE)

## End(Not run)
```

getMCMCscore *Extracting score from MCMC object*

Description

This function extracts the score of a maximum DAG sampled in the MCMC run.

Usage

```
getMCMCscore(x)
```

Arguments

x	object of class 'orderMCMC','partitionMCMC' or 'iterativeMCMC'

Value

a score of a maximum-scoring DAG found/sampled in one MCMC run

Examples

```
myscore<-scoreparameters("bge", Boston)
## Not run:
itfit<-learnBN(myscore,algorithm="orderIter")
getMCMCscore(itfit)

## End(Not run)
```

`getRuntime` *Extracting runtime*

Description

This function extracts runtime of a particular step of order and partition MCMC.

Usage

```
getRuntime(x, which = 0)
```

Arguments

`x`	object of class 'orderMCMC'or 'partitionMCMC'
`which`	integer, defines if the runtime is extracted for: computing score tables (which = 1), running MCMC chain (which = 2)

Value

runtime of a particular step of MCMC scheme or total runtime

Examples

```
myscore<-scoreparameters("bge",Boston)
## Not run:
orderfit<-sampleBN(myscore,algorithm="order")
(getRuntime(orderfit,1))
(getRuntime(orderfit,2))

## End(Not run)
```

getSpace *Extracting scorespace from MCMC object*

Description

This function extracts an object of class 'scorespace' from the result of the MCMC run when the parameter 'scoreout' was set to TRUE; otherwise extracts only adjacency matrix of the final search space without the score tables.

Usage

```
getSpace(x)
```

Arguments

x	object of class 'orderMCMC','partitionMCMC' or 'iterativeMCMC'

Value

an object of class 'scorespace' or an adjacency binary matrix corresponding to a search space last used in MCMC

Examples

```
myscore<-scoreparameters("bge", Boston)
## Not run:
itfit<-learnBN(myscore,algorithm="orderIter",scoreout=TRUE)
itspace<-getSpace(itfit)

## End(Not run)
```

getSubGraph *Deriving subgraph*

Description

This function derives an adjacency matrix of a subgraph based on the adjacency matrix of a full graph and a list of nodes

Usage

```
getSubGraph(adj, nodes)
```

Arguments

adj	square adjacency matrix with elements in {0,1}, representing a graph
nodes	vector of node names of the subgraph; should be a subset of column names of 'adj'

Value

adjacency matrix of a subgraph which includes all 'nodes'

Examples

```
getSubGraph(Asiamat,c("E","B","D","X"))
```

`getTrace` *Extracting trace from MCMC object*

Description

This function extracts a trace of

- DAG scores
- DAG adjacency matrices
- orders
- order scores

from the result of the MCMC run. Note that the last three options work only when the parameter 'scoreout' was set to TRUE.

Usage

```
getTrace(x, which = 0)
```

Arguments

`x`	object of class 'orderMCMC','partitionMCMC' or 'iterativeMCMC'
`which`	integer, indication which trace is returned: DAG scores (which = 0), DAGs (which = 1), orders (which = 2), order scores (which = 3)

Value

a list or a vector of objects representing MCMC trace, depends on parameter 'which'; by default, the trace of DAG scores is returned

Examples

```
myscore<-scoreparameters("bge",Boston)
## Not run:
orderfit<-sampleBN(myscore,algorithm="order")
DAGscores<-getTrace(orderfit,which=0)
DAGtrace<-getTrace(orderfit,which=1)
orderscores<-getTrace(orderfit,which=3)

## End(Not run)
```

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. Functions 'randomDAG' and 'rmvDAG' from R-packages 'pcalg' were used to generate the data.

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. Functions 'randomDAG' and 'rmvDAG' from R-packages 'pcalg' were used to generate the data.

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

`interactions` *interactions dataset*

Description

A data frame containing possible interactions between genes from `kirp` and `kirc` data sets

Usage

```
interactions
```

Format

A data frame with 179 rows and 3 columns;

- node1 character, name of a gene
- node2 character, name of a gene
- combined_score interaction score, reflecting confidence in the fact that interaction between gene1 and gene2 is possible

each row represents a possible interaction between two genes

Source

https://string-db.org/

`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 (arXiv:1803.07859v3). 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,
  MAP = TRUE,
  posterior = 0.5,
  softlimit = 9,
  hardlimit = 12,
  alpha = 0.05,
  gamma = 1,
  verbose = TRUE,
  chainout = FALSE,
  scoreout = FALSE,
  cpdag = FALSE,
  mergetype = "skeleton",
  iterations = NULL,
  moveprobs = NULL,
  stepsave = NULL,
  startorder = NULL,
  accum = FALSE,
  compress = TRUE,
  plus1it = NULL,
  startspace = NULL,
  blacklist = NULL,
  addspace = NULL,
  scoretable = NULL,
  alphainit = NULL
)

## S3 method for class 'iterativeMCMC'
plot(
  x,
  ...,
  main = "iterative MCMC, DAG scores",
  xlab = "MCMC step",
  ylab = "DAG logscore",
  type = "l",
  col = "blue"
)

## S3 method for class 'iterativeMCMC'
print(x, ...)

## S3 method for class 'iterativeMCMC'
summary(object, ...)
```

Arguments

scorepar	an object of class scoreparameters, containing the data and scoring 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; 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
softlimit	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	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	numerical significance value in {0,1} for the conditional independence tests in the PC-stage
gamma	tuning parameter which transforms the score by raising it to this power, 1 by default
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; three options are available 'dag', 'cpdag', 'skeleton'; 'skeleton' by default
iterations	integer, the number of MCMC steps, the default value is $3.5n^2 \log n$
moveprobs	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
stepsave	integer, thinning interval for the MCMC chain, indicating the number of steps between two output iterations, the default is iterations/1000
startorder	integer vector of length n, which will be used as the starting order in the MCMC algorithm, the default order is random

`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

`compress` logical, if TRUE adjacency matrices representing sampled graphs will be stored as a sparse Matrix (recommended); TRUE by default

`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

`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

- "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

`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) object of class `scorespace`. When not NULL, parameters `startspace` and `addspace` are ignored.

`alphainit` (optional) numerical, defines alpha that is used by the PC algorithm to learn initial structure of a DBN, ignored in static case

`x` object of class 'iterativeMCMC'

`...` ignored

`main` name of the graph; "iterative MCMC, DAG scores" by default

`xlab` name of x-axis; "MCMC step"

`ylab` name of y-axis; "DAG logscore"

`type` type of line in the plot; "l" by default

`col` colour of line in the plot; "blue" by default

`object` object of class 'iterativeMCMC'

Value

Object of class `iterativeMCMC`, which contains log-score trace as well as adjacency matrix of the maximum scoring DAG, its score and the order score. The output can optionally include DAGs sampled in MCMC iterations and the score tables. Optional output is regulated by the parameters `chainout` and `scoreout`. See `iterativeMCMC class` for a detailed class structure.

Note

see also extractor functions `getDAG`, `getTrace`, `getSpace`, `getMCMCscore`.

Author(s)

Polina Suter, Jack Kuipers

References

P. Suter, J. Kuipers, G. Moffa, N.Beerenwinkel (2023) <doi:10.18637/jss.v105.i09>

Kuipers J, Super P and Moffa G (2020). Efficient Sampling and Structure Learning of Bayesian Networks. (arXiv:1803.07859v3)

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("bge",Boston)
itfit<-iterativeMCMC(Bostonpar, chainout=TRUE, scoreout=TRUE)
plot(itfit)

## End(Not run)
```

`iterativeMCMC class` *iterativeMCMC class structure*

Description

The structure of an object of S3 class `iterativeMCMC`.

Details

An object of class `iterativeMCMC` is a list containing at least the following components:

- DAG: adjacency matrix of a maximum scoring DAG found/sampled in MCMC.
- CPDAG: adjacency matrix representing equivalence class of a maximum scoring DAG found/sampled in MCMC.
- score: score of a maximum scoring DAG found/sampled in MCMC.
- maxorder: order of a maximum scoring DAG found/sampled in MCMC.
- maxtrace: a list of maximum score graphs uncovered at each expansion of the search space; their scores and orders
- info: a list containing information about parameters and results of MCMC
- trace: a list of vectors containing log-scores of sampled DAGs, each element of the list corresponds to a single expansion of a search space
- startspace: adjacency matrix representing the initial core space where MCMC was ran
- endspace: adjacency matrix representing the final core space where MCMC was ran

  Optional components:

  - `traceadd`: list which consists of three elements:
    - incidence: list containg adjacency matrices of sampled DAGs
    - order: list of orders from which the DAGs were sampled
    - orderscores: a list of vectors with order log-scores
  - `scoretable`: object of class `scorespace class`

Author(s)

Polina Suter

itercomp *Performance assessment of iterative MCMC scheme against a known Bayesian network*

Description

This function compute 8 different metrics of structure fit of an object of class `iterativeMCMC` to the ground truth DAG (or CPDAG). Object of class `iterativeMCMC` stores MAP graph at from each search space expansion step. This function computes structure fit of each of the stored graphs to the ground truth one. Computed metrics include: TP, FP, TPR, FPR, FPRn, FDR, SHD. See metrics description in see also `compareDAGs`.

Usage

```
itercomp(MCMCmult, truedag, cpdag = TRUE, p = 0.5, trans = TRUE)

## S3 method for class 'itercomp'
plot(x, ..., vars = c("FP", "TP"), type = "b", col = "blue", showit = c())

## S3 method for class 'itercomp'
print(x, ...)

## S3 method for class 'itercomp'
summary(object, ...)
```

Arguments

`MCMCmult`	an object which of class `iterativeMCMC`, see also `iterativeMCMC`)
`truedag`	ground truth DAG which generated the data used in the search procedure; represented by an object of class `graphNEL` or an adjacency matrix
`cpdag`	logical, if TRUE (FALSE by default) all DAGs are first converted to their respective equivalence classes (CPDAG)
`p`	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
`x`	object of class 'itercomp'
`...`	ignored
`vars`	a tuple of variables which will be used for 'x' and 'y' axes; possible values: "SHD", "TP", "FP", "TPR", "FPR", "FPRn", "FDR", "score"
`type`	type of line in the plot;"b" by default

col	colour of line in the plot; "blue" by default
showit	(optional) vector of integers specifying indices of search expansion iterations to be labelled; by default no iterations are labelled
object	object of class 'itercomp'

Value

an object if class itersim, a matrix with the number of rows equal to the number of expansion iterations in iterativeMCMC, and 8 columns reporting for the maximally scoring DAG uncovered at each iteration: the number of true positive edges ('TP'), the number of false positive edges ('FP'), the true positive rate ('TPR'), the structural Hamming distance ('SHD'), false positive rate ('FPR'), false discovery rate ('FDR') and the score of the DAG ('score').

Author(s)

Polina Suter

Examples

```
gsim.score<-scoreparameters("bge", gsim)
## Not run:
MAPestimate<-learnBN(gsim.score,"orderIter")
itercomp(MAPestimate, gsimmat)

## End(Not run)
```

---

kirc *kirc dataset*

Description

Mutation data from TCGA kidney renal clear cell cohort (KIRC). Mutations are picked according to q-value computed by MutSig2CV (q<0.1) or connected in networks discovered by Kuipers et al. 2018.

Usage

```
kirc
```

Format

An object of class matrix (inherits from array) with 476 rows and 70 columns.

Details

Each variable represents a gene. If in sample i gene j contains a mutation, than j-th element in row i equals 1, and 0 otherwise. The rows are named according to sample names in TCGA. The columns are named according to gene symbols.

References

https://portal.gdc.cancer.gov/

http://firebrowse.org/iCoMut/?cohort=kirc

Lawrence, M. et al. Mutational heterogeneity in cancer and the search for new cancer-associated genes. Nature 499, 214-218 (2013)

`kirp` *kirp dataset*

Description

Mutation data from TCGA kidney renal papillary cell cohort (KIRP). Mutations are picked according to q-value computed by MutSigCV (q<0.1) or connected in networks discovered by Kuipers et al. 2018.

Usage

```
kirp
```

Format

An object of class `matrix` (inherits from `array`) with 282 rows and 70 columns.

Details

Each variable represents a gene. If in sample i gene j contains a mutation, than j-th element in row i equals 1, and 0 otherwise. The rows are named according to sample names in TCGA. The columns are named according to gene symbols.

References

https://portal.gdc.cancer.gov/

http://firebrowse.org/iCoMut/?cohort=kirp

Lawrence, M. et al. Mutational heterogeneity in cancer and the search for new cancer-associated genes. Nature 499, 214-218 (2013)

learnBN *Bayesian network structure learning*

Description

This function can be used finding the maximum a posteriori (MAP) DAG using stochastic search relying on MCMC schemes. Due to the superexponential size of the search space, it must be reduced. 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. Order MCMC scheme (algorithm="order") performs the search of a maximum scoring order and selects a maximum scoring DAG from this order as MAP. To avoid discovering a suboptimal graph due to the absence of some of the true positive edges in the search space, the function includes the possibility to expand the default or input search space, by allowing each node in the network to have one additional parent (plus1="TRUE"). This offers improvements in the learning of Bayesian networks. The iterative MCMC (algorithm="orderIter") scheme allows for iterative expansions of the search space. This is useful in cases when the initial search space is poor in a sense that it contains only a limited number of true positive edges. Iterative expansions of the search space efficiently solve this issue. However this scheme requires longer runtimes due to the need of running multiple consecutive MCMC chains. This function is a wrapper for the individual structure learning functions that implement each of the described algorithms; for details see orderMCMC, and iterativeMCMC.

Usage

```
learnBN(
  scorepar,
  algorithm = c("order", "orderIter"),
  chainout = FALSE,
  scoreout = ifelse(algorithm == "orderIter", TRUE, FALSE),
  alpha = 0.05,
  moveprobs = NULL,
  iterations = NULL,
  stepsave = NULL,
  gamma = 1,
  verbose = FALSE,
  compress = TRUE,
  startspace = NULL,
  blacklist = NULL,
  scoretable = NULL,
  startpoint = NULL,
  plus1 = TRUE,
 iterpar = list(softlimit = 9, mergetype = "skeleton", accum = FALSE, plus1it = NULL,
    addspace = NULL, alphainit = NULL),
 cpdag = FALSE,
 hardlimit = 12
```

)

Arguments

scorepar	an object of class scoreparameters, containing the data and score parameters, see constructor function scoreparameters
algorithm	MCMC scheme to be used for MAP structure learning; possible options are "order" (orderMCMC) or "orderIter" (iterativeMCMC)
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 for "order", TRUE for "orderIter"
alpha	numerical significance value in {0,1} for the conditional independence tests at the PC algorithm stage
moveprobs	a numerical vector of 4 (for "order" and "orderIter" algorithms) or 5 values (for "partition" algorithm) representing probabilities of the different moves in the space of order and partitions accordingly. The moves are described in the corresponding algorithm specific functions orderMCMC and partitionMCMC
iterations	integer, the number of MCMC steps, the default value is $6n^2 \log n$ orderMCMC, $20n^2 \log n$ for partitionMCMC and $3.5n^2 \log n$ for iterativeMCMC; where n is the number of nodes in the Bayesian network
stepsave	integer, thinning interval for the MCMC chain, indicating the number of steps between two output iterations, the default is iterations/1000
gamma	tuning parameter which transforms the score by raising it to this power, 1 by default
verbose	logical, if TRUE messages about the algorithm's progress will be printed, FALSE by default
compress	logical, if TRUE adjacency matrices representing sampled graphs will be stored as a sparse Matrix (recommended); TRUE by default
startspace	(optional) a square sparse or ordinary 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 sparse or ordinary 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.
scoretable	(optional) object of class scorespace containing list of score tables calculated for example by the last iteration of the function iterativeMCMC. When not NULL, parameter startspace is ignored.
startpoint	(optional) integer vector of length n (representing an order when algorithm="order" or algorithm="orderIter") or an adjacency matrix or sparse adjacency matrix (representing a DAG when algorithm="partition"), which will be used as the starting point in the MCMC algorithm, the default starting point is random

plus1	logical, if TRUE (default) the search is performed on the extended search space; only changable for orderMCMC; for other algorithms is fixed to TRUE
iterpar	addition list of parameters for the MCMC scheme implemeting iterative expansions of the search space; for more details see iterativeMCMC; list(posterior = 0.5, softlimit = 9, mergetype = "skeleton", accum = FALSE, plus1it = NULL, addspace = NULL, alphainit = NULL)
cpdag	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
hardlimit	integer, limit on the size of parent sets in the search space; by default 14 when MAP=TRUE and 20 when MAP=FALSE

Value

Depending on the value or the parameter algorithm returns an object of class orderMCMC or iterativeMCMC which contains log-score trace of sampled DAGs as well as adjacency matrix of the maximum scoring DAG(s), its score and the order or partition score. The output can optionally include DAGs sampled in MCMC iterations and the score tables. Optional output is regulated by the parameters chainout and scoreout. See orderMCMC class, iterativeMCMC class for a detailed description of the classes' structures.

Note

see also extractor functions getDAG, getTrace, getSpace, getMCMCscore.

Author(s)

Polina Suter, Jack Kuipers, the code partly derived from the order MCMC implementation from Kuipers J, Moffa G (2017) <doi:10.1080/01621459.2015.1133426>

References

P. Suter, J. Kuipers, G. Moffa, N.Beerenwinkel (2023) <doi:10.18637/jss.v105.i09>

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:
myScore<-scoreparameters("bge",Boston)
mapfit<-learnBN(myScore,"orderIter")
summary(mapfit)
plot(mapfit)

## 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)
```

`mapping` *mapping dataset*

Description

A data frame containing mapping between names of genes used in `kirp`/`kirc` data sets and names used in STRING interactions list (see `interactions`).

Usage

```
mapping
```

Format

A data frame with 46 rows and two columns:

- queryItem character, name used for structure learning
- preferredName character, name used in STRING interactions data set

Source

https://string-db.org/

`modelp` *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

```
modelp(MCMCchain, p, pdag = FALSE, burnin = 0.2)
```

Arguments

`MCMCchain`	object of class `partitionMCMC`, `orderMCMC` or `iterativeMCMC`, representing the output of structure sampling function `partitionMCMC` or `orderMCMC` (the latter when parameter `chainout`=TRUE;
`p`	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`	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

Author(s)

Polina Suter

Examples

```
Bostonscore<-scoreparameters("bge", Boston)
## Not run:
partfit<-sampleBN(Bostonscore, "partition")
hdag<-modelp(partfit, p=0.9)

## End(Not run)
```

---

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,
  chainout = FALSE,
  scoreout = FALSE,
  moveprobs = NULL,
  iterations = NULL,
  stepsave = NULL,
  alpha = 0.05,
  cpdag = FALSE,
  gamma = 1,
  hardlimit = ifelse(plus1, 14, 20),
  verbose = FALSE,
  compress = TRUE,
  startspace = NULL,
  blacklist = NULL,
  startorder = NULL,
  scoretable = NULL
)
```

```
## S3 method for class 'orderMCMC'
plot(
  x,
  ...,
  burnin = 0.2,
  main = "DAG logscores",
  xlab = "iteration",
  ylab = "logscore",
  type = "l",
  col = "#0c2c84"
)

## S3 method for class 'orderMCMC'
print(x, ...)

## S3 method for class 'orderMCMC'
summary(object, ...)
```

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
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
moveprobs	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	integer, the number of MCMC steps, the default value is $6n^2 \log n$
stepsave	integer, thinning interval for the MCMC chain, indicating the number of steps between two output iterations, the default is iterations/1000
alpha	numerical significance value in {0,1} for the conditional independence tests at the PC algorithm stage
cpdag	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	tuning parameter which transforms the score by raising it to this power, 1 by default

hardlimit	integer, limit on the size of parent sets in the search space; by default 14 when MAP=TRUE and 20 when MAP=FALSE
verbose	logical, if TRUE messages about the algorithm's progress will be printed, FALSE by default
compress	logical, if TRUE adjacency matrices representing sampled graphs will be stored as a sparse Matrix (recommended); TRUE 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.
startorder	(optional) integer vector of length n, which will be used as the starting order in the MCMC algorithm, the default order is random
scoretable	(optional) object of class `scorespace` containing list of score tables calculated for example by the last iteration of the function `iterativeMCMC`. When not NULL, parameter `startspace` is ignored.
x	object of class 'orderMCMC'
...	ignored
burnin	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
main	name of the graph; "DAG logscores" by default
xlab	name of x-axis; "iteration"
ylab	name of y-axis; "logscore"
type	type of line in the plot; "l" by default
col	colour of line in the plot; "#0c2c84" by default
object	object of class 'orderMCMC'

Value

Object of class `orderMCMC`, which contains log-score trace of sampled DAGs as well as adjacency matrix of the maximum scoring DAG, its score and the order score. The output can optionally include DAGs sampled in MCMC iterations and the score tables. Optional output is regulated by the parameters `chainout` and `scoreout`. See `orderMCMC class` for a detailed class structure.

Note

see also extractor functions `getDAG`, `getTrace`, `getSpace`, `getMCMCscore`.

Author(s)

Polina Suter, Jack Kuipers, the code partly derived from the order MCMC implementation from Kuipers J, Moffa G (2017) <doi:10.1080/01621459.2015.1133426>

References

P. Suter, J. Kuipers, G. Moffa, N.Beerenwinkel (2023) <doi:10.18637/jss.v105.i09>

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("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)
```

---

orderMCMC class *orderMCMC class structure*

Description

The structure of an object of S3 class `orderMCMC`.

Details

An object of class `orderMCMC` is a list containing at least the following components:

- DAG: adjacency matrix of a maximum scoring DAG found/sampled in the MCMC scheme.
- CPDAG: adjacency matrix representing equivalence class of a maximum scoring DAG found/sampled in MCMC.

- score: score of a maximum scoring DAG found/sampled in MCMC.
- maxorder: order of a maximum scoring DAG found/sampled in MCMC.
- info: a list containing information about parameters and results of MCMC.
- trace: a vector containing log-scores of sampled DAGs.
  Optional components:
  - `traceadd`: list which consists of three or four elements (depending on MCMC scheme used for sampling):
    - incidence: list containg adjacency matrices of sampled DAGs
    - order: list of orders from which the DAGs were sampled
    - orderscores: order log-scores
  - `scoretable`: object of class `scorespace class`

Author(s)

Polina Suter

`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,
  moveprobs = NULL,
  iterations = NULL,
  stepsave = NULL,
  alpha = 0.05,
  gamma = 1,
  verbose = FALSE,
  scoreout = FALSE,
  compress = TRUE,
  startspace = NULL,
  blacklist = NULL,
  scoretable = NULL,
  startDAG = NULL
)
```

```
## S3 method for class 'partitionMCMC'
plot(
  x,
  ...,
  burnin = 0.2,
  main = "DAG logscores",
  xlab = "iteration",
  ylab = "logscore",
  type = "l",
  col = "#0c2c84"
)

## S3 method for class 'partitionMCMC'
print(x, ...)

## S3 method for class 'partitionMCMC'
summary(object, ...)
```

Arguments

scorepar	an object of class scoreparameters, containing the data and scoring parameters; see constructor function scoreparameters.
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	integer, the number of MCMC steps, the default value is $20n^2 \log n$
stepsave	integer, thinning interval for the MCMC chain, indicating the number of steps between two output iterations, the default is iterations/1000
alpha	numerical significance value in {0,1} for the conditional independence tests at the PC algorithm stage
gamma	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
scoreout	logical, if TRUE the search space and score tables are returned, FALSE by default
compress	logical, if TRUE adjacency matrices representing sampled graphs will be stored as a sparse Matrix (recommended); TRUE 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]=1` it means that the edge from node `i` to node `j` is excluded from the search space
`scoretable`	(optional) object of class `scorespace` containing list of score tables calculated for example by the last iteration of the function `iterativeMCMC`. When not NULL, parameter `startspace` is ignored
`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
`x`	object of class 'partitionMCMC'
`...`	ignored
`burnin`	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
`main`	name of the graph; "DAG logscores" by default
`xlab`	name of x-axis; "iteration"
`ylab`	name of y-axis; "logscore"
`type`	type of line in the plot; "l" by default
`col`	colour of line in the plot; "#0c2c84" by default
`object`	object of class 'partitionMCMC'

Value

Object of class `partitionMCMC`, which contains log-score trace as well as adjacency matrix of the maximum scoring DAG, its score and the order score. Additionally, returns all sampled DAGs (represented by their adjacency matrices), their scores, orders and partitions See `partitionMCMC class`.

Note

see also extractor functions `getDAG`, `getTrace`, `getSpace`, `getMCMCscore`.

Author(s)

Jack Kuipers, Polina Suter, the code partly derived from the partition MCMC implementation from Kuipers J, Moffa G (2017) <doi:10.1080/01621459.2015.1133426>

References

P. Suter, J. Kuipers, G. Moffa, N.Beerenwinkel (2023) <doi:10.18637/jss.v105.i09>

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("bge", Boston)
partfit<-partitionMCMC(myScore)
plot(partfit)

## End(Not run)
```

`partitionMCMC class` *partitionMCMC class structure*

Description

The structure of an object of S3 class `partitionMCMC`.

Details

An object of class `partitionMCMC` is a list containing at least the following components:

- DAG: adjacency matrix of a maximum scoring DAG found/sampled in the MCMC scheme.
- CPDAG: adjacency matrix representing equivalence class of a maximum scoring DAG found/sampled in MCMC.
- score: score of a maximum scoring DAG found/sampled in MCMC.
- maxorder: order of a maximum scoring DAG found/sampled in MCMC.
- info: a list containing information about parameters and results of MCMC.
- trace: a vector containing log-scores of sampled DAGs.
  Optional components:
  - `traceadd`: list which consists of three or four elements (depending on MCMC scheme used for sampling):
    - incidence: list containg adjacency matrices of sampled DAGs
    - order: list of orders from which the DAGs were sampled
    - partition: list of partition from which the DAGs were sampled
    - partitionscores: partition log-scores
  - `scoretable`: object of class `scorespace class`

Author(s)

Polina Suter

`plot2in1` *Highlighting similarities between two graphs*

Description

This function plots nodes and edges from two graphs in one and indicates similarities between these graphs.

Usage

```
plot2in1(graph1, graph2, name1 = NULL, name2 = NULL, bidir = FALSE, ...)
```

Arguments

`graph1`	binary adjacency matrix of a graph
`graph2`	binary adjacency matrix of a graph, column names should coincide with column names of 'graph1'
`name1`	character, custom name for 'graph1'; when NULL no legend will be plotted
`name2`	character, custom name for 'graph2'
`bidir`	logical, defines if arrows of bidirected edges are drawn; FALSE by defauls.
`...`	optional parameters passed to **Rgraphviz** plotting functions e.g. `main`, `fontsize`

Value

plots the graph which includes nodes and edges two graphs; nodes which are connected to at least one other node in both graphs are plotted only once and coloured orange, edges which are shared by two graphs are coloured orange; all other nodes and edges a plotted once for each 'graph1' and 'graph2' and coloured blue and green accordingly.

Author(s)

Polina Suter

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"), b = 0, shape = "circle", ...)
```

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; acceptable values are init or trans
b	number of static variables in the DBN, 0 by default; note that for function to work correctly all static variables have to be in the first b columns of the matrix
shape	string, defining the shape of the box around each node; possible values are circle, ellipse, box
...	optional parameters passed to Rgraphviz plotting functions e.g. main, fontsize

Value

plots the DBN defined by the adjacency matrix 'DBN' and number of static and dynamic variables. When 'struct' equals "trans" the transition structure is plotted, otherwise initial structure is plotted

Author(s)

Polina Suter

Examples

```
plotDBN(DBNmat, "init", b=3)
plotDBN(DBNmat, "trans", b=3)
```

plotdiffs *Plotting difference between two graphs*

Description

This function plots edges from two graphs in one and indicates similarities and differences between these graphs. It is also possible to use this function for plotting mistakes in estimated graph when the ground truth graph is known.

Usage

```
plotdiffs(
  graph1,
  graph2,
  estimated = TRUE,
  name1 = "graph1",
  name2 = "graph2",
  clusters = NULL,
  ...
)
```

Arguments

graph1	object of class graphNEL or its adjacency matrix
graph2	object of class graphNEL or its adjacency matrix
estimated	logical, indicates if graph1 is estimated graph and graph2 is ground truth DAG, TRUE by default; this affects the legend and colouring of the edges
name1	character, custom name for 'graph1'
name2	character, custom name for 'graph2'
clusters	(optional) a list of nodes to be represented on the graph as clusters
...	optional parameters passed to Rgraphviz plotting functions e.g. main, fontsize

Value

plots the graph which includes edges from graph1 and graph2; edges which are different in graph1 compared to graph2 are coloured according to the type of a difference

Author(s)

Polina Suter

Examples

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

`plotdiffsDBN` *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

```
plotdiffsDBN(
  eDBN,
  trueDBN,
  struct = c("init", "trans"),
  b = 0,
  showcl = TRUE,
  orientation = "TB",
  ...
)
```

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
`b`	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 b columns of the matrix
`showcl`	logical, when TRUE (default) nodes are shown in clusters according to the time slice the belong to
`orientation`	orientation of the graph layout, possible options are 'TB' (top-bottom) and 'LR' (left-right)
`...`	optional parameters passed to `Rgraphviz` plotting functions e.g. `main`, `fontsize`

Value

plots the graph highlights differences between 'eDBN' (estimated DBN) and 'trueDBN' (ground truth); edges which are different in 'eDBN' compared to 'trueDBN' are coloured according to the type of a difference: false-positive, false-negative and error in direction.

Author(s)

Polina Suter

Examples

```
dbnscore<-scoreparameters("bge",DBNdata,
dbnpar = list(samestruct=TRUE, slices=5, b=3),
DBN=TRUE)
## Not run:
orderDBNfit<-learnBN(dbnscore,algorithm="order")
iterDBNfit<-learnBN(dbnscore,algorithm="orderIter")
plotdiffsDBN(getDAG(orderDBNfit),DBNmat,struct="trans",b=3)
plotdiffsDBN(getDAG(iterDBNfit),DBNmat,struct="trans",b=3)

## End(Not run)
```

---

plotpcor *Comparing posterior probabilitites of single edges*

Description

This function can be used to compare posterior probabilities of edges in a graph

Usage

```
plotpcor(pmat, highlight = 0.3, printedges = FALSE, cut = 0.05, ...)
```

Arguments

pmat	a list of square matrices, representing posterior probabilities of single edges in a Bayesian network; see edgep for obtaining such a matrix from a single MCMC run
highlight	numeric, defines maximum acceptable difference between posterior probabilities of an edge in two samples; points corresponding to higher differences are highlighted in red
printedges	when TRUE the function also returns squared correlation and RMSE of posterior probabilities higher than the value defined by the argument 'cut' as well as the list of all edges whose posterior probabilities in the first two matrices differ more than 'highlight'; FALSE by default
cut	numeric value corresponding to a minimum posterior probabilitity which is included into calculation of squared correlation and MSE when 'printedges' equals TRUE

... prameters passed further to the `plot` function (e.g. `xlab`, `ylab`, `main`) in case when the length of `pmat` equals 2

Value

plots concordance of posterior probabilitites of single edges based on several matrices (minimum 2 matrices); highlights the edges whose posterior probabilities in a pair of matrices differ by more than 'highlight'; when 'printedges' set to TRUE, the function returns also squared correlation and RMSE of posterior probabilities higher than the value defined by the argument 'cut' as well as the list of all edges whose posterior probabilities in the first two matrices differ by more than 'highlight'.

Author(s)

Polina Suter

Examples

```
Asiascore<-scoreparameters("bde", Asia)
## Not run:
orderfit<-list()
orderfit[[1]]<-sampleBN(Asiascore,algorithm="order")
orderfit[[2]]<-sampleBN(Asiascore,algorithm="order")
orderfit[[3]]<-sampleBN(Asiascore,algorithm="order")
pedges<-lapply(orderfit,edgep,pdag=TRUE)
plotpcor(pedges, xlab="run1", ylab="run2",printedges=TRUE)

## End(Not run)
```

---

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,
  highlight = 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 which equal 1 will be plotted
highlight	(optional) binary matrix, edges corresponding to entries which equal 1 are highlighted with "red"
...	(optional) parameters passed to the plot function

Value

plots posterior probabilities of edges in the graph as a function of MCMC iterations

Author(s)

Polina Suter

Examples

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

## End(Not run)
```

---

sampleBN *Bayesian network structure sampling from the posterior distribution*

Description

This function can be used for structure sampling using three different MCMC schemes. Order MCMC scheme (algorithm="order") is the most computationally efficient however it imposes a non-uniform prior in the space of DAGs. Partition MCMC (algorithm="partition") is less computationally efficient and requires more iterations to reach convergence, however it implements sampling using a uniform prior in the space of DAGs. 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. The iterative MCMC scheme (`algorithm="orderIter"`) allows for iterative expansions of the search space. This is useful in cases when the initial search space is poor in a sense that it contains only a limited number of true positive edges. Iterative expansions of the search space efficiently solve this issue. However this scheme requires longer runtimes due to the need of running multiple consecutive MCMC chains. This function is a wrapper for the three individual structure learning and sampling functions that implement each of the described algorithms; for details see `orderMCMC`, `partitionMCMC`,`iterativeMCMC`.

Usage

```
sampleBN(
  scorepar,
  algorithm = c("order", "orderIter", "partition"),
  chainout = TRUE,
  scoreout = FALSE,
  alpha = 0.05,
  moveprobs = NULL,
  iterations = NULL,
  stepsave = NULL,
  gamma = 1,
  verbose = FALSE,
  compress = TRUE,
  startspace = NULL,
  blacklist = NULL,
  scoretable = NULL,
  startpoint = NULL,
  plus1 = TRUE,
  cpdag = FALSE,
  hardlimit = 12,
 iterpar = list(posterior = 0.5, softlimit = 9, mergetype = "skeleton", accum = FALSE,
    plus1it = NULL, addspace = NULL, alphainit = NULL)
)
```

Arguments

`scorepar`	an object of class `scoreparameters`, containing the data and score parameters, see constructor function `scoreparameters`
`algorithm`	MCMC scheme to be used for sampling from posterior distribution; possible options are "order" (`orderMCMC`), "orderIter" (`iterativeMCMC`) or "partition" (`partitionMCMC`)
`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
`alpha`	numerical significance value in `{0,1}` for the conditional independence tests at the PC algorithm stage

`moveprobs` a numerical vector of 4 (for "order" and "orderIter" algorithms) or 5 values (for "partition" algorithm) representing probabilities of the different moves in the space of order and partitions accordingly. The moves are described in the corresponding algorithm specific functions `orderMCMC` and `partitionMCMC`

`iterations` integer, the number of MCMC steps, the default value is $6n^2 \log n$ orderMCMC, $20n^2 \log n$ for partitionMCMC and $3.5n^2 \log n$ for iterativeMCMC; where n is the number of nodes in the Bayesian network

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

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

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

`compress` logical, if TRUE adjacency matrices representing sampled graphs will be stored as a sparse Matrix (recommended); TRUE by default

`startspace` (optional) a square sparse or ordinary 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 sparse or ordinary 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.

`scoretable` (optional) object of class `scorespace` containing list of score tables calculated for example by the last iteration of the function `iterativeMCMC`. When not NULL, parameter `startspace` is ignored.

`startpoint` (optional) integer vector of length n (representing an order when `algorithm="order"` or `algorithm="orderIter"`) or an adjacency matrix or sparse adjacency matrix (representing a DAG when `algorithm="partition"`), which will be used as the starting point in the MCMC algorithm, the default starting point is random

`plus1` logical, if TRUE (default) the search is performed on the extended search space; only changable for orderMCMC; for other algorithms is fixed to TRUE

`cpdag` 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

`hardlimit` integer, limit on the size of parent sets in the search space;

`iterpar` addition list of parameters for the MCMC scheme implemeting iterative expansions of the search space; for more details see `iterativeMCMC`; list(posterior = 0.5, softlimit = 9, mergetype = "skeleton", accum = FALSE, plus1it = NULL, addspace = NULL, alphainit = NULL)

Value

Depending on the value or the parameter `algorithm` returns an object of class `orderMCMC`, `partitionMCMC` or `iterativeMCMC` which contains log-score trace of sampled DAGs as well as adjacency matrix of the maximum scoring DAG(s), its score and the order or partition score. The output can optionally include DAGs sampled in MCMC iterations and the score tables. Optional output is regulated by the parameters `chainout` and `scoreout`. See `orderMCMC class`, `partitionMCMC class`, `iterativeMCMC class` for a detailed description of the classes' structures.

Note

see also extractor functions `getDAG`, `getTrace`, `getSpace`, `getMCMCscore`.

Author(s)

Polina Suter, Jack Kuipers, the code partly derived from the order MCMC implementation from Kuipers J, Moffa G (2017) <doi:10.1080/01621459.2015.1133426>

References

P. Suter, J. Kuipers, G. Moffa, N.Beerenwinkel (2023) <doi:10.18637/jss.v105.i09>

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:
Asiascore <- scoreparameters("bde", Asia)
iterativefit <- learnBN(Asiascore, algorithm = "orderIter")
orderfit <- sampleBN(Asiascore, scoretable = iterativefit)

myScore<-scoreparameters("bge",Boston)
MCMCchains<-list()
MCMCchains[[1]]<-sampleBN(myScore,"partition")
MCMCchains[[2]]<-sampleBN(myScore,"partition")
edge_posterior<-lapply(MCMCchains,edgep,pdag=TRUE)
plotpcor(edge_posterior)

## End(Not run)
```

samplecomp *Performance assessment of sampling algorithms against a known Bayesian network*

Description

This function compute 8 different metrics of structure fit of an object of classes orderMCMC and partitionMCMC to the ground truth DAG (or CPDAG). First posterior probabilities of single edges are calculated based on a sample stores in the object of class orderMCMC or partitionMCMC. This function computes structure fit of each of the consensus graphs to the ground truth one based on a defined range of posterior thresholds. Computed metrics include: TP, FP, TPR, FPR, FPRn, FDR, SHD. See metrics description in see also compareDAGs.

Usage

```
samplecomp(
  MCMCchain,
  truedag,
  p = 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
)

## S3 method for class 'samplecomp'
plot(x, ..., vars = c("FP", "TP"), type = "b", col = "blue", showp = NULL)

## S3 method for class 'samplecomp'
print(x, ...)

## S3 method for class 'samplecomp'
summary(object, ...)
```

Arguments

MCMCchain	an object of class partitionMCMC or orderMCMC, 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
p	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 $p = c(0.99, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2)$
pdag	logical, if TRUE (default) all DAGs in the MCMCchain are first converted to equivalence class (CPDAG) before the averaging

burnin	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
x	object of class 'samplecomp'
...	ignored
vars	a tuple of variables which will be used for 'x' and 'y' axes; possible values: "SHD", "TP", "FP", "TPR", "FPR", "FPRn", "FDR"
type	type of line in the plot; "b" by default
col	colour of line in the plotl; "blue" by default
showp	logical, defines if points are labelled with the posterior threshold corresponding to the assessed model
object	object of class 'samplecomp'

Value

an object if class samplesim, a matrix with the number of rows equal to the number of elements in 'p', and 8 columns reporting for the consensus graphss (corresponfing to each of the values in 'p') the number of true positive edges ('TP'), the number of false positive edges ('FP'), the number of false negative edges ('FN'), the true positive rate ('TPR'), the structural Hamming distance ('SHD'), false positive rate ('FPR'), false discovery rate ('FDR') and false positive rate normalized by TP+FN ('FPRn').

Author(s)

Polina Suter

Examples

```
gsim.score<-scoreparameters("bge", gsim)
## Not run:
MAPestimate<-learnBN(gsim.score,"orderIter",scoreout=TRUE)
ordersample<-sampleBN(gsim.score, "order", scoretable=getSpace(MAPestimate))
samplecomp(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,
  onlymain = FALSE,
  bdecatCvec = NULL
)
```

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) defines, whether to use the posterior mean for scoring (default) or to marginalise over the posterior distribution (more computationally costly)
onlymain	(optional), defines the the score is computed for nodes excluding 'bgnodes'; FALSE by default
bdecatCvec	(optional for categorical data)

Value

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

Author(s)

Jack Kuipers, Polina Suter

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

`scoreagainstDBN` *Score against DBN*

Description

Scoring observations against a DBN structure

Usage

```
scoreagainstDBN(
  scorepar,
  incidence,
  datatoscore = NULL,
  marginalise = FALSE,
  onlymain = FALSE,
  datainit = NULL
)
```

Arguments

`scorepar`	object of class 'scoreparameters'
`incidence`	adjacency matrix of a DAG
`datatoscore`	matrix or vector containing observations to be scored
`marginalise`	(logical) should marginal score be used?
`onlymain`	(logical) should static nodes be included in the score?
`datainit`	optional, in case of unbalanced design, the mean score of available samples for T0 are computed

Value

vector of log-scores

Author(s)

Polina Suter

scoreparameters *Initializing 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(
  scoretype = c("bge", "bde", "bdecat", "usr"),
  data,
  bgepar = list(am = 1, aw = NULL, edgepf = 1),
  bdepar = list(chi = 0.5, edgepf = 2),
  bdecatpar = list(chi = 0.5, edgepf = 2),
 dbnpar = list(samestruct = TRUE, slices = 2, b = 0, stationary = TRUE, rowids = NULL,
    datalist = NULL, learninit = TRUE),
  usrpar = list(pctesttype = c("bge", "bde", "bdecat")),
  mixedpar = list(nbin = 0),
  MDAG = FALSE,
  DBN = FALSE,
  weightvector = NULL,
  bgnodes = NULL,
  edgepmat = NULL,
  nodeslabels = NULL
)

## S3 method for class 'scoreparameters'
print(x, ...)

## S3 method for class 'scoreparameters'
summary(object, ...)
```

Arguments

scoretype	the score to be used to assess the DAG structure: "bge" for Gaussian data, "bde" for binary data, "bdecat" for categorical data, "usr" for a user defined score; when "usr" score is chosen, one must define a function (which evaluates the log score of a node given its parents) in the following format: usrDAGcorescore(j,parentnodes,n,param), where 'j' is node to be scores, 'parentnodes' are the parents of this node, 'n' number of nodes in the netwrok and 'param' is an object of class 'scoreparameters'
data	the data matrix with n columns (the number of variables) and a number of rows equal to the number of observations
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
- edgepf (optional) a positive numerical value providing the edge penalization factor to be combined with the BGe score, 1 by default (no penalization)

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
- b the number of static variables; all static variables have to be in the first b columns of the data; for DBNs static variables have the same meaning as bgnodes for usual Bayesian networks; for DBNs parameters parameter bgnodes is ignored
- rowids optional vector of time IDs; usefull for identifying data for initial time slice
- datalist indicates is data is passed as a list for a two step DBN; useful for unbalanced number of samples in timi slices

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

- pctesttype (optional) conditional independence test ("bde","bge","bdecat")

mixedpar — a list which contains parameters for the BGe and BDe score for mixed data

- nbin a positive integer number of binary nodes in the network (the binary nodes are always assumed in first nbin columns of the data)

MDAG — logical, when TRUE the score is initialized for a model with multiple sets of parameters but the same structure

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

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 vector that contains column indices in the data defining the nodes that are forced to be root nodes in the sampled graphs; root 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 and defined via element b of the parameters dbnpar; parameter bgnodes is ignored for DBNs

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
x	object of class 'scoreparameters'
...	ignored
object	object of class 'scoreparameters'

Value

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

Author(s)

Polina Suter, Jack kuipers

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("bge", myData)
```

---

scorespace *Prints 'scorespace' object*

Description

Prints 'scorespace' object

Summary of object of class 'scorespace'

Usage

```
scorespace(
  scorepar,
  alpha = 0.05,
  hardlimit = 14,
  plus1 = TRUE,
  cpdag = TRUE,
  startspace = NULL,
  blacklist = NULL,
  verbose = FALSE
)

## S3 method for class 'scorespace'
print(x, ...)

## S3 method for class 'scorespace'
summary(object, ...)
```

Arguments

scorepar	an object of class scoreparameters, containing the data and score scorepareters, see constructor function scoreparameters
alpha	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 $n > 50$)
hardlimit	integer, limit on the size of parent sets in the search space; by default 14 when MAP=TRUE and 20 when MAP=FALSE
plus1	logical, if TRUE (default) the search is performed on the extended search space
cpdag	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
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 messages about the algorithm's progress will be printed, FALSE by default
x	object of class 'scorespace'
...	ignored
object	object of class 'scorespace'

Value

Object of class scorespace, a list of three objects: 'adjacency' matrix representiong the search space, 'blacklist' used to exclude edges from the search space and 'tables' containing score quantities for each node needed to run MCMC schemes

Author(s)

Polina Suter, Jack Kuipers

References

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

Examples

```
#' #find a MAP DAG with search space defined by PC and plus1 neighbourhood
Bostonscore<-scoreparameters("bge",Boston)
Bostonspace<-scorespace(Bostonscore, 0.05, 14)
## Not run:
orderfit<-orderMCMC(Bostonscore, scoretable=Bostonspace)
partitionfit<-orderMCMC(Bostonscore, scoretable=Bostonspace)

## End(Not run)
```

---

scorespace class *scorespace class structure*

Description

The structure of an object of S3 class scorespace.

Details

An object of class scorespace is a list containing at least the following components:

- adjacency: adjacency martrix representing the core search space
- blacklist: adjacency martrix representing the blacklist used for computing score tables tables
- tables: a list of matrices (for core search space) or a list of lists of matrices (for extended search space) containing quantities needed for scoring orders and sampling DAGs in MCMC schemes; this list corresponds to adjacency and blacklist

Author(s)

Polina Suter

`string2mat` *Deriving interactions matrix*

Description

This transforms a list of possible interactions between proteins downloaded from STRING database into a matrix which can be used for blacklisting/penalization in BiDAG.

Usage

```
string2mat(curnames, int, mapping = NULL, type = c("int"), pf = 2)
```

Arguments

`curnames`	character vector with gene names which will be used in `BiDAG` learning function
`int`	data frame, representing a interactions between genes/proteins downloaded from STRING (https://string-db.org/); two columns are necessary 'node1' and 'node2'
`mapping`	(optional) data frame, representing a mapping between 'curnames' (gene names, usually the column names of 'data') and gene names used in interactions downloaded from STRING (https://string-db.org/); two columns are necessary 'queryItem' and 'preferredName'
`type`	character, defines how interactions will be reflected in the output matrix; `int` will result in a matrix whose entries equal 1 if interaction is present in the list of interactions `int` and 0 otherwise; `blacklist` results in a matrix whose entries equal 0 when interaction is present in the list of interactions and 1 otherwise; `pf` results in a matrix results in a matrix whose entries equal 1 is interaction is present in the list of interactions `int` and `pf` otherwise$ "int" by default
`pf`	penalization factor for interactions, needed if `type=pf`

Value

square matrix whose entries correspond to the list of interactions and parameter `type`

Examples

```
curnames<-colnames(kirp)
intmat<-string2mat(curnames, mapping, interactions, type="pf")
```

Index