

Package ‘gRapHD’

January 2, 2012

Version 0.2.2

Date 2009-09-15

Title Efficient selection of undirected graphical models for high-dimensional datasets

Author

Gabriel Coelho Goncalves de Abreu <abreu_ga@yahoo.com.br>, Rodrigo Labouriau <Rodrigo.Labouriau@agrsci.dk>, wards <David.Edwards@agrsci.dk>.

Maintainer Gabriel Coelho Goncalves de Abreu <abreu_ga@yahoo.com.br>

Depends R (>= 2.9.0), methods

Suggests graph

Imports graph

LazyLoad yes

Description gRapHD is designed for efficient selection of high-dimensional undirected graphical models. The package provides tools for selecting trees, forests and decomposable models minimizing information criteria such as AIC or BIC, and for displaying the independence graphs of the models. It has also some useful tools for analysing graphical structures. It supports the use of discrete, continuous, or both types of variables.

License GPL (>= 3)

Repository CRAN

Date/Publication 2011-05-26 20:48:43

R topics documented:

gRapHD-package	2
adjMat	3
calcStat	4
ccoeff	5
chStat	6
CI.test	8
convData	9
Degree	10
DFS	11
dsCont	12
dsDiscr	12
dsMixed	13
findEd	13
fit	15
gRapHD-class	16
jTree	18
MCS	20
minForest	21
modelDim	23
modelFormula	24
neighbourhood	26
neighbours	27
perfSets	28
plot.gRapHD	29
randTree	31
rowProds	33
shortPath	34
sortMat	35
stepw	36
SubGraph	39
Index	41

gRapHD-package

*The gRapHD package***Description**

The gRapHD package is designed for efficient selection of high-dimensional undirected graphical models. The package provides tools for selecting trees, forests and decomposable models minimizing information criteria such as AIC or BIC, and for displaying the independence graphs of the models. It has also some useful tools for analysing graphical structures. It supports the use of discrete, continuous, or both types of variables.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>
 Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>
 David Edwards (<David.Edwards@agrsci.dk>)

References

Gabriel C. G. Abreu, David Edwards, Rodrigo Labouriau (2010)., High-Dimensional Graphical Model Search with the gRapHD R Package., Journal of Statistical Software, 37(1), 1-18., URL <http://www.jstatsoft.org/v37/i01/>.

adjMat	<i>Adjacency matrix</i>
--------	-------------------------

Description

Returns the adjacency matrix based on a list of edges, supplied in a gRapHD object or as a matrix.

Usage

```
adjMat(model=NULL, edges=NULL, p=NULL)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Column 1 contains the vertex with lower index.
p	number of vertices.

Details

The dimension of the matrix is given by `model$p` or by the maximum value between `max(edges)` and `p`.

Value

matrix	<code>p</code> by <code>p</code> .
--------	------------------------------------

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
edges <- SubGraph(edges=m1@edges, v=1:10)@edges
adjMat(edges=edges, p=10)
```

calcStat *Pairwise weights*

Description

Calculates pairwise statistics ($-2 \cdot \log$ -LR, AIC, or BIC) for each variable pair (edge) in the dataset.

Usage

```
calcStat(dataset, homog=TRUE, forbEdges=NULL, stat="LR")
```

Arguments

dataset	matrix or data frame (nrow(dataset) observations and ncol(dataset) variables).
homog	TRUE for homogeneous covariance structure, FALSE for heterogeneous. This is only meaningful with mixed models. Default is homogeneous (TRUE).
forbEdges	list with edges that should not be considered. Matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Default is NULL.
stat	measure to be minimized: LR ($-2 \cdot \log$ -likelihood), AIC, or BIC. Default is LR. It can also be a user defined function with format: FUN(newEdge, numCat, dataset); where numCat is a vector with number of levels for each variable (0 if continuous); newEdge is a vector with length two; and dataset is a matrix (n by p).

Details

Calculates pairwise statistics ($-2 \cdot \log$ -LR, AIC, or BIC) for all possible edges, returning the values sorted in descending order.

Value

A matrix with $p(p-1)/2$ lines and 4 columns, where each line refers to a possible edge, and the columns are: vertex 1, vertex 2, value of the statistic, and number of estimated parameters (degrees of freedom) associated with the edge.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)
David Edwards (<David.Edwards@agrsci.dk>)

Examples

```

set.seed(7,kind="Mersenne-Twister")
dataset <- matrix(rnorm(1000),nrow=100,ncol=10)
m <- calcStat(dataset,stat="BIC")

data(dsCont)
# m1 <- calcStat(dataset,homog=TRUE,forbEdges=NULL,stat="LR")
#       1. in this case, there is no use for homog
#       2. no forbidden edges
#       3. the measure used is the LR (the result is a tree)
v <- calcStat(dsCont,homog=TRUE,forbEdges=NULL,stat="LR")

# result
head(v)
# column 1: first vertex of the edge
# column 2: second vertex of the edge
# column 3: in this case, -LR
# column 4: number of parameters for the edge
#       [,1] [,2] [,3] [,4]
# [1,]  17  27 393.0072  1
# [2,]  21  27 343.5780  1
# [3,]  22  25 306.0097  1
# [4,]  17  21 302.9414  1
# [5,]  27  32 300.0275  1
# [6,]  21  32 289.4179  1

```

ccoeff

Clustering coefficient

Description

Returns the clustering coefficients of the vertices in a graph.

Usage

```
ccoeff(model=NULL,edges=NULL,p=NULL)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Column 1 contains the vertex with lower index.
p	number of vertices. If NULL, the $p=\max(\text{edges})$.

Details

The clustering coefficient is given by $C_i=2*e_i/(k_i*(k_i-1))$, where k_i is the number of neighbours the vertex i has, and e_i is the number of edges between the neighbours of i .

Value

A vector with length p with the clustering coefficient of each vertex.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
data(dsCont)
m <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="BIC")
m1 <- stepw(m, dsCont)
cc <- ccoeff(edges=m1@edges, p=m1@p)
mean(cc)
```

chStat

Internal use

Description

Calculates the deviance associated with the addition of each add-eligible edge. Called from [stepw](#).

Usage

```
chStat(model, dataset, previous=NULL, forbEdges=NULL)
```

Arguments

model	a gRapHD object.
dataset	matrix (nrow(dataset) observations and ncol(dataset) variables).
previous	result of a previous run of chStat.
forbEdges	list with edges that should not be considered. Matrix with 2 columns, each row representing an edge, and each column a vertex. Default is NULL.

Details

The deviance and degrees of freedom associated with each add-eligible edge is returned. If previous results are specified these are reused as appropriate, and a concatenated result list is returned. This function is used by [stepw](#).

Value

A list with:

```
edges.to.test  matrix (k by 5), with columns:
                1 - first vertex of the tested edge
                2 - second vertex (the two values are ordered)
                3 - index for the separator in S
                4 - deviance statistic associated with adding this edge
                5 - degrees of freedom associated with the deviance

S              list of separators.
```

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
 Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)
 David Edwards (<David.Edwards@agrsci.dk>)

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
ch <- findEd(m1@edges, m1@p, NULL, 0)
ch <- chStat(m1, dsCont, ch, forbEdges=0)
str(ch)
# List of 2
# $ edges.to.test: num [1:53, 1:5] 1 11 19 11 17 2 2 10 10 2 ...
# $ S              :List of 53
# ..$ : int 11
# ..$ : int 21
# ..$ : int 21
# ..$ : int 21
# ..$ : int 27
# ..$ : int 17
# ...
head(ch$edges.to.test)
#      [,1] [,2] [,3]      [,4] [,5]
# [1,]   1   21   1 -0.61733689   1
# [2,]  11  19   2 -0.24637623   1
# [3,]  19  27   3 -0.47194908   1
# [4,]  11  27   4 -7.00259895   1
# [5,]  17  21   5 -11.09310305   1
# [6,]   2  27   6 -0.04690911   1

# the columns in ch$edges.to.test
# 1: first vertex in the edge
# 2: second vertex in the edge
# 3: index os the separator in ch$S
# 4: change in the LR for the edge
# 5: number of parameters for the edge
```

CI.test	<i>Test of conditional independence</i>
---------	---

Description

Test of conditional independence.

Usage

```
CI.test(x,y,S,dataset,homog=TRUE)
```

Arguments

x	one of the variables.
y	the other variable.
S	separator (possibly NULL).
dataset	matrix or data frame (nrow(dataset) observations and ncol(dataset) variables).
homog	TRUE for homogeneous covariance structure, FALSE for heterogeneous. This is only meaningful with mixed models. Default is homogeneous (TRUE).

Details

Performs a test of conditional independence of x and y given a set of variables S. The variables are specified as column numbers of the dataset. Under the alternative the variables are assumed to follow an unrestricted (mixed) graphical model. If x and y are discrete, S must also be discrete. Note that the model dimension returned by the `fit` function assumes that all parameters are estimable, which may not be the case for high-dimensional sparse data. However, here and in the search functions we use the adjusted degrees of freedom, which need no such assumptions and are believed to be correct.

Value

A list with the deviance (deviance) and the adjusted degrees of freedom (numP).

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)
David Edwards (<David.Edwards@agrsci.dk>)

References

Lauritzen, S.L. (1996) *Graphical Models*, Oxford University Press.
Edwards, D. (2000) *Introduction to Graphical Modelling*, Springer-Verlag New York Inc.

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="BIC")
CI.test(20, 29, c(9, 11), dsCont)
#$deviance
#[1] 0.7617515263220724
#
#$numP
#[1] 1
```

convData	<i>Converts dataset</i>
----------	-------------------------

Description

Converts a dataset to a structure required by other functions.

Usage

```
convData(dataset)
```

Arguments

dataset matrix or data frame (discrete variables must be factors).

Details

Convert the dataset to the structure required by the other functions.

Value

List:

dataset Matrix.

numCat Vector with the number of levels in each variable (0 if continuous).

vertNames - vector with the original vertices' names.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
data(dsDiscr)
ds <- convData(dsDiscr)
```

Degree	<i>Degree</i>
--------	---------------

Description

Returns the degree of a set of vertices.

Usage

```
Degree(model=NULL, edges=NULL, v=NULL)
```

Arguments

model	gRaphD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Column 1 contains the vertex with lower index.
v	set of vertices.

Details

Calculates the degree of each vertex in v. If v=NULL, it returns the degree of all vertices in edges. If v contains a vertex not in edges, the corresponding value is NA.

Value

vector	length(v).
--------	------------

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
Degree(model=m1)
```

DFS *Depth-first search*

Description

Returns all vertices reachable from one specific vertex (assuming that there are no cycles).

Usage

```
DFS(model=NULL, edges=NULL, v, p=NULL)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge.
v	initial vertex ($0 < v \leq p$).
p	number of vertices.

Details

Given a list of edges, and a specific vertex v , returns a vector with all vertices in the connected component containing v . The function assumes that the graph is acyclic.

Value

Vector with all vertices reachable from v , or 0 if v is an isolated vertex.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

References

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. *Introduction to Algorithms*, 2nd Edition. MIT Press and McGraw-Hill, 2001, pp.540:9.

Examples

```
set.seed(7, kind="Mersenne-Twister")
dataset <- matrix(rnorm(1000), nrow=100, ncol=10)
m <- minForest(dataset, stat="BIC")

DFS(edges=m@edges, v=1, p=10)
# [1] 5 2 9 8
#####
data(dsDiscr)
m1 <- minForest(dsDiscr, homog=TRUE, forbEdges=NULL, stat="BIC")
```

```

vertices <- DFS(edges=m1@edges, v=1, p=m1@p)

# result
vertices
# numeric(0)
# meaning that 1 is an isolated vertex

# OR
m1 <- minForest(dsDiscr, homog=TRUE, forbEdges=NULL, stat="LR")
vertices <- DFS(edges=m1@edges, v=1, p=m1@p)

# result
vertices
# [1] 4 8 12 19 18 14 7 17 5 3 10 13 15 9 6 20 16 11 2
# meaning that 1 reaches all vertices (a tree)

```

dsCont

Test dataset

Description

Test dataset with only continuous variables.

Format

Matrix containing 174 observations (rows) and 34 variables (columns).

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)

David Edwards (<David.Edwards@agrsci.dk>)

Examples

```
data(dsCont)
```

dsDiscr

Test dataset

Description

Test dataset with only discrete variables.

Format

Data frame containing 200 observations and 20 variables.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)
David Edwards (<David.Edwards@agrsci.dk>)

Examples

```
data(dsDiscr)
```

dsMixed

Test dataset

Description

Test dataset with continuous and discrete variables.

Format

Data frame containing 200 observations and 15 variables. The first 5 variables are discrete, and the last 10 continuous.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)
David Edwards (<David.Edwards@agrsci.dk>)

Examples

```
data(dsMixed)
```

findEd

Finds add-eligible edges

Description

Finds the edges that can be added to a (strongly) triangulated graph such that the result is also (strongly) triangulated.

Usage

```
findEd(edges,p,previous=NULL,numCat,from=0,exact=FALSE,join=FALSE)
```

Arguments

edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge.
p	number of vertices.
previous	result of a previous run of findEd.
numCat	vector with number of levels for each variable (0 if continuous).
from	initial vertex to be used in MCS.
exact	logical indicating if the exact algorithm for finding add-eligible edges is to be used. Default is FALSE.
join	logical indicating if the disjoint components can be joined. Default is FALSE.

Details

Returns all add-eligible edges for a given triangulated graph, that is, edges that preserve the triangulated property when added. In the case of a mixed graph, only edges that do not result in forbidden paths are returned.

The argument `from` can be used to indicate the initial vertex used in the MCS algorithm. If 0, the first vertex is used.

If `exact` is FALSE, the edge list may contain a few edges that are not add-eligible. Further tests (for example `mcs`) will be required before adding edges. Otherwise, the list contains only edges that preserve triangularity. That is, each edge that may be added to the graph such that the resulting graph is triangulated.

For graphs with both discrete and continuous vertices, the graph should be triangulated and contain no forbidden paths, and the edges that may be added preserving both properties are returned. See Lauritzen (1996), p. 11-13.

Value

A list with:

edges.to.test	matrix (k by 5), with columns: 1 - first vertex of the tested edge 2 - second vertex (the two values are ordered) 3 - index for the separator in S 4 - change in the LR by adding this edge 5 - number of parameters for that edge
S	list with the separators.

Author(s)

Gabriel Coelho Goncalves de Abreu (<Gabriel.Abreu@agrsci.dk>)
Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)
David Edwards (<David.Edwards@agrsci.dk>)

References

Lauritzen, S.L. (1996) *Graphical Models*, Oxford University Press.

Examples

```
edges <- matrix(c(1,2,2,3,2,4,2,5,2,6,3,4,4,5,5,6),ncol=2,byrow=TRUE)
addEligible <- findEd(edges=edges,p=6,previous=NULL,numCat=rep(0,6),
                      from=1)

# > str(addEligible)
# List of 2
# $ edges: num [1:7, 1:5] 1 1 3 1 4 3 1 3 4 5 ...
# $ S :List of 6
# ..$ : int 2
# ..$ : int [1:2] 2 4
# ..$ : int 2
# ..$ : int [1:2] 2 5
# ..$ : int 2
# ..$ : int 2
# > addEligible$edges
#      [,1] [,2] [,3] [,4] [,5]
# [1,]  1  3  1  0  0
# [2,]  1  4  1  0  0
# [3,]  3  5  2  0  0
# [4,]  1  5  3  0  0
# [5,]  4  6  4  0  0
# [6,]  3  6  5  0  0
# [7,]  1  6  6  0  0

# the columns in addEligible$edges
# 1: first vertex in the edge
# 2: second vertex in the edge
# 3: index os the separator in addEligible$$
# 4: change in the LRT for the edge (used if previous != NULL)
# 5: number of parameters for the edge (used if previous != NULL)

# note that the edge 3-6 (row 6) is actually a "false positive".
# If it's used from=3,4,5, or 6, this does not happen.
```

fit

*Log-likelihood, AIC, BIC***Description**

Calculates $-2 \times \log$ -likelihood, AIC, and BIC for a triangulated graph (decomposable model).

Usage

```
fit(model=NULL, edges=NULL, dataset, homog=NULL)
```

Arguments

model gRapHD object.

edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge.
dataset	matrix or data frame (nrow(dataset) observations and ncol(dataset) variables).
homog	only used in the mixed model case. TRUE if the model is homogeneous. The default is NULL, indicating that the attribute homog of the model parameter must be used (or TRUE if only edges is provided).

Value

Vector with: model dimension (no of free parameters), $-2 \cdot \log$ -likelihood, AIC, and BIC. Note that all parameters are assumed to be estimable in the dimension calculation.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

References

Lauritzen, S.L. (1996) *Graphical Models*, Oxford University Press.

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
fit(edges=m1@edges, dataset=dsCont)
```

gRapHD-class

Class "gRapHD"

Description

S4 class.

Details

As the gRapHD class does not use variables names, but only the column indexes in the dataset, the conversion may change the variables references. When converting a gRapHD object to a graphNEL object, the nodes names in the new object are only the original indexes converted to character. When doing the reverse conversion, the nodes indexes in the new gRapHD object are the respective indexes in the original element nodes in the graphNEL object. See the example below.

Objects from the Class

Objects can be created, for example, by `new("gRapHD", ...)`. Where ... may contain the initial attribute names and values, as described in the Slots section.

Slots

edges: integer, matrix with 2 columns, each row representing one edges and each column one of the vertices in the edge. Column 1 contains the vertex with lower index.

homog: logical, TRUE if the covariance is homogeneous.

minForest: integer, first and last edges found with minForest.

numCat: integer, vector with number of levels for each variable (0 if continuous).

numP: integer, vector with number of estimated parameters for each edge.

p: integer, number of variables (vertices) in the model.

stat.minForest: character, measure used (LR, AIC, or BIC).

stat.stepw: character, measure used (LR, AIC, or BIC).

stat.user: character, user defined.

statSeq: numeric, vector with value of stat.minForest for each edge.

stepw: integer, first and last edges found with stepw.

userDef: integer, first and last edges defined by the user.

vertNames: character, vector with vertices' names.

Methods

```
setMethod("initialize", "gRapHD", initialize.gRapHD)
setMethod("summary", signature(object="gRapHD"), summary.gRapHD)
setMethod("print", signature(x="gRapHD"), print.gRapHD)
setMethod("show", signature(object="gRapHD"), show.gRapHD)
setMethod("plot", signature(x="gRapHD"), plot.gRapHD)
setAs(from="matrix", to="gRapHD", def=matrix.gRapHD)
setAs(from="gRapHD", to="graphNEL", def=gRapHD.graphNEL)
setAs(from="graphNEL", to="gRapHD", def=graphNEL.gRapHD)
```

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
David Edwards (<David.Edwards@agrsci.dk>)

References

R. Gentleman, Elizabeth Whalen, W. Huber and S. Falcon. graph: A package to handle graph data structures. R package version 1.22.2.

Examples

```

# conversion from gRapHD to graphNEL
edges <- matrix(c(1,2,1,3,1,4),,2,byrow=TRUE)
g <- as(edges,"gRapHD")
#List of 9
# $ edges      : num [1:3, 1:2] 1 1 1 2 3 4
# $ p          : int 4
# $ stat.user  : chr "LR"
# $ statSeq    : num [1:3] NA NA NA
# $ varType    : int [1:4] 0 0 0 0
# $ numCat     : int [1:4] 0 0 0 0
# $ homog      : logi TRUE
# $ numP       : num [1:3] NA NA NA
# $ userDef    : num [1:2] 1 3
# - attr(*, "class")= chr "gRapHD"
g1 <- as(g,"graphNEL")
# A graphNEL graph with undirected edges
# Number of Nodes = 4
# Number of Edges = 3
g1@nodes # the nodes names

as(matrix(integer(0),,2),"gRapHD")
# note that the vertices must be numbered consecutively from 1. In the
# following, vertex 2 is added as an isolated vertex.
m1 <- as(matrix(c(1,3,1,4),,2,byrow=TRUE),"gRapHD")
## Not run: plot(m1)

```

jTree

Junction tree

Description

Finds a junction tree.

Usage

```
jTree(model)
```

Arguments

model object of gRapHD class.

Details

Returns one possible junction tree. Note that each edge is associated to one separator in the list, and a separator may be contained in other(s) separator(s). To identify which separator is associated to each edge is enough to check `ind<-indSepOrig[which(indSepOrig!=1)]`. In this way, the edge `juncTree[i,]` is associated with separator `ind[i]`.

Value

A list with:

separators	list with unique minimal separators.
juncTree	edges in the tree (each vertex is a clique in the list below).
sepSubSetOfSep	list in which each element gives all the separators which contain this respective separator.
indSepOrig	index of the original separator (in the MCS result) in the list above.
cliques	list with cliques.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
edges <- matrix(c(1,2,2,3,2,4,2,5,2,6,3,4,4,
                 5,5,6,7,8,7,9,8,9,8,10,9,10),ncol=2,byrow=TRUE)
m <- new("gRapHD",edges=edges)
jT <- jTree(m)
str(jT)
# List of 5
# $ separators      :List of 5
# ..$ : int(0)
# ..$ : int 2
# ..$ : int [1:2] 2 4
# ..$ : int [1:2] 2 5
# ..$ : int [1:2] 8 9
# $ juncTree        : int [1:4, 1:2] 1 2 3 5 2 3 4 6
# $ sepSubSetOfSep:List of 5
# ..$ : int [1:4] 2 3 4 5
# ..$ : int [1:2] 3 4
# ..$ : int(0)
# ..$ : int(0)
# ..$ : int(0)
# $ indSepOrig      : int [1:6] 1 2 3 4 1 5
# $ cliques         :List of 6
# ..$ : int [1:2] 1 2
# ..$ : int [1:3] 2 3 4
# ..$ : int [1:3] 2 4 5
# ..$ : int [1:3] 2 5 6
# ..$ : int [1:3] 7 8 9
# ..$ : int [1:3] 8 9 10
```

MCS

Maximum cardinality search

Description

Returns a perfect ordering of the edges.

Usage

```
MCS(model=NULL, edges=NULL, v=0, p=NULL)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge.
v	initial vertex ($0 \leq v \leq p$). If $v=0$, the algorithm starts from vertex 1.
p	number of vertices.

Details

Returns a perfect ordering of the vertices. For mixed graphs, the discrete vertices appear before the continuous vertices, in each connected component.

Value

Zero if the graph is not triangulated, or a vector with the number of each vertex in the perfect ordering.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

References

Tarjan, R.E., Yannakakis, M. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, Vol 13, 3:566-79.

Leimer, H. Triangulated graphs with marked vertices. *Ann. Discr. Maths.*, Vol 41, 311-324.

Examples

```
set.seed(7, kind="Mersenne-Twister")
dataset <- matrix(rnorm(1000), nrow=100, ncol=10)
m <- minForest(dataset, stat="BIC")
```

```
MCS(edges=m@edges, v=1, p=10)
```

minForest	<i>Minimum forest</i>
-----------	-----------------------

Description

Returns the forest that minimises the $-2 \cdot \log$ -likelihood, AIC, or BIC using Chow-Liu algorithm.

Usage

```
minForest(dataset, homog=TRUE, forbEdges=NULL, stat="BIC",
          cond=NULL, ...)
```

Arguments

dataset	matrix or data frame (nrow(dataset) observations and ncol(dataset) variables).
homog	TRUE for homogeneous covariance structure, FALSE for heterogeneous. This is only meaningful with mixed models. Default is homogeneous (TRUE).
forbEdges	matrix specifying edges that should not be considered. Matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Default is NULL.
stat	measure to be minimized: LR ($-2 \cdot \log$ -likelihood), AIC, or BIC. Default is LR. It can also be a user defined function with format: FUN(newEdge, numCat, dataset); where numCat is a vector with number of levels for each variable (0 if continuous); newEdge is a vector with length two; and dataset is a matrix (n by p).
cond	list with complete sets of vertices, to specify mandatory edges.
...	arguments to be passed to the user function in stat.

Details

Returns for the tree or forest that minimizes the $-2 \cdot \log$ -likelihood, AIC, or BIC. If the log-likelihood is used, the result is a tree, if AIC or BIC is used, the result is a tree or forest. The dataset contains variables (vertices) in the columns, and observations in the rows. The result has vertices numbered according to the column indexes in vertNames.

All discrete variables must be factors. All factor levels must be represented in the data. Missing values are not allowed.

Value

A list containing:

edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Column 1 contains the vertex with lower index.
p	number of variables (vertices) in the model.
stat.minForest	measure used (LR, AIC, or BIC).

statSeq	vector with value of stat.minForest for each edge.
vertNames	vector with the original vertices' names. If there are no names in dataset then the vertices will be named according to the original column indexes in dataset.
numCat	vector with number of levels for each variable (0 if continuous).
homog	TRUE if the covariance is homogeneous.
numP	vector with number of estimated parameters for each edge.
minForest	first and last edges found with minForest.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>
 Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>
 David Edwards (<David.Edwards@agrsci.dk>

References

Chow, C.K. and Liu, C.N. (1968) Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, Vol. IT-14, 3:462-7.
 Edwards, D., de Abreu, G.C.G. and Labouriau, R. (2010). Selecting high- dimensional mixed graphical models using minimal AIC or BIC forests. *BMC Bioinformatics*, 11:18.

Examples

```
set.seed(7,kind="Mersenne-Twister")
dataset <- matrix(rnorm(1000),nrow=100,ncol=10)
m <- minForest(dataset,stat="BIC")

#####
# Example with continuous variables
data(dsCont)
# m1 <- minForest(dataset,homog=TRUE,forbEdges=NULL,stat="LR")
#       1. in this case, there is no use for homog
#       2. no forbidden edges
#       3. the measure used is the LR (the result is a tree)
m1 <- minForest(dsCont,homog=TRUE,forbEdges=NULL,stat="LR")
plot(m1,numIter=1000)

#####
# Example with discrete variables
data(dsDiscr)
# m1 <- minForest(dataset,homog=TRUE,forbEdges=NULL,stat="LR")
#       1. in this case, there is no use for homog
#       2. no forbidden edges
#       3. the measure used is the LR (the result is a tree)
m1 <- minForest(dsDiscr,homog=TRUE,forbEdges=NULL,stat="LR")
plot(m1,numIter=1000)

#####
# Example with mixed variables
data(dsMixed)
```

```

# m1 <- minForest(dataset,homog=TRUE,forbEdges=NULL,stat="LR")
#       1. it is to be considered homogeneous
#       2. no forbidden edges
#       3. the measure used is the LR (the result is a tree)
m1 <- minForest(dsMixed,homog=TRUE,forbEdges=NULL,stat="LR")
plot(m1,numIter=1000)

#####
# Example using a user defined function
# The function userFun calculates the same edges weights as the
# option stat="LR". It means that the final result, using either
# option, is the same.
userFun <- function(newEdge,numCat,dataset)
{
  sigma <- var(dataset[,newEdge])
  v <- nrow(dataset)*log(prod(diag(sigma))/det(sigma))
  return(c(v,1))
}

data(dsCont)
m <- minForest(dsCont,stat="LR")
m1 <- minForest(dsCont,stat=userFun)
identical(m@edges,m1@edges)

#####
# Example with mandatory edges (the so-called conditional Chow-Liu
# algorithm). The edges (1,2), (1,3) and (2,3) are specified as
# mandatory. The algorithm returns the optimal graph containing the
# mandatory edges such that only cycles with mandatory edges are
# allowed.
data(dsCont)
m1 <- minForest(dsCont,cond=list(1:3))
## Not run: plot(m1)

```

modelDim

Model's dimension

Description

Calculates the number of free parameters in the model.

Usage

```
modelDim(model)
```

Arguments

model gRapHD class.

Details

See Lauritzen (1996), pages 202-203, and 215-216 for more details.

Value

Number of free parameters in the model.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)

References

Lauritzen, S.L. (1996) *Graphical Models*, Oxford University Press.

Examples

```
data(dsCont)
m <- minForest(dsCont, stat="BIC")
modelDim(m)
# 102
```

```
m <- stepw(m, dsCont, stat="BIC")
modelDim(m)
# 149
```

modelFormula

Model's formula

Description

Returns the formula of the model.

Usage

```
modelFormula(model)
```

Arguments

model gRapHD class.

Details

See Lauritzen (1996), pages 202-203, and 215-216 for more details.

Value

List with the generators of the model:

discrete	terms (d, \emptyset)
linear	terms (d, γ^2)
quadratic	terms (d, γ) and $(d, \{\gamma, \mu\})$
quadratic2	terms (d, c^2)

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

References

Lauritzen, S.L. (1996) *Graphical Models*, Oxford University Press.

Examples

```

data(dsMixed)
m <- minForest(dsMixed, homog=TRUE, stat="LR")
str(modelFormula(m))
#List of 4
# $ discrete :List of 4
# ..$ : int [1:2] 1 3
# ..$ : int [1:2] 3 4
# ..$ : int [1:2] 3 5
# ..$ : int [1:2] 5 2
# $ linear : list()
# $ quadratic :List of 5
# ..$ : num [1:2] 5 8
# ..$ : num [1:2] 5 9
# ..$ : num [1:2] 4 11
# ..$ : num [1:2] 5 14
# ..$ : num [1:2] 2 15
# $ quadratic2:List of 6
# ..$ : num [1:2] 6 11
# ..$ : num [1:2] 7 8
# ..$ : num [1:2] 9 10
# ..$ : num [1:2] 9 13
# ..$ : num [1:2] 12 15
# ..$ : num 14

m <- minForest(dsMixed, homog=FALSE, stat="LR")
str(modelFormula(m))
#List of 4
# $ discrete :List of 4
# ..$ : int [1:2] 1 3
# ..$ : int [1:2] 3 4
# ..$ : int [1:2] 3 5
# ..$ : int [1:2] 5 2

```

```

# $ linear      :List of 10
# ..$ : num [1:2] 2 6
# ..$ : num [1:2] 5 7
# ..$ : num [1:2] 5 8
# ..$ : num [1:2] 5 9
# ..$ : num [1:2] 5 10
# ..$ : num [1:2] 2 11
# ..$ : num [1:2] 3 12
# ..$ : num [1:2] 2 13
# ..$ : num [1:2] 5 14
# ..$ : num [1:2] 2 15
# $ quadratic : list()
# $ quadratic2: list()

```

neighbourhood

Neighbourhood of a vertex

Description

Finds the set of vertices with up to a given distance from a given vertex.

Usage

```
neighbourhood(model=NULL, edges=NULL, orig=NULL, rad=1)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Column 1 contains the vertex with lower index.
orig	central vertex.
rad	distance.

Details

Finds the set of vertices with up to a given distance from a given vertex.

Value

Returns a list with:

subEdges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Column 1 contains the vertex with lower index.
v	matrix with 2 columns, the first indicating the vertex index, and the second the distance to the orig.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
aux <- neighbourhood(model=m1, orig=27, rad=2)
plot(new("gRapHD", edges=aux$edges, p=m1@p), vert=aux$v[,1])
```

neighbours*Finds all direct neighbours*

Description

Finds all direct neighbours of a given vertex in a given graph.

Usage

```
neighbours(model=NULL, edges=NULL, v)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge.
v	reference vertex.

Details

Returns all vertices with a direct connection with vertex v in edges.

Value

Vector with all neighbours of vertex v.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
neigh <- neighbours(edges=m1@edges, v=22)
# > neigh
# [1] 3 9 24 25
```

perfSets *Finds a perfect sequence*

Description

Finds a perfect sequence, returning the cliques, histories, residuals, and separators of a given triangulated graph.

Usage

```
perfSets(model=NULL, edges=NULL, p=NULL, varType=0, from=0)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge.
p	number of vertices.
varType	vector indicating the type of each variable: 0 if continuous, or 1 if discrete.
from	initial vertex to be used in MCS.

Details

Based on the perfect numbering of mcs, returns the perfect sequence.

The sequence is given by the cliques in the graph: $C_j = \text{closure}(\alpha_j) \cap \{\alpha_1, \dots, \alpha_j\}$, $j \geq 1$.

The other sets are given by:

- Histories: $H_j = C_1 \cup \dots \cup C_j$
- Residuals: $R_j = C_j \setminus H_{j-1}$
- Separators: $S_j = H_{j-1} \cap C_j$

Value

A list containing:

cliques	list.
histories	list.
residuals	list.
separators	list.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

References

Lauritzen, S.L. (1996) *Graphical Models*, Oxford University Press.

Examples

```
edges <- matrix(c(1,2,2,3,2,4,2,5,2,6,3,4,4,5,5,6),ncol=2,byrow=TRUE)
setList <- perfSets(edges=edges, p=6, varType=0, from=1)
# > str(setList)
# List of 4
# $ cliques :List of 4
# ..$ : int [1:2] 1 2
# ..$ : int [1:3] 2 3 4
# ..$ : int [1:3] 2 4 5
# ..$ : int [1:3] 2 5 6
# $ histories :List of 4
# ..$ : int [1:2] 1 2
# ..$ : int [1:4] 1 2 3 4
# ..$ : int [1:5] 1 2 3 4 5
# ..$ : int [1:6] 1 2 3 4 5 6
# $ separators:List of 4
# ..$ : NULL
# ..$ : int 2
# ..$ : int [1:2] 2 4
# ..$ : int [1:2] 2 5
# $ residuals :List of 4
# ..$ : int [1:2] 1 2
# ..$ : int [1:2] 3 4
# ..$ : int 5
# ..$ : int 6
```

plot.gRapHD

Plots a gRapHD object

Description

[Methods](#) for class gRapHD.

Usage

```
## S3 method for class 'gRapHD'
plot(x, vert=NULL, numIter=50, main="", plotVert=TRUE, plotEdges=TRUE,
     energy=FALSE, useWeights=FALSE, vert.hl=NULL, col.hl="red",
     vert.radii=0.01, coord=NULL, col.ed="darkgray", lty.ed=1, lwd.ed=1,
     lwd.vert=1, border=0, symbol.vert=1, cex.vert.label=.40,
     vert.labels=TRUE, asp=NA, disp=TRUE, font=par("font"),
     col.labels=NULL, add=FALSE, ...)
```

Arguments

x	a gRapHD object.
vert	vector of vertices to be plotted. If NULL, all vertices will be plotted
numIter	number of iterations for the Fruchterman-Reingold algorithm.

<code>main</code>	main title.
<code>plotVert</code>	if TRUE the vertices are plotted.
<code>plotEdges</code>	if TRUE the edges are plotted.
<code>energy</code>	if TRUE use the minimum energy as initial values.
<code>useWeights</code>	if TRUE use the <code>model\$statSeq</code> as edge length weights).
<code>vert.hl</code>	vector of vertices to be highlighted.
<code>col.hl</code>	colour to be used in the highlighted vertices.
<code>vert.radii</code>	radii of the vertices (scalar or vector with length equal to the number of vertices). See 'Details'.
<code>coord</code>	initial coordinate values for the vertices.
<code>col.ed</code>	colour of the edges (scalar or vector).
<code>lty.ed</code>	type of line for the edges (scalar or vector).
<code>lwd.ed</code>	width of the line for the edges (scalar or vector).
<code>lwd.vert</code>	width of the vertices' border (scalar or vector).
<code>border</code>	colour of the vertices borders (scalar or vector).
<code>symbol.vert</code>	symbol to be used in the vertices (length 1 or number of vertices): 0 is a ellipse ($a=2*vs, b=vs$), 1 a circle, 2 a square, 3 or higher represents the number of sides (scalar or vector).
<code>cex.vert.label</code>	numeric character expansion factor for the labels; multiplied by <code>par</code> yields the final character size. NULL and NA are equivalent to 1.0 (scalar or vector).
<code>vert.labels</code>	labels to be used in the vertices. If FALSE, the vertices are not labeled (scalar or vector).
<code>asp</code>	numeric, giving the aspect ratio y/x (see <code>plot.window</code> for more details).
<code>disp</code>	if TRUE (default), the graph is plotted.
<code>font</code>	an integer which specifies which font to use for the labels. If possible, device drivers let 1 correspond to plain text (the default), 2 to bold face, 3 to italic and 4 to bold italic.
<code>col.labels</code>	colour of the labels. Default is NULL, using black for continuous vertices and white for discrete (scalar or vector).
<code>add</code>	if <code>add</code> is TRUE, the graph is added to an existing plot (the ACTIVE one), otherwise a new plot is created.
<code>...</code>	further arguments passed to or from other methods.

Details

Plot a graph based on the list of edges.

Only one (`model` or `edges`) should be provided. If `model`, the function uses also the information about the type of variables (discrete or continuous). If `edges`, then all variables are plotted as continuous (circles).

The plotting area is square, ranging from 0 to 1. The unit of parameter `vs` follow the axes.

The algorithm proposed by Fruchterman & Reingold (1991) is used to determine the position of each vertex. It is not initialised randomly, but in a regular grid.

The graph is always positioned in a square centred in (0.5,0.5) with sides of length one.

Value

The coordinates for the vertices are returned, as a matrix. This may be used in subsequent calls to plot using the coord argument, usually with numIter=0.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)

References

Fruchterman, T.M.J. and Reingold, E.M. (1991) Graph Drawing by Force-directed Placement. *Software-Practice and Experience*, Vol. 21(11), 1129-1164.

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. <http://igraph.sf.net>

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
plot(m1, numIter=1000)

# or
plot(m1, numIter=1000, plotVert=FALSE, labelVert=FALSE)

#####
r <- 3
edges <- rep(1, r)
x <- 2+r-1
edges <- c(edges, sort(rep(2:x, r-1)))
edges <- c(edges, sort(rep((x+1):(x+(x-1)*(r-1)), r-2)))
edges <- c(edges, sort(rep((x+(x-1)*(r-1)+1):(x+(x-1)*(r-1)+
(x-1)*(r-1)*(r-2)), r-3)))
edges <- cbind(edges, 2:(length(edges)+1))
a <- neighbourhood(edges=edges, orig=1, rad=r)
vs <- a$V[, 2]
vs <- 1/vs
vs[1] <- 2
vs <- vs/30
model <- new("gRapHD", edges=edges)
plot(model, numIter=200, col.hl=colours()[386:383][a$V[, 2]+1],
      vert.hl=a$V[, 1], vert.radii=vs, border="black", lwd.vert=2)
```

 randTree

Random tree

Description

Generates a random tree

Usage

```
randTree(p, seed=1)
```

Arguments

p	number of vertices.
seed	seed (for the random generator, see set.seed).

Value

A list containing:

edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge.
seed	seed.
p	number of vertices.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

References

Rodionov, A.S., Choo, H. On Generating Random Network Structures: Trees. *Springer-Verlag Lecture Notes in Computer Science*, vol. 2658, pp. 879-887, June 2003.

Examples

```
tree <- randTree(p=10, seed=1)
plot(new("gRapHD", edges=tree$edges))
tree
# $edges
#      [,1] [,2]
# [1,]   3   4
# [2,]   3   5
# [3,]   2   3
# [4,]   4   6
# [5,]   6   9
# [6,]   1   6
# [7,]   6  10
# [8,]   6   7
# [9,]   5   8
#
# $seed
# [1] 1
#
# $p
# [1] 10
```

rowProds	<i>Row products</i>
----------	---------------------

Description

Form row products for numeric arrays.

Usage

```
rowProds(x, na.rm=TRUE)
```

Arguments

x	matrix.
na.rm	logical. Whether missing values (including NaN) are omitted from the calculations.

Details

Equivalent to use of apply with FUN = prod and MARGIN = 1, but is faster.

Value

Vector with length nrow(x).

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
set.seed(1, kind="Mersenne-Twister")
a <- matrix(rnorm(100), nrow=5)
x <- rowProds(x=a, na.rm=TRUE)
# x
# [1] -3.359208e-07 -2.861043e-10 -2.831108e-08
# [4] -5.451996e-07 3.057436e-04
```

shortPath	<i>Shortest path</i>
-----------	----------------------

Description

Shortest paths between one vertex and all other vertices.

Usage

```
shortPath(model=NULL, edges=NULL, v=NULL, p=NULL)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge. Column 1 contains the vertex with lower index.
v	vertex.
p	number of vertices. If NULL, the number of vertices will be considered as equal the maximum vertices' index, i.e., $p = \max(\text{edges})$.

Details

Calculates the shortest path between the vertex v and all other vertices.

Value

vector with length equal p , with the shortest path length from v to each other vertex.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
shortPath(edges=m1@edges, v=1)
```

sortMat	<i>Sort matrix</i>
---------	--------------------

Description

Sorts the rows of a matrix by given columns.

Usage

```
sortMat(mat,cols)
```

Arguments

mat	matrix.
cols	sequence os columns to sort by.

Details

It is just a interface to the function [order](#).

Value

Matrix.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
set.seed(1,kind="Mersenne-Twister")
a <- matrix(c(sample(2,6,TRUE),sample(3,6,TRUE),sample(2,6,TRUE)),
            nrow=6)
x <- sortMat(mat=a, cols=c(1:3))
a
#      [,1] [,2] [,3]
# [1,]  1   3   2
# [2,]  1   2   1
# [3,]  2   2   2
# [4,]  2   1   1
# [5,]  1   1   2
# [6,]  2   1   2
x
#      [,1] [,2] [,3]
# [1,]  1   1   2
# [2,]  1   2   1
# [3,]  1   3   2
# [4,]  2   1   1
# [5,]  2   1   2
# [6,]  2   2   2
```

stepw	<i>Stepwise forward selection</i>
-------	-----------------------------------

Description

Stepwise forward selection.

Usage

```
stepw(model, dataset, stat="BIC", saveCH=NULL, forbEdges=NULL,
       exact=FALSE, initial=NULL, threshold=0, join=FALSE)
```

Arguments

model	gRapHD object.
dataset	matrix (nrow(dataset) observations and ncol(dataset) variables).
stat	measure to be minimized: LR, AIC, or BIC. Default is BIC. It can also be a user defined function with format: FUN(model, dataset, previous, forbEdges); where the parameters are defined as in chStat . The function must return a structure as in chStat .
saveCH	pattern of a file name to save each iteration results. NULL not to save (default).
forbEdges	list with edges that should not be considered. Matrix with 2 columns, each row representing one edge, and each column a vertex. Default is NULL.
exact	logical indicating whether the exact algorithm for finding add-eligible edges is to be used. Default is FALSE.
initial	continue the algorithm from a previous point. The parameter must have the same structure as the result of chStat .
threshold	values greater than the threshold are not included in the model. Default is 0.
join	logical indicating whether disjoint components can be joined. Default is FALSE.

Details

Performs a stepwise forward selection of edges to be added to a triangulated graph. Only edges preserving the triangularity are considered ([findEd](#)). At each step the edge giving the greatest improvement in the chosen statistic is added. The process ends when no further improvement is possible.

If `exact` is FALSE, the list of edges returned may contain a few edges that do not preserve triangularity, requiring further tests (for example `mcs`) before adding an edge. Otherwise, the list will only contain edges that preserve triangularity. That is, each edge that may be added to the graph such that the resulting graph is triangulated.

For graphs with both discrete and continuous vertices, the graph should be triangulated and contain no forbidden paths, and the edges that may be added must preserve both properties. See Lauritzen (1996), p. 11-13.

Value

The same structure as `model`, but adding the new edges, and updating other relevant information. See [minForest](#) for the description of the structure.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)
 Rodrigo Labouriau (<Rodrigo.Labouriau@agrsci.dk>)
 David Edwards (<David.Edwards@agrsci.dk>)

Examples

```
set.seed(7,kind="Mersenne-Twister")
dataset <- matrix(rnorm(1000),nrow=100,ncol=10)
m <- minForest(dataset,stat="BIC")

M <- stepw(m,dataset,stat="LR",NULL,NULL)

#####
# Example with continuous variables
data(dsCont)
# m1 <- minForest(dataset,homog=TRUE,forbEdges=NULL,stat="LR")
#       1. in this case, there is no use for homog
#       2. no forbidden edges
#       3. the measure used is the LR (the result is a tree)
m1 <- minForest(dsCont,homog=TRUE,forbEdges=NULL,stat="LR")
plot(m1,numIter=1000)

# m2 <- stepw(m1,dataset,stat="BIC",saveCH=NULL,forbEdges=NULL)
#       1. m1 is the result of minForest
#       2. the same dataset (this is not checked)
#       3. the default is BIC
#       4. if saveCH="XXX", a file "XXX_00000i.RData"
#           is saved for each iter
#       5. no forbidden edges
m2 <- stepw(m1,dsCont,stat="BIC",saveCH=NULL,forbEdges=NULL)
plot(m2,numIter=1000)

#####
# Example with discrete variables
data(dsDiscr)
# m1 <- minForest(dataset,homog=TRUE,forbEdges=NULL,stat="LR")
#       1. in this case, there is no use for homog
#       2. no forbidden edges
#       3. the measure used is the LR (the result is a tree)
m1 <- minForest(dsDiscr,homog=TRUE,forbEdges=NULL,stat="LR")
plot(m1,numIter=1000)

# m2 <- stepw(m1,dataset,stat="BIC",saveCH=NULL,forbEdges=NULL)
#       1. m1 is the result of minForest
#       2. the same dataset (this is not checked)
#       3. the default is BIC
```

```

#           4. if saveCH="XXX", a file "XXX_00000i.RData"
#           is saved for each iter
#           5. no forbidden edges
m2 <- stepw(m1,dsDiscr,stat="BIC",saveCH=NULL,forbEdges=NULL)
plot(m2,numIter=1000)

#####
# Example with mixed variables
data(dsMixed)
# m1 <- minForest(dataset,homog=TRUE,forbEdges=NULL,stat="LR")
#           1. it is to be considered homogeneous
#           2. no forbidden edges
#           3. the measure used is the LR (the result is a tree)
m1 <- minForest(dsMixed,homog=TRUE,forbEdges=NULL,stat="LR")
plot(m1,numIter=1000)

# m2 <- stepw(m1,dataset,stat="BIC",saveCH=NULL,forbEdges=NULL)
#           1. m1 is the result of minForest
#           2. the same dataset (this is not checked)
#           3. the default is BIC
#           4. if saveCH="XXX", a file "XXX_00000i.RData"
#           is saved for each iter
#           5. no forbidden edges
m2 <- stepw(m1,dsMixed,stat="BIC",saveCH=NULL,forbEdges=NULL)
plot(m2,numIter=1000)

#####
# Example using a user defined function
# The function userFun calculates the same edges weights as the
# option stat="BIC". It means that the final result, using either
# option, is the same.
userFun <- function(model,dataset,previous=NULL,forbEdges=NULL)
{
  p <- ncol(dataset) # number of variables (vertices)
  n <- nrow(dataset) # number of observations
  edges.to.test <- previous$edges.to.test
  SS <- previous$S # minimal separators
  rm(previous)

  num <- nrow(edges.to.test)

  if (num > 0)
    for (i in 1:num)
    {
      x <- edges.to.test[i,1]
      y <- edges.to.test[i,2]
      if ((edges.to.test[i,4]==0) &
          (!is.element((x-1)*p-(x-1)*x/2+y-x,forbEdges)))
      {
        S <- SS[[edges.to.test[i,3]]]
        clique <- c(edges.to.test[i,1:2],S)
        CM <- cov(dataset[,clique],use="pairwise.complete.obs")*
              (nrow(dataset)-1)
      }
    }
}

```

```

    a <- match(edges.to.test[i,1:2],clique)
    d <- log(det(CM)) +
      log(det(matrix(CM[-a,-a],length(clique)-2)))
    d <- d - (log(det(matrix(CM[-a[1],-a[1]],length(clique)-1))) +
      log(det(matrix(CM[-a[2],-a[2]],length(clique)-1))))
    edges.to.test[i,4] <- nrow(dataset)*d + log(n)
    edges.to.test[i,5] <- 1
  }
}
return(list(edges.to.test=edges.to.test,S=SS))
}

data(dsCont)
m <- minForest(dsCont,stat="BIC")
m1 <- stepw(m,dsCont,stat="BIC")
m2 <- stepw(m,dsCont,stat=userFun)
identical(m1@edges,m2@edges)

#####
# Example of joining disjoint components
data(dsCont)
m <- minForest(dsCont,stat="BIC")
m1 <- stepw(m,dsCont)
m2 <- m
m2@edges <- matrix(integer(0),,2)
m3 <- stepw(m2,dsCont,join=TRUE)
identical(sortMat(m1@edges),sortMat(m3@edges))
# TRUE

```

SubGraph

Generates a subgraph

Description

Generates a sub-graph.

Usage

```
SubGraph(model=NULL, edges=NULL, v=NULL, p=0)
```

Arguments

model	gRapHD object.
edges	matrix with 2 columns, each row representing one edge, and each column one of the vertices in the edge.
v	list of vertices in the sub-graph.
p	Number of vartices (used only if edges is not NULL).

Details

Based on a list of vertices, generate a sub-graph.

Value

Returns a gRapHD object, in which the edge list contains only edges where both vertices are in *v*.

Author(s)

Gabriel Coelho Goncalves de Abreu (<abreu_ga@yahoo.com.br>)

Examples

```
data(dsCont)
m1 <- minForest(dsCont, homog=TRUE, forbEdges=NULL, stat="LR")
plot(m1, numIter=1000)

v <- c(1, 11, 21, 19, 30, 25, 22, 24, 34, 9, 20, 29)
subM1 <- SubGraph(model=m1, v=v)
plot(subM1, numIter=1500)
```

Index

- *Topic **array**
 - rowProds, [33](#)
 - sortMat, [35](#)
- *Topic **classes**
 - gRapHD-class, [16](#)
- *Topic **datasets**
 - dsCont, [12](#)
 - dsDiscr, [12](#)
 - dsMixed, [13](#)
- *Topic **dplot**
 - plot.gRapHD, [29](#)
- *Topic **graphs**
 - adjMat, [3](#)
 - calcStat, [4](#)
 - ccoeff, [5](#)
 - chStat, [6](#)
 - CI.test, [8](#)
 - convData, [9](#)
 - Degree, [10](#)
 - DFS, [11](#)
 - findEd, [13](#)
 - fit, [15](#)
 - jTree, [18](#)
 - MCS, [20](#)
 - minForest, [21](#)
 - modelDim, [23](#)
 - modelFormula, [24](#)
 - neighbourhood, [26](#)
 - neighbours, [27](#)
 - perfSets, [28](#)
 - plot.gRapHD, [29](#)
 - randTree, [31](#)
 - shortPath, [34](#)
 - stepw, [36](#)
 - SubGraph, [39](#)
- adjMat, [3](#)
- calcStat, [4](#)
- ccoeff, [5](#)
- chStat, [6](#), [36](#)
- CI.test, [8](#)
- convData, [9](#)
- Degree, [10](#)
- DFS, [11](#)
- dsCont, [12](#)
- dsDiscr, [12](#)
- dsMixed, [13](#)
- findEd, [13](#), [36](#)
- fit, [8](#), [15](#)
- gRapHD (gRapHD-package), [2](#)
- gRapHD-class, [16](#)
- gRapHD-package, [2](#)
- gRapHD.graphNEL (gRapHD-class), [16](#)
- graphNEL.gRapHD (gRapHD-class), [16](#)
- initialize.gRapHD (gRapHD-class), [16](#)
- jTree, [18](#)
- matrix.gRapHD (gRapHD-class), [16](#)
- MCS, [14](#), [20](#), [28](#)
- Methods, [29](#)
- minForest, [21](#), [37](#)
- modelDim, [23](#)
- modelFormula, [24](#)
- neighbourhood, [26](#)
- neighbours, [27](#)
- order, [35](#)
- par, [30](#)
- perfSets, [28](#)
- plot.gRapHD, [29](#)
- plot.window, [30](#)
- print.gRapHD (gRapHD-class), [16](#)
- randTree, [31](#)

rowProds, [33](#)

set.seed, [32](#)

shortPath, [34](#)

show.gRapHD (gRapHD-class), [16](#)

sortMat, [35](#)

stepw, [6](#), [36](#)

SubGraph, [39](#)

summary.gRapHD (gRapHD-class), [16](#)