Package 'deal'

May 8, 2026

Version 1.2-42

Date 2022-11-09

Title Learning Bayesian Networks with Mixed Variables

Author Susanne Gammelgaard Bottcher, Claus Dethlefsen.

Maintainer Claus Dethlefsen <rpackage.deal@gmail.com>

Depends R (>= 2.0.0)

Description Bayesian networks with continuous and/or discrete variables can be learned and compared from data. The method is described in Boettcher and Dethlefsen (2003), <doi:10.18637/jss.v008.i20>.

License GPL (>= 2)

Collate addarrows.R autosearch.R conditional.R cycletest.R drawnetwork.R findex.R generic.R genlatex.R heuristic.R inspectprob.R jointcont.R jointdisc.R jointprior.R learning.R makesimprob.R fullsimprob.R maketrylist.R master.R network.R networkfamily.R node.R numbermixed.R perturb.R postc.R postc0.R postdist.R readnet.R rnetwork.R savenet.R unique.R

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-11-09 21:50:23 UTC

Contents

`autosearch` *Greedy search*

Description

From initial network, does local perturbations to increase network score.

Usage

```
autosearch(initnw,data,prior=jointprior(network(data)),maxiter=50,
           trylist= vector("list",size(initnw)),trace=TRUE,
           timetrace=TRUE,showban=FALSE,removecycles=FALSE)

heuristic(initnw,data,prior=jointprior(network(data)),
          maxiter=100,restart=10,degree=size(initnw),
          trylist= vector("list",size(initnw)),trace=TRUE,
          timetrace=TRUE,removecycles=FALSE)
gettable(x)
```

Arguments

`initnw`	an object of class `network`, from which the search is started.
`data`	a data frame used for learning the network, see `network`.
`prior`	a list containing parameter priors, generated by `jointprior`.
`maxiter`	an integer, which gives the maximum number of steps in the search algorithm.
`restart`	an integer, which gives the number of times to perturb `initnw` and rerun the search.
`degree`	an integer, which gives the degree of perturbation, see `perturb`.
`trylist`	a list used internally for reusing learning of nodes, see `maketrylist`.
`trace`	a logical. If `TRUE`, plots the accepted networks during search.
`timetrace`	a logical. If `TRUE`, prints some timing information on the screen.

showban	a logical passed to the plot method for network objects. If FALSE, the banned arrows are not shown in the plots (if trace is TRUE).
removecycles	a logical. If TRUE, all networks explored in the search is returned, except for networks containing a cycle. If FALSE, all networks are returned, including cyclic networks.
x	an output object from a search.

Details

In autosearch, a list of networks is in each step created with either one arrow added, one arrow deleted or one arrow turned (if a cycle is not generated). The network scores of all the proposal networks are calculated and the network with the highest score is chosen for the next step in the search. If no proposed network has a higher network score than the previous network, the search is terminated. The network with the highest network score is returned, along with a list containing all tried networks (depending on the value of removecycles).

heuristic restarts by perturbing initnw degree times and calling autosearch again. The number of restarts is given by the option restart.

Value

autosearch and heuristic returns a list with three elements, that may be accessed using getnetwork, gettable and gettrylist. The elements are

nw	an object of class network, which gives the network with the highest score.
table	a table with all tried networks. If removecycles is FALSE, the networks may contain cycles. The table contains two columns: model with a string representation of the model and score with the corresponding log network score. The table can be translated to a networkfamily using makenw.
trylist	an updated list used internally for reusing learning of nodes, see maketrylist.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

See Also

perturb

Examples

```
data(rats)
fit       <- network(rats)
fit.prior <- jointprior(fit,12)
fit       <- getnetwork(learn(fit,rats,fit.prior))
fit       <- getnetwork(insert(fit,2,1,rats,fit.prior))
fit       <- getnetwork(insert(fit,1,3,rats,fit.prior))
hisc      <- autosearch(fit,rats,fit.prior,trace=FALSE)
hisc      <- autosearch(fit,rats,fit.prior,trace=FALSE,removecycles=TRUE) # slower
```

```
plot(getnetwork(hisc))

hisc2     <- heuristic(fit,rats,fit.prior,restart=10,trace=FALSE)
plot(getnetwork(hisc2))
print(modelstring(getnetwork(hisc2)))
plot(makenw(gettable(hisc2),fit))
```

---

drawnetwork *Graphical interface for editing networks*

Description

drawnetwork allows the user to specify a Bayesian network through a point and click interface.

Usage

```
drawnetwork(nw,df,prior,trylist=vector("list",size(nw)),
            unitscale=20,cexscale=8,
            arrowlength=.25,nocalc=FALSE,
            yr=c(0,350),xr=yr,...)
```

Arguments

nw	an object of class network to be edited.
df	a data frame used for learning the network, see network.
prior	a list containing parameter priors, generated by jointprior.
trylist	a list used internally for reusing learning of nodes, see maketrylist.
cexscale	a numeric passed to the plot method for network objects. Measures the scaled size of text and symbols.
arrowlength	a numeric passed to the plot method for network objects. Measures the length of the edges of the arrowheads.
nocalc	a logical. If TRUE, no learning procedure is called, see eg. rnetwork.
unitscale	a numeric passed to the plot method for network objects. Scale parameter for chopping off arrow heads.
xr	a numeric vector with two components containing the range on x-axis.
yr	a numeric vector with two components containing the range on y-axis.
...	additional plot arguments, passed to the plot method for network objects.

Details

To insert an arrow from node 'A' to node 'B', first click node 'A' and then click node 'B'. When the graph is finished, click 'stop'.

To specify that an arrow must not be present, press 'ban' (a toggle) and draw the arrow. This is shown as a red dashed arrow. It is possible to ban both directions between nodes. The ban list is stored with the network in the property `banlist`. It is a matrix with two columns. Each row is the 'from' node index and the 'to' node index, where the indices are the column number in the data frame.

Note that the network score changes as the network is re-learned whenever a change is made (unless `nocalc` is `TRUE`).

Value

A list with two elements that may be accessed using `getnetwork` and `gettrylist`. The elements are

`nw`	an object of class `network` with the final network.
`trylist`	an updated list used internally for reusing learning of nodes, see `maketrylist`.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

See Also

`network`

Examples

```
data(rats)
rats.nw    <- network(rats)
rats.prior <- jointprior(rats.nw,12)
rats.nw    <- getnetwork(learn(rats.nw,rats,rats.prior))

## Not run: newrat  <- getnetwork(drawnetwork(rats.nw,rats,rats.prior))
```

---

genlatex *From a network family, generate LaTeX output*

Description

The networks in a network family is arranged as pictex-graphs in a LaTeX-table.

Usage

```
genlatex(nwl,outdir="pic/",prefix="scoretable",picdir="",picpre="pic",
         ncol=5,nrow=7,width=12/ncol,vadjust=-1.8)
genpicfile (nwl,outdir="pic/",prefix="pic",w=1.6,h=1.6,bigscale=3)
```

Arguments

nwl	object of class networkfamily containing a list of objects of class network.
outdir	character string, the directory for storing output.
prefix	character string, the filename (without extension) of the LaTeX file. The filenames of the picfiles begin with the given prefix.
picdir	character string, the directory where pic-files are stored.
picpre	character string, prefix for pic-files.
ncol	integer, the number of columns in LaTeX table.
nrow	integer, the number of rows in LaTeX table.
width	numeric, the width of each cell in the LaTeX table.
vadjust	numeric, the vertical adjustment in LaTeX table.
w	numeric, the width of pictex objects
h	numeric, the height of pictex objects
bigscale	numeric, the scaling of the best network, which is output in 'nice.tex'

Value

Files:

{outdir}{picpre}xx.tex
: one pictex file for each network in the network family, indexed by xx.

{outdir}{prefix}.tex
: LaTeX file with table including all pictex files.

{outdir}{picpre}nice.tex
: pictex file with the best network.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

See Also

networkfamily, pictex

Examples

```
data(rats)
allrats <- getnetwork(networkfamily(rats,network(rats)))
allrats <- nwfsort(allrats)

## Not run: dir.create("c:/temp")
## Not run: genpicfile(allrats,outdir="c:/temp/pic/")
## Not run: genlatex(allrats,outdir="c:/temp/pic/",picdir="c:/temp/pic/")

## LATEX FILE:
#\documentclass{article}
#\usepackage{array,pictex}
#\begin{document}
#\input{scoretable}
#\input{picnice}
#\end{document}

#data(ksl)
#ksl.nw  <- network(ksl)
#ksl.prior <- jointprior(ksl.nw,64)
#mybanlist <- matrix(c(5,5,6,6,7,7,9,
#                    8,9,8,9,8,9,8),ncol=2)
#banlist(ksl.nw) <- mybanlist
#ksl.nw <- getnetwork(learn(ksl.nw,ksl,ksl.prior))
#ksl.search <- autosearch(ksl.nw,ksl,ksl.prior,
#                      trace=TRUE)
#ksl.searchlist <- makenw(ksl.search$table,ksl.search$nw)
#ksl.searchlist <- nwfsort(ksl.searchlist)
## Not run: genpicfile(ksl.searchlist)
## Not run: genlatex(ksl.searchlist)
```

`insert` *Insert/remove an arrow in network*

Description

Inserts/removes one arrow in a network (if legal)

Usage

```
insert (nw,j,i,df,prior,nocalc=FALSE,trylist=vector("list",size(nw)))
remover(nw,j,i,df,prior,nocalc=FALSE,trylist=vector("list",size(nw)))
```

Arguments

`nw`	an object of class `network`.
`j`	integer, giving the index of the 'from' node.
`i`	integer, giving the index of the 'to' node.

df	a data frame used for learning the network, see network.
prior	a list describing parameter priors, generated by jointprior.
nocalc	a logical. If TRUE, learning is not called.
trylist	a list, used internally for reusing learning of nodes, see maketrylist.

Details

Examines if the arrow from j to i is legal according to the following criteria

Arrows from/to the same node are not legal.

Arrows from continous nodes to discrete nodes are not legal.

Arrows banned in ban list are not legal, see drawnetwork.

Arrows already existing in the network are not legal.

If the arrow is not legal, a NULL network is returned. Otherwise, the arrow is inserted/removed, the network is re-learned (if nocalc is FALSE). The trylist is updated.

Value

A list with two elements

nw	an object of class network with the arrow added/removed if this is possible. If not, NULL is returned.
trylist	an updated list, used internally for reusing learning of nodes, see maketrylist.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

Examples

```
data(rats)
rats.nw    <- network(rats)
rats.nw    <- getnetwork(insert(rats.nw,2,1,nocalc=TRUE))
rats.prior <- jointprior(rats.nw,12)

rats.nw2   <- network(rats)
rats.nw2   <- getnetwork(learn(rats.nw2,rats,rats.prior))
rats.nw2   <- getnetwork(insert(rats.nw2,1,2,rats,rats.prior))

rats.nw3   <- getnetwork(remover(rats.nw2,1,2,rats,rats.prior))
```

jointprior *Calculates the joint prior distribution*

Description

Given a network with a prob property for each node, derives the joint probability distribution. Then the quantities needed in the local master procedure for finding the local parameter priors are deduced.

Usage

```
jointprior(nw,N=NA,phiprior="bottcher",timetrace=FALSE)
```

Arguments

nw	an object of class network. Each node must have a prob property to describe the local probability distribution. The prob property is created using prob method for network objects, which is called by the network function.
N	an integer, which gives the size of the imaginary data base. If this is too small, NA's may be created in the output, resulting in errors in learn. If no N is given, the procedure tries to set a value as low as possible.
phiprior	a string, which specifies how the prior for phi is calculated. Either phiprior="bottcher" or phiprior="heckerman" can be used.
timetrace	a logical. If TRUE, prints some timing information on the screen.

Details

For the discrete part of the network, the joint probability distribution is calculated by multiplying together the local probability distributions. Then, jointalpha is determined by multiplying each entry in the joint probability distribution by the size of the imaginary data base N.

For the mixed part of the network, for each configuration of the discrete variables, the joint Gaussian distribution of the continuous variables is constructed and represented by jointmu (one row for each configuration of the discrete parents) and jointsigma (a list of matrices – one for each configuration of the discrete parents). The configurations of the discrete parents are ordered according to findex. The algorithm for constructing the joint distribution of the continuous variables is described in Shachter and Kenley (1989).

Then, jointalpha, jointnu, jointrho, mu and jointphi are deduced. These quantities are later used for deriving local parameter priors.

For each configuration i of the discrete variables,

$$\nu_i = \rho_i = \alpha_i$$

and

$$\phi_i = (\nu_i - 1)\Sigma_i$$

if phiprior="bottcher", see Bottcher(2001) and

$$\phi_i = \nu_i(\rho_i - 2)\Sigma_i/(\nu_i + 1)$$

if phiprior="heckerman", see Heckerman, Geiger and Chickering (1995).

Value

A list with the following elements,

jointalpha	a table used in the local master procedure for discrete variables.
jointnu	a table used in the local master procedure for continuous variables.
jointrho	a table used in the local master procedure for continuous variables.
jointmu	a numeric matrix used in the local master procedure for continuous variables.
jointsigma	a list of numeric matrices (not used in further calculations).
jointphi	a list of numeric matrices used in the local master procedure for continuous variables.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

References

Bottcher, S.G. (2001). Learning Bayesian Networks with Mixed Variables, Artificial Intelligence and Statistics 2001, Morgan Kaufmann, San Francisco, CA, USA, 149-156.

Heckerman, D., Geiger, D. and Chickering, D. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, 20: 197-243.

Shachter, R.D. and Kenley, C.R. (1989), Gaussian influence diagrams. Management Science, 35:527-550.

See Also

network, prob

Examples

```
data(rats)
rats.nw    <- network(rats)
rats.prior <- jointprior(rats.nw,12)

## Not run: savenet(rats.nw,file("rats.net"))
## Not run: rats.nw <- readnet(file("rats.net"))
## Not run: rats.nw <- prob(rats.nw,rats)
## Not run: rats.prior <- jointprior(rats.nw,12)
```

ksl *Health and social characteristics*

Description

Data from a study measuring health and social characteristics of representative samples of Danish 70 year olds, taken in 1967 and 1984.

Format

A data frame with variables of both discrete and continuous types.

FEV Forced ejection volume

Kol Cholesterol

Hyp Hypertension (no/yes)

logBMI Logarithm of Body Mass Index

Smok Smoking (no/yes)

Alc Alcohol consumption (seldom/frequently)

Work Working (yes/no)

Sex male/female

Year Survey year (1967/1984)

learn *Estimation of parameters in the local probability distributions*

Description

Updates the distributions of the parameters in the network, based on a prior network and data. Also, the network score is calculated.

Usage

```
learn (nw, df, prior=jointprior(nw),
               nodelist=1:size(nw),
               trylist=vector("list",size(nw)),
               timetrace=FALSE)
```

Arguments

nw	an object of class network.
df	a data frame used for learning the network, see network.
prior	a list containing parameter priors, generated by jointprior.
nodelist	a numeric vector of indices of nodes to be learned.
trylist	a list used internally for reusing learning of nodes, see maketrylist.
timetrace	a logical. If TRUE, prints some timing information on the screen.

Details

The procedure learn determines the master prior, local parameter priors and local parameter posteriors, see Bottcher (2001). It may be called on all nodes (default) or just a single node.

From the joint prior distribution, the marginal distribution of all parameters in the family consisting of the node and its parents can be determined. This is the master prior, see localmaster.

The local parameter priors are now determined by conditioning in the master prior distribution, see conditional. The hyperparameters associated with the local parameter prior distribution is attached to each node in the property condprior.

Finally, the local parameter posterior distributions are calculated (see post) and attached to each node in the property condposterior.

A so-called trylist is maintained to speedup the learning process. The trylist consists of a list of matrices for each node. The matrix for a given node holds previously evaluated parent configurations and the corresponding log-likelihood contribution. If a node with a certain parent configuration needs to be learned, it is checked, whether the node has already been learned. The previously learned nodes are given as input in the trylist parameter and is updated in the learning procedure.

When one or more nodes in a network have been learned, the network score is updated and attached to the network in the property score.

The learning procedure is called from various functions using the principle, that networks should always be updated with their score. Thus, e.g.\ drawnetwork keeps the network updated when the graph is altered.

Value

A list with two elements that may be accessed using getnetwork and gettrylist. The elements are

nw	an object of class network, with the condposterior properties updated for the nodes. Also, the property score is updated and contains the network score. The contribution to the network score for each node is contained in the property loglik for each node.
trylist	an updated list used internally for reusing learning of nodes, see maketrylist.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

References

Bottcher, S.G. (2001). Learning Bayesian Networks with Mixed Variables, Artificial Intelligence and Statistics 2001, Morgan Kaufmann, San Francisco, CA, USA, 149-156.

See Also

networkfamily, jointprior, maketrylist, network

Examples

```
data(rats)
fit       <- network(rats)
fit.prior <- jointprior(fit,12)
fit.learn <- learn(fit,rats,fit.prior,timetrace=TRUE)
fit.nw    <- getnetwork(fit.learn)
fit.learn2<- learn(fit,rats,fit.prior,trylist=gettrylist(fit.learn),timetrace=TRUE)
```

`makesimprob` *Make a suggestion for simulation probabilities*

Description

Creates local probability distributions reflecting the graph of the network. These are attached as a `simprob` property to each node in the network and can be edited and used for `rnetwork`.

Usage

```
makesimprob(nw,
                          s2=function(idx,cf) {
                            cf <- as.vector(cf)
                            xs <- (1:length(cf))
                            log(xs%*%cf+1)
                          },
                          m0=function(idx,cf) {
                            cf <- as.vector(cf)
                            xs <- (1:length(cf))^2
                            .69*(xs%*%cf)
                            },
                          m1=function(idx,cf) {
                            cf <- as.vector(cf)
                            xs <- (1:length(cf))*10
                            idx*(cf%*%xs)
                            })
```

Arguments

`nw`	an object of class `network`.
`s2`	function that returns the variance as a function of the node index and the configuration of the discrete variables.
`m0`	function that returns the intercept as a function of the node index and the configuration of the discrete variables.
`m1`	function that returns the regression coefficients as a function of the node index and the configuration of the discrete variables.

Details

For each node, the local simprob is determined. If the node is discrete, the probability distribution is uniform (and thus not reflecting the dependence in the graph, as it should). If the node is continuous, one mean and variance is attached per configuration of the discrete parents. The mean depends on the continuos parents and is the regression coefficients determined by the functions m0 (intercept) and m1 (regression coefficients). The variance is determined by the function s2.

Value

The network object nw, where each node has attached the property simprob.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

See Also

rnetwork

---

maketrylist *Creates the full trylist*

Description

For faster learning, a trylist is maintained as a lookup table for a given parent configuration of a node.

Usage

```
maketrylist(initnw,data,prior=jointprior(network(data)),timetrace=FALSE)
```

Arguments

initnw	an object of class network, from which the search is started.
data	a data frame used for learning the network, see network.
prior	a list containing parameter priors, generated by jointprior.
timetrace	a logical. If TRUE, prints some timing information on the screen.

Details

This procedure is included for illustrative purposes. For each node in the network, all possible parent configurations are created and learned. The result is called a trylist. To create the full trylist is very time-consuming, and a better choice is to maintain a trylist while searching and indeed this is automatically done. The trylist is given as output to all functions that call the learning procedure and can be given as an argument.

Value

A list with one element per node in the network. In the list, element *i* is a matrix with two columns: a string with the indices of the parent nodes, separated by ":", and a numeric with the log-likelihood contribution of the node given the parent configuration. Whenever learning is performed of a node given a parent configuration, the trylist is consulted to yield faster learning, especially useful when using autosearch or heuristic.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

See Also

networkfamily, autosearch heuristic

Examples

```
data(rats)
rats.nw <- network(rats)
rats.pr <- jointprior(rats.nw,12)
rats.nw <- getnetwork(learn(rats.nw,rats,rats.pr))
rats.tr <- maketrylist(rats.nw,rats,rats.pr)

rats.hi <- getnetwork(heuristic(rats.nw,rats,rats.pr,trylist=rats.tr))
```

---

network *Bayesian network data structure*

Description

A Bayesian network is represented as an object of class network. Methods for printing and plotting are defined.

Usage

```
network(df,specifygraph=FALSE,inspectprob=FALSE,
        doprob=TRUE,yr=c(0,350),xr=yr)
## S3 method for class 'network'
print(x,filename=NA,condposterior=FALSE,
                          condprior=FALSE,...)
## S3 method for class 'network'
 plot(x,arrowlength=.25,
                        notext=FALSE,
                        sscale=7,showban=TRUE,yr=c(0,350),xr=yr,
                        unitscale=20,cexscale=8,...)
```

Arguments

df	a data frame, where the columns define the variables. A continuous variable should have type numeric and discrete varibles should have type factor.
specifygraph	a logical. If TRUE, provides a call to drawnetwork to interactively specify a directed acyclic graph and possibly a ban list (see below).
inspectprob	a logical. If TRUE, provides a plot of the graph and possibility to inspect the calculated probability distribution by clicking on the nodes.
doprob	a logical. If TRUE, do not calculate a probability distribution. Used for example in rnetwork.
x	an object of class network.
filename	a string or NA. If not NA, output is printed to a file.
condprior	a logical. If TRUE, the conditional prior is printed, see conditional.
condposterior	a logical. If TRUE, the conditional posterior is printed, see learn.
sscale	a numeric. The nodes are initially placed on a circle with radius sscale.
unitscale	a numeric. Scale parameter for chopping off arrow heads.
cexscale	a numeric. Scale parameter to set the size of the nodes.
arrowlength	a numeric containing the length of the arrow heads.
xr	a numeric vector with two components containing the range on x-axis.
yr	a numeric vector with two components containing the range on y-axis.
notext	a logical. If TRUE, no text is displayed in the nodes on the plot.
showban	a logical. If TRUE, banned arrows are shown in red.
...	additional plot arguments, passed to plot.node.

Value

The netork creator function returns an object of class network, which is a list with the following elements (properties),

nodes	a list of objects of class node. If doprob is TRUE, the nodes are given the property prob which is the initial probability distribution used by jointprior.
n	an integer containing the number of nodes in the network.
discrete	a numeric vector of indices of discrete nodes.
continuous	a numeric vector of indices of continuous nodes.
banlist	a numeric matrix with two columns. Each row contains the indices i -> j of arrows that may not be allowed in the directed acyclic graph.
score	a numeric added by learn and is the log network score.
relscore	a numeric added by nwfsort and is the relative network score – compared with the best network in a network family.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

See Also

networkfamily, node, rnetwork, learn, drawnetwork, jointprior, heuristic, nwequal

Examples

```
A <- factor(rep(c("A1","A2"),50))
B <- factor(rep(rep(c("B1","B2"),25),2))
thisnet <- network( data.frame(A,B) )

set.seed(109)
sex     <- gl(2,4,label=c("male","female"))
age     <- gl(2,2,8)
yield   <- rnorm(length(sex))
weight  <- rnorm(length(sex))
mydata  <- data.frame(sex,age,yield,weight)
mynw    <- network(mydata)

# adjust prior probability distribution
localprob(mynw,"sex")   <- c(0.4,0.6)
localprob(mynw,"age")   <- c(0.6,0.4)
localprob(mynw,"yield") <- c(2,0)
localprob(mynw,"weight")<- c(1,0)

print(mynw)
plot(mynw)

prior <- jointprior(mynw)
mynw  <- getnetwork(learn(mynw,mydata,prior))
thebest <- getnetwork(autosearch(mynw,mydata,prior))

print(mynw,condposterior=TRUE)

## Not run: savenet(mynw,file("yield.net"))
```

Network tools *Tools for manipulating networks*

Description

Various extraction/replacement functions for networks

Usage

```
modelstring(x)
makenw(tb,template)
as.network(nwstring,template)
size(x)
banlist(x)
banlist(x) <- value
```

```
getnetwork(x)
gettrylist(x)
```

Arguments

x	an object of class network.
tb	a table output from autosearch or heuristic in the list property table. Can be translated into a networkfamily.
template	an object of class network with the same nodes as the networks described in the table tb.
nwstring	a string representing the network.
value	a numeric matrix with two columns. Each row contains the indices i -> j of arrows that may not be allowed in the directed acyclic graph.

Details

The string representation of a network is a minimal size representation to speed up calculations. The functions modelstring, as.network and makenw converts between the string represention and network objects.

size extracts the number of nodes in a network object.

banlist extracts the banlist from a network object.

getnetwork and gettrylist are accessor function that extracts a network object or trylist from the result from autosearch, heuristic, learn, perturb, networkfamily, drawnetwork.

---

networkfamily *Generates and learns all networks for a set of variables.*

Description

Method for generating and learning all networks that are possible for a given set of variables. These may be plotted or printed. Also, functions for sorting according to the network score (see nwfsort) and for making a network family unique (see the unique method for networkfamily objects) are available.

Usage

```
networkfamily(data,nw=network(data), prior=jointprior(nw),
              trylist=vector("list",size(nw)), timetrace=TRUE)

## S3 method for class 'networkfamily'
print(x,...)
## S3 method for class 'networkfamily'
plot(x,layout=,
        cexscale=5,arrowlength=0.1,sscale=7,...)
```

Arguments

nw	an object of class network. This should be the empty network for the set of variables.
data	a data frame used for learning the network, see network.
prior	a list containing parameter priors, generated by jointprior.
trylist	a list used internally for reusing learning of nodes, see maketrylist.
timetrace	a logical. If TRUE, prints some timing information on the screen.
x	an object of class networkfamily.
layout	a numeric two dimensional vector with the number of plots in the rows and columns of each plotting page. Default set to rep(min(1+floor(sqrt(length(x))),5),2).
cexscale	a numeric. A scaling parameter to set the size of the nodes.
arrowlength	a numeric, which gives the length of the arrow heads.
sscale	a numeric. The nodes are initially placed on a circle with radius sscale.
...	additional plot arguments passed to the plot method for network objects.

Details

networkfamily generates and learns all possible networks with the nodes given as in the initial network nw. This is done by successively trying to generate the networks with all possible arrows to/from each node (see addarrows). If there is a ban list present in nw (see network), then this is respected, as are the restrictions described in insert.

After generation of all possible networks, a test for cycles (see cycletest) is performed and only networks with directed acyclic graphs are returned.

Value

The function networkfamily returns a list with two components,

nw	an object of class networkfamily.
trylist	an updated list used internally for reusing learning of nodes, see maketrylist.

Note

Generating all possible networks can be *very* time consuming!

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

See Also

network, genlatex, heuristic, nwfsort, unique.networkfamily, elementin, addarrows, cycletest

Examples

```
data(rats)
allrats <- getnetwork(networkfamily(rats))
plot(allrats)
print(allrats)
```

---

node *Representation of nodes*

Description

An important part of a network is the list of nodes. The nodes summarize the local properties of a node, given the parents of the node.

Usage

```
node (idx,parents,type="discrete",name=paste(idx),
                levels=2,levelnames=paste(1:levels),position=c(0,0))
## S3 method for class 'node'
 print(x,filename=NA,condposterior=TRUE,condprior=TRUE,...)
## S3 method for class 'node'
 plot(x,cexscale=10,notext=FALSE,...)
nodes(nw)
nodes(nw) <- value
```

Arguments

x	an object of class node.
parents	a numeric vector with indices of the parents of the node.
idx	an integer, which gives the index of the node (the column number of the corresponding data frame).
type	a string, which gives the type of the node. Either "discrete" (for factors) or "continuous" (for numeric).
name	a string, which gives the name used when plotting and printing. Defaults to the column name in the data frame.
levels	an integer. If type is "discrete", this is the number of levels for the discrete variable.
levelnames	if type is "discrete", this is a vector of strings (same length as levels) with the names of the levels. If type is "continuous", the argument is ignored.
position	a numeric vector with coordinates where the node should appear in the plot. Usually set by network and drawnetwork.
nw	an object of class network.
value	a list of elements of class node.
filename	a string or NA. If not NA, output is printed to a file.

condprior	a logical. If TRUE, the conditional prior is printed, see conditional.
condposterior	a logical. If TRUE, the conditional posterior is printed, see learn.
cexscale	a numeric. Scale parameter to set the size of the nodes.
notext	a logical. If TRUE, no text is displayed in the nodes on the plot.
...	additional plot arguments.

Details

The operations on a node are typically done when operating on a network, so these functions are not to be called directly.

When a network is created with network, the nodes in the nodelist are created using the node procedure.

Local probability distributions are added as the property prob to each node using prob.node. If the node is continuous, this is a numeric vector with the conditional variance and the conditional regression coefficients arising from a regression on the continuous parents, using data. If the node has discrete parents, prob is a matrix with a row for each configuration of the discrete parents. If the node is discrete, prob is a multiway array which gives the conditional probability distribution for each configuration of the discrete parents. The generated prob can be replaced to match the prior information available.

nodes gives the list of nodes of a network. localprob gives the probability distribution for each node in the network.

Value

The node creator function returns an object of class node, which is a list with the following elements (properties),

idx	an integer. A unique index for this node. It MUST correspond to the column index of the variable in the data frame.
name	a string. The printed name of the node.
type	a string. Either "continuous" or "discrete".
levels	an integer. If the node is of type "discrete", this integer is the number of levels of the node.
levelnames	if type is "discrete", this is a vector of strings (same length as levels) with the names of the levels. If type is "continuous", the node does not have this property.
parents	a vector of indices of the parents to this node. It is best to manage this vector using the insert function.
prob	a numeric vector, matrix or multiway array, giving the initial probability distribution. If the node is discrete, prob is a multiway array. If the node is continuous, prob is a matrix with one row for each configuration of the discrete parents, reducing to a vector if the node has no discrete parents.
condprior	a list, generated by conditional giving the parameter priors deduced from jointprior using the master prior procedure (see localmaster).
condposterior	a list, which gives the parameter posteriors obtained from learnnode.

loglik	a numeric giving the log likelihood contribution for this node, calculated in learnnode.
simprob	a numeric vector, matrix or multiway array similar to prob, added by makesimprob and used by rnetwork.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

---

numbermixed *The number of possible networks*

Description

Calculates the number of different directed acyclic graphs for a set of discrete and continuous nodes.

Usage

```
numbermixed(nd,nc)
```

Arguments

nd	an integer, which gives the number of discrete nodes.
nc	an integer, which gives the number of continuous nodes.

Details

No arrows are allowed from continuous nodes to discrete nodes. Cycles are not allowed. The number of networks is given by Bottcher (2003), using the result in Robinson (1977).

When nd+nc>15, the procedure is quite slow.

Value

A numeric containing the number of directed acyclic graphs with the given node configuration.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

References

Bottcher, S.G. (2003). Learning Conditional Gaussian Networks. Aalborg University, 2003.

Robinson, R.W. (1977). Counting unlabeled acyclic digraphs, Lecture Notes in Mathematics, 622: Combinatorial Mathematics.

Examples

```
numbermixed(2,2)
## Not run: numbermixed(5,10)
```

`nwfsort` *Sorts a list of networks*

Description

According to the `score` property of the networks in a network family, the networks are sorted and the relative score, i.e.\ the score of a network relative to the highest score, is attached to each network as the `relscore` property.

Usage

```
nwfsort(nwf)
```

Arguments

`nwf`	an object of class `networkfamily`.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

`perturb` *Perturbs a network*

Description

Randomly insert/delete/turn arrows to obtain another network.

Usage

```
perturb(nw,data,prior,degree=size(nw),trylist=vector("list",size(nw)),
        nocalc=FALSE,timetrace=TRUE)
```

Arguments

`nw`	an object of class `network`, from which arrows are added/removed/turned.
`data`	a data frame used for learning the network, see `network`.
`prior`	a list containing parameter priors, generated by `jointprior`.
`degree`	an integer, which gives the number of attempts to randomly insert/remove/turn an arrow.
`trylist`	a list used internally for reusing learning of nodes, see `maketrylist`.
`nocalc`	a logical. If `TRUE` no learning procedure is called, see eg. `rnetwork`.
`timetrace`	a logical. If `TRUE`, prints some timing information on the screen.

Details

Given the initial network, a new network is constructed by randomly choosing an action: remove, turn, add. After the action is chosen, we choose randomly among all possibilities of that action. If there are no possibilites, the unchanged network is returned.

Value

A list with two elements that may be accessed using `getnetwork` and `gettrylist`. The elements are

`nw`	an object of class `network` with the generated network.
`trylist`	an updated list used internally for reusing learning of nodes, see `maketrylist`.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

Examples

```
set.seed(200)
data(rats)
fit       <- network(rats)
fit.prior <- jointprior(fit)
fit       <- getnetwork(learn(fit,rats,fit.prior))
fit.new   <- getnetwork(perturb(fit,rats,fit.prior,degree=10))

data(ksl)
ksl.nw    <- network(ksl)
ksl.rand  <- getnetwork(perturb(ksl.nw,nocalc=TRUE,degree=10))
plot(ksl.rand)
```

---

prob *Local probability distributions*

Description

Methods for accessing or changing the local probability distributions and for accessing the local prior and posterior distributions

Usage

```
prob(x,df,...)

## S3 method for class 'node'
 prob(x,df,nw,...)
## S3 method for class 'network'
 prob(x,df,...)
```

```
localprob(nw)
localprob(nw,name) <- value

localprior(node)
localposterior(node)
```

Arguments

x	an object of class node or network.
df	a data frame, where the columns define the variables. A continuous variable should have type numeric and discrete varibles should have type factor.
nw	an object of class network.
node	an object of class node.
name	a string, which gives the node name.
...	additional arguments for specific methods.
value	If the node is continuous, this is a numeric vector with the conditional variance and the conditional regression coefficients arising from a regression on the continuous parents, using data. If the node has discrete parents, it is a matrix with a row for each configuration of the discrete parents. If the node is discrete, it is a multiway array which gives the conditional probability distribution for each configuration of the discrete parents.

Details

The prob methods add local probability distributions to each node. If the node is continuous, this is a numeric vector with the conditional variance and the conditional regression coefficients arising from a regression on the continuous parents, using data. If the node has discrete parents, prob is a matrix with a row for each configuration of the discrete parents. If the node is discrete, prob is a multiway array which gives the conditional probability distribution for each configuration of the discrete parents. The generated prob can be replaced to match the prior information available.

localprob returns the probability distribution for each node in the network.

In a learned network, the local prior and posterior can be accessed for each node using localprior and localposterior.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

rats *Weightloss of rats*

Description

An artificial data set. 24 rats (12 female, 12 male) have been randomized to use one of three drugs (products for loosing weight). The weightloss for each rat is noted after one and two weeks.

Format

A data frame with 4 variables.

Sex a factor with two levels: "M" (male), "F" (female)

Drug a factor with three levels: "D1", "D2", "D3" (three types)

W1 a numeric: weightloss, week one.

W2 a numeric: weightloss, week 2.

References

Morrison, D.F. (1976). Multivariate Statistical Methods. McGraw-Hill, USA.

Edwards, D. (1995). Introduction to Graphical Modelling, Springer-Verlag. New York.

readnet *Reads/saves .net file*

Description

Reads/saves a Bayesian network specification in the `.net` language used by Hugin.

Usage

```
readnet(con=file("default.net"))
savenet(nw, con=file("default.net"))
```

Arguments

`con`	a connection.
`nw`	an object of class `network`.

Details

`readnet` reads only the structure of a network, i.e.\ the directed acyclic graph.

`savenet` exports the `prob` property for each node in the network object along with the network structure defined by the parents of each node.

Value

readnet creates an object of class network with the nodes specified as in the .net connection. The network has not been learned and the nodes do not have prob properties (see prob.network).

savenet writes the object to the connection.

Note

The call to readnet(savenet(network)) is *not* the identity function as information is thrown away in both savenet and readnet.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

See Also

network

Examples

```
data(rats)
nw <- network(rats)
## Not run: savenet(nw,file("default.net"))
## Not run: nw2 <- readnet(file("default.net"))
## Not run: nw2 <- prob(nw2,rats)
```

---

rnetwork *Simulation of data sets with a given dependency structure*

Description

Given a network with nodes having the simprob property, rnetwork simulates a data set.

Usage

```
rnetwork(nw, n=24, file="")
```

Arguments

nw	an object of class network, where each node has the property simprob (see makesimprob).
n	an integer, which gives the number of cases to simulate.
file	a string. If non-empty, the data set is stored there.

Details

The variables are simulated one at a time in an order that ensures that the parents of the node have already been simulated. For discrete variables a multinomial distribution is used and for continuous variables, a Gaussian distribution is used, according to the `simprob` property in each node.

Value

A data frame with one row per case. If a file name is given, a file is created with the data set.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

Examples

```
A  <- factor(NA,levels=paste("A",1:2,sep=""))
B  <- factor(NA,levels=paste("B",1:3,sep=""))
c1 <- NA
c2 <- NA
df <- data.frame(A,B,c1,c2)

nw <- network(df,doprob=FALSE) # doprob must be FALSE
nw <- makesimprob(nw)          # create simprob properties

set.seed(944)
sim <- rnetwork(nw,n=100)    # create simulated data frame
```

---

score *Network score*

Description

Accessor for the score from a node or network

Usage

```
score(x,...)

## S3 method for class 'node'
    score(x,...)
## S3 method for class 'network'
 score(x,...)
```

Arguments

`x`	an object of class `node` or `network`.
`...`	additional arguments for specific methods.

Value

For networks, the log network score is returned. For nodes, the contribution to the log network score is returned.

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

---

unique.networkfamily *Makes a network family unique.*

Description

Removes networks that are equal or equivalent to networks already in the network family.

Usage

```
## S3 method for class 'networkfamily'
unique(x,incomparables=FALSE,equi=FALSE,timetrace=FALSE,epsilon=1e-12,...)
```

Arguments

x	an object of class networkfamily.
incomparables	a logical, but has no effect.
equi	a logical. If TRUE, also equivalent networks are thrown out (*i.e.* if their score is within epsilon from another network).
timetrace	a logical. If TRUE, prints some timing information on the screen.
epsilon	a numeric, which measures how close network scores are allowed to be from each other to be 'equivalent'.
...	further arguments (no effect)

Author(s)

Susanne Gammelgaard Bottcher,
Claus Dethlefsen <rpackage.deal@gmail.com>.

Examples

```
data(rats)
rats.nwf <- networkfamily(rats)
rats.nwf2<- unique(getnetwork(rats.nwf),equi=TRUE)
```

Index