Package 'sdnet'

April 30, 2019

Title Soft-Discretization-Based Bayesian Network Inference

Version 2.4.1

Author Nikolay Balov

Description Fitting discrete Bayesian networks using soft-discretized data. Soft-discretization is based on mixture of normal distributions. Also implemented is a supervised Bayesian network learning employing Kullback-Leibler divergence. For more information see Balov (2013) <DOI:10.1186/1755-8794-6-S3-S1>.

Maintainer Nikolay Balov <nhbalov@gmail.com>

License GPL (>= 2)

Depends R (>= 3.0.2)

Imports methods, stats, utils, graphics

Suggests

Collate class.R def.R graph2catnet.R dags.R probs.R joint.prob.R marginal.prob.R samples.R loglik.R entropy.R categor.R dist.R plot.R find.R search.R eval.R predict.R chisq.R histo.R cluster.R bif.R xdsl.R quant.R expres.r zzz.R

LazyLoad yes

Repository CRAN

Date/Publication 2019-04-30 07:10:09 UTC

URL http://www.biomedcentral.com/1755-8794/6/S3/S1

NeedsCompilation yes

R topics documented:

sdnet-package *sdnet: Soft Discretization-based Bayesian Network Inference*

Description

sdnet Fitting discrete Bayesian networks using marginal mixture distributions. A Bayesian network is defined by a graphical structure in form of directed acyclic graph and a probability model given as a set of conditional distributions, one for each node in the network. Considered in the package are only categorical Bayesian networks - networks which nodes represent discrete random variables. The searching functions implemented in sdnet output sets of networks with increasing complexity that fit the data according to the MLE criterion. These optimal networks is believed to explain and represent the relations between the node-variables. The final network selection is left to the user.

Before starting to use the package, we suggest the user to take a look at some of the main objects used in sdnet such as catNetwork and catNetworkEvaluate and then familiarize with the main search functions cnSearchOrder and cnSearchSA. More details and examples can be found in the manual pages and the vignettes accompanying the package.

Since sdnet does not have its own plotting abilities, the user needs to setup some external tools in order to visualize networks, or more precisely, catNetwork objects. There are two options: first one is to use igraph package and second, and better one, is to use Graphviz library. Graphviz is not a R-package but a platform independent library that the user have to install in advance on its machine in order to use this option.

If the user choose the first option, igraph, the only thing he/she has to do is to install the library in R and set the environment variable R_SDNET_USE_IGRAPH to TRUE. A convenient place to do this is in the R .First function

.First <- function() {

.....

Sys.setenv(R_SDNET_USE_IGRAPH=TRUE)

}

In order to use Graphviz, in addition to installing the library, the user has to register a environmental variable with name R_DOTVIEWER with the path to the Dot executable file of Graphviz. The Dot routine generates a postscript or pdf-file from a text dot-file. Also, the user needs a postscript and pdf-viewer. The full path to it has to be given in another variable with name R_PDFVIEWER. Note that R_PDFVIEWER variable might be already setup. To check this call Sys.getenv("R_PDFVIEWER") in R.

The variables R_DOTVIEWER and eventually R_PDFVIEWER can be registered in the .First function residing in the .Rprofile initializing file.

Below we give two examples. On UNIX platform the user may use code like this one

.First <- function() {

.....

Sys.setenv(R_DOTVIEWER="/usr/bin/dot")

}

On Windows platform the user may have the following two lines in its .First function

.First <- function() {

.....

Sys.setenv(R_PDFVIEWER="\"C:/Program Files (x86)/Adobe/Reader 9.0/Reader/AcroRd32\"")

Sys.setenv(R_DOTVIEWER="\"C:/Program Files (x86)/Graphviz 2.26.3/bin/Dot\"")

}

Note that all paths in Windows should be embraced by comment marks, "\"".

Author(s)

N. Balov

`catNetwork-class` *Class "catNetwork"*

Description

This is the base class in the `catnet` package for representing Bayesian networks with categorical values. It stores both the graph and probability structure of categorical Bayesian networks. Technically, `catNetwork` is a S4 type of R-class implemented in object-oriented style, with slots representing object components and members for accessing and manipulating class objects. Below we list the slots of `catNetwork` and some of its main members along with the functions for creating `catNetwork` objects.

Details

The `catNetwork` class provides a comprehensive general structure for representing discrete Bayesian networks by describing both the graph and probability structures. Although available for direct access, the class components, its slots, should not be manipulated directly but using the class members instead. A `catNetwork` object integrity can always be checked by calling `is(object, "catNetwork")`.

Objects from the Class

Objects can be created by calls of

`cnNew(nodes, cats, pars, probs)`

`cnRandomCatnet(numnodes, maxpars, numCategories)`

`cnCatnetFromEdges(nodes, edges, numCategories)`

`cnCatnetFromSif(file)` `cnCatnetFromBif(file)` `cnCatnetFromXdsl(file)`

Slots

`objectName` an optional object name of class `character`.

`numnodes`: an integer, the number of nodes in the object.

`nodes`: a `vector` specifying the node names.

`pars`: a `list` specifying the node pars. The list `pars` must be the same length as `nodes`. Parents are kept as indices in the `nodes` vector.

`cats`: a `list` of characters specifying a set of categorical values for each node.

`probs`: a numerical `list` that for each node specifies a discrete probability distribution - the distribution of the node conditional on its parent set. The elements of `probs` are lists themselves. See `cnProb` function for more details.

`maxpars`: an integer, the maximum number of node pars.

`maxcats`: an integer, the maximum number of node cats.

`meta`: an object of class `character` storing some meta-data information.

`nodecomplx`: a numerical `vector`, the node complexities.

`nodelik`: a numerical `vector`, the node likelihoods of the sample being used for estimation.

`complx`: an `integer`, the network complx

`loglik`: a `numerical`, the total loglik of the sample being used for estimation

`nodeSampleSizes`: a numerical `vector`, if the object is an estimate, the node sample sizes.

Methods

cnNew `signature(nodes="vector", cats="list", pars="list", probs="list")`: Creating a new class object.

cnRandomCatnet `signature(numnodes="integer", maxpars="integer", numCategories="integer")`: Creating a random class object.

cnCatnetFromEdges `signature(nodes="vector", edges="list", numCategories="integer")`: Deriving a class object from a list of edges.

cnCatnetFromSif `signature(file="character")`: Creating a class object from a file.

cnNumNodes `signature(object="catNetwork")`:

cnNodes `signature(object="catNetwork", which="vector")`:...

cnSubNetwork `signature(object="catNetwork", nodeIndices="vector", indirectEdges="logical")`:...

cnReorderNodes `signature(object="catNetwork", nodeIndices="vector")`:...

cnParents `signature(object="catNetwork", which="vector")`:...

cnMatParents `signature(object="catNetwork", nodeorder="vector")`:...

cnEdges `signature(object="catNetwork", which="vector")`:...

cnMatEdges `signature(object="catNetwork")`:...

cnProb `signature(object="catNetwork")`:...

cnSetProb `signature(object="catNetwork", psamples="matrix")`:...

cnPlot `signature(object="catNetwork")`:...

cnDot signature(object="catNetwork", file="character"):...

cnSamples signature(object="catNetwork", nsamples="integer"):...

cnSamplesPert signature(object="catNetwork", nsamples="integer", perturbations="matrix"):...

cnOrder signature(object="catNetwork"):...

cnLoglik signature(object="catNetwork", psamples="matrix"):...

cnComplexity signature(object="catNetwork"):...

cnEvaluate signature(object="catNetwork", psamples="matrix", perturbations="matrix", maxComplexity="integer"):...

cnPredict signature(object="catNetwork", psamples="matrix"):...

cnCompare signature(object1="catNetwork", object2="catNetwork"):...

Author(s)

N. Balov

See Also

cnRandomCatnet, cnCatnetFromEdges, cnNew, cnNodes, cnEdges, cnComplexity, cnPlot

Examples

```
set.seed(123)
cnet <- cnRandomCatnet(numnodes=10, maxpars=2, numcats=2)
cnet
```

---

catNetworkDistance-class

Class "catNetworkDistance"

Description

This class contains a list of catNetworks and it is the output format of cnEvaluate function

Details

See in the manual of cnCompare function for description of different distance criteria.

Slots

hamm: an integer, the hamming distance between the parent matrices of the found networks and the original network.

hammexp: an integer, the hamming distance between the exponents of the parent matrices.

tp: an integer, the number of true positives directed edges.

fp: an integer, the number of false positives directed edges.

`fn`: an `integer`, the number of false negatives directed edges.

`pr`: a `numeric`, precision.

`sp`: a `numeric`, specificity.

`sn`: a `numeric`, sensitivity(recall).

`fscore`: a `numeric`, the F-score.

`skel.tp`: an `integer`, the number of true positives undirected edges.

`skel.fp`: an `integer`, the number of false positives undirected edges.

`skel.fn`: an `integer`, the number of false negatives undirected edges.

`order.fp`: an `integer`, the number of false positive order relations.

`order.fn`: an `integer`, the number of false negative order relations.

`markov.fp`: an `integer`, the number of false positive Markov pairs.

`markov.fn`: an `integer`, the number of false negative Markov pairs.

`KLdist`: a `numerical`, the KL distance, currently inactive.

Methods

cnPlot `signature(object="catNetworkDistance")`: Draw some distance plots.

Author(s)

N. Balov

See Also

`catNetwork-class`, `catNetworkEvaluate-class`, `cnCompare`, `cnPlot`

---

catNetworkEvaluate-class

Classes "catNetworkEvaluate" and "dagEvaluate"

Description

This class contains a list of `catNetworks` together with some diagnostic metrics and information. `catNetworkEvaluate` objects are created automatically as result of calling `cnEvaluate` or one of the `cnSearch` functions.

Details

The class catNetworkEvaluate is used to output the result of two functions: cnEvaluate and cnSearchSA. The usage of it in the first case is explained next. The complexity and log-likelihood of the networks listed in nets slots are stored in complexity and loglik slots. Function cnEvaluate and cnCompare fills all the slots from hamm to markov.fn by comparing these networks with a given network. See in the manual of cnCompare function for description of different distance criteria. By calling cnPlot upon a catNetworkEvaluate object, some relevant comparison information can be plotted.

When catNetworkEvaluate is created by calling cnSearchSA or cnSearchSAcluster functions, complexity and loglik contains the information not about the networks in the nets list, but about the optimal networks found during the stochastic search process. Also, the slots from hamm to markov.fn are not used.

Slots

numnodes: an integer, the number of nodes in the network.

numsamples: an integer, the sample size used for evaluation.

nets: a list of resultant networks.

complexity an integer vector, the network complexity.

loglik a numerical vector, the likelihood of the sample being evaluated.

hamm: an integer vector, the hamming distance between the parent matrices of the found networks and the original network.

hammexp: an integer vector, the hamming distance between the exponents of the parent matrices.

tp: an integer vector, the number of true positives directed edges.

fp: an integer vector, the number of false positives directed edges.

fn: an integer vector, the number of false negatives directed edges.

pr: a numeric vector, precision.

sp: a numeric vector, specificity.

sn: a numeric vector, sensitivity(recall).

fscore: a numeric vector, the F-score.

skel.tp: an integer vector, the number of true positives undirected edges.

skel.fp: an integer vector, the number of false positives undirected edges.

skel.fn: an integer vector, the number of false negatives undirected edges.

order.fp: an integer vector, the number of false positive order relations.

order.fn: an integer vector, the number of false negative order relations.

markov.fp: an integer vector, the number of false positive Markov pairs.

markov.fn: an integer vector, the number of false negative Markov pairs.

KLdist: a numerical vector, the KL distance, currently inactive.

time: a numerical, the processing time in seconds.

Methods

cnFind `signature(object="catNetworkEvaluate", complexity="integer")`: Finds a network in the list `nets` with specific complexity.

cnFindAIC `signature(object="catNetworkEvaluate")`: Finds the optimal network according to AIC criterion.

cnFindBIC `signature(object="catNetworkEvaluate")`: Finds the optimal network according to BIC criterion.

cnPlot `signature(object="catNetworkEvaluate")`: Draw distance plots.

Author(s)

N. Balov

See Also

`catNetwork-class`, `catNetworkDistance-class`, `cnCompare`, `cnPlot`

---

cnCatnetFromEdges *catNetwork from Edges*

Description

Creates a `catNetwork` object from list of nodes and edges.

Usage

```
cnCatnetFromEdges(nodes, edges, numcats=2)
```

Arguments

`nodes`	a `vector` of node names
`edges`	a `list` of node edges
`numcats`	an `integer`, the number of categories per node

Details

The function uses a list of nodes and directional edges to create a `catNetwork` with specified (fixed) number of node categories. A random probability model is assigned, which can be changed later by `cnSetProb` for example. Note that `cnSetProb` takes a given data sample and changes both the node categories and their conditional probabilities according to it.

Value

A `catNetwork` object

Author(s)

N. Balov

See Also

cnNew, cnCatnetFromSif, cnSetProb

---

cnCatnetFromSif *Categorical Network from Simple Interaction File (SIF) and Bayesian Networks Interchange Format (BIF)*

Description

Creates a catNetwork object from a SIF/BIF file.

Usage

```
cnCatnetFromSif(file, numcats=2)
cnCatnetFromBif(file)
cnCatnetFromXdsl(file)
```

Arguments

file	a file name
numcats	an integer, the number of node categories

Details

The function imports a graph structure from a SIF file by assigning equal number numcats of categories for each of its nodes and a random probability model. Subsequently, the probability model can be changed by calling cnSetProb function.

Value

A catNetwork object

Author(s)

N. Balov

See Also

cnNew, cnCatnetFromEdges, cnSetProb

cnCluster-method *Network Clustering*

Description

Retrieving the clusters, the connected sub-networks, of a given network. Estimating the clusters from data.

Usage

```
cnCluster(object)
cnClusterSep(object, data, pert=NULL)
cnClusterMI(data, pert=NULL, threshold=0)
```

Arguments

object	a catNetwork
data	a matrix in row-nodes format or a data.frame in column-nodes format
pert	a binary perturbation matrix with the dimensions of data
threshold	a numeric value

Details

The function cnCluster constructs a list of subsets of nodes of the object, each representing a connected sub-network. Isolated nodes, these are nodes not connected to any other, are not reported. Thus, every element of the output list contains at least two nodes. The function cnClusterMI clusters the nodes of the data using the pairwise mutual information and critical value threshold.

Value

A list of named nodes.

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=30, maxpars=2, numcats=2)
cnCluster(object=cnet)
```

cnCompare-method *Network Comparison*

Description

Compares two catNetwork objects by several criteria

Usage

```
cnCompare(object1, object2, extended = FALSE)
```

Arguments

object1	a catNetwork object
object2	a catNetwork object, matrix, list of catNetworks or catNetworkEvaluate object
extended	a logical parameter, specifying whether basic but quicker or extended comparison to be performed

Details

Comparison can be performed only between networks with the same sets of nodes. The function considers several topology-related comparison metrics.

First, directed edge comparison is performed and the true positives (TP), the false positive (FP) and the false negatives (FN) are reported assuming object1 to be the 'true' network.

Second, the difference between the binary parent matrices of the two objects is measured as the number of positions at which they differ. This is the so called Hamming distance and it is coded as hamm. Also, when extended parameter is set to TRUE, the difference between the exponents of the parent matrices is calculated, hammexp.

Third, the node order difference between the two networks is measured as follows. Let us call 'order pair' a pair of indices (i,j) such that there is a directed path from j-th node to i-th node in the network, which sometimes is denoted by j>i. The order comparison is done by counting the false positive and false negative order pairs.

The fourth criteria accounts for the so called 'Markov blanket'. The term 'Markov pair' is used to denote a pair of indices which corresponding nodes have a common child. In case of extended comparison, the numbers of false positive and false negative Markov pairs are calculated.

The cnCompare function returns an object with the following slots: 1) the number of true positive edges tp; 2) the number of false positive edges fp; 3) the number of false negative edges fn; 4) precision pr; 5) specificity sp; 6) sensitivity(recall) sn; 7) the F-score, which is the harmonic average of precision and recall 8) the number of different elements in the corresponding parent matrices hamm; 9) the total number of different elements between all powers of the parent matrices hammexp;

Next three numbers identify the difference in the objects' skeletons (undirected graph structure)

7) the number of true positive undirected edges tp; 8) the number of false positive undirected edges fp; 9) the number of false negative undirected edges fn;

10) the number of false positive order pairs `order.fp`; 11) the number of false negative order pairs `order.fn`; 12) the number of false positive Markov pairs `markov.fp`; and 13) the number of false positive Markov pairs `markov.fn`. It is assumed that the first object represents the ground truth with respect to which the comparison is performed.

If `extended` is set off (FALSE) only the edge (TP, FP, FN) and skeleton (TP, FP, FN) numbers are reported, otherwise all distance parameters are calculated. Turning off the `extended` option is recommended for very large networks (e.g. with number of nodes > 500), since the calculation of some of the distance metrics involve matrix calculations for which the function is not optimized and can be very slow.

Value

A `catNetworkDistance` if `object2` is `catNetwork` and `catNetworkEvaluate` otherwise.

Author(s)

N. Balov

See Also

`catNetworkEvaluate-class`

Examples

```
cnet1 <- cnRandomCatnet(numnodes=10, maxpars=2, numcats=2)
cnet2 <- cnRandomCatnet(numnodes=10, maxpars=2, numcats=2)
dist <- cnCompare(object1=cnet1, object2=cnet2)
dist
```

---

cnComplexity-method *Network Complexity*

Description

Returns the complexity of a network

Usage

```
cnComplexity(object, node=NULL, include.unif=TRUE)
cnKLComplexity(object, node=NULL)
```

Arguments

`object`	a `catNetwork` object
`node`	an `integer`, node index
`include.unif`	a `logical`

Details

Complexity is a network characteristics that depends both on its graphical structure and the categorization of its nodes.

If node is specified, then the function returns that node complexity, otherwise the total complexity of object, which is the sum of its node complexities, is reported. A node complexity is determined by the number of its parents and their categories. For example, a node without parents has complexity 1. A node with k parents with respected number of categories c1,c2, ...,ck, has complexity c1*c2*...*ck. Complexity is always a number that is equal or greater than the number of nodes in the network. For a network with specified graph structure, its complexity determines the number of parameters needed to define its probability distribution and hence the importance of complexity as network characteristic.

If include.unif is set to FALSE

Value

An integer

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnComplexity(object=cnet)
```

---

cnDiscretize *Data Categorization*

Description

Soft and hard discretization of numerical data.

Usage

```
cnDiscretize(data, numcats=3, mode = "soft", marginal = "quantile",
learnset=NULL, cover=0.95, maxiter=100, eps=1e-8, weights=NULL)
```

Arguments

data	a numerical matrix or data.frame
numcats	an integer, the number of categories per node
mode	a character, the discretization method to be used, "soft" or "hard"
marginal	a character, the marginal model, "quantile", "uniform" or "gauss"
learnset	observation indices to be used for setting the discretization parameters

cover	a numerical between 0 and 1, proportion of the sample to be used for setting the discretization parameters
maxiter	an integer, maximum iterations for the mixture of Gaussians EM algorithm
eps	a numerical, convergence precision number
weights	a numerical, sample record weights

Details

The numerical data is discretized into given number of categories, numcats, using the empirical node quantiles. As in all functions of catnet package that accept data, if the data parameter is a matrix then it is organized in the row-node format. If it is a data.frame, the column-node format is assumed.

The mode specifies the discretization model. Currently, two hard discretization methods are supported - "quantile" and "uniform", which is the default choice. The quantile-based discretization method is applied as follows. For each node, the sample node distribution is constructed, which is then represented by a sum of non-intersecting classes separated by the quantile points of the sample distribution. Each node value is assigned the class index in which it falls into.

The uniform discretization breaks the range of values of each node into numcats equal intervals or of lengths proportional to the corresponding qlevels values.

Currently, the function assigns equal number of categories for each node of the data.

Value

A matrix or data.frame of indices.

Author(s)

N. Balov

See Also

cnSamples

Examples

```
ps <- t(sapply(1:10, function(i) rnorm(20, i, 0.1)))
dps1 <- cnDiscretize(data=ps, numcats=3, mode="hard", marginal="quantile")
hist(dps1[1,])
```

cnDot-method *Network Description File*

Description

The function generates a dot-file, the native storage format for Graphviz software package, that describes the graph structure of a catNetwork object.

Usage

```
cnDot(object, file=NULL, format="ps", nodestyle=NULL, edgestyle=NULL)
```

Arguments

object	a catNetwork, a list of catNetworks or a parent matrix
file	a character, an optional output file name
format	a character, an optional output file format, "ps" or "pdf"
nodestyle	a list of triplets, nodes' shape, color and edge-color
edgestyle	a list of triplets, nodes' shape, color and edge-color

Details

The function generates a dot-text file as supported by Graphviz library. In order to draw a graph the user needs a dot-file converter and pdf/postscript viewer. The environment variables R_DOTVIEWER and R_PDFVIEWER specify the corresponding executable routines.

If Graphviz is installed and the variable R_DOTVIEWER is set with the full path to the dot executable file (the routine that converts a dot-text file to a postscript or pdf), a pdf or postscript file is created depending on the value of the format parameter.

If the file variable is not specified, then the function just prints out the resulting string which otherwise would be written into a dot file. Next, if a pdf-viewer is available, the created postscript or pdf file is shown.

Value

A character or a dot-file

Author(s)

N. Balov

See Also

sdnet-package, cnPlot

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnDot(object=cnet, file="cnet")
```

---

cnEdges-method *Network Edges*

Description

Returns the set of directed edges of a catNetwork object.

Usage

```
cnEdges(object, which)
```

Arguments

object	a catNetwork
which	a vector of node indices or node names

Details

The edges of a catNetwork are specified as parent-to-child vectors. The function returns a list that for each node with index in the vector which contains its set of children. If which is not specified, the children of all nodes are listed.

Value

A list of nodes' children.

Author(s)

N. Balov

See Also

cnParents

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnEdges(object=cnet)
```

cnEntropy *Pairwise Node Entropy*

Description

Calculates the matrix of conditional entropy for each pair of nodes.

Usage

```
cnEntropy(data, pert=NULL)
cnEdgeDistanceKL(data, pert)
cnEdgeDistancePearson(data, pert)
cnEntropyOrder(data, pert=NULL)
```

Arguments

data	a matrix in row-nodes format or a data.frame in column-nodes format
pert	a binary matrix with the dimensions of data. A value 1 designates the corresponding node in the sample as perturbed.

Details

The conditional entropy of node X with respect to Y is defined as -P(X|Y)logP(X|Y), where P(X|Y) is the sample conditional probability, and this is the value at the (X,Y)'th position in the resulting matrix.

Value

A matrix

Author(s)

N. Balov

See Also

cnParHist

cnFind-method *Find Network by Complexity*

Description

This is a model selection routine that finds a network in a set of networks for a given complx.

Usage

```
cnFind(object, complx = 0, alpha=0, factor=1)
cnFindKL(object, numsamples)
```

Arguments

object	catNetworkEvaluate or dagEvaluate or list of catNetworks
complx	an integer, target complx
alpha	a character or numeric
factor	a numeric
numsamples	an integer

Details

The complx must be at least the number of nodes of the networks. If no network with the requested complx exists in the list, then the one with the closest complx is returned. Alternatively, one can apply some standard model selection with alpha="BIC" and alpha=AIC.

Value

A catNetwork object.

Author(s)

N. Balov

See Also

cnFindAIC, cnFindBIC

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=2, numcats=2)
psamples <- cnSamples(object=cnet, numsamples=100)
netlist <- cnSearchOrder(data=psamples, maxParentSet=2)
bnet <- cnFind(object=netlist, complx=cnComplexity(cnet))
bnet
```

cnFindAIC-method *Find Network by AIC*

Description

This is a model selection routine that finds a network in a set of networks using the AIC criteria.

Usage

```
cnFindAIC(object, numsamples)
```

Arguments

object	A list of catNetwork objects or catNetworkEvaluate or dagEvaluate
numsamples	an integer

Details

The function returns the network with maximal AIC value from a list of networks as obtained from one of the search-functions cnSearchOrder, cnSearchSA and cnSearchSAcluster. The formula used for the AIC is log(Likelihood) - Complexity.

Value

A catNetwork object with optimal AIC value.

Author(s)

N. Balov

See Also

cnFind, cnFindBIC

Examples

```
  cnet <- cnRandomCatnet(numnodes=12, maxpars=3, numcats=2)
  psamples <- cnSamples(object=cnet, numsamples=10)
  nodeOrder <- sample(1:12)
  nets <- cnSearchOrder(data=psamples, pert=NULL,
maxParentSet=2, maxComplexity=36, nodeOrder)
  aicnet <- cnFindAIC(object=nets)
  aicnet
```

cnFindBIC-method *Find Network by BIC*

Description

This is a model selection routine that finds a network in a set of networks using the BIC criteria.

Usage

```
cnFindBIC(object, numsamples)
```

Arguments

object	A list of catNetworkNode objects or catNetworkEvaluate or dagEvaluate
numsamples	The number of samples used for estimating object

Details

The function returns the network with maximal BIC value from a list of networks as obtained from one of the search-functions cnSearchOrder, cnSearchSA and cnSearchSAcluster. The formula used for the BIC is log(Likelihood) - 0.5*Complexity*log(numNodes).

Value

A catNetwork object with optimal BIC value.

Author(s)

N. Balov

See Also

cnFindAIC, cnFind

Examples

```
  cnet <- cnRandomCatnet(numnodes=12, maxpars=3, numcats=2)
  psamples <- cnSamples(object=cnet, numsamples=10)
  nodeOrder <- sample(1:12)
  nets <- cnSearchOrder(data=psamples, pert=NULL,
maxParentSet=2, maxComplexity=36, nodeOrder)
  bicnet <- cnFindBIC(object=nets, numsamples=dim(psamples)[2])
  bicnet
```

cnLoglik-method *Sample Log-likelihood*

Description

Calculate the log-likelihood of a sample with respect to a given catNetwork object

Usage

```
cnLoglik(object, data, pert=NULL, bysample=FALSE, softmode=FALSE)
```

Arguments

object	a catNetwork object
data	a data matrix given in the column-sample format, or a data.frame in the row-sample format
pert	a binary matrix with the dimensions of data. A value 1 designates the corresponding node in the sample as perturbed.
bysample	a logical
softmode	a logical, turns on/off the soft quantization mode

Details

If bysample is set to TRUE, the function output is a vector of log-likelihoods of the individual sample records. Otherwise, the total average of the log-likelihood of the sample is reported.

Value

A numeric value

Author(s)

N. Balov

See Also

cnNodeLoglik

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars = 3, numcats = 2)
psamples <- cnSamples(object=cnet, numsamples=100)
cnLoglik(object=cnet, data=psamples)
```

cnMatEdges-method *Network Edge Matrix*

Description

Returns a matrix representing the edges of a catNetwork object.

Usage

```
cnMatEdges(object)
```

Arguments

object	a catNetwork object

Details

The resulting matrix has two columns and the number of edges rows. Edges are given as ordered pairs of the elements of the first and second columns.

Value

A matrix of characters.

Author(s)

N. Balov

See Also

cnEdges, cnMatParents

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnMatEdges(object=cnet)
```

cnMatParents-method *Network Parent Matrix*

Description

Returns the binary matrix of parent-child relations of a catNetwork object.

Usage

```
cnMatParents(object, nodeorder)
```

Arguments

object	a catNetwork or catNetworkFit object
nodeorder	an integer vector specifying the order of the nodes to be taken

Details

The resulting matrix has a value 1 at row i and column j if i-th node has j-th node as a parent, and 0 otherwise.

Value

A matrix

Author(s)

N. Balov

See Also

cnParents, cnMatEdges

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnMatParents(object=cnet)
```

cnNew *New catNetwork*

Description

Creates a new catNetwork with specified nodes, categories, parent sets and probability structure.

Usage

```
cnNew(nodes, cats, pars, probs=NULL, p.delta1=0.01, p.delta2=0.01, dagonly=FALSE)
```

Arguments

nodes	a vector of nodes names
cats	a list of node categories
pars	a list of node pars
probs	a list of probabilities
p.delta1	a numeric
p.delta2	a numeric
dagonly	a logical, selects between catNetwork and DAG

Details

If probs is not specified, then a random probability model is assigned with conditional probability values in the union of the intervals [p.delta1, 0.5-p.delta2] and [0.5+p.delta2, 1-p.delta1]. Because of the nested list hierarchy of the probability structure, specifying the probability argument explicitly can be very elaborated task for large networks. In the following example we create a small network with only three nodes. The first node has no pars and only its marginal distribution is given, c(0.2,0.8). Note that all inner most vectors in the probs argument, such as (0.4,0.6), represent conditional distributions and thus sum to 1.

Value

A catNetwork object.

Author(s)

N. Balov

See Also

catNetwork-class, cnRandomCatnet

Examples

```
cnet <- cnNew(
nodes = c("a", "b", "c"),
cats = list(c("1","2"), c("1","2"), c("1","2")),
pars = list(NULL, c(1), c(1,2)),
probs = list( c(0.2,0.8),
list(c(0.6,0.4),c(0.4,0.6)),
list(list(c(0.3,0.7),c(0.7,0.3)),
list(c(0.9,0.1),c(0.1,0.9))))
)
```

---

cnNodeLoglik *Node Log-likelihood*

Description

For a given data sample, the function calculates the log-likelihood of a node with respect to a specified parent set.

Usage

```
cnNodeLoglik(object, node, data, pert=NULL, softmode=FALSE, klmode=FALSE)
```

Arguments

object	a catNetwork object
node	an integer or a list of integers, node indices in the data
data	a matrix or data.frame of categories
pert	an optional perturbation matrix or data.frame
softmode	a logical, turns on/off the soft quantization mode
klmode	a logical, use the negative KL-distance or the log-likelihood

Value

a numeric value

Author(s)

N. Balov

See Also

cnLoglik

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
psamples <- cnSamples(object=cnet, numsamples=100)
cnNodeLoglik(cnet, node=5, data=psamples)
```

cnNodeMarginalProb-method
Probability Calculations

Description

Marginal probability of a node, joint probability of a set of nodes or conditional probability of two sets of nodes.

Usage

```
cnNodeMarginalProb(object, node)
cnJointProb(object, nodes)
cnCondProb(object, x, y)
```

Arguments

object	a catNetwork
node	an integer, a node index in object
nodes	a vector of node names or indices in object
x,y	vectors of node categories (either characters or indices) named after nodes of object

Details

cnJointProb returns a matrix with probability values for each combinations of categories arranged in columns. cnCondProb calculates the value of P(X=x|Y=y).

Value

a numerical or numerical matrix

Author(s)

N. Balov

See Also

cnProb

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnNodeMarginalProb(cnet, node=5)
cnCondProb(cnet, x=c("N1"=1, "N2"=2), y=c("N3"=1, "N4"=2, "N5"=2))
```

cnNodes-method *Netwok Nodes*

Description

Returns the list of nodes of a catNetwork object.

Usage

```
cnNodes(object, which)
```

Arguments

object	a catNetwork object
which	a vector of node indices

Details

Nodes are represented by characters. When a random catNetwork object is constructed, it takes the default node names N#, where # are node indices. The function returns the node names with indices given by parameter which, and all node names if which is not specified.

Value

a list of characters, the node names

Author(s)

N. Balov

See Also

cnNumNodes

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnNodes(object=cnet)
```

cnNodeSampleLoglik *Node Log-likelihood*

Description

For a given data sample, the function calculates the log-likelihood of a node with respect to a specified parent set.

Usage

```
cnNodeSampleLoglik(node, pars, data, pert=NULL)
cnNodeSampleProb(node, pars, data, pert=NULL)
```

Arguments

node	an integer or a list of integers, node indices in the data
pars	an integer or a list of integers, vector of parent indices for the nodes
data	a matrix or data.frame of categories
pert	an optional perturbation matrix or data.frame

Value

a numeric value

Author(s)

N. Balov

See Also

cnLoglik

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
psamples <- cnSamples(object=cnet, numsamples=100)
cnNodeSampleLoglik(node=5, pars=c(1,2), data=psamples)
```

cnNumNodes-method *Network Size*

Description

Returns the number of nodes of a catNetwork object.

Usage

```
cnNumNodes(object)
```

Arguments

object	a catNetwork

Value

an integer

Author(s)

N. Balov

See Also

cnNodes

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnNumNodes(object=cnet)
```

cnOrder-method *Network Node Order*

Description

The function returns an order of the nodes of a network that is compatible with its parent structure.

Usage

```
cnOrder(object)
```

Arguments

object	a catNetwork or a list of node parents.

Details

An order is compatible with the parent structure of a network if each node has as parents only nodes appearing earlier in that order. That such an order exists is guaranteed by the fact that every `catNetwork` is a DAG (Directed Acyclic Graph). The result is one order out of, eventually, many possible.

Value

a list of node indices.

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=20, maxpars=3, numcats=2)
cnOrder(object=cnet)
```

---

cnParents-method *Network Parent Structure*

Description

Returns the list of parents of selected nodes of a `catNetwork` object. If `which` is not specified, the parents of all nodes are listed.

Usage

```
cnParents(object, which)
```

Arguments

`object`	a `catNetwork` object
`which`	a vector of node indices

Value

A list of named nodes.

Author(s)

N. Balov

See Also

`cnMatParents`, `cnEdges`

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnParents(object=cnet)
```

---

cnParHist-method *Parenthood Histogram*

Description

Calculates the histogram of parent-child edges for a catNetworkEvaluate object or a list of catNetworks

Usage

```
cnParHist(object)
```

Arguments

object	catNetworkEvaluate or list of catNetwork objects

Value

a numerical matrix

Author(s)

N. Balov

Examples

```
  cnet <- cnRandomCatnet(numnodes=20, maxpars=3, numcats=2)
  psamples <- cnSamples(cnet, 100)
  nodeOrder <- sample(1:20)
  nets <- cnSearchOrder(psamples, pert=NULL,
maxParentSet=2, maxComplexity=50, nodeOrder)
  cnParHist(object=nets)
```

cnPearsonTest-method *Goodness of Fit Test*

Description

The function calculates the Pearson's chi-square statistics for all nodes of a network.

Usage

```
cnPearsonTest(object, data)
```

Arguments

object	a catNetwork
data	a data matrix or data.frame

Details

For given data and network object, the function reports both the chi-square statistics and the degree of freedom for each node in the network for the purpose of performing goodness of fit tests.

Value

A list

Author(s)

N. Balov

cnPlot-method *Plot Network*

Description

Draws the graph structure of catNetwork object or some diagnostic plots associated with a catNetworkEvaluate

Usage

```
cnPlot(object, file=NULL)
```

Arguments

object	catNetwork or catNetworkEvaluate object
file	a file name

Details

First we consider the case when object is a catNetwork. There are two visualization options implemented - one using 'igraph' and the other 'Graphviz'. The usage of these two alternatives is controlled by two environment variables - the logical one R_CATNET_USE_IGRAPH and the character one R_DOTVIEWER, correspondingly. If igraph is installed and R_CATNET_USE_IGRAPH is set to TRUE, the function constructs an igraph compatible object corresponding to the object and plot it.

If igraph is not found, the function generates a dot-file with name file.dot, if file is specified, or unknown.dot otherwise. Furthermore, provided that Graphviz library is found and R_DOTVIEWER points to the dot-file executable, the created earlier dot-file will be compiled to pdf or postscript, if object is a list. Finally, if the system has pdf or postscript rendering capabilities and R_PDFVIEWER variable shows the path to the pdf-rendering application, the resulting pdf-file will be shown.

In case object is of class catNetworkEvaluate, then the function draws six relevant plots: likelihood vs. complexity, Hamming (hamm) and exponential Hamming (hammexp) distances, Markov neighbor distance (FP plus FN), and the false positive (fp) and false negative (fn) edges vs. complexity.

Value

A R-plot or dot-file or pdf-file.

Author(s)

N. Balov

See Also

cnDot, catNetworkEvaluate-class, cnCompare

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnPlot(object=cnet)
```

---

cnPredict-method *Prediction*

Description

Predicts the 'not-available' elements in an incomplete sample.

Usage

```
cnPredict(object, data)
```

Arguments

object	a catNetwork
data	a data matrix or data.frame

Details

Data should be a matrix or data frame of categorical values or indices. If it is a matrix then the rows should represent object's nodes; otherwise, the columns represent the nodes. Data's values represent object's categories either as characters or indices. Indices should be integers in the range from 1 to the number of categories of the corresponding node. Prediction is made for those nodes that are marked as not-available (NA) in the data and is based on maximum probability criterion. For each data instance, the nodes are traversed in their topological order in object and the categorical values with the maximum probability are assigned.

Value

An updated sample matrix

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=3)
## generate a sample of size 2 and set nodes 8, 9 and 10 as not-available
psamples <- matrix(as.integer(1+rbinom(10*2, 2, 0.4)), nrow=10)
psamples[8, ] <- rep(NA, 2)
psamples[9, ] <- rep(NA, 2)
psamples[10, ] <- rep(NA, 2)
## make show sample rows are named after the network's nodes
rownames(psamples) <- cnNodes(cnet)
## predict the values of nodes 8, 9 and 10
newsamples <- cnPredict(object=cnet, data=psamples)
```

---

cnProb-method *Conditional Probability Structure*

Description

Returns the list of conditional probabilities of nodes specified by which parameter of a catNetwork object. Node probabilities are reported in the following format. First, node name and its parents are given, then a list of probability values corresponding to all combination of parent categories (put in brackets) and node categories. For example, the conditional probability of a node with two parents, such that both the node and its parents have three categories, is given by 27 values, one for each of the 3*3*3 combination.

Usage

```
cnProb(object, which=NULL)
cnPlotProb(object, which=NULL)
```

Arguments

object	a catNetwork object
which	a vector of indices

Value

A named list of probability tables.

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnProb(object=cnet)
cnPlotProb(object=cnet)
```

---

cnRandomCatnet *Random Network*

Description

Creates a random catNetwork with specified number of nodes, number of parents and categories per node.

Usage

```
cnRandomCatnet(numnodes, maxpars, numcats, p.delta1=0.01, p.delta2=0.01)
```

Arguments

numnodes	an integer, the number of nodes
maxpars	an integer, the maximum number of parents per node
numcats	an integer, the number of categories for each node. It is the function limitation to support only constant number of node categories.
p.delta1	a numeric
p.delta2	a numeric

Details

A random set of parents, no more than maxpars, is assigned to each node along with a random conditional probability distribution with values in the union of [p.delta1, 0.5-p.delta2] and [0.5+p.delta2, 1-p.delta1]. Also, each node is assigned a fixed, thus equal, number of categories, numcats.

The function is designed for evaluation and testing purposes only thus lacking much user control over the networks it create. Once created with cnRandomCatnet, a network can be further modified manually node by node. However, this requires direct manipulation of the object's slots and may result in a wrong network object. It is recommended that after any manual manipulation a call is(object, "catNetwork") is performed to check the object's integrity.

Value

A catNetwork object

Author(s)

N. Balov

See Also

cnNew

Examples

```
cnet <- cnRandomCatnet(numnodes=20, maxpars=3, numcats=2)
```

---

cnReorderNodes-method *Reorder Network Nodes*

Description

The function rearranges the nodes of a network according to a new order.

Usage

```
cnReorderNodes(object, nodeIndices)
```

Arguments

object	a catNetwork
nodeIndices	a vector representing the new node order

Details

Node reordering affects the list of node names, parents and probabilities. It is a useful operation in cases when comparison of two networks is needed.

Value

A catNetwork object.

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnMatParents(cnet)
cnet1 <- cnReorderNodes(object=cnet, nodeIndices=cnOrder(cnet))
cnNodes(object=cnet1)
cnMatParents(cnet1)
```

---

cnSamples-method *Samples from Network*

Description

Generates samples from of a catNetwork object.

Usage

```
cnSamples(object, numsamples = 1, pert = NULL, output="frame", as.index=FALSE, naRate=0)
```

Arguments

object	a catNetwork
numsamples	an integer, the number of samples to be generated
pert	a vector, node pert
output	a character, the output format. Can be a data.frame or matrix.
as.index	a logical, the output categorical format
naRate	a numeric, the proportion of NAs per sample instance

Details

If the output format is "matrix" then the resulting sample matrix is in row-node format - the rows correspond to the object's nodes while the individual samples are represented by columns. If the output format is "frame", which is by default, the result is a data frame with columns representing the nodes and levels the set of categories of the respected nodes. If as.index is set to TRUE, the output sample consists of categorical indices, otherwise, and this is by default, of characters specifying the categories.

A perturbed sample is a sample having nodes with predefined, thus fixed, values. Non-perturbed nodes, the nodes which values have to be set, are designated with zeros in the perturbation vector

and their values are generated conditional on the values of their pars. While the non-zero values in the perturbation vector are carried on unchanged to the output.

If `naRate` is positive, then `floor(numnodes*naRate)` NA values are randomly placed in each sample instance.

Value

A `matrix` or `data.frame` of node categories as integers or characters

Author(s)

N. Balov

See Also

`cnPredict`

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=3)
## generate a sample of size 100 from cnet
psamples <- cnSamples(object=cnet, numsamples=100, output="frame", as.index=FALSE)
## perturbed sample
nsamples <- 20
pert <- rbinom(10, 2, 0.4)
## generate a perturbed sample of size 100 from cnet
psamples <- cnSamples(object=cnet, numsamples=nsamples, pert, as.index=TRUE)
```

---

cnSearchHist *Parent Histogram Matrix*

Description

Estimation of the parent matrix of nodes from data. The frequency of node edges is obtained by fitting networks consistent to randomly generated node orders.

Usage

```
cnSearchHist(data, pert=NULL,
maxParentSet=1, parentSizes=NULL, maxComplexity=0,
nodeCats=NULL, parentsPool=NULL, fixedParents=NULL,
score = "BIC", weight="loglik",
maxIter=32, numThreads=2, echo=FALSE)
```

Arguments

data	a matrix in row-nodes format or a data.frame in column-nodes format
pert	a binary matrix with the dimensions of data. A value 1 designates the corresponding node in the sample as perturbed
maxParentSet	an integer, the maximal number of parents per node
parentSizes	an integer vector, maximal number of parents per node
maxComplexity	an integer, the maximal network complexity for the search
nodeCats	a list of node categories
parentsPool	a list of parent sets to choose from
fixedParents	a list of parent sets to choose from
score	a character, network selection score such as "AIC" and "BIC"
weight	a character, specifies how the
maxIter	an integer, the number of single order searches to be performed
numThreads	an integer value, the number of parallel threads
echo	a boolean that sets on/off some functional progress and debug information

Details

The function performs niter calls of cnSearchOrder for randomly generated node orders (uniformly over the space of all possible node orders), selects networks according to score and sum their parent matrices weighted by weight. Three scoring criteria are currently supported: "BIC", "AIC" and maximum complexity for any other value of score. The weight can be 1) "likelihhod", then the parent matrices are multiplied by the network likelihood, 1) "score", then the parent matrices are multiplied by the exponential of the network score, 3) any other value of weight uses multiplier 1. In this case the entries in the output matrix count the presence of the corresponding parent-child pairs.

The function can runs numThreads number of parallel threads each processing different order. cnSearchHist function can be useful for empirical estimation of the relationships in some multivariate categorical data.

Value

A matrix

Author(s)

N. Balov

See Also

cnMatParents, cnSearchOrder

Examples

```
  library(sdnet)
  cnet <- cnRandomCatnet(numnodes=8, maxpars=3, numcats=2)
  psamples  <- cnSamples(object=cnet, numsamples=100)
  mhisto <- cnSearchHist(data=psamples, pert=NULL,
maxParentSet=2, maxComplexity=100)
  mhisto
```

`cnSearchOrder` *Network Search for Given Node Order*

Description

The function implements a MLE based algorithm to search for optimal networks complying with a given node order. It returns a list of networks, with complexities up to some maximal value, that best fit the data.

Usage

```
cnSearchOrder(data, pert=NULL,
maxParentSet=0, parentSizes=NULL, maxComplexity=0,
nodeOrder=NULL,
nodeCats=NULL, parentsPool=NULL, fixedParents=NULL, edgeProb=NULL,
echo=FALSE, softmode=FALSE, dagsOnly = FALSE, classes = NULL, clsdist=1)
```

Arguments

`data`	a `matrix` in row-nodes format or a `data.frame` in column-nodes format
`pert`	a binary matrix with the dimensions of `data`. A value 1 marks that the node in the corresponding sample as perturbed
`maxParentSet`	an `integer`, maximal number of parents for all nodes
`parentSizes`	an `integer` vector, maximal number of parents per node
`maxComplexity`	an `integer`, the maximal network complexity for the search
`nodeOrder`	a `vector` specifying a node order; the search is among the networks consistent with this topological order
`nodeCats`	a `list` of node categories
`parentsPool`	a `list` of parent sets to choose from
`fixedParents`	a `list` of parent sets to choose from
`edgeProb`	a square `matrix` of length the number of nodes specifying prior edge probabilities
`echo`	a `logical`, turns on/off some progress information
`softmode`	a `logical`, turns on/off the soft quantization mode
`dagsOnly`	a `logical`, selects between catNetwork and DAG only search
`classes`	a binary matrix with the dimensions of `data` that assigns a class to each node-observation
`clsdist`	class separation distance function, currently 1('chisq') and 2('kl') are supported

Details

The data can be a matrix of character categories with rows specifying the node-variables and columns assumed to be independent samples from an unknown network, or a data.frame with columns specifying the nodes and rows being the samples.

The number of node categories are obtained from the sample. If given, the nodeCats is used as a list of categories. In that case, nodeCats should include the node categories presented in the data.

The function returns a list of networks, one for each admissible complexity within the specified range. The networks in the list are the Maximum Likelihood estimates in the class of networks having the given topological order of the nodes and complexity. When maxComplexity is not given, thus zero, its value is reset to the maximum possible complexity for the given parent set size. When nodeOrder is not given or NULL, the order of the nodes in the data is taken, 1,2,....

The parameters parentsPool and fixedParents allow the user to put some exclusion/inclusion constrains on the possible parenthood of the nodes. They should be given as lists of index vectors, one for each node.

The rows in edgeProb correspond to the nodes in the sample. The [i,j]-th element in edgeProb specifies a prior probability for the j-th node to be a parent of the i-th one. In calculating the prior probability of a network all edges are assumed independent Bernoulli random variables. The elements of edgeProb are cropped in the range [0,1], such that the zero probabilities effectively exclude the corresponding edges, while the ones force them.

Value

A catNetworkEvaluate object

Author(s)

N. Balov

Examples

```
  cnet <- cnRandomCatnet(numnodes=12, maxpars=3, numcats=2)
  psamples <- cnSamples(object=cnet, numsamples=100)
  nodeOrder <- sample(1:12)
  nets <- cnSearchOrder(data=psamples, pert=NULL,
maxParentSet=2, maxComplexity=36, nodeOrder)
  ## next we find the network with complexity of the original one and plot it
  cc <- cnComplexity(object=cnet)
  cnFind(object=nets, complx=cc)
```

---

cnSetProb-method *Set Probability from Data*

Description

The function sets the probability structure of a network from data according to the Maximum Likelihood criterion.

Usage

```
cnSetProb(object, data, pert=NULL, nodeCats=NULL, softmode=FALSE)
```

Arguments

object	a catNetwork
data	a data matrix or data.frame
pert	a binary matrix with the dimensions of data
nodeCats	a list of node categories
softmode	a logical, turns on/off the soft quantization mode

Details

The function generates a new probability table for object and returns an updated catNetwork. The graph structure of the object is kept unchanged.

The data can be a matrix in the node-rows format, or a data.frame in the node-column format. If given, the nodeCats is used as a list of categories. In that case, nodeCats should include the node categories presented in the data.

Value

catNetwork

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=3)
psamples <- matrix(as.integer(1+rbinom(10*100, 2, 0.4)), nrow=10)
rownames(psamples) <- cnet@nodes
newcnet <- cnSetProb(object=cnet, data=psamples)
```

---

cnSetSeed *Random Generator Seed*

Description

Sets a seed for the random number generator.

Usage

```
cnSetSeed(seed)
```

Arguments

seed	an integer

Details

Setting a fixed seed before any stochastic function guaratees repeated results.

Value

NA

Author(s)

N. Balov

---

cnSubNetwork-method *Sub-Network*

Description

Returns a sub-network of a given catNetwork object.

Usage

```
cnSubNetwork(object, nodeIndices, indirectEdges)
```

Arguments

object	a catNetwork
nodeIndices	a vector, the subset of nodes to be taken
indirectEdges	a logical, should the indirect connectivity be preserved

Details

The function creates a new network from a given one using a subset of its nodes, specified by nodeIndices. If indirectIndices is set to TRUE, then the resulting network contains edges between all nodes that are connected by chains of directed edges in the original one. The default value of indirectIndices is FALSE, thus the new set of edges is subset of the original one.

Value

A catNetwork object.

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=10, maxpars=3, numcats=2)
cnet1 <- cnSubNetwork(object=cnet, nodeIndices=c(1,2,3,4,5), indirectEdges=TRUE)
cnNodes(object=cnet)
cnNodes(object=cnet1)
```

CPDAG-class *Class CPDAG*

Description

Base class implementing Complete Partially Directed Acyclic Graphs (CPDAGs)

Slots

numnodes: an integer, the number of nodes

nodes: a vector of node names

edges: a list of graph edges

Author(s)

N. Balov

See Also

dag2cpdag

---

dag2cpdag-method *Complete Network Representation*

Description

Generate the complete graphical structure for a catNetwork object.

Usage

```
dag2cpdag(object)
```

Arguments

object	a catNetwork object

Value

A non-DAG catNetwork object.

Author(s)

N. Balov

expres *Naive Classifier of Gene Expression Profiles*

Description

Implements a naive classifier using soft discretization.

Usage

```
sdnLearn(data, cls, clslevs = NULL, ncats = 3, nodeCats = NULL, quant="uniform", std=TRUE)
sdnPredict(model, data, std=TRUE)
sdnEvaluate(train, test, ncats = 3, nodeCats = NULL, std=FALSE)
```

Arguments

data	a numerical matrix in row-genes format
train	a numerical matrix in row-genes format
test	a numerical matrix in row-genes format
cls	a factor or integer, the sample labels
clslevs	an optional vector of labels, should include training data's labels
ncats	an integer, the number of categories per node
nodeCats	a list, custom node categories
quant	quantization method
std	a logical, should the data rows be standardized
model	a list of components for the training model

Details

The model contains a vector of gene names geneset, a vector of sample labels clslevs, class catNetworks: nets, a list of node categories nodeCats and a training quantization model quant.

Value

sdnPredict returns the log-ratio of the class conditional probabilities for each test observation. sdnEvaluate handles 2-class problems and returns the prediction accuracy and predicted classes.

Author(s)

N. Balov

isDAG *Check Direct Acyclic Graph (DAG) Condition*

Description

For a pair of node and parent lists, the function checks whether the DAG condition holds or not.

Usage

```
isDAG(lnodes, lpars)
```

Arguments

lnodes	a list of nodes
lpars	a list of node parents

Details

The DAG verification algorithm is based on the topological ordering of the graph nodes. If node ordering is not possible, the graph is not a DAG.

Value

A logical TRUE/FALSE value.

Author(s)

N. Balov

Examples

```
cnet <- cnRandomCatnet(numnodes=20, maxpars=3, numcats=2)
isDAG(lnodes=cnet@nodes, lpars=cnet@pars)
```

tyroid *Tyroid cancer data*

Description

These are subsets of RMA normalized GEO datasets GSE3467 and GSE3678. Each containes the expression of 95 genes selected by t-test significance. The data is utilized in the tyroid demo.

Usage

```
data(tyr1)
```

Format

List containing normalized data ('cdata' slot) and class labels ('cls' slot).

Source

"http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE3467", "http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE

Index