Package 'sparsebnUtils'

May 29, 2019

Title Utilities for Learning Sparse Bayesian Networks

Version 0.0.7

Date 2019-05-28

Maintainer Bryon Aragam <sparsebn@gmail.com>

Description A set of tools for representing and estimating sparse Bayesian networks from continuous and discrete data, as described in Aragam, Gu, and Zhou (2017) <arXiv:1703.04025>.

Depends R (>= 3.2.3)

Imports Matrix, stats, utils, methods, nnet

Suggests bnlearn, graph, igraph, network, RCy3, testthat

URL https://github.com/itsrainingdata/sparsebnUtils

BugReports https://github.com/itsrainingdata/sparsebnUtils/issues

License GPL (>= 2)

RoxygenNote 6.1.1

NeedsCompilation no

Author Bryon Aragam [aut, cre],
Jiaying Gu [aut]

Repository CRAN

Date/Publication 2019-05-29 08:00:09 UTC

R topics documented:

`as.data.frame.sparsebnData` *Convert a sparsebnData object back to a data.frame*

Description

Convert a sparsebnData object back to a data.frame

Usage

```
## S3 method for class 'sparsebnData'
as.data.frame(x, ...)
```

Arguments

x	a sparsebnData object.
...	(optional) additional argument to as.data.frame.

---

as.edgeList *as.edgeList*

Description

Methods for coercing other R objects to edgeList objects.

Usage

```
as.edgeList(x)
```

Arguments

x	A compatible R object.

Value

edgeList

---

as.sparse *as.sparse*

Description

Methods for coercing other R objects to sparse objects.

Usage

```
as.sparse(x, index = "R", ...)
```

Arguments

x	A compatible R object.
index	"R" or "C", depending on whether to use R- or C-style indexing.
...	other parameters.

Value

sparse

coerce_discrete *Recode discrete data*

Description

Recodes discrete data so that the levels correspond to `0...n-1`, where `n` is the total number of levels in a discrete factor.

Usage

```
coerce_discrete(x)

## S3 method for class 'factor'
coerce_discrete(x)

## S3 method for class 'numeric'
coerce_discrete(x)

## S3 method for class 'integer'
coerce_discrete(x)

## S3 method for class 'character'
coerce_discrete(x)

## S3 method for class 'data.frame'
coerce_discrete(x)

## S3 method for class 'sparsebnData'
coerce_discrete(x)
```

Arguments

`x`	an R object to coerce.

Details

Assumes data is unordered. Ordered factors are not supported at this time.

Examples

```
x <- 1:5
coerce_discrete(x) # output: 0 1 2 3 4

x <- c("high", "normal", "high", "low")
coerce_discrete(x) # output: 0 2 0 1
```

count.interventions *Count the number of rows under intervention*

Description

Returns the number of rows with at least one intervention

Usage

```
count.interventions(data)
```

Arguments

data	a sparsebnData object.

count.levels *Count the number of levels per variable*

Description

Returns the number of levels per variable as an ordered vector.

Usage

```
count.levels(data)
```

Arguments

data	a sparsebnData object.

degrees *Degree distribution of a graph*

Description

Returns a data.frame with summary statistics on the total degree, in-degree, and out-degree of each node in the network.

Usage

```
degrees(x)
```

Arguments

x	an edgeList object

edgeList *edgeList class*

Description

Convenience wrapper class for a (column-major) edge list. Each component of the list corresponds to a node, and each component is an integer vector whose components are the parents of this node in the graph.

Usage

```
is.edgeList(x)

## S3 method for class 'edgeList'
print(x, maxsize = 20, ...)

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

edgeList(x)
```

Arguments

x	A list containing parents for each node in a graph. The length of this list should be the same as the number of nodes in the graph.
maxsize	Maximum number of nodes to print out. If num.nodes(x) > maxsize, then a simple summary will be printed instead.
...	(optional) additional arguments.
object	an object of type edgeList

Details

Also inherits from list.

Methods

get.adjacency.matrix, num.nodes, num.edges

estimate.parameters *Estimate the parameters of a Bayesian network*

Description

Given the structure of a Bayesian network, estimate the parameters (weights) using ordinary least squares (for Gaussian data) or logistic regression (for discrete data).

Usage

```
estimate.parameters(fit, data, ...)
```

Arguments

fit	fitted sparsebnFit or sparsebnPath object containing the Bayesian network structure to fit.
data	Data to use for fitting.
...	(optional) additional arguments to pass to lm or glm.

Details

The low-level fitting methods are fit_glm_dag (for continuous data) and fit_multinom_dag (for discrete data).

fit_glm_dag *Inference in Bayesian networks*

Description

Basic computing engine called by estimate.parameters for fitting parameters in a Bayesian network. Should not be used directly unless by experienced users.

Usage

```
fit_glm_dag(parents, dat, call = "lm.fit", ...)
```

Arguments

parents	edgeList object.
dat	Data.
call	Either "lm.fit" or "glm.fit".
...	If call = "glm.fit", specify family here. Also allows for other parameters to lm.fit and glm.fit.

Details

Can call either lm.fit or glm.fit, with any choice of family.

`fit_multinom_dag` *Inference in discrete Bayesian networks*

Description

Given the structure of a Bayesian network, estimate the parameters using multinomial logistic regression. For each node i, regress i onto its parents set using `multinom` in package `nnet`.

Usage

```
fit_multinom_dag(parents, dat)
```

Arguments

`parents`	An `edgeList` object.
`dat`	Data, a dataframe or matrix

Value

A list with with one component for each node in the graph. Each node is a coefficient matrix for the parents of that node.

Examples

```
### construct a random data set
x <- c(0,1,0,1,0)
y <- c(1,0,1,0,1)
z <- c(0,1,2,1,0)
a <- c(1,1,1,0,0)
b <- c(0,0,1,1,1)
dat <- data.frame(x, y, z, a, b)

### randomly construct an edgelist of a graph
nnode <- ncol(dat)
li <- vector("list", length = nnode)
li[[1]] <- c(2L,4L)
li[[2]] <- c(3L,4L,5L)
li[[3]] <- integer(0)
li[[4]] <- integer(0)
li[[5]] <- integer(0)
edgeL <- edgeList(li)

### run fit_multinom_dag
fit.multinom <- fit_multinom_dag(edgeL, dat)
```

`generate.lambdas` *generate.lambdas*

Description

Convenience function for creating a grid of lambdas.

Usage

```
generate.lambdas(lambda.max, lambdas.ratio = 0.001,
  lambdas.length = 50, scale = "linear")
```

Arguments

`lambda.max`	Maximum value of lambda; in terms of the algorithm this is the initial value of the regularization parameter in the solution path.
`lambdas.ratio`	Ratio between the maximum lambda value and the minimum lambda value in the solution path.
`lambdas.length`	Number of values to include.
`scale`	Which scale to use: Either `"linear"` or `"log"`.

Details

See Section 5.3 of Aragam and Zhou (2015) for a discussion of regularization paths (also, solution paths).

`get.adjacency.matrix.edgeList` *get.adjacency.matrix*

Description

Extracts the adjacency matrix of the associated graph object.

Usage

```
## S3 method for class 'edgeList'
get.adjacency.matrix(x)

get.adjacency.matrix(x)

## S3 method for class 'sparsebnFit'
get.adjacency.matrix(x)

## S3 method for class 'sparsebnPath'
get.adjacency.matrix(x)
```

Arguments

x	any R object.

Value

matrix

Methods (by class)

- edgeList: Convert internal edgeList representation to an adjacency matrix
- sparsebnFit: Retrieves edges slot and converts to an adjacency matrix
- sparsebnPath: Retrieves all edges slots in the solution path, converts to an adjacency matrix, and returns as a list

---

get.covariance *Covariance and precision matrices*

Description

Methods for computing covariance and precision matrices given an estimated directed graph.

Usage

```
get.covariance(x, data, ...)

get.precision(x, data, ...)
```

Arguments

x	fitted sparsebnFit or sparsebnPath object.
data	data as sparsebnData object.
...	(optional) additional parameters

Details

For Gaussian data, the precision matrix corresponds to an undirected graphical model for the distribution. This undirected graph can be tied to the corresponding directed graphical model; see Sections 2.1 and 2.2 (equation (6)) of Aragam and Zhou (2015) for more details.

Value

Covariance (or precision) matrix as Matrix object.

`get.lambdas` *get.lambdas*

Description

Extracts the lambda values from a `sparsebnPath` object.

Usage

```
get.lambdas(x)

## S3 method for class 'sparsebnPath'
get.lambdas(x)
```

Arguments

`x`	a `sparsebnPath` object.

Value

Vector of `numeric` lambda values in fitted object.

Methods (by class)

- `sparsebnPath`: Returns a vector of lambda values defining the solution path of a `sparsebnPath` object.

`get.nodes` *get.nodes*

Description

Returns the node names associated with a fitted object.

Usage

```
get.nodes(x)

## S3 method for class 'sparsebnFit'
get.nodes(x)

## S3 method for class 'sparsebnPath'
get.nodes(x)
```

Arguments

x	a sparsebnFit or sparsebnPath object.

Value

Vector of character names.

Methods (by class)

- sparsebnFit: Returns the node names from a sparsebnFit object.
- sparsebnPath: Returns the node names from a sparsebnPath object.

---

get.solution *Select solutions from a solution path*

Description

Choose solutions from a solution path based on number of edges, value of regularization parameter lambda, or index.

Usage

```
get.solution(x, edges, lambda, index)
```

Arguments

x	a sparsebnPath object.
edges	number of edges to search for.
lambda	value of regularization parameter to search for.
index	integer index to select.

Details

For edges (resp. lambda), the solution with the closest number of edges (resp. regularization parameter) is returned. If there is no match within a tolerance of 0.1 for lambda, nothing is returned. Fuzzy matching is not used for when selecting by index.

If there is more than one match (for example, by number of edges), then the first such estimate is returned. Note that select(x, index = j) is equivalent to (but slightly slower than) x[[j]].

is.obs *Check if data is observational*

Description

Returns TRUE if the data contains no interventions, i.e. is purely observational

Usage

```
is.obs(data)
```

Arguments

data	a sparsebnData object.

is.zero.edgeList *is.zero*

Description

Determines whether or not the object is the same as the null or zero object from its class.

Usage

```
## S3 method for class 'edgeList'
is.zero(x)

is.zero(x)
```

Arguments

x	a fitted object.

Value

TRUE or FALSE.

Methods (by class)

- edgeList: Determines whether or not the object represents a null graph with no edges.

num.edges.edgeList *num.edges*

Description

Extracts the number of edges of the associated graph object.

Usage

```
## S3 method for class 'edgeList'
num.edges(x)

num.edges(x)

## S3 method for class 'sparsebnFit'
num.edges(x)

## S3 method for class 'sparsebnPath'
num.edges(x)
```

Arguments

x	a sparsebnFit or sparsebnPath object.

Value

Number of edges as integer.

Methods (by class)

- edgeList: Extracts the number of edges of edgeList object.
- sparsebnFit: Extracts the number of edges of sparsebnFit object.
- sparsebnPath: Extracts the number of edges of sparsebnPath object.

num.nodes.edgeList *num.nodes*

Description

Extracts the number of nodes of the associated graph object.

Usage

```
## S3 method for class 'edgeList'
num.nodes(x)

num.nodes(x)

## S3 method for class 'sparsebnFit'
num.nodes(x)

## S3 method for class 'sparsebnPath'
num.nodes(x)
```

Arguments

x	a `sparsebnFit` or `sparsebnPath` object.

Value

Number of nodes as `integer`.

Methods (by class)

- `edgeList`: Extracts the number of nodes of edgeList object.
- `sparsebnFit`: Extracts the number of nodes of sparsebnFit object.
- `sparsebnPath`: Extracts the number of nodes of sparsebnPath object.

`num.samples` *num.samples*

Description

Extracts the number of samples used to estimate the associated object.

Usage

```
num.samples(x)

## S3 method for class 'sparsebnData'
num.samples(x)

## S3 method for class 'sparsebnFit'
num.samples(x)

## S3 method for class 'sparsebnPath'
num.samples(x)
```

Arguments

x	a sparsebnFit or sparsebnPath object.

Value

Number of samples as integer.

Methods (by class)

- sparsebnData: Extracts the number of samples of sparsebnData object.
- sparsebnFit: Extracts the number of samples of sparsebnFit object.
- sparsebnPath: Extracts the number of samples of sparsebnPath object.

---

openCytoscape *Display graphs in Cytoscape*

Description

NOTE: This method is currently experimental and under development!

Usage

```
openCytoscape(x, title, ...)
```

Arguments

x	A sparsebnFit object or other graph object.
title	A character string, this is the name you will see on the Cytoscape network window. Multiple windows with the same name are not permitted. See createNetworkFromGraph for more details.
...	Other arguments to createNetworkFromGraph.

Details

Displays the selected graph in the Cytoscape application. Note that this requires that Cytoscape is installed on the user's system, and that the RCy3 package is installed and properly configured. Cytoscape can be downloaded at http://www.cytoscape.org/.

permute.nodes *Permute the order of nodes in a graph*

Description

Randomize the order of the nodes in a graph.

Usage

```
permute.nodes(x, perm = NULL)
```

Arguments

x	Graph as edgeList object.
perm	Permutation to use.

Details

Useful for obfuscating the topological sort in a DAG, which is often the default output of methods that generate a random DAG. Output is graph isomorphic to input.

Value

Permuted graph as edgeList object.

pick_family *Utility functions*

Description

Various utility functions for packages in the sparsebn family

Usage

```
pick_family(x)

reIndexC(x)

reIndexR(x)

default_max_iters(numnode)

default_alpha()

check_if_matrix(m)
```

```
check_if_data_matrix(df)

check_if_complete_data(df)

check_if_numeric_data(df)

check_null(x)

check_na(x)

count_nas(df)

list_classes(li)

auto_generate_levels(df)

auto_count_levels(df)

check_list_class(li, check.class)

check_list_numeric(li)

check_list_names(li, check.names)

col_classes(X)

capitalize(string)

recode_levels(x)

convert_factor_to_discrete(x)

cor_vector_ivn(data, ivn = NULL)

pmatch_numeric(x, table, tol = 0.1)

zero_threshold()
```

Arguments

x	a compatible object.
numnode	integer number of nodes.
m	a matrix.
df	a data.frame.
li	a list.
check.class	character class name to compare against.

check.names	character names to compare against.
X	a matrix.
string	a character string.
data	a data.frame.
ivn	list of interventions (see sparsebnData).
table	table of values to compare against.
tol	maximum tolerance used for matching.

---

plot.edgeList *Plot a fitted Bayesian network object*

Description

Plots the graph object associated with the output of a learning algorithm.

Usage

```
## S3 method for class 'edgeList'
plot(x, ...)
```

Arguments

x	fitted object to plot.
...	(optional) additional arguments to plotting mechanism.

Details

plot.sparsebnFit uses some default settings to make large graphs easier to interpret, but these settings can be over-ridden.

See Also

setPlotPackage, getPlotPackage

random.dag *Generate random DAGs*

Description

Generate a random DAG with fixed number of edges.

Usage

```
random.dag(nnode, nedge, FUN = NULL, permute = TRUE)
```

Arguments

nnode	Number of nodes in the DAG.
nedge	Number of edges in the DAG.
FUN	Optional function to be used as a random number generator.
permute	If TRUE, order of nodes will be randomly permuted. If FALSE, output will be ordered according to its topological sort, i.e. with a lower-triangular adjacency matrix.

Details

FUN can be any function whose first argument is called n. This allows for both random and deterministic outputs.

Value

An (weighted) adjacency matrix.

random.graph *Generate random DAGs*

Description

Generate a random graph with fixed number of edges.

Usage

```
random.graph(nnode, nedge, acyclic = TRUE, loops = FALSE,
  permute = TRUE)
```

Arguments

nnode	Number of nodes in the graph.
nedge	Number of edges in the graph.
acyclic	If TRUE, output will be an acyclic graph.
loops	If TRUE, output may include self-loops.
permute	If TRUE, order of nodes will be randomly permuted. If FALSE, output will be ordered according to its topological sort, i.e. with a lower-triangular adjacency matrix.

Value

An edgeList object containing a list of parents for each node.

---

random.spd *Generate a random positive definite matrix*

Description

Generate a random positive definite matrix

Usage

```
random.spd(nnode, eigenvalues = NULL, num.ortho = 10)
```

Arguments

nnode	Number of nodes in the matrix.
eigenvalues	Vector of eigenvalues desired in output. If this has fewer than nnode values, the remainder are filled in as zero.
num.ortho	Number of random Householder reflections to compose.

---

resetGraphPackage *Change data structure for representing graphs internally*

Description

Changes the output of the main algorithms to be compatible with other packages in the R ecosystem.

Usage

```
resetGraphPackage(coerce = TRUE)

setGraphPackage(pkg, matchPlot = TRUE, coerce = FALSE)

getGraphPackage()
```

Arguments

coerce	If TRUE, then all sparsebnFit and sparsebnPath objects in the global environment will be coerced to be compatible with the selected package. This will overwrite your existing data.
pkg	The desired package; default value is NULL corresponding to edgeList. Possible values are "sparsebn", "igraph", "graph", "bnlearn", and "network".
matchPlot	Force the underlying plotting mechanism to match the selected package (see setPlotPackage).

Details

sparsebn is compatible with four different data structures for representing graphs: edgeList (default), graphNEL-class (from the graph package), igraph (from the igraph package), and network (from network-package). edgeList is provided by default in sparsebn, however, the other three options require that extra packages are installed.

Functions

- resetGraphPackage: Reset all data to default edgeList format and set graph package back to default "sparsebn".
- getGraphPackage: Returns the current choice of graph package (NULL corresponds to no selection)

See Also

setPlotPackage, getPlotPackage

---

select *Select solutions from a solution path*

Description

Choose solutions from a solution path based on number of edges, value of regularization parameter lambda, or index.

Usage

```
select(x, edges, lambda, index)
```

Arguments

x	a sparsebnPath object.
edges	number of edges to search for.
lambda	value of regularization parameter to search for.
index	integer index to select.

Details

For edges (resp. lambda), the solution with the closest number of edges (resp. regularization parameter) is returned. If there is no match within a tolerance of 0.1 for lambda, nothing is returned. Fuzzy matching is not used for when selecting by index.

If there is more than one match (for example, by number of edges), then the first such estimate is returned. Note that select(x, index = j) is equivalent to (but slightly slower than) x[[j]].

---

select.parameter *Tuning parameter selection*

Description

Choose the best DAG model according to the criterion described in Fu and Zhou (2013) (Section 3.4).

Usage

```
select.parameter(x, data, type = "profile", alpha = 0.1)
```

Arguments

x	sparsebnPath object.
data	sparsebnData containing the original data.
type	either "profile" or "full", default is profile.
alpha	tuning parameter for selection between 0.05 and 0.1, default is 0.5 (see equation (11) in Fu and Zhou (2013)).

Details

A sparsebnPath objects represents a *solution path* which depends on the regularization parameter lambda. Model selection is usually based on an estimated prediction error, and commonly used model selection methods include the Bayesian information criterion (BIC) and cross-validation (CV) among others. It is well-known that these criteria tend to produce overly complex models in practice, so instead we employ an empirical model selection criterion that works well in practice. As lambda is decreased and thus the model complexity increases, the log-likelihood of the estimated graph will increase. An increase in model complexity, which is represented by an increase in the total number of predicted edges, is desirable only if there is a substantial increase in the log-likelihood. In order to select an optimal parameter, this method computes successive difference ratios between the increase in log-likelihood and the increase in number of edges and balances these quantities appropriately. For specific details, please see Section 3.4 in Fu and Zhou (2013).

setPlotPackage *Change default plotting mechanism*

Description

Changes the default plotting mechanism used by sparsebn to plot output and fitted objects.

Usage

```
setPlotPackage(pkg)

getPlotPackage()
```

Arguments

pkg	The desired package; default value is igraph.

Details

For plotting, sparsebn can use one of three packages: graph (see also Rgraphviz), igraph (see plot.igraph), and network-package (see plot.network). Note that plotting requires that (at least one of) these extra packages are installed.

Functions

- getPlotPackage: Returns the current choice of plotting mechanism

See Also

setGraphPackage, getGraphPackage

show.parents *Inspect subgraph*

Description

Print out the edge list corresponding to a subset of nodes in a graph. Useful for inspecting particular nodes of interest in a large graph. Out is indexed by children, with the parents of each node listed to the right of each child.

Usage

```
show.parents(x, nodes, nchar = 4)
```

Arguments

x	sparsebnFit object.
nodes	character vector containing names of nodes to show.
nchar	integer indicating how many characters of each parent to show in printed output. Use this to control how the output appears on screen, larger numbers allow for longer node names but may present formatting issues for large graphs. Defaults to 4.

Details

Uses partial matching, duplicates are OK and will be duplicated in output.

---

sparse *sparse class*

Description

Low-level representation of sparse matrices.

Usage

```
sparse(x, ...)

is.sparse(x)
```

Arguments

x	Various R objects.
...	(optional) additional arguments.

Details

An alternative data structure for storing sparse matrices in R using the (row, column, value) format. Internally it is stored as a list with three components, each vectors, that contain the rows / columns / values of the nonzero elements.

sparsebn-messages *Messages*

Description

Warning and error messages for use in the sparsebn family

Usage

```
input_not_sparsebnData(data)

alg_input_data_frame()

has_missing_values(count)

invalid_pkg_specification()

pkg_not_installed(pkg)

global_coerce_warning(pkg)

feature_not_supported(feature)

invalid_class(actual, expected)

dag_summary(nnode, nedge)

empty_dag_summary(nnode)

data_not_numeric(indices)

invalid_type_input(types)
```

Arguments

data	data object.
count	number of missing values.
pkg	package name.
feature	feature name.
actual	class input by user.
expected	class input expected by function.
nnode	number of nodes in a DAG.
nedge	number of edges in a DAG.
indices	invalid indices
types	valid input types

sparsebnData *sparsebnData class*

Description

This class stores data that may contain interventions on some or all of the observations. It also allows for the degenerate case with no interventions, i.e. purely observational data.

Usage

```
sparsebnData(x, ...)

is.sparsebnData(x)

## S3 method for class 'data.frame'
sparsebnData(x, type, levels = NULL, ivn = NULL,
  ...)

## S3 method for class 'matrix'
sparsebnData(x, type, levels = NULL, ivn = NULL, ...)

## S3 method for class 'sparsebnData'
print(x, n = 5L, ...)

## S3 method for class 'sparsebnData'
summary(object, n = 5L, ...)

## S3 method for class 'sparsebnData'
plot(x, ...)
```

Arguments

x	a data.frame or matrix object.
...	(optional) additional arguments.
type	either 'discrete' or 'continuous'.
levels	(optional) list of levels for each node. If omitted, levels will be automatically detected from unique.
ivn	(optional) list of interventions for each observation. If omitted, data is assumed to be purely observational.
n	(optional) number of rows from data matrix to print.
object	an object of type sparsebnData

Details

The structure of a sparsebnData object is very simple: It contains a data.frame object, a type identifier (i.e. discrete or continuous), a list of factor levels, and a list of interventions.

- The levels list should be the same size as the number of nodes and consist of names of the different levels for each node. Each level should be coded to be from 0...k-1 where k is the number of levels for a particular variable (see below for more).
- The ivn list should be the same size as the number of rows in the dataset, and each component indicates which column(s) in the dataset is (are) under intervention. If an observation has no interventions, then the corresponding component is NULL. Thus, if the data is purely observational, this list should contain only NULL values.

Presently, only levels coded as 0,1,...,k-1 are supported (k = the number of levels for a variable). Future releases are planned to support more general factor levels. The level 0 corresponds to the baseline level or measurement.

Also inherits from list.

Slots

data (data.frame) Dataset.

type (character) Type of data: Either "continuous", "discrete", or "mixed".

levels (list) List of levels for each column in data.

ivn (list) List of columns under intervention for each row in data.

Methods

print num.samples is.obs count.levels count.interventions as.data.frame

Examples

```
### Generate a random continuous dataset
mat <- matrix(rnorm(1000), nrow = 20)
dat <- sparsebnData(mat, type = "continuous") # purely observational data with continuous variables

### Discrete data
mat <- rbind(c(0,2,0),
             c(1,1,0),
             c(1,0,3),
             c(0,1,0))
dat.levels <- list(c(0,1), c(0,1,2), c(0,1,2,3))
dat <- sparsebnData(mat,
                    type = "discrete",
                 levels = dat.levels) # purely observational data with discrete variables

dat.ivn <- list(c(1),   # first observation was intervened at node 1
                c(1),   # second observation was intervened at node 1
                c(2,3), # third observation was intervened at nodes 2 and 3
                c(1,3)) # fourth observation was intervened at nodes 1 and 3
```

```
dat <- sparsebnData(mat,
                    type = "discrete",
                    levels = dat.levels,
                    ivn = dat.ivn) # specify intervention rows
```

`sparsebnFit` *sparsebnFit class*

Description

Main class for representing DAG estimates. Represents a single DAG estimate in a solution path.

Usage

```
sparsebnFit(x)

is.sparsebnFit(x)

## S3 method for class 'sparsebnFit'
print(x, maxsize = 20, ...)

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

## S3 method for class 'sparsebnFit'
plot(x, ...)
```

Arguments

`x`	A `list` or an object of type `sparsebnFit`. Should only be used internally.
`maxsize`	If the number of nodes in a graph is $\leq$ `maxsize`, then the entire graph is printed to screen, otherwise a short summary is displayed instead.
`...`	(optional) additional arguments.
`object`	an object of type `sparsebnFit`

Details

This is the main class for storing and manipulating the output of `estimate.dag`. The main slot of interest is `edges`, which stores the graph as an `edgeList` object. If desired, this slot can be changed to hold a `graphNEL`, `igraph`, or `network` object if desired (see `setGraphPackage`). For anything beyond simply inspecting the graph, it is recommended to use one of these packages.

Since `edgeLists` do not contain information on the node names, the second slot `nodes` stores this information. The indices in `edges` are in one-to-one correspondence with the names in the `nodes` vector. The `lambda` slot stores the regularization parameter used to estimate the graph.

Other slots include nedge, for the number of edges; pp, for p = number of nodes; nn, for n = number of samples, and time, for the time in seconds needed to estimate this graph. Note that these slots are mainly for internal use, and in particular it is best to query the number of nodes via num.nodes, the number of edges via num.edges, and the number of samples via num.samples.

By default, only small graphs are printed, but this behaviour can be overridden via the maxsize argument to print. To view a list of parents for a specific subset of nodes, use show.parents.

Generally speaking, it should not be necessary to construct a sparsebnFit object manually. Furthermore, these estimates should always be wrapped up in a sparsebnPath object, but can be handled separately if desired (be careful!).

Slots

edges (edgeList) Edge list of estimated DAG (see edgeList).

nodes (character) Vector of node names.

lambda (numeric) Value of lambda for this estimate.

nedge (integer) Number of edges in this estimate.

pp (integer) Number of nodes.

nn (integer) Number of observations this estimate was based on.

time (numeric) Time in seconds to generate this estimate.

Methods

get.adjacency.matrix, num.nodes, num.edges, num.samples, show.parents

Examples

```
## Not run:
### Learn the cytometry network
library(sparsebn)
data(cytometryContinuous) # from the sparsebn package
cyto.data <- sparsebnData(cytometryContinuous[["data"]], type = "continuous")
cyto.learn <- estimate.dag(cyto.data)

### Inspect the output
class(cyto.learn[[1]])
print(cyto.learn[[2]])
show.parents(cyto.learn[[1]], c("raf", "mek", "plc"))

### Manipulate a particular graph
cyto.fit <- cyto.learn[[7]]
num.nodes(cyto.fit)
num.edges(cyto.fit)
show.parents(cyto.fit, c("raf", "mek", "plc"))
plot(cyto.fit)

### Use graph package instead of edgeLists
setGraphPackage("graph", coerce = TRUE) # set sparsebn to use graph package
cyto.edges <- cyto.fit$edges
```

```
degree(cyto.edges)       # only available with graph package
isConnected(cyto.edges)  # only available with graph package

## End(Not run)
```

`sparsebnPath` *sparsebnPath class*

Description

Convenience wrapper class for solution paths of DAG learning algorithms: This class represents an entire solution path of an algorithm. Its components are of type `sparsebnFit`. Also inherits from `list`.

Usage

```
sparsebnPath(x)

is.sparsebnPath(x)

## S3 method for class 'sparsebnPath'
print(x, verbose = FALSE, ...)

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

## S3 method for class 'sparsebnPath'
plot(x, labels = FALSE, ...)
```

Arguments

`x`	A `list` or an object of type `sparsebnPath`. Should only be used internally.
`verbose`	If `TRUE`, then each estimate in the solution path is printed separately. Do not use for large graphs or large solution paths. (default = `FALSE`)
`...`	(optional) additional arguments.
`object`	an object of type `sparsebnPath`
`labels`	`TRUE` or `FALSE`. Whether or not to print out labels with summary information for each plot in the solution path.

Details

Each value of lambda in the (discrete) solution path corresponds to a single DAG estimate (see Aragam and Zhou (2015) for details). Internally, this estimate is represented by a `sparsebnFit` object. The full solution path is then represented as a `list` of `sparsebnFit` objects: This class is essentially a wrapper for this list.

Most methods for sparsebnPath objects simply apply lapply to the object in question. The exceptions to this rule apply when the output will always be the same for every component; e.g. num.nodes and num.samples.

Methods

get.adjacency.matrix, get.lambdas, num.nodes, num.edges, num.samples

Examples

```
## Not run:
### Learn the cytometry network
library(sparsebn)
data(cytometryContinuous) # from the sparsebn package
cyto.data <- sparsebnData(cytometryContinuous[["data"]], type = "continuous")
cyto.learn <- estimate.dag(cyto.data)

### Inspect the output
class(cyto.learn)
print(cyto.learn)
plot(cyto.learn)

## End(Not run)
```

---

sparsebnUtils *sparsebnUtils: Utilities for the sparsebn package.*

Description

A set of tools for representing and estimating sparse Bayesian networks from continuous and discrete data.

Details

This package provides various S3 classes for making it easy to estimate graphical models from data:

- sparsebnData for managing experimental data with interventions.
- sparsebnFit for representing the output of a DAG learning algorithm.
- sparsebnPath for representing a solution path of estimates.

The package also provides methods for manipulating these objects and for estimating parameters in graphical models:

- estimate.parameters for directed graphs.
- get.precision for undirected graphs.
- get.covariance for covariance matrices.

Internally, all graph objects may be stored as edgeLists (default), or using graphNEL, igraph, bnlearn, or network objects.

---

specify.prior *Build a black list based on prior knowledge*

Description

Utility for specifying known root and leaf nodes in a network, to be used in conjunction with the blacklist argument of network estimation methods.

Usage

```
specify.prior(roots = NULL, leaves = NULL, nodes, indices = FALSE)
```

Arguments

roots	Vector of root nodes. May be character or integer.
leaves	Vector of leaf nodes. May be character or integer.
nodes	Full vector of node names of the entire network. Both roots and leaves must be a subset of this vector.
indices	Logical: Return indices or character names?

Details

Builds an (m+k)x2 matrix, where m is the number of user-specified root nodes and k is the number of user-specified leaf nodes.

- A *root* node is any node without any parents, i.e. with no incoming edges.
- A *leaf* node is any node without any children, i.e. with no outgoing edges.

---

to_bn *Conversion between graph types*

Description

These methods convert graph objects (e.g. edgeList) and objects containing graph data (e.g. sparsebnFit, sparsebnPath) to other formats including igraph, graphNEL, network, and bn-class.

Only graph objects are modified with these methods. For example, if the input is either sparsebnFit or sparsebnPath, the output will still be a sparsebnFit or sparsebnPath object. Only the edges slots will be converted to a different graph type. This will be the case for the default output from estimate.dag, so that metadata from the learning phase is not lost during conversion. If, on the other hand, the input is already an edgeList, then the output will directly be a graph object.

Usage

```
to_bn(x)

to_graphNEL(x)

to_igraph(x)

to_network(x)
```

Arguments

x	An object of type sparsebnPath, sparsebnFit, edgeList, igraph, graphNEL, network, or bn-class.

Details

to_igraph converts sparsebn objects to igraph-compatible objects.

to_graph converts sparsebn objects to graphNEL-compatible objects.

to_network converts sparsebn objects to network-compatible objects.

to_bn converts sparsebn objects to bn-class-compatible objects.

Examples

```
## Not run:
### Learn the cytometry network
library(sparsebn)
data(cytometryContinuous)
cyto.data <- sparsebnData(cytometryContinuous[["data"]],
                          type = "continuous",
                          ivn = cytometryContinuous[["ivn"]])
cyto.learn <- estimate.dag(data = cyto.data)

### The output is a sparsebnPath object, which is a list of sparsebnFit objects
class(cyto.learn)
class(cyto.learn[[1]])

### Convert to igraph
cyto.igraph <- to_igraph(cyto.learn)
class(cyto.igraph)       # not an igraph object!
class(cyto.igraph[[1]]$edges) # the graph data in the 'edges' slot is converted to igraph
gr <- cyto.igraph[[1]]$edges

### Different behaviour when input is already an edgeList
edgeL <- cyto.learn[[1]]$edges
gr <- to_igraph(edgeL) # input is edgeList, not sparsebnFit or sparsebnPath
class(gr)              # igraph object

## End(Not run)
```

`to_edgeList` *Conversion to edgeList object*

Description

`to_edgeList` converts an object to an `edgeList` object. Works on both fitted objects and graphs themselves. In the first case, every underlying 'edges' component is converted to `edgeList`. In the second, the conversion applies directly to the object.

Usage

```
to_edgeList(x)

## S3 method for class 'sparsebnFit'
to_edgeList(x)
```

Arguments

x	An object of type `sparsebnPath`, `sparsebnFit`, `graphNEL-class`, `igraph`, or `network`.

Methods (by class)

- `sparsebnFit`: description

Index