Package: bnpa (via r-universe)

October 5, 2024

Type Package

Title Bayesian Networks & Path Analysis

Version 0.3.0

Imports bnlearn, fastDummies, lavaan, Rgraphviz, semPlot, xlsx

Author Elias Carvalho, Joao R N Vissoci, Luciano Andrade, Wagner Machado, Emerson P Cabrera, Julio C Nievola

Maintainer Elias Carvalho <ecacarva@gmail.com>

Description This project aims to enable the method of Path Analysis to infer causalities from data. For this we propose a hybrid approach, which uses Bayesian network structure learning algorithms from data to create the input file for creation of a PA model. The process is performed in a semi-automatic way by our intermediate algorithm, allowing novice researchers to create and evaluate their own PA models from a data set. The references used for this project are: Koller, D., & Friedman, N. (2009). Probabilistic graphical models: principles and techniques. MIT press. <doi:10.1017/S0269888910000275>. Nagarajan, R., Scutari, M., & Lèbre, S. (2013). Bayesian networks in r. Springer, 122, 125-127. Scutari, M., & Denis, J. B. <doi:10.1007/978-1-4614-6446-4>. Scutari M (2010). Bayesian networks: with examples in R. Chapman and Hall/CRC. <doi:10.1201/b17065>. Rosseel, Y. (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1 - 36. <doi:10.18637/jss.v048.i02>.

URL https://sites.google.com/site/bnparp/.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-08-01 23:20:02 UTC

Contents

`boot.strap.bn` *Executes a bootstrap during the learning of a BN structure*

Description

This function receives a list of parameters, executes the bootstrap process and learn the Bayesian Network (BN) from the data set, then executes the process of model averaging to extract the final BN structure and print it.

Usage

```
boot.strap.bn(bn.algorithm, bn.score.test, data.to.work, black.list,
  white.list, nreplicates = 1000, type.of.algorithm, outcome.var)
```

Arguments

`bn.algorithm`	is a list of algorithms to learn the BN structure.
`bn.score.test`	is list of conditional independence tests and the network scores to be used.
`data.to.work`	is a data from which the BN structure will be learned.

black.list	is a list of forbiden connections of BN structure to be created.
white.list	is a list of mandatory connections of BN structure to be created.
nreplicates	is the number of replications to be done in the bootstrap process.
type.of.algorithm	is the type of algorithm to learn the BN sctructure, it would be constrained or score based.
outcome.var	is the variable to be used as outcome (dependent) and be highlighted in the BN.

Value

The final BN structure learned.

Author(s)

Elias Carvalho

References

Claeskens N, Hjort N (2009) Model selection and model avaraging. Cambridge University Press, Cambridge, England.

Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT Press, Cambridge.

Scutari M (2017). Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package. Journal of Statistical Software, 77(2), 1-20.

Examples

```
## Not run:
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQualiN)
# Start the cluster
cl <- bnpa::create.cluster()
# Set the number of replications
nreplicates=1000
# Set the algorithm to be used
bn.algorithm="hc"
# Executes a parallel bootstrap process
data.bn.boot.strap=bnlearn::boot.strength(data = dataQualiN, R = nreplicates, algorithm =
bn.algorithm, cluster=cl, algorithm.args=list(score="bic"), cpdag = FALSE)
# Release the cluster
parallel::stopCluster(cl)
```

```
head(data.bn.boot.strap)

## End(Not run)
```

`check.algorithms` *Verifies the BN learning algorithms*

Description

This function receives a list of algorithms of bnlearn package and check if it would be used in bnpa package.

Usage

```
check.algorithms(bn.learn.algorithms)
```

Arguments

`bn.learn.algorithms`
: is a list of algorithms (present in bnlearn package) to be used in BN structure learning pocess in bnpa.

Author(s)

Elias Carvalho

References

Scutari M (2017). Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package. Journal of Statistical Software, 77(2), 1-20.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("~/your working directory")
# Load packages
library(bnpa)
# Set what BN learning algorithms will be used
bn.learn.algorithms <- c("gs", "hc")
# Check these algorithms
check.algorithms(bn.learn.algorithms)
```

check.dichotomic.one.var *Verify if one specific variable of a data set is dichotomic*

Description

This function receives a data set and the name of a specific variable and verify if it is dichotomic or not. If 'yes' then the function return TRUE.

Usage

```
check.dichotomic.one.var(data.to.work, variable.name)
```

Arguments

data.to.work	is a data set containing the variables to be checked.
variable.name	is the name of a variable to be checked.

Value

TRUE or FALSE

Author(s)

Elias Carvalho

References

HAYES, A F; PREACHER, K J. Statistical mediation analysis with a multicategorical independent variable. British Journal of Mathematical and Statistical Psychology, v. 67, n. 3, p. 451-470, 2014.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQuantC)
head(dataQuantC)
# Show the structure of data set
str(dataQuantC)
# Set variable name
variable.name = "A"
# data set has not dichotomic variables and function will return FALSE
check.dichotomic.one.var(dataQuantC, variable.name)
```

```
# Adding dichotomic data to dataQuantC
dataQuantC$Z <- round(runif(500, min=0, max=1),0)
# Show the new structure of data set
str(dataQuantC)
# Set variable name
variable.name = "Z"
# Now data set has dichotomic variables and function will return TRUE
check.dichotomic.one.var(dataQuantC, variable.name)
```

`check.levels.one.variable` *Check the levels of a categorical variable*

Description

This function receives a data set and a variable name, check the type of variable to be sure it is categorical (factor) and then count the number of levels it has.

Usage

```
check.levels.one.variable(data.to.work, variable.name)
```

Arguments

`data.to.work`	is a data set with variable.
`variable.name`	is the name of variable to be checked.

Author(s)

Elias Carvalho

References

GUJARATI, Damodar N. Basic econometrics. Tata McGraw-Hill Education, 2009.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQualiN)
head(dataQualiN)
# Adding random data to dataQualiN, function will return TRUE
dataQualiN$Z <- round(runif(500, min=0, max=1000),2)
```

```
# Converting the numeric variable into factor
dataQualiN$Z <- factor(dataQualiN$Z)
# Set the variable name to a non categorical one
variable.name = "Z"
# Count the number o levels of a specific variable
number.of.levels <- check.levels.one.variable(dataQualiN, variable.name)
number.of.levels
# Set the variable name to a categorical variable
variable.name = "A"
# Count the number o levels of a specific variable
number.of.levels <- check.levels.one.variable(dataQualiN, variable.name)
number.of.levels
```

---

check.na *Verify variables with NA*

Description

This function receives a data set and calculates the number of NAs to each variable, then calculates the percentual of existing NAs and inform the variables, number/percent of NAs.

Usage

```
check.na(data.to.work)
```

Arguments

data.to.work	is a data set containing the variables to check NAs.

Value

the number and percent of NAs.

Author(s)

Elias Carvalho

References

LITTLE, R J A; RUBIN, D B. Statistical analysis with missing data. John Wiley & Sons, 2014.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
```

```
# Use working data sets from package
data(dataQuantC)
head(dataQuantC)
# Adding NAs to dataQuantC # credits for the random NA code for: https://goo.gl/Xj6caY
dataQuantC <- as.data.frame(lapply(dataQuantC, function(cc) cc[ sample(c(TRUE, NA),
                            prob = c(0.85, 0.15), size = length(cc), replace = TRUE) ]))
# Checking the Nas
check.na(dataQuantC)
```

`check.ordered.one.var` *Verify if one specific variable of a data set is an ordered factor*

Description

Receives a data set, the name of a specific variable and verify if it is an ordered factor or not. If 'yes' then the function return TRUE.

Usage

```
check.ordered.one.var(data.to.work, var.name)
```

Arguments

`data.to.work`	is a data set containing the variables to be checked.
`var.name`	is the name of variable to be checked.

Value

TRUE or FALSE

Author(s)

Elias Carvalho

References

HAYES, A F; PREACHER, K J. Statistical mediation analysis with a multicategorical independent variable. British Journal of Mathematical and Statistical Psychology, v. 67, n. 3, p. 451-470, 2014.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
```

```
data(dataQualiN)
head(dataQualiN)
# Transform variable A into ordered factor
dataQualiN$A <- ordered(dataQualiN$A)
# Check variable A and return TRUE
var.name <- "A"
check.ordered.one.var(dataQualiN, var.name)
# Check variable B and return FALSE
var.name <- "B"
check.ordered.one.var(dataQualiN, var.name)
```

`check.ordered.to.pa` *Verifies if there are ordered factor variables to be declared in the pa model building process*

Description

Receives a BN structure and a data set, then verifies if there are ordered variables. In a positive case return TRUE.

Usage

```
check.ordered.to.pa(bn.structure, data.to.work)
```

Arguments

`bn.structure`	is a BN structure learned from data used to identify if the variable is endogenous or exogenous when building the PA model.
`data.to.work`	is a data set containing the variables of the BN.

Value

a data frame with ordered variables.

Author(s)

Elias Carvalho

References

HAYES, A F; PREACHER, K J. Statistical mediation analysis with a multicategorical independent variable. British Journal of Mathematical and Statistical Psychology, v. 67, n. 3, p. 451-470, 2014.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("~/your working directory")
# Load packages
library(bnpa)
# Load the dataset
data(dataQualiN) # Pre-Loaded
# Build the BN structure
bn.structure<-bnlearn::hc(dataQualiN)
# Show the BN structure learned
bnlearn::graphviz.plot(bn.structure)
# Tranforms variables A and B in ordered factor
dataQualiN$A <- as.ordered(dataQualiN$A)
dataQualiN$B <- as.ordered(dataQualiN$B)
# Generates a list with variables to be ordered and exogenous variables
cat.var.to.use.in.pa <- bnpa::check.ordered.to.pa(bn.structure, dataQualiN)
# Show the variables
cat.var.to.use.in.pa
```

---

check.outliers *Indentifies and gives an option to remove outliers*

Description

This function receives a data set, scan all variables e for each one, verifies if there are outliers and ask if we wish to remove them. We can pass a parameter where we set if the function remove it automatically or will ask before.

Usage

```
check.outliers(data.to.work, ask.before)
```

Arguments

data.to.work	is a data set with variables to be checked.
ask.before	control if the process will ask for confirmation or not.

Author(s)

Elias Carvalho

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Load the data set
data(dataQuantC) # Pre-Loaded
# Set a variable to ask before remove outlier or not
ask.before = "Y" # or ask.before = "N"
# Call the procedure to check if there are outliers
dataQuantC <- check.outliers(dataQuantC, ask.before)
```

`check.type.one.var` *Verify the type of one variable*

Description

Receives a specific variable and return a number indicating its type. The variables can be 1 is integer, 2 is numeric, 3 is factor, 8 is character.

Usage

```
check.type.one.var(data.to.work, show.message = 0, variable.name)
```

Arguments

`data.to.work`	is a data set containing the variables to be verified.
`show.message`	is a parameter indicating if the function will or not show a message.
`variable.name`	is the name of variable to be checked.

Value

A variable with the code indicating the type of variable and a message (or not).

Author(s)

Elias Carvalho

References

GUJARATI, Damodar N. Basic econometrics. Tata McGraw-Hill Education, 2009.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQuantC)
head(dataQuantC)
# Adding random data to dataQuantC, function will return TRUE
dataQuantC$Z <- round(runif(500, min=0, max=1000),2)
# Converting the numeric variable into factor
dataQuantC$Z <- factor(dataQuantC$Z)
# Check and return a numeric value correspondig to the variable type
# Set the variable name
variable.name = "A"
# identify the type
check.type.one.var(dataQuantC, show.message=0, variable.name)
# Set the variable name
variable.name = "Z"
# identify the type
check.type.one.var(dataQuantC, show.message=0, variable.name)
```

---

check.types *Verify types of variable*

Description

This function receives a data set as parameter and check each type of variable returning a number indicating the type of variables in the whole data set. The variables can be 1=integer, 2=numeric, 3=factor, 4=integer and numeric, 5=integer and factor, 6=numeric and factor, 7=integer, numeric and factor, 8=character.

Usage

```
check.types(data.to.work, show.message = 0)
```

Arguments

data.to.work	is a data set containing the variables to be verified.
show.message	is a parameter indicating if the function will or not show a message.

Value

A variable with the code indicating the type of variable and a message (or not)

Author(s)

Elias Carvalho

References

GUJARATI, Damodar N. Basic econometrics. Tata McGraw-Hill Education, 2009.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQuantC)
# Show first lines of data set
head(dataQuantC)
# Check and return a numeric value
show.message <- 1
bnpa::check.types(dataQuantC, show.message)
# Adding random data to dataQuantC, function will return TRUE
dataQuantC$Z <- round(runif(500, min=0, max=1000),2)
# Converting the numeric variable into factor
dataQuantC$Z <- factor(dataQuantC$Z)
# Check and return a numeric value correspondig to: 1=integer, 2=numeric, 3=factor, 4=integer and
# numeric, 5=integer and  factor, 6=numeric and factor or 7=integer, numeric and factor.
show.message <- 1
bnpa::check.types(dataQuantC, show.message)
# Supressing the message
show.message <- 0
bnpa::check.types(dataQuantC, show.message)
```

---

check.variables.to.be.ordered

Check if the variables need to be ordered

Description

This function receives a data set and check the level of each factor variable, if they have more than 2 levels the function recommend to check the need to transform it to ordered factor.

Usage

```
check.variables.to.be.ordered(data.to.work)
```

Arguments

`data.to.work` is a data set with variables to check.

Value

TRUE or FALSE if need or not to tranform the variable into ordered factor.

Author(s)

Elias Carvalho

References

HAYES, A F; PREACHER, K J. Statistical mediation analysis with a multicategorical independent variable. British Journal of Mathematical and Statistical Psychology, v. 67, n. 3, p. 451-470, 2014.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQualiN)
# Show first lines of data set
head(dataQualiN)
# Insert categorical variables with more than 2 levels
dataQualiN$test.variable[dataQualiN$A == "yes"] <- "low"
dataQualiN$test.variable[dataQualiN$B == "yes"] <- "medium"
dataQualiN$test.variable[dataQualiN$X == "yes"] <- "high"
# Transform it to factor variable
dataQualiN$test.variable <- as.factor(dataQualiN$test.variable)
# Check the necessity to transform in ordered variables
bnpa::check.variables.to.be.ordered(dataQualiN)
```

---

convert.confusion.matrix

Converts the position of any element of confusion matrix to VP, FP, FN, VN

Description

This function receives a confusion matrix and the matrix values to keep the order VP, FP, FN, VN.

Usage

```
convert.confusion.matrix(confusion.matrix, cm.position)
```

Arguments

confusion.matrix	is the confusion matrix to be converted.
cm.position	is the position of your VP, FP, FN, VN at the confusion matrix.

Value

a new confusion matrix

Author(s)

Elias Carvalho

References

STORY, Michael; CONGALTON, Russell G. Accuracy assessment: a user's perspective. Photogrammetric Engineering and remote sensing, v. 52, n. 3, p. 397-399, 1986.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Creates a confusion matrix
confusion.matrix <-matrix(c(12395, 4, 377, 1), nrow=2, ncol=2, byrow=TRUE)
# Creates a vector with the position of VP, FP, FN, VN
cm.position <- c(4,3,2,1)
# Shows the original confusion matrix
confusion.matrix
# Converts the confusion matrix
confusion.matrix <- convert.confusion.matrix(confusion.matrix, cm.position)
# Shows the converted confusion matrix
confusion.matrix
```

---

create.cluster *Create a Parallel Socket Cluster*

Description

This function counts the number of cores of your computer processor and mount a parallel socket cluster. It always creates N-1 node to the cluster to let 1 core to the other tasks.

Usage

```
create.cluster()
```

Value

an object of class "cluster"

Author(s)

Elias Carvalho

References

R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Examples

```
## Not run:
## Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQualiN)
# Start the cluster
cl <- bnpa::create.cluster()
# Set the number of replications
R=1000
# Set the algorithm to be used
algorithm="hc"
# Executes a parallel bootstrap process
data.bn.boot.strap=boot.strength(data=dataQualiN,R,algorithm,cluster=cl,
                                 algorithm.args=list(score="bic"),cpdag = FALSE)
# Release the cluster
parallel::stopCluster(cl)

## End(Not run)
```

---

create.dummies *Creates dummy variables in the data set and remove master variables*

Description

This function receives a data set and the name of variables to be transformed into dummies. Then it create the dummy variables, transform it into numeric (to work with PA generation) and remove the master variables that originates the dummies.

Usage

```
create.dummies(data.to.work, dummy.vars)
```

Arguments

data.to.work	is a data set containing the variables to tranform.
dummy.vars	are the variables to be transformed.

Value

the new data set

Author(s)

Elias Carvalho

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2),1-36.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQualiN)
# Show the structure before
str(dataQualiN)
# Set possible dummy variables
dummy.vars <- c("A", "B")
# Create dummies
dataQualiN <- bnpa::create.dummies(dataQualiN, dummy.vars)
# Show the structure before
str(dataQualiN)
```

---

dataQualiN *A qualitative data set to test functions*

Description

This is data set with qualitative nominal variables containing 500 obs extracted from ASIA Bayesian Networks of bnlearn package repository.

Usage

```
dataQualiN
```

Format

A data frame with 500 rows and 7 variables:

A a categorical variale

S a categorical variale

T a categorical variale

L a categorical variale

B a categorical variale

E a categorical variale

X a categorical variale

D a categorical variale ...

Source

http://www.bnlearn.com/bnrepository//

`dataQuantC` *A quantiative data set to test functions*

Description

This is data set with quantiative continuous variables containing 500 obs extracted from ASIA Bayesian Networks of bnlearn package repository.

Usage

```
dataQuantC
```

Format

A data frame with 500 rows and 7 variables:

A a numeric variale

B a numeric variale

C a numeric variale

D a numeric variale

E a numeric variale

F a numeric variale

G a numeric variale ...

Source

http://www.bnlearn.com/bnrepository//

`gera.bn.structure` *Learn the Bayesian Network structure from data and build a PA model*

Description

This function receives a data set, a list of parameters to learn the BN structure based on this data set. Then with the BN ready it will build a PA model if required. The process will then save the graphs of BN and PA and PA parameters.

Usage

```
gera.bn.structure(data.to.work, white.list = "", black.list = "",
  nreplicates = 1000, cb.algorithms = c("gs", "iamb", "fast.iamb",
  "inter.iamb", "mmpc", "si.hiton.pc"), sb.algorithms = c("hc", "tabu"),
  cb.tests = "", sb.tests = "", optimized.option = "FALSE",
  outcome.var, build.pa)
```

Arguments

`data.to.work`	is a data from which the BN structure will be learned.
`white.list`	is a list of mandatory connections of BN structure to be created.
`black.list`	is a list of forbiden connections of BN structure to be created.
`nreplicates`	is how many times the boostrap will run.
`cb.algorithms`	the name of constrained-based algorithms.
`sb.algorithms`	the name of score-based algorithms.
`cb.tests`	the name of tests for constrained-based algorithms.
`sb.tests`	the name of network scores for score-based algorithms.
`optimized.option`	a paremeter of bnlearn package to optmize the BN learn structre learning.
`outcome.var`	is the outcome (dependent) variable.
`build.pa`	indicates if the process will bulld a PA model or not.

Author(s)

Elias Carvalho

References

Scutari M (2017). Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package. Journal of Statistical Software, 77(2), 1-20.

Examples

```
## Not run:
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# setwd("To your working directory")
# Load packages
library(bnpa)
# Load Data
data(dataQualiN)
# Set variables to work
nreplicates = 1000
white.list <- NULL
black.list <- "L-T"
cb.algorithms = c("gs")
sb.algorithms = c("hc")
cb.tests = "jt"
sb.tests = "aic"
optimized.option="FALSE"
outcome.var = "E"
build.pa = 0
# Learn the BN from data and save results (data & images)
gera.bn.structure(dataQualiN, white.list, black.list, nreplicates, cb.algorithms,sb.algorithms,
                  cb.tests, sb.tests, optimized.option, outcome.var, build.pa)

## End(Not run)
```

---

gera.pa *Generates a PA model*

Description

This function receives a BN structure learned, the data set and some parameters and build a PA input model string. Then run the PA model using Structural Equation Model functions and export a PA graph and a PA model summary information.

Usage

```
gera.pa(bn.structure, data.to.work, pa.name, pa.imgname, bn.algorithm,
  bn.score.test, outcome.var)
```

Arguments

bn.structure	is a BN structure learned from data.
data.to.work	is a data frame containing the variables of the BN.
pa.name	is a variable to store the name of file to save PA parameters.
pa.imgname	is a variable to store the name of file to save PA graph.

bn.algorithm	is a list of algorithms to learn the BN structure.
bn.score.test	is a list of tests to be used during BN structure learning.
outcome.var	is the outcome variable.

Author(s)

Elias Carvalho

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2),1-36.

Examples

```
## Not run:
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("To your working directory")
# Load packages
library(bnpa)
# Load data sets from package
data(dataQualiN)
# Show first lines
head(dataQualiN)
# Learn BN structure
bn.structure <- bnlearn::hc(dataQualiN)
bnlearn::graphviz.plot(bn.structure)
# Set variables
pa.name<-"docPAHC"
pa.imgname<-"imgPAHC"
bn.algorithm<-"hc"
bn.score.test<-"aic-g"
outcome.var<-"D"
# Generates the PA model from bn structure
gera.pa(bn.structure, dataQualiN, pa.name, pa.imgname, bn.algorithm, bn.score.test, outcome.var)

## End(Not run)
```

---

gera.pa.model *Generates PA input model*

Description

This function is called from 'gera.pa' function. It receives a BN structure and a data set, build a PA input model string based on BN structure and return it.

Usage

```
gera.pa.model(bn.structure, data.to.work)
```

Arguments

bn.structure	is a BN structure learned from data.
data.to.work	is a data set containing the variables of the BN.

Value

the PA input modeo string

Author(s)

Elias Carvalho

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2),1-36.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("To your working directory")
# Load packages
library(bnpa)
library(bnlearn)
# load data sets from package
data(dataQualiN)
# Show first lines
head(dataQualiN)
# Learn BN structure
bn.structure <- hc(dataQualiN)
bnlearn::graphviz.plot(bn.structure)
# Set variables
# Generates the PA model from bn structure
pa.model <- gera.pa.model(bn.structure, dataQualiN)
pa.model
```

mount.wl.bl.list *Mounts a white or black list*

Description

This function receives a simple list with one or more couple of variables and mount a new data frame in "bnlearn" syntax. The final result must return an object similar to the result of bnlearn command "data.frame(from = c('B', 'F'), to = c('F', 'B'))" that is more complex syntax.

Usage

```
mount.wl.bl.list(black_or_white_list)
```

Arguments

black_or_white_list	is a list of couple of variables.

Value

A new data frame with the 'from' and 'to' variables

Author(s)

Elias Carvalho

References

Scutari M (2017). Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package. Journal of Statistical Software, 77(2), 1-20.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("To your working directory")
# Load packages
library(bnpa)
library(bnlearn)
# Load data sets from package
data(dataQuantC)
# Show the first lines of data
head(dataQuantC)
# Learn the BN structure without black and white list
bn.structure <- hc(dataQuantC)
# Split graph panel in 2 columns
```

```
par(mfrow=c(1,2))
# Show the BN structure
bnlearn::graphviz.plot(bn.structure)
# Mounting the black list
black.list <- ("A-C,D-F")
black.list <- mount.wl.bl.list(black.list)
black.list
white.list <- ("A-B,D-G")
white.list <- mount.wl.bl.list(white.list)
white.list
# Learn the BN structure with black and white list
bn.structure <- hc(dataQuantC, whitelist = white.list, blacklist = black.list)
# Show the BN structure
bnlearn::graphviz.plot(bn.structure)
```

`outcome.predictor.var` *Builds a black list of predictor and/or outcome variable*

Description

This function receives a data set, an outcome/predictor variable, the type of variable and a black list. If this variable is classfied as type outcome the function builds a black list from it to all other variable. If it is classified as type predictor builds a list from all other variables to it. You can pass a previously black list and then this function will append a new list in the end of it.

Usage

```
outcome.predictor.var(data.to.work, var.name, type.var, black.list)
```

Arguments

`data.to.work`	is a data set containing the variables to build a list.
`var.name`	is the outcome/predictor variable name.
`type.var`	is a type of variable: <o>utcome or <p>redictor.
`black.list`	is a previous black list, it would be empty or loaded.

Value

a black list with from - to variables

Author(s)

Elias Carvalho

References

KATZ, M H. Multivariable analysis: a primer for readers of medical research. Annals of internal medicine, v. 138, n. 8, p. 644-650, 2003.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("To your working directory")
# Load packages
library(bnpa)
library(bnlearn)
# Load data sets from package
data(dataQuantC)
# Show first lines
head(dataQuantC)
# Create an empty list or fill it before start
black.list <- ""
# Setting the type of var as typical "outcome" what means it will not point to any var
type.var <- "o"
# Setting variable "A" as "outcome" will create a black from this variable to all others
var.name <- "A"
# Creating the black list
black.list <- outcome.predictor.var(dataQuantC, var.name, type.var, black.list)
black.list
# Setting the type of var as typical "predictor" it will not be pointed from any other var
type.var <- "p"
# Setting variable "D" as "predictor" will create a blacklist from all others to it
var.name <- "D"
# Creating the black list
black.list <- outcome.predictor.var(dataQuantC, var.name, type.var, black.list)
black.list
```

---

preprocess.outliers *Extract information of outliers*

Description

This function receives a data set, the variable content and name, analyzes their content and extract outliers information, showing a boxplot and a histogram.

Usage

```
preprocess.outliers(data.to.work, variable.content, variable.name)
```

Arguments

`data.to.work`	is a data frame containing the variables.
`variable.content`	is a variable with all content of variable in the data set.
`variable.name`	is the name of variable to be verified.

Value

a list with number of outliers and the variable content

Author(s)

Elias Carvalho

References

GUJARATI, Damodar N. Basic econometrics. Tata McGraw-Hill Education, 2009.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Load data sets from package
data(dataQuantC)
# Set parameters to function
variable.content <- dataQuantC$A
variable.name <- "A"
# Preprocess information
preprocess.information <- preprocess.outliers(dataQuantC, variable.content, variable.name)
num.outliers <- preprocess.information[[1]]
variable.content <- preprocess.information[[2]]
mean.of.outliers <- preprocess.information[[3]]
```

---

transf.into.ordinal *Transform categorical variables into ordinal*

Description

This function receives a data set with categorical variables, scan all variables and transform it into odered factors.

Usage

```
transf.into.ordinal(data.to.work)
```

Arguments

data.to.work	is a data set where all variables will be transformed into odered factors.

Value

The data set transformed

Author(s)

Elias Carvalho

References

GUJARATI, Damodar N. Basic econometrics. Tata McGraw-Hill Education, 2009.

Examples

```
# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
#Load Data
data(dataQualiN)
# Transform all variables into ordinal
dataQualiN <- bnpa::transf.into.ordinal(dataQualiN)
str(dataQualiN)
```

Index