Package ‘bnpa’

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:

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
R topics documented:

boot.strap.bn ........................................... 2
check.algorithms ....................................... 4
check.dichotomic.one.var ............................... 5
check.levels.one.variable ............................. 6
check.na .................................................. 7
check.ordered.one.var ................................. 8
check.ordered.to.pa .................................... 9
check.outliers .......................................... 10
check.type.one.var ..................................... 11
check.types ............................................. 12
check.variables.to.be.ordered ......................... 13
convert.confusion.matrix ............................... 14
create.cluster .......................................... 15
create.dummies ......................................... 16
dataQualiN ............................................... 17
dataQuantC ............................................... 18
ger. bn. structure ......................................... 19
ger. pa ................................................... 20
ger. pa.model ............................................ 21
mount.wl.bl.list ....................................... 23
outcome.predictor.var .................................. 24
preprocess.outliers ..................................... 25
transf. into. ordinal .................................... 26

Index 28

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 forbidden 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 structure, 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


Examples

```r
## Not run:
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Use working data sets from package
data(dataQualiN)
# Start the cluster
c1 <- 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=c1, algorithm.args=list(score="bic"), cpdag = FALSE)
# Release the cluster
parallel::stopCluster(c1)
```
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 process in bnpa.

Author(s)

Elias Carvalho

References


Examples

# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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

Examples
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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**

```r
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**


**Examples**

```r
# 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 of 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 of 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

```r
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


### Examples

```r
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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**


**Examples**

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

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

check.outliers

Identifies and gives an option to remove outliers

Description

This function receives a data set, scans all variables for each one, verifies if there are outliers and asks if we wish to remove them. We can pass a parameter where we set if the function removes 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
check.type.one.var

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

**Examples**

```r
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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**

```r
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)
check.variables.to.be.ordered

**Author(s)**

Elias Carvalho

**References**


**Examples**

```r
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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**

```r
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 transform the variable into ordered factor.

Author(s)

Elias Carvalho

References


Examples

# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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

Converting 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


Examples

# Clean environment
closeAllConnections()
rm(list=ls())
# Set enviroment
# setwd("to your working directory")
# Load packages
library(bnpa)
# Creates a confusion matrix
c 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
c 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()
create.dummies

Create 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 transform.
- `dummy.vars` are the variables to be transformed.

**Value**

the new data set

**Author(s)**

Elias Carvalho

**References**


**Examples**

```r
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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
dataQuantC

Format

A data frame with 500 rows and 7 variables:

A a categorical variable
S a categorical variable
T a categorical variable
L a categorical variable
B a categorical variable
E a categorical variable
X a categorical variable
D a categorical variable ...

Source

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

dataQuantC

A quantitative data set to test functions

Description

This is data set with quantitative 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 variable
B a numeric variable
C a numeric variable
D a numeric variable
E a numeric variable
F a numeric variable
G a numeric variable ...

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

```r
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 forbidden connections of BN structure to be created.
- **nreplicates**
  - is how many times the bootstrap 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 parameter of bnlearn package to optimize the BN learn structure learning.
- **outcome.var**
  - is the outcome (dependent) variable.
- **build.pa**
  - indicates if the process will build a PA model or not.

Author(s)

Elias Carvalho

References

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.
### gera.pa.model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bn.algorithm</td>
<td>is a list of algorithms to learn the BN structure.</td>
</tr>
<tr>
<td>bn.score.test</td>
<td>is a list of tests to be used during BN structure learning.</td>
</tr>
<tr>
<td>outcome.var</td>
<td>is the outcome variable.</td>
</tr>
</tbody>
</table>

#### Author(s)

Elias Carvalho

#### References


#### Examples

```r
## Not run:
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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
gergera.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


Examples

# Clean environment
closeAllConnections()
rm(list=ls())

# Set environment
# 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


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
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 classified as type outcome the function builds a black list from it to all other variables. 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>outcome or <p>predictor.
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

preprocess.outliers

Examples

# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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.
transf.into.ordinal

Value

a list with number of outliers and the variable content

Author(s)

Elias Carvalho

References


Examples

# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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)
um.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 ordered factors.

Usage

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

Arguments

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

Value

The data set transformed
Author(s)
Elias Carvalho

References

Examples

```r
# Clean environment
closeAllConnections()
rm(list=ls())
# Set environment
# 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

*Topic datasets
  dataQualiN, 17
  dataQuantC, 18

boot strap bn, 2

check algorithms, 4
check dichotomic one var, 5
check levels one variable, 6
check na, 7
check ordered one var, 8
check ordered to pa, 9
check outliers, 10
check type one var, 11
check types, 12
check variables to be ordered, 13
convert confusion matrix, 14
create cluster, 15
create dummies, 16

dataQualiN, 17
dataQuantC, 18

ger a bn structure, 19
ger a pa, 20
ger a pa model, 21

mount wl bl list, 23

outcome predictor var, 24
preprocess outliers, 25
transf into ordinal, 26