Package 'IPToolbox'

May 28, 2018

Type Package

Title IPToolbox: an R package

Version 1.0

Author Louison Fresnais and Nicolas TCHITCHEK

Maintainer Nicolas TCHITCHEK <nicolas.tchitchek@gmail.com>

Description IPToolbox, an R package performing Data integration and prediction on data.

License GPL-3 | file LICENSE

Imports biclust,
gridExtra,
data.table,
diptest,
evtree,
flowCore,
ggdendro,
ggfortify,
ggplot2,
ggRandomForests,
ggrepel,
grDevices,
grid,
gridExtra,
gtable,
gtools,
igraph,
MASS,
methods,
packcircles,
plyr,
randomForestSRC,
reshape2,
survival,
WGCNA,
pROC

LazyData true

Suggests knitr

VignetteBuilder knitr

RoxygenNote 6.0.1

biocViews FlowCytometry, Visualization, StatisticalMethod, Clustering, MultidimensionalScaling, Regression

R topics documented:

`cor.mtest` *A correlation statistical test taking a matrix*

Description

This function will compute the pvalues of a correlation test between variables in a matrix

Usage

```
cor.mtest(mat, ...)
```

Arguments

`mat`	: a matrix of quantitatives values
`...`	: Any usual parameter for the cor.test function from R

Details

Call example : correlation(data, "group", "X.tissus")

Value

return the coefficient of correlation

correlation *Global correlation method*

Description

This function will compute the coefficient of correlation of two variables (given by their names) and according to their type. This function will be automatically called by the modelise function if it detects that there is only one explanation variable.

Usage

```
correlation(dataframe, var1, var2)
```

Arguments

dataframe	: a dataframe from list_type function.
var1	: The name of a variable, between double quotes
var2	: The name of a variable, between double quotes

Details

Call example : correlation(data, "group", "X.tissus")

Value

return the coefficient of correlation

create_correlation_table *A correlation table function*

Usage

```
create_correlation_table(data, cutoff, level = 0.95, na, method,
  computepvalues)
```

Arguments

data	: a dataframe with variables in column and numeric values
cutoff	: a value of absolute difference or similarity that will choose if edges are included in the edgelist or not
level:	the percentage of confidence (0.95 for 95
	\itemna:default is "omit", select the na action
	\itemmethod:default is "pearson", the name of the correlation technique. available: pearson, spearman, kendall, bicor, hoeffding
	\itemcomputepvalues:default is FALSE, if you want the function to compute pvalues set TRUE. Computation will be longer.
	return a edge list table

This function will compute the coefficients of correlation and the associated p-values between all variables in the dataframe
Call example : create_correlation_table(dataframe)

`create_correlation_table_double_df` *A correlation table function*

Usage

```
create_correlation_table_double_df(data, data2 = NULL, cutoff, level = 0.95,
  na, method = "pearson", computepvalues = FALSE, type = "differential")
```

Arguments

`data`	: a dataframe with variables in column and numeric values
`data2`	: a second dataframe with variables in column and numeric values
`cutoff`	: a value of absolute difference or similarity that will choose if edges are included in the edgelist or not
`type`	: default is "differential", in differential mode, edges with a abs(coef1-coef2) > cutoff will be included in the edge list. If type = "common", edges with abs(coef1-coef2) < cutoff will be included.
`level:`	the percentage of confidence (0.95 for 95 \itemna:default is "omit", select the na action \itemmethod:default is "pearson", the name of the correlation technique. available: pearson, spearman, kendall, bicor, hoeffding \itemcomputepvalues:default is FALSE, if you want the function to compute pvalues set TRUE. Computation will be longer. return a edge list table This function will compute the coefficients of correlation and return a differential or common edge list Call example : create_correlation_table_double_df(groupA, groupB, 0.7, 0.95, "omit", "spearman", FALSE, "common")

`create_report` *Report creation for IPToolbox R package*

Usage

```
create_report(dataframe, dependantvar, explanationvar, directory,
  reportname = NULL)
```

Arguments

`dataframe`	: The dataframe used in the model the user wants to report
`dependantvar`	: The name of the dependant variable used in the model the user wants to report.
`explanationvar`	: The list of explanation variable used in the model the user wants to report.
`directory`	: The working directory (where the Rdata file is located) (Currently seeking an alternative to this method)

Details

Called by the model function

Value

An invitation to comile the .tex file. The autocompilation in a pdf is implemented but if the user doesn't have pdftex it will fail.

`hub_bottlenecks` *HUB and Bottlenecks computation function*

Usage

```
hub_bottlenecks(graph, filename.hub, filename.bottlenecks, top = TRUE)
```

Arguments

`graph`	: A igraph object
`filename.hub`	: the name of hubs files whithout extension
`filename.bottlenecks`	: the name of bottlenecks files whitout extention
`top`	: default = true, if top is true compute the top 100 of hubs and bottlenecks (if there is more than 100 hubs and/or 100 bottlenecks)

Value

write tsv files with hubs and bottlenecks

`learning_sample` *Random sampling in a dataframe in order to create learning and validation samples*

Usage

```
learning_sample(dataframe, reset)
```

Arguments

`dataframe`	: a dataframe with raw data : individuals are in rows, variables in columns
`reset`	: TRUE if we want to replace the drawn value, FALSE otherwise

Value

return the learning sample in a dataframe

list_type *List the type of variables in a dataframe*

Description

This function analyse variables types avalaible in the dataframe These types are added to a list The order of this list is the order of variables in the dataframe

Usage

```
list_type(dataframe)
```

Arguments

dataframe	a dataframe with raw data

Value

a object containing a list and the input dataframe

model *Global modelisation method*

Usage

```
model(dataframe, dependantvar, mode = "everyone", level = 0.95,
  explanationvar, switch, response = NULL)
```

Arguments

dataframe	: a dataframe from list_type function.
dependantvar	: the name between quotes of the dependant variable.
mode	: the sample technique to use.
level	: the level of confidence for the prediction ex: 0.95 (equal alpha risk 5 percent).
explanationvar	: a vector with the names of columns refering to explanation variables.

Details

Call example : model(data, "group", "random_test", 0.95, vecteur_explanationvar). mode take "everyone" by default. everyone mean that the predict will be made on every available individual. if mode = random_reset, a learning sample with reset will me made and the predict data will be the validation sample. if mode = random, a learning sample will me made by randomly sampling patients in the dataframe. Predict data will be the validation sample. if mode = a vector with patients id's for the learning sample, these patients will be in the learning sample and the predict data will be the validation sample.

Value

return the model, objects with data for the report creation.

if the dependant variable is binary, then also return roc object

model.linear *linear modelisation method*

Usage

```
model.linear(data.model, data.predict, dependantvar, level = 0.95,
  explanationvar, switch)
```

Arguments

data.predict	: a dataframe with data to predict from learning.sample
dependantvar	: the name between quotes of the dependant variable : values needs to be continuous
level	: the level of confidence for the prediction ex: 0.95 (equal alpha risk 5 percent).
explanationvar	: a vector with the names of columns refering to explanation variables.
switch	: If TRUE and dependantvar
dataframe	: a dataframe from list_type function.

Details

Call example : model.linear(data, data.predict, "group", 0.95, vecteur_explanationvar, FALSE).

Value

return the model, objects with data for the report creation.

if the dependant variable is binary, then also return roc object

model.logistic *Logistic regression function*

Usage

```
model.logistic(data.model, data.predict, dependantvar, level = 0.95,
  explanationvar, switch, response)
```

Arguments

dependantvar	: the name between quotes of the dependant variable : value needs to be binary or categorial
level	: the level of confidence for the prediction ex: 0.95 (equal alpha risk 5 percent).
explanationvar	: a vector with the names of columns refering to explanation variables.
dataframe	: a dataframe from list_type function.
mode	: the sample technique to use.

Details

Call example : model.logistic(data, "group", 0.95, vecteur_explanationvar).

Value

return the model, objects with data for the report creation.

if the dependant variable is binary, then also return roc object

---

`model.multinomial.logistic` *Multinomial Logistic regression function*

Usage

```
model.multinomial.logistic(data.model, data.predict, dependantvar,
  level = 0.95, explanationvar, switch, response)
```

Arguments

`dependantvar`	: the name between quotes of the dependant variable : value needs to be binary or categorial
`level`	: the level of confidence for the prediction ex: 0.95 (equal alpha risk 5 percent).
`explanationvar`	: a vector with the names of columns refering to explanation variables.
`dataframe`	: a dataframe from list_type function.
`mode`	: the sample technique to use.

Details

Call example : model.logistic(data, "group", 0.95, vecteur_explanationvar).

Value

return the model, objects with data for the report creation.

if the dependant variable is binary, then also return roc object

---

`model_selection` *Automatized model selection*

Usage

```
model_selection(dataframe, dependantvar, mode = "everyone", level = 0.95,
  explanationvar, switch, response)
```

Arguments

`dataframe`	: a dataframe from list_type function.
`dependantvar`	: the name between quotes of the dependant variable.
`mode`	: the sample technique to use.
`level`	: the level of confidence for the prediction ex: 0.95 (equal alpha risk 5 percent).
`explanationvar`	: a vector with the names of columns refering to explanation variables.

Details

Call example : model_selection(data, "group", "random_test", 0.95, vecteur_explanationvar, FALSE). mode take "everyone" by default. everyone mean that the predict will be made on every available individual if mode = random_reset, a learning sample with reset will me made and the predict data will be the validation sample. if mode = random, a learning sample will me made by randomly sampling patients in the dataframe. Predict data will be the validation sample. if mode = a vector with patients id's for the learning sample, these patients will be in the learning sample and the predict data will be the validation sample.

Value

return the optimal model (selection based on pvalues)

Number.ROC *Number of ROC curves for a repsonse vector*

Usage

```
Number.ROC(response)
```

Arguments

response:	a vector containg each possible response

Value

return the number of ROC curves (possible unique association in response)

---

plot.coexpression *A co-expression network plotting function*

Description

This function will plot a co-expression network

Usage

```
plot.coexpression(edge.table, directed = TRUE, title,
  matrix.foldchange = NULL, color = "fold-change", directory,
  filename = "random_filename.pdf", vertex.size = 3.5, vertex.label = NA,
  edge.arrow.size = NA)
```

Arguments

edge.table	: a edge table (or edge list) from the function create_correlation_table (or create_correlation_table_double_df)
directed	: Takes TRUE (default) if the graph as to be directed or FALSE if not
title	: the title of the graph
matrix.foldchange	: default is NULL, is coloring technique is fold-change then a matrix of fold-change is needed
color	: default is "fold-change", define the color technique
directory	: set a path where to store the pdf
filename	: default is "random_filename.pdf", set the name of the pdf file
vertex.size	: default is 3.5, set the size of nodes in the co-expression network
vertex.label	: default is NA, set the labels of nodes in the co-expression network
edge.arrow.size	: default is NA, set the arrow size in the co-expression network

Details

Call example : plot.coexpression(myedgetable, "This is the title of my co-expression network", mymatrixoffoldchange, color = "fold-change", "path to a folder", "The name of the file")

Value

return a dataframe

---

plot.correlogram *A correlogram plotting function*

Description

This function will call the correlogram plotting function from corrplot package

Usage

```
plot.correlogram(pdf.name, matrix.coef, matrix.pvalues, level = 0.95,
  method = "number", number.size = 0.35, pvalues.size = 0.3,
  title = pdf.name, mar = c(0, 0, 1, 0), label.size = 0.5)
```

Arguments

pdf.name	: the name of the pdf file (and by default also title of the plot)
matrix.coef	: A square matrix with all coefficients from square_matrix_from_edge_list
matrix.pvalues	: A square matrix with all pvalues from square_matrix_from_edge_list
level	: the value of confidence (0.95 by default)
method	: the corrplot method, number by default
number.size	: a numeric value that will act as a multiplier on the number size
pvalues.size	: a numeric value that will act as a multiplier on the pvalues size
title	: the title of the plot, by default the pdf anme
mar	: margin values. By default 0,0,1,0
label.size	: a numeric value that will act as a multiplier on the label size

Details

Call example : plot.correlogram("mypdfname",matric.coefficients, matrix.pvalues, 0.95, "number")

Value

return a corrplot

Recode.response *Recode categorial variable*

Description

Recode a numeric categorial variable with strings from a response vector

Usage

```
Recode.response(data, col.number, response)
```

Arguments

data:	a dataframe containing variables to recode
col.number:	the numeric position of the column in the datafrale
response:	a vector containg each possible response

Value

return the number of ROC curves (possible unique association in response)

---

square_matrix_from_edge_list *A square matrix creation function*

Description

This function will compute a square matrix from an edgelist (from correlation tables functions)

Usage

```
square_matrix_from_edge_list(data, content = "coefficients")
```

Arguments

data	: a edge list dataframe from create_correlation table function
content	: default is "coefficients", determine what is in the square matrix

Details

Call example : square_matrix_from_edge_list(edge.list, "coefficients")

Value

return a square matrix

validation_sample *Create the validation sample from data not in the learning sample*

Usage

```
validation_sample(learning_sample, dataframe)
```

Arguments

dataframe	: The dataframe used with the Learning_Sample function
Learning_sample	: The Learning dataframe creted with the Learning_Sample function

Value

return the validation dataframe associated with the parameter: learning_sample

Index