Package ‘mechkar’

March 5, 2020

Type Package
Title Useful Tools for Scientific Research
Version 1.14.6
Date 2020-02-27

Description
Utilities that help researchers in various phases of their research work. This package offers a function that automates the exploratory data analysis, a function for the generation of Table 1 (required on many epidemiological papers) which can be exported to Excel. There is also a function for generation of a forestplot with a table and relative risks or odds ratios that can be used for publication. Additionally, there is a function that generates train/test random partitions used for model evaluation that checks for the balance of the partitions.

URL https://github.com/karpatit/mechkar

Depends R (>= 3.3.0)
Imports stats, grDevices, rlang, utils, car, Hmisc, ROSE, pROC, ResourceSelection, InformationValue, gmodels, dplyr, tibble, givitiR, coxphw, ggplot2, writexl, MASS

License GPL-3

NeedsCompilation no
LazyLoad yes

Author Tomas Karpati [aut, cre] (<https://orcid.org/0000-0003-2650-6192>)
Maintainer Tomas Karpati <karpati@it4biotech.com>

Repository CRAN
Date/Publication 2020-03-05 11:10:02 UTC

R topics documented:

age_adjusted ................................................................. 2
exploreData ................................................................. 3
getMissingness ............................................................. 5
getModelCutoffs ........................................................... 6
MeanCI ....................................................................... 7
This function is intended to calculate the age-adjusted rates given a dataset that includes age and an binary outcome. It returns the crude rates with their confidence interval and the age-adjusted rates with its confidence interval.

**Usage**

```r
age_adjusted(dataset, outcome, age, agemin=0, agemax=130, source="who", alpha=0.05)
```

**Arguments**

- `dataset` the name of the dataframe
- `outcome` the name of the outcome variable
- `age` the name of the age variable
- `agemin` the minimum age to be used to calculate the weights (default=0)
- `agemax` the maximum age to be used to calculate the weights (default=130)
- `source` the name of the standard to be used for the weights (see the details section). Default: 'who'
- `alpha` the probability of a type I error. Default: 0.05 for the 95% CI (1-alpha)

**Details**

This function return a list containing nine values (see values section). It includes three types of standards: the European (euro), the American (us) and the World Health Organization (who). The confidence interval (CI) is calculated using the binomial probabilities using the binconf function of the Hmisc package. The default is to calculate the 95% CI (using an alpha of 0.05).
exploreData

Value

This function returns a list containing the following fields:

- outcome - the name of the outcome variable
- population - the total population on the dataset
- events - the number of events for the outcome
- crude - a list with the crude rates. This contains a list we three elements:
  - rate - the value of the crude rate
  - CImin - the value of the lower CI
  - CImax - the value of the higher CI
- weighted - a list with the weighted rates. This contains a list we three elements:
  - rate - the value of the crude rate
  - CImin - the value of the lower CI
  - CImax - the value of the higher CI

Author(s)

Tomas Karpati M.D.

References


Examples

age <- floor(runif(150,min=30,max=65))
outcome <- round(runif(150,min=0,max=1),0)
df <- data.frame(age=age,disease=outcome)
aa <- age_adjusted(df,"disease","age",agemin=30,agemax=65,source="who",alpha=0.0)
exploreData

Description
This function lets the user to have a panoramic view of a dataset. All variables with less than a specified unique values will be treated as a factor. The number of unique values to be used for the determination of factors in numeric variables can be changed with the factorSize argument. The default value for this argument is 10. String variables (including dates) will also be treated as factors. The output of the function is an html page showing four or five columns (in case if a dependent variable is included). The first column show the variable name. The second column a graph with the main distribution of the variable. In cases where the variable is continuous, an histogram will be shown and in cases when the variable is a factor, a bar graph is shown. The third column contains basic descriptive statistics, including the data type, the number of valid and missing values. For continuous variables, the mean and standard deviation, median and interquartile range and the minimum and maximal values are shown. For non-continuous variables, the number of levels and frequencies for the most common levels are shown. The fourth column shows a graph of the individual values and the number of outliers, if they exist. This column may be of help to show undesirable patterns, e.g. may point to problems in data recollection.

Usage
exploreData(data = data, y = NULL, rn = NULL, factorSize = 10, dir = tempdir(), debug = FALSE, ...)  

Arguments

data  
The name of the dataset

y  
The dependent variable. If present, a fifth column showing a plot of each variable and the dependent variable. This may be helpful for early discovery of relationships between a variable and the outcome variable.

rn  
a string vector containing the text that will replace variables defined in argument x. If y is not provided, the original variable name will be used.

factorSize  
The maximal number of unique values on a continuous variable for considering it as a factor.

dir  
The name of the directory to write on the report. If dir is not supplied, a temporal directory will be used

debug  
Normally, the exploreData function will show a progress bar when running. Sometimes the command fails because some anomaly in a variable that was not detected by the function. In such case, setting debug to TRUE will print each variable that is processed. In case of error, the last printed variable must be checked. The default to this parameter is FALSE

... additional arguments

Value
This function creates a folder called report in the main user directory, and html file called report.html in the report directory, and a `fig` folder where all the images will be saved. We decided to use html as the output format, and not other type (e.g. PDF), because we wanted to make available to the user the whole set of graphs that may be of help when writing the preliminary reports.
getMissingness

Author(s)

Tomas Karpati is an Data Researcher at the Clalit Research Institute.

Examples

```r
## --- Getting data stats for a given dataset ---
data(iris)
examineData(data=iris, dir=tempdir())

## --- Getting data stats for a given dataset, defining also a dependent variable ---
examineData('Species', data=iris, dir=tempdir())
```

---

getMissingness

Determine the degree of missings on a dataset

Description

This function calculates the missingness of the supplied dataset. The function returns a list of variables, in descending order, showing their number and percentage of missing values. It also reports the number of rows with complete data. If getRows is set to true, it will also return a list of indexes of the rows containing complete data. This list can be used to retrieve a new dataset containing only complete data or a dataset with all the data having at least one missing value.

Usage

```r
getMissingness(data, getRows=FALSE)
```

Arguments

- `data`: the dataset to be checked. It must be a data.frame, data.table or tibble.
- `getRows`: indicates if a list of row indexes has to be returned. Default is FALSE

Details

The function getMissingness returns a list that include the list of variables and their percentage of missing values. You can retrieve any of the objects in the list by their names.

Value

A list that include the following objects:

- `missingness`: a dataframe containing the variable names and the percentage of missingness
- `msg`: a message string stating the number of rows (and their percent) having complete data
- `rows`: a vector with the indexes of the rows having complete data only
Author(s)

Tomas Karpati M.D.

Examples

```r
### Show only the list of variables with missing values and the number of complete rows.
data <- airquality
data[4:10,3] <- rep(NA,7)
data[1:5,4] <- NA
aa <- getMissingness(data)

### Shows the list of variables with missing values and the number of complete rows
aa <- getMissingness(data,getRows=TRUE)
ind <- aa[[3]]
complete <- data[ind,]
incomplete <- data[-ind,]
```

getModelCutoffs

Function for selection of the prediction cutoffs by each quantile of the prediction

Description

This function is intended to calculate the most important statistics that may help in making decisions on the optimal cutoff of the predictions for our needs.

Usage

```r
getModelCutoffs(pred, obs, div=10)
```

Arguments

- `pred` a vector containing the predicted values of a model
- `obs` a vector containing the values of the outcome variable
- `div` the number of quantiles to calculate. Default=10

Details

This function return a list containing nine values (see values section). It includes three types of standards: the European (euro), the American (us) and the World Health Organization (who). The confidence interval (CI) is calculated using the binomial probabilities using the binconf function of the Hmisc package. The default is to calculate the 95% CI (using an alpha of 0.05).
Value

This function returns a table containing the following statistics:

- TP - the number of true negatives
- FP - the number of false positives
- FN - the number of false negatives
- TN - the number of true positives
- sensitivity
- specificity
- PPV - positive predictive value
- NVP - negative predictive value
- accuracy
- error
- prevalence
- lift
- precision (same as PPV)
- recall (same as sensitivity)
- F1_score (harmonic mean of precision and recall)

Author(s)

Tomas Karpati M.D.

Examples

```r
mod <- glm(am ~ mpg + cyl + hp + wt, data=mtcars, family="binomial")
pred <- as.numeric(predict(mod, newdata = mtcars, type="response"))
tab2 <- getModelCutoffs(pred, mtcars$am)
```

MeanCI

*Calculates the mean and 95% confidence interval of a numeric vector*

Description

This function return a vector containing the mean and 95% confidence interval of a numeric vector.

Usage

```r
MeanCI(x, round=3)
```

Arguments

- `x` a numeric vector
- `round` the number of decimals to be returned. Default is 3
**Value**

This function returns a named numeric vector with the mean and the lower and upper confidence interval.

**Author(s)**

Tomas Karpati M.D.

**Examples**

```r
### calculates the mean and 95% CI of a given vector
x <- c(1,0,1,0,0,0,1,1,0)
mn <- MeanCI(x)
```

---

**modelValidity**

*Return discriminative and calibration measures for predictive models*

**Description**

This function returns common statistical validity metrics used for testing the discriminative power and calibration of a predictive model.

**Usage**

```r
modelValidity(data, model, class, train = FALSE, calib.graph = FALSE)
```

**Arguments**

- **data**: a data frame containing the data to be tested on the model
- **model**: the model to be tested
- **class**: the name (as a string) of the outcome variable
- **train**: a logical value indicating if the data comes from the same dataset used to train the model. Defaults is FALSE
- **calib.graph**: a logical value indicating if the calibration graph has to be generated. Default is FALSE

**Details**

The `modelValidity` function returns a summary table with the validity metrics most commonly used in predictive modeling.
Value

A character matrix containing the following statistical metrics:

- **auc**: the area under the curve of the model
- **cimin**: the lower 95% confidence interval of the auc
- **cimax**: the higher 95% confidence interval of the auc
- **SRME**: the square root mean error
- **precision**: the precision of the model. This is also known in epidemiology as the positive predictive value (PPV)
- **recall**: the recall of the model. Also known in epidemiology as the sensitivity
- **fscore**: the armonic mean of precision and recall (F1-score).
- **NPV**: the negative predictive value
- **D**: the Tjur's discriminative measure
- **TN**: the true negative value
- **mmce**: the mean missclassification error
- **Hosmer_Lemeshow**: the Hosmer-Lemeshow index

Author(s)

Tomas Karpati M.D.

Examples

```r
set.seed(123)
n <- 100
x <- rnorm(n)
xb <- x
pr <- exp(xb)/(1+exp(xb))
y <- 1*(runif(n) < pr)
mod <- glm(y~x, family="binomial")
vt <- modelValidity(data.frame(x=x,y=y),mod,"y")
```

---

**PropCI**

*Calculates the proportion and 95% confidence interval of a binary or dichotomic vector*

Description

This function returns a vector containing the Proportion and 95% confidence interval of a vector. The returning vector will contain the category that was used to calculate the proportion. If the user prefers that the reported category will be that used as reference, then must set second to FALSE.
Usage

```
PropCI(x, round=3, multi=100, ref=2)
```

Arguments

- **x**: a numeric vector
- **round**: the number of decimals to be returned. Default is 3
- **multi**: a number indicating the way the proportion will be reported. The default is 100 for percent. If the proportion required is per 10,000 set multi=10000
- **ref**: determine which category will be used as reference to be reported. The default is 2 so the proportion for the second category will be reported. If the proportion for the first category must be reported, then set ref to 1

Value

This function returns a named numeric vector with the percentage of the highest value of the binary vector and its lower and upper confidence interval.

Author(s)

Tomas Karpati M.D.

Examples

```r
### calculates the proportion and 95% CI of a numeric dichotomic vector
x <- c(1,0,1,0,0,0,1,1,0)
pp <- PropCI(x)

### calculate the proportion and 95% CI of a character dichotomic vector
n <- c("a","a","b","b","a","b","a")
pp <- PropCI(n)
### change the report to the other category
pp <- PropCI(n, ref=1)
```

Description

This function creates a table that complies with the table 1 required on most of the research papers. The function create descriptive statistics for the entire population and, if a categorical variable is defined for strata, then additional descriptive statistics will be added to the table and a p-value will be calculated. Additionally, if desired, the table may be exported to an excel file, were the table could be edited to adapt to the formatting required by the journal it will be published.
Usage

Table1(x=NULL, y=NULL, rn=NULL, data=NULL, miss=3, catmiss=TRUE, formatted=TRUE, categorize=FALSE, factorVars=NULL, maxcat=10, delzero=TRUE, decimals = 1, messages = TRUE,excel=0, excel_file=NULL, debug = FALSE)

Arguments

  x  a string vector containing the name of the variables for which descriptive statistics will be calculated.

  y  the name of the categorical variable that defines the stratification that will be used.

  rn  a string vector containing the text that will replace variables defined in argument x. If y is not provided, the original variable name will be used.

  data  the name of the dataset to be used.

  miss  determines if missingness will be shown in the table for the variables. The possible values are: 0=dont add missing statistics; 1=add missing statistics for continuous (numerical) values only; 2=add missing statistics for categorical (factor) values only;3=add missing statistics for both numerical and categorical values;

  catmiss  On categorical variables, adds a new category (Missing) to the available categories. Default is FALSE. For activation change this to TRUE. If TRUE, the "miss" parameter will not be used for category variables

  formatted  As default, the table output is formatted as text with values that include parenthesis and percentages, e.g. 153 (26.5%). If you are interested that the table return each numeric value as aseparate cell, set this variable to FALSE.

  categorize  If there are categorical variables that are defined as numeric we can force the function to take them as categorical (factor) by changing this to TRUE. Default is FALSE.

  factorVars  If categorize is set to TRUE, a list of variables to be considered as categoricals may be given. In this case, maxcat will not be used and only the variables in the list will be converted to factors.

  maxcat  If we force categorize to be TRUE, maxcat will be used to define the maximum number of unique values permitted for a variable to be considered as categorical. Default is 10.

  delzero  For dichotomic variables, the default behaviour is to delete the rows with the first value (0,"No","Normal",etc.). If you want that all the values are presented change it to FALSE.

  decimals  Determinate the number of decimal places of numerical data. Default is 1

  messages  This switch will show the iterations of the function through the variables. In case you want to suppress those message set it to FALSE.

  excel  indicates if the table will be exported to excel. The default is not to export (excel). For exporting to excel set excel=1.

  excel_file  A string variable defining the name of the excel file. If the directory path is not included in the file name, the file will be saved on the current path directory.
debug

Normally, the Table1 function will show a progress bar when running. Sometimes the command fails because some anomaly in a variable that was not detected by the function. In such case, setting debug to TRUE will print each variable that is processed. In case of error, the last printed variable must be checked. The default to this parameter is FALSE.

Details

The getTable1 function generates a descriptive statistical summary appropriate for publishing in a scientific paper.

Value

The Table1 function returns a data frame object containing the descriptive statistics for each of the variables defined on the x parameters (if provided) or all the variables (if x was not provided). If the parameter y is not defined, only the column for the total population will be returned. If y is defined (and providing that it is categorical with less than six categories), additional columns will be added for each of the categories. If the parameter excel is true and a path/file is defined, an excell file will be generated with the data in the table.

Author(s)

Tomas Karpati M.D.

Examples

```r
### get the table 1 with original variables name
rv <- names(trees)
tab1 <- Table1(x=rv, data=trees)

### get the table 1 with the specified given variables text
rv <- names(swiss)
rn <- rv
rn[6] <- "Infant Mortality"
tab1 <- Table1(x=rv, rn=rn, data=swiss)

### get the table 1 and stratify by one variable
tab1 <- Table1(data=data.frame(HairEyeColor), y="Sex")
```

Table2

| Print a formatted table with the Odds/Hazard ratios from a given model |

Description

This function creates a table with the Odds/Hazard ratios (depending on the model type) with their confidence intervals and p-values.
Usage

Table2(mod, rv=NULL, level = 0.95, decimals = 3)

Arguments

mod    the model name
rv     (Optional) the text to be used for each of the variables to be used. If omitted, the original variable names will be used.
level  the confidence level to be used. Default=0.95
decimals the number of decimals to be used. Default=3

Details

The Table2 function generates a summary table containing the exponent of the model coefficients and the respective confidence intervals and p-values.

Value

This function returns a data frame object with the table of the variables with the coefficients of the model and their p-value.

Author(s)

Tomas Karpati M.D.

Examples

df <- MASS::biopsy[,2:11]
mod <- glm(class ~ ., data=df, family = "binomial")
vn <- c("Intercept","clump thickness","uniformity cell size","uniformity cell shape",
         "marginal adhesion","epithelial cell size","bare nuclei",
         "bland chromatin","normal nucleoli","mitoses")
rn <- data.frame(vars=vn, cats=rep("",10))
tab2 <- Table2(mod=mod, rv = vn)

Table2.forestplot  Generate a paper ready plot with a risk table and a foresplot

Description

This function plots a table with the Odds/Hazard ratios (depending on the model type) with their confidence intervals and p-values and insert in the middle a foresplot.

Usage

Table2.forestplot(mod, nr=NULL)
Arguments

mod the model name
nr (Optional) A data frame containing two columns: the first column with the text to be used for each of the variables, and the second column with the names of the categories for nominal variables. If omitted, the original variable names as appear in the model will be used.

Details

The Table2.foresplot function plots a summary table containing the model’s risks and its respective confidence intervals and p-values. A forestplot graph is inserted in the middle of the table. This function generates a plot that is paper ready. We recommend to save it as PDF file, which let modifying its size without blurring the text.

Value

This function returns a data frame object with the table of the variables with the coefficients of the model and their p-value. Additionally, a graph which includes the table and a forestplot that shows graphically the coefficients is generated.

Author(s)

Tomas Karpati M.D.

Examples

df <- MASS::biopsy[,2:11]
mod <- glm(class ~ ., data=df ,family = "binomial")
vn <- c("Intercept","clump thickness","uniformity cell size","uniformity cell shape",
       "marginal adhesion","epithelial cell size","bare nuclei",
       "bland chromatin","normal nucleoli","mitoses")n <- data.frame(vars=vn,cats=rep("",10))

Table2(mod,rv = vn)
Table2.forestplot(mod)
Table2.forestplot(mod,nr=rn)

train_test (data=NULL)  Generates a training and test dataset

Description

This function generates a training and test datasets by randomly assigning individuals to each dataset.

Usage

train_test(data=NULL,train_name=NULL,test_name=NULL,prop=NULL,seed=123,tableone = FALSE)
ValidityTest

Arguments

data original dataset
train_name a string that defines the name to be assigned to the train dataset object
test_name a string that defines the name to be assigned to the test dataset object
prop the proportion of the training dataset. The value is a fractional number between 0 and 1. The value default value is set to 0.6, indicating that the training dataset will contain 60% of the cases and the test dataset will contain the 40% of the cases.
seed the desired seed. Using a constant seed value allows to obtain the same individuals on each group when running many times (important feature needed for replicability)
tableone a logical value indicating if the Table1 function has to be generated for comparing the train and test division. Default is FALSE

Value

This function creates new variables using the names entered for the train and test partitions. Additionally, it returns the a table (based on the Table1 function) comparing all the available variables by partition. This helps understanding if the partition is balanced.

Author(s)

Tomas Karpati M.D.

Examples

### the following example will generate a train dataset named "train" which
### includes 70% of the records, while generating a test
### dataset called "test" and that includes 30% of the the original dataset.
df <- Theoph
df$Subject <- NULL
train_test(data=df,train_name="train",test_name="test",prop=0.7,seed=2)

---

ValidityTest Print a formatted table with the validity test for the results of a screening or a predictive model

Description

This function creates a table showing the most common statistical validity metrics used for testing the validity of a screening test or a predictive model. The way that the output is formatted is as follows:

############################ Validity Test ############################

Observed
<table>
<thead>
<tr>
<th>Predicted</th>
<th>+ TP</th>
<th>FP</th>
<th>PPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td>b</td>
<td>e (e1-e2)</td>
</tr>
<tr>
<td>- FN</td>
<td></td>
<td>TN</td>
<td>NPV</td>
</tr>
<tr>
<td>c</td>
<td></td>
<td>d</td>
<td>f (f1-f2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Prevalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>g (g1-g2)</td>
<td>h (h1-h2)</td>
<td>i (i1-i2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(FP+FN)</td>
<td>(TP+FP+FN+TN)</td>
<td></td>
</tr>
<tr>
<td>Accuracy:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(TP+TN)</td>
<td>(TP+FP+FN+TN)</td>
<td></td>
</tr>
<tr>
<td>Precision:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

f1-Score*: 2*(Precision*Recall) / (Precision+Recall)

*F1-Score: Harmonic mean of precision and recall.

Usage

```
ValidityTest(a, b, c, d, multi=100, caption = "Validity of the Model/Screening")
```

Arguments

- `a`: the true positive (TP) value
- `b`: the false positive (FP) value
- `c`: the false negative (FN) value
- `d`: the true negative (TN) value
- `multi`: (Optional) The multiplier for the values. The default is 100 for calculating the percentage.
- `caption`: the text to be printed as the title.

Details

The ValidityTest function returns a summary table with the validity metrics most commonly used in epidemiology and in statistical analysis.

Value

A character matrix containing the following statistical metrics:

- **TP**: the true positive value
- **FP**: the false positive value
- **PPV**: the positive predictive value
ValidityTest

FN       the false negative value
TN       the true negative value
NPV      the negative predictive value
FN       the false negative value
TN       the true negative value
NPV      the negative predictive value
Sensitivity the sensitivity of the test
Specificity the true negative value
Prevalence the prevalence of the positive cases in the group
Error    the of incorrectly assigned cases
Accuracy the true negative value
Precision is the same as the PPV
Recall   Other name for the sensitivity of the test
F1-Score  the armonic mean of the precision and the recall

All results are given with their confidence intervals.

Author(s)

Tomas Karpati M.D.

Examples

`tab1 <- ValidityTest(110,20,80,324)`
Index

age_adjusted, 2
exploreData, 3
getMissingness, 5
getModelCutoffs, 6
MeanCI, 7
modelValidity, 8
PropCI, 9
Table1, 10
Table2, 12
Table2.forestplot, 13
train_test, 14
ValidityTest, 15