Package ‘analyzer’

June 30, 2020

Type Package
Title Data Analysis and Automated R Notebook Generation
Version 1.0.1
Maintainer Apurv Priyam <apurvpriyam@gmail.com>
Description Easy data analysis and quality checks which are commonly used in data science. It combines the tabular and graphical visualization for easier usability. This package also creates an R Notebook with detailed data exploration with one function call. The notebook can be made interactive.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports ggplot2, gridExtra, dplyr, grid
RoxygenNote 7.1.0
Suggests data.table, testthat, tidyr, reshape2, rmarkdown, MASS, shiny, nnet, knitr
VignetteBuilder knitr
NeedsCompilation no
Author Apurv Priyam [aut, cre]
Repository CRAN
Date/Publication 2020-06-30 09:20:21 UTC

R topics documented:

anderson.test .......................................................... 2
association ............................................................. 3
CCassociation .......................................................... 5
consoleBoxplot ......................................................... 7
CQassociation .......................................................... 8
Cx ................................................................. 9
CxCy ............................................................... 10
explainer ............................................................ 10
Description

`anderson.test` performs Anderson-Darling test

Usage

```r
anderson.test(x)
```

Arguments

- `x`: a numeric vector. Length must be greater than 7. Missing values are allowed.

Details

Performs the Anderson-Darling test for the composite hypothesis of normality, see e.g. Thode (2002, Sec. 5.1.4).

Value

A list with following elements:

- `statistic`: the value of Anderson-Darling test statistic
- `p.value`: p-value of the test
- `method`: Test name
- `data.name`: Vector name
### association

**Find association between variables**

**Description**

association finds association among all the variables in the data.

**Usage**

```r
association(
  tb,
  categorical = NULL,
  method1 = c("auto", "pearson", "kendall", "spearman"),
  method3 = c("auto", "parametric", "non-parametric"),
  methodMats = NULL,
  use = "everything",
  normality_test_method = c("ks", "anderson", "shapiro"),
  normality_test_pval = 0.05,
  ...
)
```

**Arguments**

- `tb` : tabular data
- `categorical` : a vector specifying the names of categorical (character, factor) columns
- `method1` : method for association between continuous-continuous variables. Values can be "auto", "pearson", "kendall", "spearman". See details for more information.
- `method3` : method for association between continuous-categorical variables. Values can be "auto", "parametric", "non-parametric". See details of `CQassociation` for more information. Parametric does t-test while non-parametric does 'Mann-Whitney' test.
- `methodMats` : This parameter can be used to define the methods for calculating correlation and association at variables pair level. The input is a square data.frame of dimension - number of columns in `tb`. The row names and column names of `methodMats` are the column names of `tb`. The values in the data.frame can be:
  - **between continuous-continuous variables** from parameter `method1` - "auto", "pearson", "kendall", "spearman"
  - **between continuous-categorical variables** from parameter `method3` - "auto", "parametric", "non-parametric"
  - **between categorical-categorical variables** from parameter `method3` - "auto", "parametric", "non-parametric"
between continuous-categorical variables from parameter method3 - "auto", "parametric", "non-parametric"

between categorical-categorical variables can be anything

Default is NULL. In that case the method used for calculating correlation and association will be the inputs from parameters.

This parameter can also take some other values. See example for more details. But its advisable to use like mentioned above.

use an optional character string giving a method for computing association in the presence of missing values. This must be (complete or an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". If use is "everything", NAs will propagate conceptually, i.e., a resulting value will be NA whenever one of its contributing observations is NA. If use is "all.obs", then the presence of missing observations will produce an error. If use is "complete.obs" then missing values are handled by case wise deletion (and if there are no complete cases, that gives an error). "na.or.complete" is the same unless there are no complete cases, that gives NA

normality_test_method method for normality test for a variable. Values can be shapiro for Shapiro-Wilk test or 'anderson' for 'Anderson-Darling' test of normality or ks for 'Kolmogorov-Smirnov'

normality_test_pval significance level for normality tests. Default is 0.05

... other parameters passed to cor, CCassociation, CQassociation and QQassociation

Details

This function calculates association value in three categories -

- between continuous variables (using CCassociation function)
- between categorical variables (using QQassociation function)
- between continuous and categorical variables (using CQassociation function)

For more details, look at the individual documentation of CCassociation, QQassociation, CQassociation

Value

A list of three tables:

- continuous_corr correlation among all the continuous variables
- continuous_pvalue Table containing p-value for the correlation test
- categorical_cramers Cramer’s V value among all the categorical variables
- categorical_pvalue Chi Sq test p-value
- continuous_categorical association value among continuous and categorical variables
- method_used A data.frame showing the method used for all pairs of variables
CCassociation

See Also

CCassociation for Correlation between Continuous variables, QQassociation for Association between Categorical variables, CQassociation for Association between Continuous-Categorical variables

Examples

tb <- mtcars
tb$cyl <- as.factor(tb$cyl)
tb$vs <- as.factor(tb$vs)
out <- association(tb, categorical = c("cyl", "vs"))

# To use the methodMats parameter, create a matrix like this
methodMats <- out$method_used

# the values can be changed as per requirement
# NOTE: in addition to the values from parameters method1 and method3,
# the values in methodMats can also be the values returned by
# association function. But its advisable to use the options from
# method1 and method3 arguments
methodMats["mpg", "disp"] <- methodMats["disp", "mpg"] <- "spearman"
out <- association(tb, categorical = c("cyl", "vs"), methodMats = methodMats)
rm(tb)

CCassociation

Association (Correlation) between Continuous (numeric) Variables

Description

CCassociation finds correlation between all the variables in data with only numeric columns

Usage

CCassociation(
  numtb,
  use = "everything",
  normality_test_method,
  normality_test_pval,
  method1 = c("auto", "pearson", "kendall", "spearman"),
  methodMat1 = NULL,
  methods_used
)

Arguments

numtb a data frame with all the numerical columns. This should have at least two columns
use

an optional character string giving a method for computing association in the presence of missing values. This must be (complete or an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". If use is "everything", NAs will propagate conceptually, i.e., a resulting value will be NA whenever one of its contributing observations is NA. If use is "all.obs", then the presence of missing observations will produce an error. If use is "complete.obs" then missing values are handled by case wise deletion (and if there are no complete cases, that gives an error). "na.or.complete" is the same unless there are no complete cases, that gives NA

normality_test_method

method for normality test for a variable. Values can be shapiro for Shapiro-Wilk test or 'anderson' for 'Anderson-Darling' test of normality or ks for 'Kolmogorov-Smirnov'

normality_test_pval

significance level for normality tests. Default is 0.05

method1

method for association between continuous-continuous variables. values can be "auto", "pearson", "kendall", "spearman". See details for more information.

methodMat1

method dataframe like methodMats from the function association

methods_used

a square data.frame which will store the type of association used between the variables. Dimension will be number of variables * number of variables.

Details

This function calls cor function to calculate the correlation values. The difference is that this doesn’t take method as parameter, instead it decides the methods itself using normality tests. If the variables satisfy the assumption of Pearson correlation, then pearson correlation is calculated. Otherwise Spearman is calculated. To learn more, check the cor

Value

a list of two tables with number of rows and column equal to number of columns in numtb:

r Table containing correlation values

r_pvalue Table containing p-value for the correlation test

See Also

association for association between any type of variables, QQassociation for Association between Categorical variables, QQassociation for Association between Continuous-Categorical variables
**consoleBoxplot**  

**Boxplot on the console**

**Description**

consoleBoxplot prints the boxplot on console.

**Usage**

```r
consoleBoxplot(x)
```

**Arguments**

- `x`: a numeric vector of length at least 3

**Details**

This function is for the numeric vectors. It prints a boxplot in a single line on the console. It automatically adjusts for the width of the console. The input vector must have a length of three, otherwise the function will throw a warning and not print any plot.

In case of any potential outliers (based on 1.5*IQR criteria), this will give a warning. This function is used in the explainer.

**Value**

prints a boxplot on the console which has:

- `|` at start and end means the minimum and maximum value respectively
- `<=*=>` The IQR region
- `*` shows the median
- `...` everything else in between

Gives a warning of potential outliers (if present)

**Examples**

```r
consoleBoxplot(mtcars$mpg)
```
CQassociation  

**Association (Correlation) between Continuous-Categorical Variables**

**Description**

CQassociation finds Association measure between one categorical and one continuous variable.

**Usage**

```r
CQassociation(
  numtb,
  factb,
  method3 = c("auto", "parametric", "non-parametric"),
  use = "everything",
  normality_test_method = c("ks", "anderson", "shapiro"),
  normality_test_pval,
  methodMat3 = NULL,
  methods_used
)
```

**Arguments**

- `numtb` a data frame with all the numerical columns. This should have at least two columns.
- `factb` a data frame with all the categorical columns. This should have at least two columns.
- `method3` method for association between continuous-categorical variables. Values can be "auto", "parametric", "non-parametric". See details for more information. Parametric does t-test while non-parametric does 'Mann-Whitney' test.
- `use` an optional character string giving a method for computing association in the presence of missing values. This must be (complete or an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". If use is "everything", NAs will propagate conceptually, i.e., a resulting value will be NA whenever one of its contributing observations is NA. If use is "all.obs", then the presence of missing observations will produce an error. If use is "complete.obs" then missing values are handled by case wise deletion (and if there are no complete cases, that gives an error). "na.or.complete" is the same unless there are no complete cases, that gives NA.
- `normality_test_method` takes values as 'shapiro' or 'anderson'. this parameter decides which test to perform for the normality test. See details of `norm_test_fun` for more information.
- `normality_test_pval` significance level for normality tests. Default is 0.05
- `methodMat3` method dataframe like methodMats from the function `association`
- `methods_used` a square data.frame which will store the type of association used between the variables. Dimension will be number of variables * number of variables.
Details

This function measures the association between one categorical variable and one continuous variable present in different dataset. Two datasets are provided as input, one data has only numerical columns while other data has only categorical columns. This performs either t-test for the parametric case and 'Mann-Whitney’ test for the non-parametric case. If the method is passed as 'auto’, the function defines the method itself based on different tests for equal variance and normality check which checks for assumptions for the t-test. If the assumptions are satisfied, then t-test (parametric) is performed, otherwise 'Mann-Whitney’ (non-parametric) test is performed.

Value

a table with number of rows equal to number of columns in numtb and number of columns equal to number of columns in factb. Table containing p-values of performed test

See Also

norm_test_fun for normality test association for association between any type of variables, CCassociation for Association between Continuous (numeric) variables, QQassociation for Association between Categorical variables

Description

This function is used by [plottr()] when independent variable is continuous. This function can be used as a template to define a custom function.

Usage

Cx(dat, xname, binwidth = NULL, inc.density = T, ...)

Arguments

dat a data.frame with only one column
xname name of independent (x) variable in dat
binwidth for the histograms (extra parameters can be added like this)
inc.density Binary. True to include the density plot on histogram
... required

Value

a grob of plot
### CxCy

*Plots for Continuous independent and dependent variables*

**Description**

This function is used by `plotrr()` when both the dependent and independent variables are continuous. This function can be used as a template to define a custom function.

**Usage**

```r
CxCy(dat, xname, yname, ...)
```

**Arguments**

- `dat`: a data.frame with two columns (including the dependent)
- `xname`: name of independent (x) variable in `dat`
- `yname`: name of dependent (y) variable in `dat`
- `...`: required

**Value**

A grob of plot

### explainer

*Generic explainer*

**Description**

Generic function for printing the details of data. Based on the data type, this calls the appropriate method.

**Usage**

```r
explainer(X, xname = NULL, ...)
```

**Arguments**

- `X`: a data.frame or a vector
- `xname`: name of the data to be printed. If missing then the name of variable passed as `X` will be used
- `...`: other parameters required for other methods of `explainer`. To see the parameters for numeric methods, use `?explainer.numeric` and similarly for other methods (`?explainer.factor` etc.)
Current methods for explainer are for data.frame, numeric, integer, character and factor vectors. To get the list of all available methods type the command `methods(explainer)`.

Prints the information on the console. For print information for the individual methods, see their documentation. Returns nothing.

# for numeric
explainer(mtcars)
explainer(mtcars$mpg) #same as explainer.numeric(mtcars$mpg)
# for factor
explainer(as.factor(mtcars$cyl)) #same as explainer.factor(as.factor(mtcars$cyl))

This is an explainer method for character vector.

## S3 method for class 'character'
explaner(X, xname = NULL, ...)

a vector of character data type

a placeholder for variable name

other parameters required

This method removes all the missing values in x before computing the summaries.

Prints the following information on console:

- vector name
- type
- number of distinct values
• number of missing values
• a frequency table and histogram. If counts of all the factor levels are less than half of length of X, then the histogram is scaled with maximum of 50

Examples

alphabets <- sample(LETTERS[1:5], 50, replace = TRUE)
explainer(alphabets)
rm(alphabets)
Explainer method for factor data types

Description
This is an explainer method for factor vector.

Usage
```r
## S3 method for class 'factor'
explainer(X, xname = NULL, ...)
```

Arguments
- `X`: a numeric (or integer) data type
- `xname`: a placeholder for variable name
- `...`: other parameters required

Details
This method removes all the missing values in `X` before computing the summaries. This calls the method `explainer.character`

Value
Prints the following information on console:
- vector name
- type
- number of distinct values
- number of missing values
- a frequency table and histogram. If counts of all the factor levels are less than half of length of `X`, then the histogram is scaled with maximum of 50

Examples
```r
alphabets <- as.factor(sample(LETTERS[1:5], 50, replace = TRUE))
explainer(alphabets)
rm(alphabets)
```
explainer.numeric  

**Explain method for numeric data types**

**Description**
This is a explainer method for numeric vector.

**Usage**
```r
## S3 method for class 'numeric'
explainer(
  X,
  xname = NULL,
  include.numeric = NULL,
  round.digit = 2,
  quant.seq = seq(0, 1, 0.2),
  trim = 0.05,
  ...
)
```

**Arguments**
- `X`  
a numeric (or integer) data type
- `xname`  
a placeholder for variable name
- `include.numeric`  
a vector having strings which is also required along with the default output. Can have values:
  - `trimmed.means` for printing the trimmed mean after removing `trim` fraction of data from each side of x. `trim` can be passed as an parameter
  - `skewness` for printing the skewness of the data. Use `?skewness` for more information
  - `kurtosis` for printing the kurtosis of the data. Use `?kurtosis` for more information
- `round.digit`  
number of decimal places required in the output.
- `quant.seq`  
vector of fractions (0 to 1) for which the quantiles are required 0.5 means median, 0 means smallest observation and 1 means largest observation
- `trim`  
the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint. This only works if `include.numeric` has a string `trimmed.means`
- `...`  
other parameters required

**Details**
This method removes all the missing values in x before computing the summaries.
freqTable

Value
Prints the following information on console:

- vector name
- type
- number of distinct values
- number of missing values
- mean
- sd (standard deviation)
- median
- quantiles based on quant.seq parameter
- other information based on include.numeric
- a box plot (only if number distinct numbers are > 2). If counts of all the factor levels are less than half of length of x, then the histogram is scaled with maximum of 50 ?consoleBoxplot for how to read the table and histogram)
- a frequency table and histogram (only if number of distinct numbers are < 11) (look at ?freqTable for how to read the table and histogram)

Examples
explainer(mtcars$mpg)
explainer(mtcars$mpg, include.numeric = c('trimmed.means', 'skewness', 'kurtosis'), round.digit = 1, quant.seq = seq(0,1,0.1), trim = 0.05)

freqTable

Frequency table and Histogram

Description
freqTable prints a frequency table and histogram of a vector.

Usage
freqTable(Value, limit = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>a vector of any type</td>
</tr>
<tr>
<td>limit</td>
<td>Upper limit of the bars in histogram. Default is NULL, for which the function will automatically find the suitable limit. This value should be in fraction (between 0 to 1)</td>
</tr>
</tbody>
</table>
Details

This function works for all type of vector type. But calling freqTable for vector with many unique values will print a very long table. If the limit parameter is left blank, then the limit of histogram is adjusted automatically and is shown at the end in brackets (e.g. 50) This function is used in the explainer.

Value

Prints a table with columns

- **Value** Value. Each row has a unique value in this table
- **Freq** The frequency count of the Value
- **Proportion** Proportion of the Value (= Freq / length(x))

This table is followed by a histogram with bars for each of the unique values present in the data.

Examples

freqTable(mtcars$cyl)
freqTable(mtcars$mpg, limit = 0.08)

GenerateReport

*Generate the report*

Description

GenerateReport generates the markdown report in one command

Usage

GenerateReport(
  dtpath,
  catVars,
  yvar = NULL,
  model = "linReg",
  title = "Report",
  output_format = "html_document",
  output_dir = tempdir(),
  normality_test_method = "ks",
  interactive.plots = FALSE,
  include.vars = NULL
)
GenerateReport

Arguments

- **dtpath**: dataset path as a string
- **catVars**: vector of categorical variables names
- **yvar**: y variable name if present else NULL
- **model**: type of model - `linReg` for linear regression `binClass` for binary classification and `multiClass` for multiclass classification
- **title**: Title of the generated report
- **output_format**: output report format. 'html_document' for html file or `pdf_document` for pdf file output. OR c("html_document", "pdf_document") for both.
- **output_dir**: Directory where the output files needs to be stored.
- **normality_test_method**: method for normality test for a variable. Values can be `shapiro` for Shapiro-Wilk test or 'anderson' for 'Anderson-Darling' test of normality or `ks` for 'Kolmogorov-Smirnov'
- **interactive.plots**: for interactive variable exploration
- **include.vars**: include only these variables from the full data

Details

This function creates a rmarkdown report which can be converted to html or pdf format file.

Value

creates a rmarkdown and html/pdf file. Returns the output directory on successful run and FALSE in case of error

Examples

```r
# Assigning the temporary folder using tempdir(). replace with required directory
GenerateReport(dtpath = mtcars,
  catVars = c("cyl", "vs", "am", "gear"),
  yvar = "vs", model = "binClass",
  output_format = NULL,
  title = "Report",
  output_dir = tempdir(), # pass the output directory
  interactive.plots = FALSE) # set TRUE for interactive
```
kurtosis

Description

kurtosis calculates the Kurtosis

Usage

kurtosis(x, na.rm = T)

Arguments

x a numeric vector, matrix or a data.frame
na.rm (logical) Should missing values be removed?

Details

This function calculates the kurtosis of data which is a measure of the "tailedness" of the probability distribution of a real-valued random variable. Like skewness, kurtosis describes the shape of a probability distribution. The formula used is:

\[
\frac{E[(X - \mu)^4]}{(E[(X - \mu)^2])^2}
\]

. This formula is the typical definition used in many older textbooks and wikipedia

Value

returns a single value if x is a vector, otherwise a named vector of size = ncol(x).

Examples

# for a single vector
kurtosis(mtcars$mpg)

# for a dataframe
kurtosis(mtcars)
linedivider

<table>
<thead>
<tr>
<th>Description</th>
<th>Draws a horizontal line on console</th>
</tr>
</thead>
</table>

**Usage**

linedivider(consolewidth = getOption("width"), st = "x")

**Arguments**

- `consolewidth` a integer
- `st` a character or symbol of length to be used for creating the line

**Value**

Prints a horizontal line of width `consolewidth`

**Examples**

linedivider(20)

mergeAnalyzer

<table>
<thead>
<tr>
<th>Description</th>
<th>Analyze data for merging</th>
</tr>
</thead>
</table>

**Usage**

mergeAnalyzer(x, y, round.digit = 2, ...)

**Arguments**

- `x` left data to merge
- `y` right data to merge
- `round.digit` integer indicating the number of decimal places to be used
- `...` other parameters needs to be passed to merge function

**Details**

Prints the summary of data retained after merge and returns the merged data. This function uses data.table (if the package is installed) for faster data merge
norm_test_fun

Value

Returns merged data with same class as that of input data. Prints summary of data retained after
merging. The summary has 6 columns:

- Column: Number of rows and union of column names of numeric columns in both the data
- $x, y$: Sum of columns in both table
- Merged: Sum of columns in merged data
- remainingWRT$x$: ratio of remaining data in merged data after merging. example - 0.5 means
  that 50 of inner join). 1.5 means value became 150 duplicates present in data
- remainingWRT$y$: same as above, but for $y$ table

Examples

# Creating two tables to merge
A <- data.frame(id = c("A", "A", "B", "D", "B"),
                valA = c(30, 83, 45, 2, 58))
B <- data.frame(id = c("A", "C", "A", "B", "C", "C"),
                valB = c(10, 20, 30, 40, 50, 60))
mergeAnalyzer(A, B, allow.cartesian = TRUE, all = FALSE)

norm_test_fun

Checks for Normality Assumption

Description

norm_test_fun checks for the normality assumption

Usage

norm_test_fun(x, method = "anderson", pval = 0.05, xn = "x", bin = FALSE)

Arguments

x a numeric vector
method shapiro for Shapiro-Wilk test or 'anderson' for 'Anderson-Darling' test of
      normality or ks for 'Kolmogorov-Smirnov'
pval significance level for normality tests. Default is 0.05
xn vector name
bin TRUE if only TRUE/FALSE is required
Details

This function checks for normality assumption using shapiro, Kolmogorov-Smirnov or Anderson Darling test. If the parameter bin is TRUE, then TRUE is returned if vector is normal, otherwise FALSE. The significance level is passed through the parameter pval.

Value

Logical TRUE/FALSE based on the performed test and pval. If the vector follows the normality assumption, then TRUE is returned.

See Also

anderson.test for Anderson Darling test

Examples

norm_test_fun(mtcars$mpg)
norm_test_fun(mtcars$mpg, method = "shapiro", pval = 0.05, xn = "mpg", bin = TRUE)

Description

This function plots the plot generated by the library analyzer.

Usage

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

Arguments

x a plot of class analyzerPlot

... extra arguments if required

Value

Displays the plot

Examples

# creating the plot
p <- plottr(mtcars)
plot(p$mpg)
plotNA

Missing value visualization using ggplot2

Description

plotNA returns a grob visualizing the missing values in data

Usage

plotNA(tb, order = T, limit = T, add_percent = T, row.level = F)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tb</td>
<td>a data.frame</td>
</tr>
<tr>
<td>order</td>
<td>(logical) Whether to order the variables based on missing values in plot</td>
</tr>
<tr>
<td>limit</td>
<td>(logical) Whether to limit the plot to maximum missing value. FALSE means</td>
</tr>
<tr>
<td></td>
<td>the limit of axis will be [0, nrow(tb)]</td>
</tr>
<tr>
<td>add_percent</td>
<td>(logical) Whether to add percent as data labels on bar plot</td>
</tr>
<tr>
<td>row.level</td>
<td>(logical) Whether to create plot at rows and variables level</td>
</tr>
</tbody>
</table>

Details

This is a function which helps in visualizing the missing values in data using plots. By default a bar plot is generated which shows the count of missing values in each variable.

If order is set as TRUE then the bars are arranged in order of missing values. If limit is set as TRUE then limit of axis is set to [0, nrow(tb)]. If add_percent is set as TRUE then percent is added as text to the bars. If row.level is set to TRUE then an additional plot is generated which shows which rows have missing values and in which variable (reshape2 (https://CRAN.R-project.org/package=reshape2) library is required for this).

Value

This function returns a grob of class 'analyzrPlot' which has a bar plot showing the count of missing value for each variable. order, limit, add_percent can be used to modify the bar plot. An additional plot will be created and added to the grob if row.level is set as TRUE

Examples

```r
p <- plotNA(airquality)
# function to show the 'analyzrPlot' class plot
plot(p)
p1 <- plotNA(airquality, order = FALSE)
plot(p1)
```
plottr

Creates plots for the variables in a data.frame

Description

plottr can be used to create plots for all the variables in a dataframe or any one vector. The output is a list of plots for each variable of class 'analyzerPlot'.

Usage

plottr(
  tb,
  yvar = NULL,
  xclasses = NULL,
  yclass = NULL,
  printall = F,
  callasfactor = 1,
  FUN1 = Cx,
  FUN2 = Qx,
  FUN3 = CxCy,
  FUN4 = QxCy,
  FUN5 = CxQy,
  FUN6 = QxQy,
  ...
)

Arguments

tb a data.frame or a vector. If yvar argument is also passed, then this should be a data.frame including the response variable (yvar)
yvar a string showing the response (dependent) variable name. Can be NULL if response variable is not present. Make sure that this variable is present in the tb
xclasses a vector of length = ncol(tb) with the data type of all the columns. Can be NULL, in such case function assigns a class to each column. The values have to be either NULL, or a vector of either 'factor' or 'numeric'. The order should be same as the actual columns in tb. In case when tb is a vector, this can be a vector of length 1.
yclass class of response variable. Can be NULL, but must have value when yvar is not NULL. Value can be 'factor' or 'numeric'
printall (logical) Whether user wants to show the plots. Setting this as FALSE will only returns a list of plots silently.
callasfactor minimum unique values needed for x to be considered as numeric. See details for more information
FUN1 an user-defined function for plotting 1 variables when the variable is Continuous. See details for more details on how to define these variables
FUN2  same as FUN1 but for categorical variable
FUN3  an user defined function for plotting 2 variables when both the independent variable (x) and dependent variable (y) are Continuous
FUN4  same as FUN3, but when independent variable (x) is Categorical and dependent variable (y) is Continuous
FUN5  same as FUN3, but when independent variable (x) is Continuous and dependent variable (y) is Categorical
FUN6  same as FUN3, but when both the independent variable (x) and dependent variable (y) are Categorical
...  extra arguments passed to functions FUN1-FUN6

Details

This is a function which helps in understanding the data through multiple visualizations. This works either for a data.frame having multiple variables or a single x variable or for a combination of predictor x and response y variables. Based on class of x and y different types of plots are automatically generated.

Please note the following points:

Defining the class of variables: If yvar is not NULL, then yclass has to be passed (which can be 'factor' for classification type problem, or 'numeric' for regression). xclasses stores the class of all the variables in the data frame in same order of columns. Note - if yvar is not NULL, then tb has to be a data.frame with at least 2 columns (including the yvar). In such case xclasses should also have the class of yvar although it is also passed through yclass. This can also be set as NULL, in such case the function assigns a class based on the contents. If variable is factor/character type, then xclasses will have 'factor' as the entry for that variable, else if x is numeric with number of unique values less than callasfactor parameter value, then xclasses will have 'factor', else 'numeric'.

DEFINING CUSTOM FUNCTIONS FOR THE PLOTS USING FUN1, FUN2, FUN3 ... FUN6:
Custom plots can be made using these functions passed as arguments. Following things must be followed while defining such functions:

- the return plot must be of type 'grob' or 'gtables' or 'ggplot'. Since these outputs will go to arrangeGrob, make sure the output plots are acceptable by arrangeGrob function. See code of CxCy for sample.
- not all 6 functions are required to be passed. Only pass those functions for which plots need to be changed.
- FUN1 and FUN2 must have 3 parameters: dat (of type data.frame for the data. Even if there is only one column, it should be passed as a data.frame of one column), xname name of column in dat and ... In addition to these three, any number of additional parameters can be added. Look into source of code of Cx for sample.
- FUN3, FUN4, FUN5 and FUN6 must have 4 parameters: dat (of type data.frame for the data. Must have two columns for independent and dependent variables), xname name of independent variable in dat, yname name of dependent variable in dat and ... In addition to these four, any number of additional parameters can be added. Look into source of code of CxCy for sample.
- ... must be added as an argument in all the functions.
To get a better idea, see the code for function \texttt{CxCy} and \texttt{Cx}

Default plots: If the \texttt{y} is \texttt{NULL}, then histogram with density is generated for numeric \texttt{x}. Boxplot is also shown in the same histogram using color and vertical lines. For factor \texttt{x}, a pie chart showing the distribution. This are the univariate plots which can be modified by using the \texttt{FUN1} and \texttt{FUN2} arguments.

If \texttt{y} is not \texttt{NULL}, then additional plots are added which can be modified by using the \texttt{FUN3}, \texttt{FUN4}, \texttt{FUN5}, \texttt{FUN6} arguments:

- \texttt{factor x, factor y}: Crosstab with heatmap (modified by using \texttt{FUN6})
- \texttt{factor x, numeric y}: histogram and boxplot of \texttt{y} for different values of \texttt{x} (modified by using \texttt{FUN4})
- \texttt{numeric x, factor y}: histogram and boxplot of \texttt{x} for different values of \texttt{y} (modified by using \texttt{FUN5})
- \texttt{numeric x, numeric y}: Scatter plot of \texttt{x} and \texttt{y} with rug plot included (modified by using \texttt{FUN3})

**Value**

A list of plots for all the variables. Each plot will have the class \texttt{analyzerPlot} and can be displayed using \texttt{plot()}. If \texttt{printall = TRUE}, then all plots will also be displayed.

**Examples**

```r
# simple use for one variable
p <- plottr(mtcars$mpg)
# To display the plot
plot(p$x)

# With complete dataframe and assuming 'mpg' as a dependent variable
p <- plottr(mtcars, yvar = "mpg", yclass = "numeric")
plot(p$disp)
```

---

**print.analyzerPlot**  
*Method for print generic*

**Description**

This function plots the plot generated by the library analyzer

**Usage**

```r
## S3 method for class 'analyzerPlot'
print(x, ...)
```
QQassociation

Arguments

- x: a plot of class analyzerPlot
- ...: other parameters

Value

Displays the plot

Examples

```r
# creating the plot
p <- plottr(mtcars$mpg)
p$x
```

---

QQassociation

<table>
<thead>
<tr>
<th>Association (Correlation) between Categorical Variables</th>
</tr>
</thead>
</table>

Description

QQassociation finds Association measure between all the variables in data with only categorical columns.

Usage

```r
QQassociation(factb, use = "everything", methods_used)
```

Arguments

- factb: a data frame with all the categorical columns. This should have at least two columns
- use: an optional character string giving a method for computing association in the presence of missing values. This must be (complete or an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". If use is "everything", NAs will propagate conceptually, i.e., a resulting value will be NA whenever one of its contributing observations is NA. If use is "all.obs", then the presence of missing observations will produce an error. If use is "complete.obs" then missing values are handled by case wise deletion (and if there are no complete cases, that gives an error). "na.or.complete" is the same unless there are no complete cases, that gives NA
- methods_used: a square data.frame which will store the type of association used between the variables. Dimension will be number of variables * number of variables.
This function measures the association between categorical variables using Chi Square test. This also returns Cramers V value which is a measure of association between two nominal variables, giving a value between 0 and +1 (inclusive). Higher number indicates higher association. Note that, unlike Pearson correlation this doesn’t give negative value.

The relation between Cramer’s V and Chi Sq test is

\[ \sqrt{\frac{\chi^2}{n \times \min(k - 1, r - 1)}} \]

where:

- \( \chi^2 \) is derived from Pearson’s chi-squared test
- \( n \) is the grand total of observations
- \( k \) being the number of columns
- \( r \) being the number of rows

The p-value for the significance of Cramer’s V is the same one that is calculated using the Pearson’s chi-squared test.

Value

- a list of two tables with number of rows and column equal to number of columns in `factb`:
  - `chisq` Table containing p-values of chi-square test
  - `cramers` Table containing Cramer’s V

See Also

- `association` for association between any type of variables, `CCassociation` for Association between Continuous (numeric) variables, `CQassociation` for Association between Continuous-Categorical variables

---

**skewness**

**Skewness**

Description

- `skewness` calculates the skewness

Usage

- `skewness(x, na.rm = T)`
Arguments

`x` a numeric vector, matrix or a data.frame

`na.rm` (logical) Should missing values be removed?

Details

This function calculates the skewness of data which is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The formula used is:

\[
\frac{E[(X - \mu)^3]}{(E[(X - \mu)^2])^{\frac{3}{2}}}
\]

This formula is the typical definition used in many older textbooks and wikipedia.

Value

returns a single value if `x` is a vector, otherwise a named vector of size = `ncol(x)`.

Examples

```r
# for a single vector
skewness(mtcars$mpg)

# for a dataframe
skewness(mtcars)
```
Index

anderson.test, 2, 21
arrangeGrob, 24
association, 3, 6, 9, 27
CCassociation, 4, 5, 5, 9, 27
consoleBoxplot, 7
cor, 6
CQassociation, 3–6, 8, 27
Cx, 9, 24, 25
CxCy, 10, 24, 25
explainer, 10
explainer.character, 11
explainer.data.frame, 12
explainer.factor, 13
explainer.numeric, 14
freqTable, 15
GenerateReport, 16
kurtosis, 18
linedivider, 19
mergeAnalyzer, 19
norm_test_fun, 3, 8, 9, 20
plot.analyzerPlot, 21
plotNA, 22
plottr, 23
print.analyzerPlot, 25
QQassociation, 4–6, 9, 26
skewness, 27