Package ‘exploreR’

October 13, 2022

Type Package

Title Tools for Quickly Exploring Data

Version 0.1

Date 2016-02-10

Description Simplifies some complicated and labor intensive processes involved in exploring and explaining data. Allows you to quickly and efficiently visualize the interaction between variables and simplifies the process of discovering covariation in your data. Also includes some convenience features designed to remove as much redundant typing as possible.

Depends R (>= 3.2.3)

License GPL-3

LazyData TRUE

RoxygenNote 5.0.1

Imports ggplot2, grDevices, stats

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Michael Coates [aut, cre]

Maintainer Michael Coates <azhain@gmail.com>

Repository CRAN

Date/Publication 2016-02-10 18:57:17

R topics documented:

masslm *Mass Linear Regression*

Description

This function takes in a dataframe, the dependent variable, and optionally a character vector of independent variables you want the function to ignore. It then produces a dataframe of regression results.

Usage

```
masslm(data, dv.var, ignore = NULL, p.round = TRUE, c.round = TRUE)
```

Arguments

`data`	data.frame object that contains both the dependent variable and predictor variables you want to regress.
`dv.var`	single dependent variable you want to regress your predictors on.
`ignore`	accepts a character vector of one or more variables you want the function to skip. If nothing is passed through this option, the function will attempt to run a regression between the dependent variable and every other column of data.
`p.round`	set to TRUE by default. If left TRUE, will round off the P.value outputs to their 6 significant digits. Can be a problem for numbers larger than 999999, set to false to return the raw number.
`c.round`	set to TRUE by default. If left TRUE, will round off the Coefficient outputs to their 6 significant digits. Can be a problem for numbers larger than 999999, set to false to return the raw number.

Value

data.frame containing three columns of data, IV, Coefficient, and P.Value. If one of the columns of data not excluded from the function contained character type data, the function will print an error recommending the user attempt to convert the variable to a factor.

Examples

```
exam.df <- iris
masslm(exam.df, "Sepal.Width", ignore = "Species")
masslm(exam.df, "Sepal.Width", ignore = c("Species", "Petal.Width"))
```

`massregplot` *Mass Regression Plot*

Description

This function takes in a dataframe, the dependent variable, and optionally a character vector of independent variables you want the function to ignore, and produces a regression plot of every variable compared to the dependent variable passed into the function. It will ignore columns which contain characters and (also optional) factors.

Usage

```
massregplot(data, dv.var, ignore = NULL, save = NULL,
  include.factors = FALSE, include.se = TRUE)
```

Arguments

`data`	data.frame object that contains both the dependent variable and predictor variables you want to plot.
`dv.var`	single dependent variable you want to plot your predictors against.
`ignore`	accepts a character vector of one or more variables you want the function to skip. If nothing is passed through this option, the function will attempt to create a graph plotting the dependent variable and every other column of data.
`save`	accepts a character. If the function recieves a character, it will create a pdf file with that name and leave the plots in there.
`include.factors`	if TRUE, will also plot factor variables against your dv. Otherwise it will skip these as regression plots of categorical variables are of imited use.
`include.se`	if left TRUE, will shade the area around the regression line with the 95% confidence interval range. Setting to FALSE will plot only the regression line to a scatter plot for each paring of variables.

Value

Doesn't return a value, per se, but will generate side effects like plotting all the graphs created by the function. If the save option is used, it will save all generated graphs to a pdf file whose name is specified by the user.

Examples

```
exam.df <- iris
massregplot(exam.df, "Sepal.Length", ignore = "Species")
massregplot(exam.df, "Sepal.Length", ignore = c("Species", "Petal.Width"), include.se = FALSE)
```

reset *Reset R*

Description

This function simply erases the the console, detaches all packages and removes all data from the global environment. The purpose of which is to provide an easy command which can clean up the workspace. Very useful after you spend some time experimenting.

Usage

```
reset()
```

Examples

```
reset()
```

standardize *Standardize Variables*

Description

This function takes in a dataframe, the name of any number of variables. It then returns either a vector or a dataframe of scaling results. If passed a single variable name, standardize will return a the standardized variable as a vector, when passed 2 or more variable names, standardize will return a data frame containing all of the standardized variables.

Usage

```
standardize(data, variable, type = "absolute")
```

Arguments

`data`	data.frame object that contains both the dependent variable and predictor variables you want to regress.
`variable`	variable name or vector of names for variables you want standardized.
`type`	by default, 'absolute' will scale every variable from 0 to 1. "classic" is a little more complicated where the variable is rescaled the mean equaling 0 and a standard deviation is 1.

Details

Often times we are forced to compare variables which exist on different scales, but how do you compare the coefficient for a country's population to one that's much smaller? Standardizing your variables can make reading regression results more useful because it will make coeficients more comparable.

Value

if the function is passed a single variable to standardize, it will return a vector of all obeservations in the variable standardized. If the function is passed more than one variable name, it will return a data-frame containing the new observation values.

Examples

```
exam.df <- iris
standardize(exam.df, "Petal.Width")
standardize(exam.df, c("Petal.Width", "Petal.Length"), type = "classic")
```

Index