Package ‘rstatsToolkit’

November 4, 2014

Title Bundles up all my most used functions for doing statistical analysis

Version 0.1

Description This package is mainly my personal collection of code that I commonly use in my research and statistical analysis. Eventually I would like to develop it into a toolkit that other graduate students (I'm a graduate student right now btw) and in the future for my own graduate students to use and develop. Until that point, I am slowly developing this package into toolkit for analyzing and exploring data.

Imports ggplot2 (>= 1.0.0),plyr (>= 1.8.1),grid (>= 3.1.1),visreg (>= 2.0.5),reshape2 (>= 1.4)

Depends R (>= 3.1.1)

License Creative Commons 0

LazyData true

R topics documented:

`bivarPlot` *Bivariate plot*

Description

In development ..

Usage

```
bivarPlot(x, y, data, ...)
```

Arguments

`x`

`y`

`data`

`...`

Details

.. content for details ..

Value

Outputs a plot

Author(s)

Luke Johnston

`diagnosticPlots` *Regression diagnostic plots and tests*

Description

Generate regression diagnostic plots and tests for linear regression models.

Usage

```
diagnosticPlots(data, y, x, covar)
```

Arguments

`data` The dataset with the variables of interest

`y` The dependent or outcome variable (that is, the `y` in the regression equation)

`x` The independent, exposure, or predictor variable (that is, the `x` in the regression equation)

`covar` The variables selected as to condition or adjust for the `y` and `x` relationship, also known as the confounding variables

Details

This function runs a linear regression on the specified variables and generates diagnostics based on the regression. Basic diagnostics include checking the normality of the residuals, assessing outliers, influence and Cook's D, and multicollinearity. Several tests have been commented out, though they can be uncommented if desired (edit the function to output these if desired). Some of the tests I don't fully understand how to interpret them, but as I learn more I will probably know. This function relies on **MASS** and **gplots**.

Value

Outputs multiple plots and textplots with diagnostic information

Author(s)

Luke Johnston

`heatmapCorr` *Correlation heatmap*

Description

Generate a matrix or non-matrix style heatmap of correlation coefficients (i.e. the number of columns and rows can be different).

Usage

```
heatmapCorr(data, x, y, leg.range = c(-1, 1), levels.xlab = NULL,
  levels.ylab = NULL, xlab = "", ylab = "", lo.color = "darkorange2",
  hi.color = "skyblue4", rm.legend = FALSE, matrix.sty = FALSE, ...)
```

Arguments

`data` Dataset that contains the variables of interest

`x` Vector of variables that will run along the x-axis

`y` Same as the `x` arg, but for the y-axis

`leg.range` Range in values for the legend, between -1 and 1 `levels.xlab` Specifies custom variable names for the x-axis (it needs to be a list object)

`levels.ylab` Same as the `levels.xlab` arg, but for the y-axis

`xlab` Sets the label for the x-axis

`ylab` Same as `xlab`, but for the y-axis

`lo.color` Color of negative correlation coefficients

`hi.color` Color of positive correlation coefficients

`rm.legend` In development. Goal is it will remove the legend

`matrix.sty` Select whether the heatmap will be a matrix style (the same variable on the x- and y- axis) or non-matrix (different variables on both axes)

`...` Potential options to add. In development

Details

This function takes two arguments, the `x` variables and the `y` variables, and generates a heatmap from the variables. A correlation matrix is computed from the data, melted (reshape package), and input into ggplot2 to generate a heatmap. The output is the correlations and the plot object.

Dependencies are: **reshape2** and **ggplot2**

Value

Outputs a plot object

Author(s)

Luke Johnston

Examples

```
xlabs <- list("X Name" = "x1", "X Name Two!" = "x2" ... )
ylabs <- list("Y Name" = "y1", "Y Name Two!" = "y2" ... )
df <- data.frame(replicate(10,sample(0:1,1000,rep=TRUE)))
plot <- heatmap.corr(data = df, x = 1:3, y = 4:10,
                     levels.xlab=xlabs, levels.ylab=ylabs)
print(plot)
```

`jitterBoxplot` *Univariate jittered boxplot*

Description

Generates a boxplot of variables on one axis with raw values "jittered" as dots underneath. The variables need to represent a similar concept or have the same units for the plot to make sense.

Usage

```
jitterBoxplot(subset.ds, dot.size = 2, dot.colour = "grey50",
  custom.var.names = NULL, xlab = NULL, ylab = NULL)
```

Arguments

`subset.ds` The dataset that only contains the series of variables that will be plotted

`dot.size` Size of the dot for `geom_jitter`

`dot.colour` Color of the dots for `geom_jitter`

`custom.var.names` List object that contains the custom (alternative) variable names for the variable (column) names you passed into the function

`xlab` The x-axis label

`ylab` The y-axis label

Details

This function is useful for exploring the distribution of a series of variables that share a common unit, such as kilogram. The values for each variable are plotted as jittered dots with a boxplot of the distribution layered on top of the dots. The function takes a subsetted dataset that contains only the series of variables that share a common unit. The output object is the plot. This function depends on **ggplot2** and **reshape2**.

Value

Outputs the plot object

Author(s)

Luke Johnston

Examples

```
varnames <- list("Name of X1" = "X1", "Name of X2" = "X2", ...)
df <- subset(ds, VN == 0, select=nefa)
names(df)
jitter.boxplot(df, xlab="Concentration (nmol/mL)",
               ylab="Fatty acid", custom.var.names=varnames)
```

multiPlot *Multiple plots on page*

Description

Lay out multiple ggplots on one frame or page.

Usage

```
multiPlot(..., plotlist = NULL, file, cols = 1, layout = NULL)
```

Arguments

`...`	Where the ggplot objects are placed to be laid out on the graph grid
`plotlist`	Can be used in place of the ... argument by specifying the ggplot objects as a list object
`file`	Not sure what this is used for
`cols`	Number of columns for the layout. For example cols=2 provides two columns and with four ggplot objects, the resulting output would be a 2 by 2 graphic
`layout`	A matrix that indicates the plot grid layout. For example, if layout = matrix(c(1, 2, 3, 3), nrow = 2, byrow = TRUE) the result would have plot 1 in the upper left, plot 2 in the upper right, and plot 3 would be go across the bottom

Details

This function, which was from http://www.cookbook-r.com/Graphs, is used to lay out several ggplot objects onto one frame or pdf page. For instance, you can have 3 plots on a page, one going vertically across the top, the other two in each corner on the bottom. This function makes up for the difficulty **ggplot2** has with outputting multiple plots on one grid. This function depends on **grid**.

Author(s)

Cookbook R

plotVisreg *Visualizing adjusted linear regression models*

Description

Generates plots of a linear regression model which includes confounding variables.

Usage

```
plotVisreg(data, y, x, covar, ylabel = x, xlabel = y, ...)
```

Arguments

`data`	Dataset with the variables of interest
`y`	The dependent or outcome variable in the regression equation
`x`	The independent or exposure variable in the regression equation
`covar`	The confounding variables, that is the variables being adjusted for
`ylabel`	The y-axis label
`xlabel`	The x-axis label
`...`	Other options. In development

Details

This function runs a linear regression on the specified variables and plots the partial residuals. This allows for visualizing the relationship between the outcome and the exposure, after adjusting for confounders. A linear slope is plotted through the partial residuals, with a confidence interval band around it. The output is a plot. This function depends on **visreg**.

Value

Outputs a plot of the regression model

Author(s)

Luke Johnston

`rstatsToolkit` *rstatsToolkit.*

Description

rstatsToolkit.

`smoothBivarPlot` *Smooth bivariate plot*

Description

In development ..

Usage

```
smoothBivarPlot(x, y, data, ...)
```

Arguments

`x`

`y`

`data`

`...`

Details

.. content for details ..

Value

Outputs a plot

Author(s)

Luke Johnston

`smoothPlot` *Smooth plot*

Description

In development ..

Usage

```
smoothPlot(x, y, data, ...)
```

Arguments

`x`

`y`

`data`

`...`

Details

.. content for details ..

Value

Outputs a plot

Author(s)

Luke Johnston

summarySE *Summarize means and standard errors of the mean*

Description

Calculates the sample size, mean, standard deviation, standard error of the mean, and the confidence interval of specified variables.

Usage

```
summarySE(data = NULL, measurevar, groupvars = NULL, na.rm = FALSE,
  conf.interval = 0.95, .drop = TRUE)
```

Arguments

`data`	A dataset (dataframe) that contains the values to be summarized
`measurevar`	The name of a column that contains the variable to be summarized
`groupvars`	A vector containing names of columns that contain grouping variables
`na.rm`	A binary (boolean) response that indicates whether to ignore missing (NA) data
`conf.interval`	Percent range of the confidence interval

Details

I took this function on 2014-01-21 from the website `http://www.cookbook-r.com/Graphs`. It basically summarizes the provided data by giving count, mean, standard deviation, standard error of the mean, and confidence interval (default 95 The dependencies are **plyr**

Value

Outputs a dataframe that contains the summarized statistics (means, etc.)

Author(s)

Cookbook R

themeWhite *Custom white ggplot theme*

Description

Creates a white, simple theme for **ggplot2** objects

Usage

```
themeWhite()
```

Details

The default **ggplot2** theme is decent for most purposes, but is visually unappealing. This function aims to correct that by setting the theme to something more similar to the default in the base R plot package. The function dependes on **ggplot2**.

Author(s)

Luke Johnston

Examples

```
## This creates a white theme
theme_white()
```

Index