Package ‘intervisu’

July 18, 2016

Type Package

Title Interactive data visualisation with shiny

Version 0.1.0

Author Cornelius Fritz

Maintainer Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Description This package supplies functions generating shiny applications with your own data for explorative descriptive data analysis.

License GPL-3

LazyData TRUE

RoxygenNote 5.0.1

Depends R (>= 3.0)

Imports shiny (>= 0.13.2),scatterplot3d (>= 0.3-36),car,mgcv,hypergea,
zoo, splines ,tseries, utils, stats, threejs, xts, dygraphs,
ggplot2, plotly, DT, tidyr, viridis

NeedsCompilation yes

RemoteType github

RemoteHost https://api.github.com

RemoteRepo intervisu

RemoteUsername corneliusfritz

RemoteRef master

RemoteSha 87aa814f53d31bd308fee6652e9d4fefa553f824

GithubRepo intervisu

GithubUsername corneliusfritz

GithubRef master

GithubSHA1 87aa814f53d31bd308fee6652e9d4fefa553f824

R topics documented:

check_package 2
factor_check 2
factor_melt 3
faktor 3
faktor2 4

Group_Boxplot 4
Metric_Single_Variable_Analysis 5
polynomfit 6
position4 7
position5 7
position6 8
range01 8
round_check 9
Scatterplotly_Matrix 9
Scatterplot_3d 10
Scatterplot_Matrix 11
Smoothing_Analysis 12
Stacked_Barplot 13
Timeseries 14
Timeseries_Stat 15

Index **17**

`check_package` *Help-Function*

Description

This function checks if a a vector of packages named each as a string are installed. Every package that is installed will be loaded and all not yet installed packages will be installed and loaded.

Usage

```
check_package(packages)
```

Arguments

`packages`	A vector of packages

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`factor_check` *Help-Function*

Description

This function tells you what variables of a given dataset are factorial variables

Usage

```
factor_check(data)
```

Arguments

`data`	The data to be analysed

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`factor_melt` *Help-Function*

Description

This function melts two factor levels into one factor level.

Usage

```
factor_melt(factor_leveln1, factor_leveln2, factor1)
```

Arguments

`factor_leveln1`	The number of the first factor level
`factor_leveln2`	The number of the second factor level
`factor1`	The factor variable itself

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`faktor` *Help-Function*

Description

This function turns all factorial variables in a dataset into metric variables and returns a logical vector what variable were factor-variables.

Usage

```
faktor(data)
```

Arguments

`data`	The data to be analysed

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`faktor2` *Help-Function*

Description

This function factorizes a variables with more than n different values.

Usage

```
faktor2(data, n)
```

Arguments

`data`	A data.frame object
`n`	A numeric value indicating from what number of different values a variable is seen as categorical variables, all variables that have more different values than n are being treated as metric values

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`Group_Boxplot` *Interactive Boxplot conditioned on a factorial variable*

Description

Takes any metric data and plots different boxplots conditioned on a factorial variable.

Usage

```
Group_Boxplot(data, n = 10, width = 600, height = 600)
```

Arguments

`data`	A data.frame object that is to be analyzed (only metric variables are shown)
`n`	A numeric value indicating the limit from what number of different values a variable is seen as categorical variable, all variables that have more than n different values are being treated as metric values
`width`	A numeric value indicating the width of the shown boxplot
`height`	A numeric value indicating the height of the shown boxplot

Details

A grouped boxplot is plotted, you can decide what metric variable is plotted and according to what factorial variable the boxplot is grouped. You can also interactively melt levels of the factorial variable together. In order to do this you at first need to select one level of the factorial variable by clicking on it with a single-click. The chosen boxplot will be highlighted. The other level that you want to melt together with the first one is identified by a double-click. Once you clicked on one level with a single-click and on one level with a double-click, those two levels are melted into one level. The labels on the x-axis are changed accordingly. You can always return to the older version of the factorial variable (with one more level than the current) by pressing the return button. You can plot a horizontal line at the global mean of the metric variable by checking the box names 'Show global mean'.

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Examples

```
if (interactive()) {
  Group_Boxplot(iris)
  }
```

Metric_Single_Variable_Analysis

Single Variable Analysis of a metric variable

Description

With this application you can graphically analyse a single metric variable. From a given data-set you can plots boxplots, density-estimations and histograms, each with many interactive elements. If you want to reproduce the same graph you built in the app, you can always get the code for the desired representation by clicking the 'Show R Code'-Button.

Usage

```
Metric_Single_Variable_Analysis(data, n = 10, a = 50, width = 700,
  height = 700)
```

Arguments

`data`	A data.frame object that is to be analyzed (only metric variables will be used in this application)
`n`	A numeric value indicating the limit from what number of different values a variable is seen as categorical variable, all variables that have more than n different values are being treated as metric values
`a`	A numeric value indicating how much freedom the slider telling the boxplot what xlim values should be allowed should have
`width`	A numeric value indicating the width of the shown plot
`height`	A numeric value indicating the height of the shown plot

Details

There are several interactive elements to each interactive data visualizations. If you want to plot a **boxplot** you can at first decide what metric variable of the chosen data should be plotted. You can also plot the points in the boxplot by pressing the button named 'Show Points'. In order to see all observed data a jitter function is used to randomize the points in the x-coordinates. Another possibility is to show a horizontal boxplot. With two sliders you can also decide what values of the observed variable should be used and what values should be ploted in the boxplot. In order to see the R-code leading to the desired graphical presentation check the 'Show R Code' button. You can copy and paste the code in you R console.

Plotting a **density estimator** also gives you many different possibilities. At first you can decide what metric variable should be plotted. You can also change the used kernel, the options are a gaussian, epanechnikov, triangular, rectangular, cosine and opt-cosine kernel. With two numeric inputs you can decide from what value to what value the slider input for the chosen bandwidth should be shown. You can also just plot only the observed values of a variable that lie within a given range (with a slider you can decide what values should be used).

The last optional graphical representation is a **histogram** of the relative frequency. With a numeric input you can set the upper limit of the y-axis. You can also change the origin of the histogram. The origin of a histogram is where the first block of the histogram starts, and to plot all optional values it can't be greater than the smallest observed value. The last interactive parameter is the used bandwidth in the plot, this is the absolute length of each single bars in the histogram.

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Examples

```
data=mtcars
if (interactive()) {
  Metric_Single_Variable_Analysis(data)
  }
```

polynomfit *Help-Function*

Description

This function calculates a polynomial-fit

Usage

```
polynomfit(d, x, y)
```

Arguments

d	The degree of the fit
x	The independent metric variable
y	The dependent metric variable

Note

The output also features a prediction of the dependent variable in the interval from min(x) to max(x)

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`position4` *Help-Function*

Description

This function is used on the Timeseries-functions and recognizes from the observed y variable of a click in what the used is clicking.

Usage

```
position4(y)
```

Arguments

`y`	The y-value of a click

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`position5` *Help-Function*

Description

This function is used on the Timeseries-functions and recognizes the y-limits of the 'brush'-interaction in the applications.

Usage

```
position5(x, ymin, ymax, ts_w)
```

Arguments

`x`	A numeric value ether 1,2,3 or 4. There are four plots shown underneath each other, what plot should be zoomed in?
`ymin`	The ymin value of the 'brush'-interaction
`ymax`	The ymax value of the 'brush'-interaction
`ts_w`	The analysed time series

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

position6 *Help-Function*

Description

This function is used on the Timeseries-functions and recognizes the x-limits of the 'brush'-interaction in the applications.

Usage

```
position6(xmin, xmax, beginning, ending)
```

Arguments

`xmin`	The ymin value of the 'brush'-interaction
`xmax`	The ymax value of the 'brush'-interaction
`beginning`	The beginning of the analysed time series
`ending`	The ending of the analysed time series

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

range01 *Help-Function*

Description

This function scales the values of a ordered vector from 0 to 1

Usage

```
range01(x)
```

Arguments

`x`	A ordered numeric vector

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`round_check` *Help-Function*

Description

This function checks if a numeric vector only includes whole numbers

Usage

```
round_check(x)
```

Arguments

`x`	The vector to be checked

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

`Scatterplotly_Matrix` *Interactive Scatterplot-Matrix*

Description

Takes any data and creates an interactive Scatterplot-Matrix with multiple features.

Usage

```
Scatterplotly_Matrix(data, metr_data = F, width = c(400, 700, 400),
  height = c(500, 700, 500))
```

Arguments

`data`	A data.frame object that is to be analyzed (all categorical variables with be transformed to metrical variables)
`metr_data`	A logical value, indicating whether a table showing the metric and categorical analogies of levels should be shown
`width`	A three-dimensional numeric value indicating the width of the three shown plots
`height`	A three-dimensional numeric value indicating the height of the three shown plots

Details

At first one has to choose the variables to be plotted in the scatterplot matrix. Once the user has choosen at least two variables (the minimum of a scatterplot), those two plots are also shown on each side of the Scatterplot-Matrix. The user though can decide what scatterplots out of the Matrix should be zoom in on each side. A conventional click on one scatterplot from the Scatterplot-Matrix triggers a bigger scatter plot on the left-hand side, while you can control the scatterplot on the right-hand side with a double click on the desired scatterplot in the Scatterplot-Matrix. On each of the small scatterplots one can select certain points with the brush option, implemented in the R-packge Shiny. Once you have chosen a cloud of data points and performed a normal click, the chosen points with be colored red in all available scatterplots. If you want to see the chosen data you have to press the "show data"-Button. A linear regression line and a loess curve can also be plotted by pressing the fitting button. In the case of a pressed "Regression"- or "Smooth"-Button. If groups have been definied by brushing actions, you can also plot the linear and loess regression by group once you pressed the Button "By Group".

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Examples

```
data=iris
if (interactive()) {
  Scatterplot_Matrix(data,metr_data = T)
  }
```

`Scatterplot_3d` *Interactive three-dimensional scatterlpot*

Description

Takes any metric data and plots a three-dimensional scatterplot and a two-dimensional scatterplot.

Usage

```
Scatterplot_3d(data, n, height = c(1500, 500))
```

Arguments

`data` A data.frame object that is to be analyzed (only metric variables are shown)

`n` A numeric value indicating the limit from what number of different values a variable is seen as categorical variable, all variables that have more than n different values are being treated as metric values

`height` A two-dimensional numeric vector indicating the height of the shown three-dimensional and two-dimensional scatterplots

Details

This function produces an application that lets you plot a interactive three-dimensional scatterplot, you can decide what metric variables should be plotted. Next to the three-dimensional scatterplot there is a two-dimensional scatterplot of the first two dimensions of the three-dimensional scatterplot (1.Variable and 2. Variable). You can change the plotted angle interactively by clicking and brushing over the shown three-dimensional scatterplot with the cursor. You can press the little 'Play'-Button and it will automatically rotate the plot once. There is also a feature added in this application, which lets you condition both plots on a fourth variable. To enter this mode press the button called 'Condition plot only values conditioned on the value of a fourth variable. You can now set a value for the fourth variable with a slider. Only 30

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Examples

```
if (interactive()) {
  Scatterplot_3d(mtcars)
}
```

Scatterplot_Matrix *Interactive Scatterplot-Matrix*

Description

Takes any data and creates an interactive Scatterplot-Matrix with multiple features.

Usage

```
Scatterplot_Matrix(data, metr_data = F, width = c(400, 700, 400),
  height = c(500, 700, 500))
```

Arguments

`data`	A data.frame object that is to be analyzed (all categorical variables with be transformed to metrical variables)
`metr_data`	A logical value, indicating whether a table showing the metric and categorical analogies of levels should be shown
`width`	A three-dimensional numeric value indicating the width of the three shown plots
`height`	A three-dimensional numeric value indicating the height of the three shown plots

Details

At first one has to choose the variables to be plotted in the scatterplot matrix. Once the user has choosen at least two variables (the minimum of a scatterplot), those two plots are also shown on each side of the Scatterplot-Matrix. The user though can decide what scatterplots out of the Matrix should be zoom in on each side. A conventional click on one scatterplot from the Scatterplot-Matrix triggers a bigger scatter plot on the left-hand side, while you can control the scatterplot on the right-hand side with a double click on the desired scatterplot in the Scatterplot-Matrix. On each of the small scatterplots one can select certain points with the brush option, implemented in the R-packge Shiny. Once you have chosen a cloud of data points and performed a normal click, the chosen points with be colored red in all available scatterplots. If you want to see the chosen data you have to press the "show data"-Button. A linear regression line and a loess curve can also be plotted by pressing the fitting button. In the case of a pressed "Regression"- or "Smooth"-Button. If groups have been definied by brushing actions, you can also plot the linear and loess regression by group once you pressed the Button "By Group".

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Examples

```
data=iris
if (interactive()) {
  Scatterplot_Matrix(data,metr_data = T)
  }
```

`Smoothing_Analysis` *Interactive smoothing with two metric variables*

Description

Takes any metric data and plots a two-dimensional scatterplot. You can interactively build a polynomial or smoothing spline fit and plot those fits with a confidence band.

Usage

```
Smoothing_Analysis(data, n = 10, height = 500)
```

Arguments

`data`	A data.frame object that is to be analyzed (only metric variables are shown)
`n`	A numeric value indicating the limit from what number of different values a variable is seen as categorical variable, all variables that have more than n different values are being treated as metric values
`height`	A numeric value indicating the height of the shown scatterplot

Details

On the sidebar you can chose two metric variables from the data set to be plotted in a scatterplot. If you wish you can also transform any of the used variables (options are: identity, log, exp, sin, cos). On the right side you see the R^2 values of the current model. To fit a polynomial model chose the 'Polynomial'-option in the type of regression.With a numeric input you can change the degree of the fit. The other option is to fit "Polynomial-Splines", one can do that by selecting the according type of regression. At first you can set the amount of knots and decide weather you want to penalize the fit. You can also use different kind of smoothing basis you want to use. There are three different possible smoothing bases: thing plate regression splines, cubic regression splinesand p-splines. If you want to penalize the fit check the box 'Penalize', you can set the numeric value lambda, indicating the importance of the penalty term. In case you are fitting a p-Spline Basis without a penalty the model is equivalent to b-splines, so you can also set the order of the fit. The last thing you are also free to change is the family and link-Function used in the the model. For further reading on these options see the documentation in the stats-Package (`family` and `make.link`).

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Examples

```
data=iris
if (interactive()) {
  Smoothing_Analysis(data,n=10,height = 500)
  }
```

`Stacked_Barplot` *Interactive Stacked Barplot*

Description

Takes any data and creates an stacked barplot with multiple interactive features.

Usage

```
Stacked_Barplot(data, n = 10, m = 5, height = c(500, 300))
```

Arguments

`data`	A data.frame object that is to be analyzed
`n`	A numeric value indicating the limit from what number of different values a variable is seen as categorical variable, all variables that have more than n different values are being treated as metric values
`m`	A numeric value telling hiow many observations need to be in each cell to perform the approximative chi^2-Test, rather than the exact Fisher-Test(default at 5)
`height`	A two-dimensional numeric value indicating the height of the barplot and boxplot

Details

In the sidebar you can choose two categorical variables and plot a stacked Barplot, each bar conditioned on the first categorical variable. If you click on one of the shown barplots, a boxplot conditioned onto that specifical level of the categorical variable will be shown underneath the barplot. You can also decide which metric variable should be plotted in the boxplot.

@details Eventually the staked barplot describes the frequency table of these two categorical variables. The independence of those variables can be testes ether with a approximate chi^2 test or a exact fisher-test. Which test is used depends on the cell counts in the contigency table and the parameter `m`, only if at least one cell in the frequency table has less observations than `m` the exact fisher-test is performed. The test can be interpreted as to weather the hypothesis of independence holds true or not. In case of a 2x2 frequency table the Odds Ratio value is given as well. You can also interactively melt levels of the factorial variable together. In order to do this you at first need to select one level of the factorial variable by clicking on it with a single-click. The chosen boxplot will be highlighted. The other level that you want to melt together with the first one is identified by a double-click. Once you clicked on one level with a single-click and on one level with a double-click, those two levels are melted into one level. The labels on the x-axis are changed accordingly. You can always return to the older version of the factorial variable (with one more level than the current) by pressing the return button.

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Examples

```
data=mtcars
if (interactive()) {
  Stacked_Barplot(data,n=10,height = c(400,400))
  }
```

Timeseries *Interactive Seasonal Decomposition of Time Series by Loess and plot it with dygraph*

Description

Takes any metric data with time series and plots a seasonal decomposition by Loess.

Usage

```
Timeseries(data, height = 200, width = 1000, n = 10)
```

Arguments

data	A data.frame object that is to be analyzed (all categorical variables with be transformed to metrical variables)
height	The height of all plots
width	The width of all plots
n	A numeric value indicating from what number of different values a variable is seen as categorical variables, all variables that have more different values than n are being treated as metric values

Details

At first you can chose what metric variable should be composed into a seasonal, trend and remainder part. Then you have to choose from what year to what year the data was observed and the frequency of observations per year. For monthly observed data this parameter should be 12. The STL algorithm is set up in two loops, one inner loop to update the seasonal and trend component and one to updated the robust weights of each oberservation. If you want to update any robust weights you can press the button "Robust Fitting". The bandwidth of the estimation used updating the seasonal component is usually the amount of observations in one period, but can also be set individually, if you don't press the "Periodicity of the Loess window for the Seasional Component". You can brush over any part of the decomposition zoom in on that specific area, perform a double click in order to return to the full data.

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

References

R. B. Cleveland, W. S. Cleveland, J.E. McRae, and I. Terpenning (1990) STL: A Seasonal-Trend Decomposition Procedure Based on Loess. Journal of Official Statistics, 6.

Examples

```
date1=as.Date(paste0(2000,"-1-1"))
date2=as.Date(paste0(2010,"-1-1"))
if (interactive()) {
  data=get.hist.quote(instrument= "BMW.DE",start = date1, end = date2 ,
                      quote=c("Open", "High", "Low", "Close",
                              "AdjClose","Volume"), provider = "yahoo",
                      compression = "m",retclass="zoo",quiet = T)
  data=as.data.frame(data)

  Timeseries(data=data)
}
```

Timeseries_Stat *Interactive Seasonal Decomposition of Time Series by Loess example application (Type 2)*

Description

For this example a working internet connection is needed. The user can chose out of a range of stocks and indicators from on yahoo finance what data from what time (in years) should be downloaded as a time series. Then a Seasonal Decomposition of Time Series by Loess is performed on the given time series.

Usage

```
Timeseries_Stat(height = 150, width = 800)
```

Arguments

`height`	The height of all plots
`width`	The width of all plots

Details

If you brush over any part of the decomposition you see a zoom plot in the sidebar panel of the shaded area.

Author(s)

Cornelius Fritz <cornelius.fritz@campus.lmu.de>

Examples

```
height = c(400,800)
if (interactive()) {
  Timeseries_Stat1(height =height )
  Timeseries_Stat2(height =height)
}
```

Index

check_package, 2

factor_check, 2
factor_melt, 3
faktor, 3
faktor2, 4
family, *13*

Group_Boxplot, 4

make.link, *13*
Metric_Single_Variable_Analysis, 5

polynomfit, 6
position4, 7
position5, 7
position6, 8

range01, 8
round_check, 9

Scatterplot_3d, 10
Scatterplot_Matrix, 11
Scatterplotly_Matrix, 9
Smoothing_Analysis, 12
Stacked_Barplot, 13

Timeseries, 14
Timeseries_Stat, 15