Internship Report

Connecting R To GroIMP

Rchart — A Chart Plugin For GroIMP Based On R/gg2plot

Lukas Gürtler

Matriculation number: 11404043

Supervisor: Prof. Dr. Winfried Kurth

University: Georg-August-Universität Göttingen

Module: M.Inf.1201 Systementwicklung in einer forschungsbezogenen Projektarbeit

Contents

1 Task setting

To plot data GroIMP internally makes use of the Java plugin JFreeChart. R is a sophisticated statistical analysis software with comprehensive data visualization capability. Plots from R are often used in publications and R makes it easy to process data in the manner of statistical computing, analysis and regression. In order to extend charting features of GroIMP it was chosen to connect it to R by especially utilizing the plugin ggplot2. The work deals with a GroIMP plugin developed by the author that accomplishes the deployment of the charting capability of R in GroIMP.

2 Results

2.1 Connecting R to GroIMP

GroIMP is platform independent since it is written in Java. There are several Java plugins that make the R language available for Java applications. These are for example Renjin, FastR, JRI, Rserve or rJava. However the existence of huge disadvantages makes the use of these plugins impossible or user-unfriendly. Some plugins require Java versions higher than 8 but GroIMP demands Java 7 compatibility. Furthermore most plugins need the installation of R or R-Studio.

Another approach to connect R to GroIMP is the usage of a server application. R then runs on a server and GroIMP communicates via a http connection to this server. Notwithstanding this technique requires a permanent internet connection and heavily is dependent on a properly working server. This results in a big unreliability issue.

Because of that it was chosen to use the R executable program RScript. GroIMP then writes scripts that are executed locally and reads the corresponding output. For the OS Windows the necessary files can be delivered within the Rchart plugin folder since portable versions of R exist. Nonetheless for Linux and Mac the prerequisite for an R-installation including necessary R plugins (like ggplot2) remains. Figure 1 illustrates the approach for Windows.

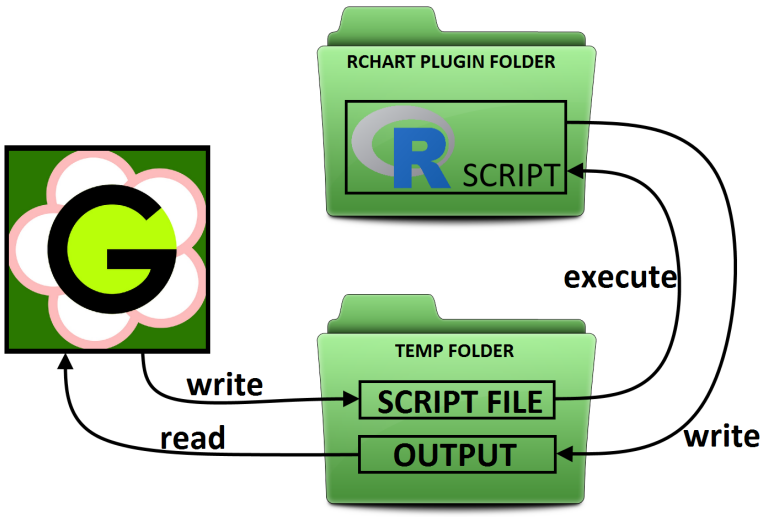

Figure 1: R to GroIMP connection scheme

2.2 Class Definition

In order to create a new plot the user has to create an instance of the class **Rchart**. For that there are 2 constructors available. Namely:

- Rchart(Dataset dataset, boolean interactive)
- Rchart(DatasetRef dsr, boolean interactive)

The variables *dataset* and *dsr* represent the data or the reference on data as the base for the plot. The variable *interactive* determines if the plot-window is event-driven. That means if set true the plot automatically updates when the dataset changes. Contrary if set to false the user manually has to call the function *show()* on the respective instance for plotting. The title for the plot is taken over from the dataset. To get or set the dataset or chage the title there are the getters and setters available accordingly. It was chosen to make the R charting plugin ggplot2 directly available as Java wrapper functions. Thus to set up the plot the user can directly call the appropriate ggplot2 functions. The only difference is that the minimum data mapping needed for the ggplot2 geom-functions needs to be done in the arguments the functions are called with. Furthermore some convenience functions have been implemented for users being inexperienced with R syntax. This holds true for general plot settings as well as for properties of single geom-functions. The whole class documentation can be found in the Rchart's Javadoc file. It is highly recommend to consider the ggplot2 documentation from R in order to see how plots are built up and also to be aware of the range of different plot types.

2.3 Features

In this section the implemented features are described.

2.3.1 Convenience functions

As mentioned in the class definition chapter, plot properties are set by calling the corresponding ggplot2 functions. However for user-friendliness it was chosen to also implement convenience functions for the most often used plot setting tasks.

General plot property settings:

- *setTitle* - set the window title, also wrapper for *ggtitle()*
- *setXAxisLabel* - set x axis label, wrapper for *xlab()*
- *setYAxisLabel* - set y axis label, wrapper for *ylab()*
- *setZAxisLabel* - set z axis label, wrapper for *zlab()*
- *setXAxisLimits* - set x axis limits, wrapper for *xlim()*
- *setYAxisLimits* - set y axis limits, wrapper for *ylim()*
- *setZAxisLimits* - set z axis limits, wrapper for *zlim()*
- *setLegendPosition* - set legend position, wrapper for *theme(legend.position='...')*

ggplot2 geom functions:

- *LinePlot* - line plot, wrapper for *geom_line()*
- *Histogram* - histogram plot, wrapper for *geom_histogram()*
- *BarPlot* - bar plot, wrapper for *geom_bar()*
- *DotPlot* - dot plot, wrapper for *geom_point()*
- *StepPlot* - step plot, wrapper for *geom_step()*
- *BoxPlot* - box plot, wrapper for *geom_boxplot()*
- *AreaPlot* - area plot, wrapper for *geom_area()*
- *FittedLinePlot* - fitted line plot, wrapper for *geom_smooth(method = 'loess')*
- *HeatMap* - heat map plot, wrapper for *geom_tile()*

For a single geom/plot option:

- *setLineColor* - set line color, wrapper for ggplot2 geom aesthetics mapping
- *setLineWidth* - set line width, wrapper for ggplot2 geom aesthetics mapping
- *setPointShape* - set point shape, wrapper for ggplot2 geom aesthetics mapping
- *setLineType* - set line type, wrapper for ggplot2 geom aesthetics mapping
- *aes* - aesthetics mapping function, adds additional arguments to ggplot2 geom aesthetics mapping

Other plot possibilities not offered by ggplot2:

- *Table* - table plot, using plugin gridExtra
- *ScatterPlot3D* - 3d scatter plot, using plugin scatterplot3d
- *LinePlot3D* - 3d line plot, using plugin plot3D

2.3.2 Plot window

The main window is the plot window. It is dock-able hence can stand alone or be docked everywhere in GroIMP. At the top the menu entries for interactive handling can be found. The caption of the window is given by the dataset or user-definition. Figure 2 shows the plot window.

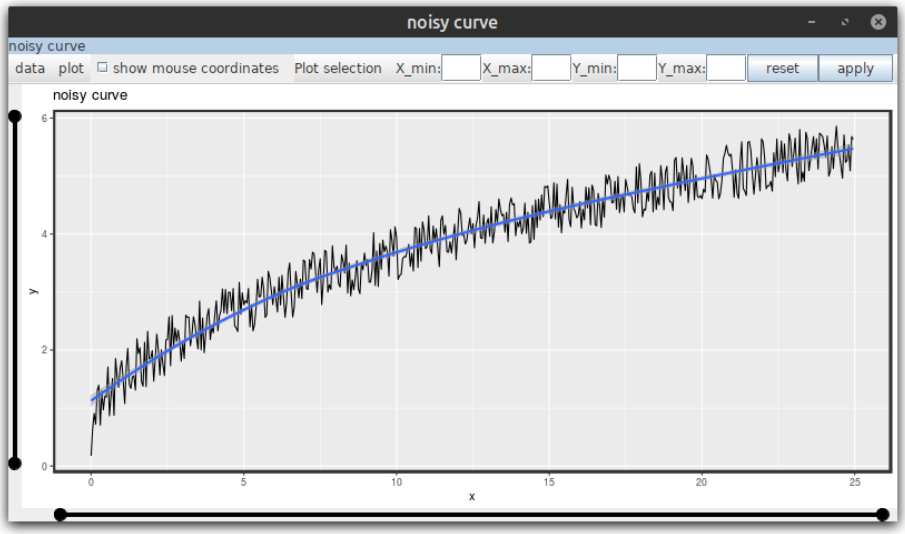

Figure 2: Plot window

2.3.3 Show mouse coordinates

At the top a checkbox with the label "show mouse coordinates" can be found. When checked the position of the mouse in the coordinate system is displayed. This option is only available for 2D plots. An example can be found in figure 3.

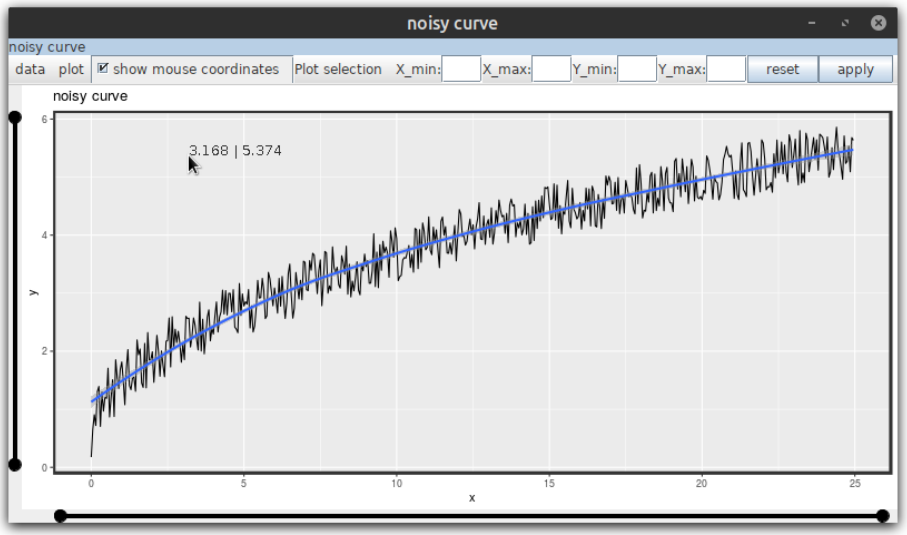

Figure 3: Mouse coordinate display

2.3.4 Axis limitation

There are three ways of setting axis limitations (also used for zooming) for 2D plots. At the top of the window in 4 text boxes the user can type in the desired range. This can be done for one or two axes. After that the apply button must be clicked. To reset the limitation the reset button can be used.
Zooming also is possible by using the mouse. By holding the left mouse button down and moving the range can be selected interactively. To reset the zoom the user can press the right mouse button or use the reset button.
Finally horizontal and vertical sliders represent the third way of limiting the plot view. All three possibilities can be seen in figure 4.

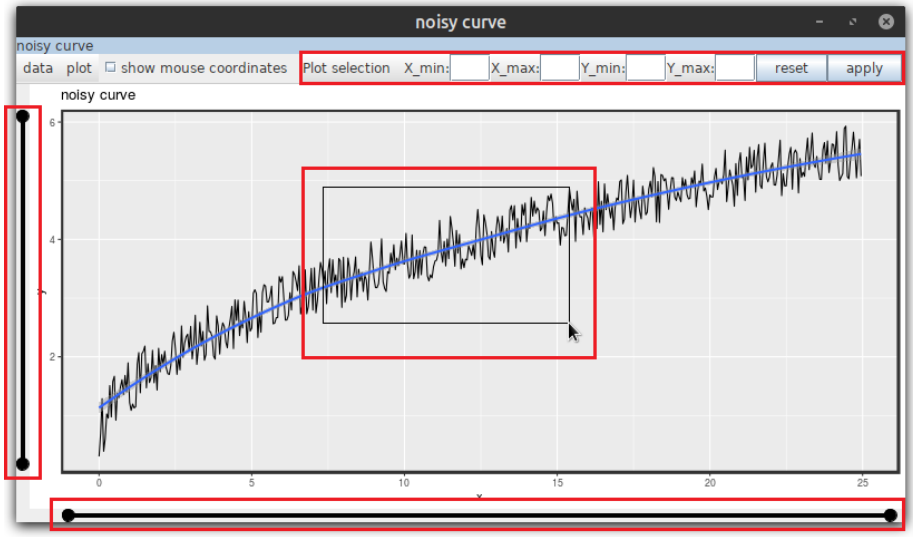

Figure 4: Axis limitation possibilities

2.3.5 Data export

To export the dataset that is associated with the plot the user must use the export menu entry. Three file types can be selected: CSV, HTML or RDATAFRAME. If intended the exported data can also be tailored to the axis limitations. Hence beneath the export menu entry the entry "export selection" can be found. Figure 5 shows the menus.

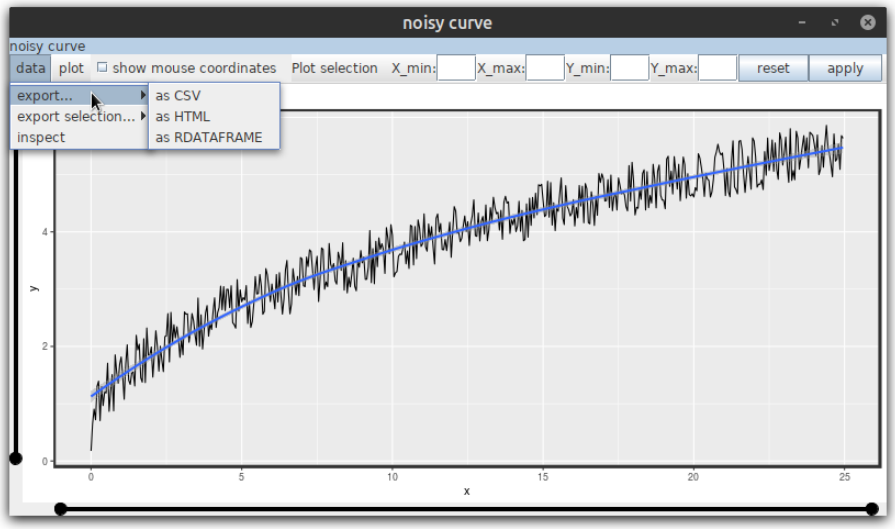

Figure 5: Data export menus

2.3.6 Plot image export

The resulting plot can be exported as an image file. In the menu "plot" the entry "save plot image" can be found. The user can chose from the following file types: JPG, PNG or GIF. Furthermore the desired image dimensions (pixels) must be keyed. In figure 6 the menu is shown.

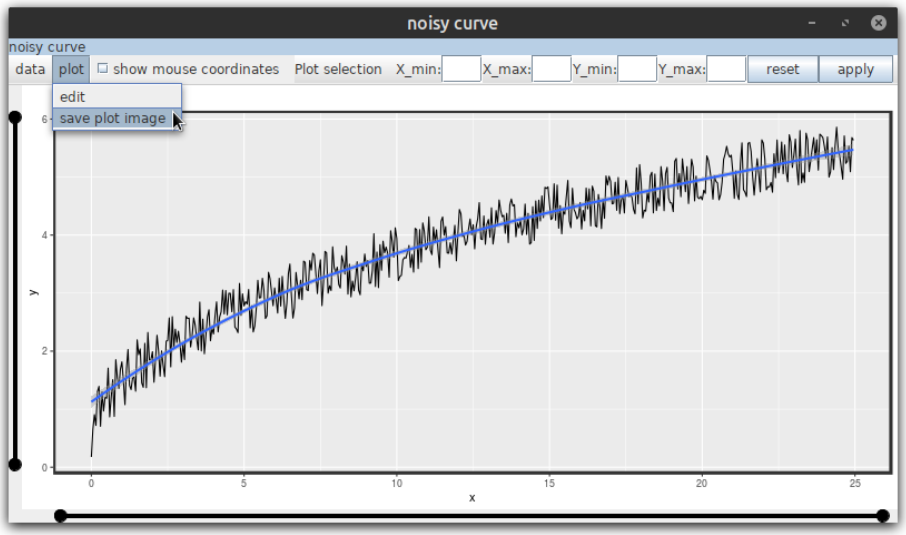

Figure 6: Plot image export menu

2.3.7 Data inspection

For the possibility of inspecting the dataset it was chosen to implement a shiny app. Thereby a platform independent browser application is provided which makes it possible to open an interactive data table. To open it the user must use the sub menu entry "inspect" in the menu "data". Within the browser application the user can select columns, search for cell entries and also restrict cells of a single column to a data range. Figure 7 gives an example for such a table.

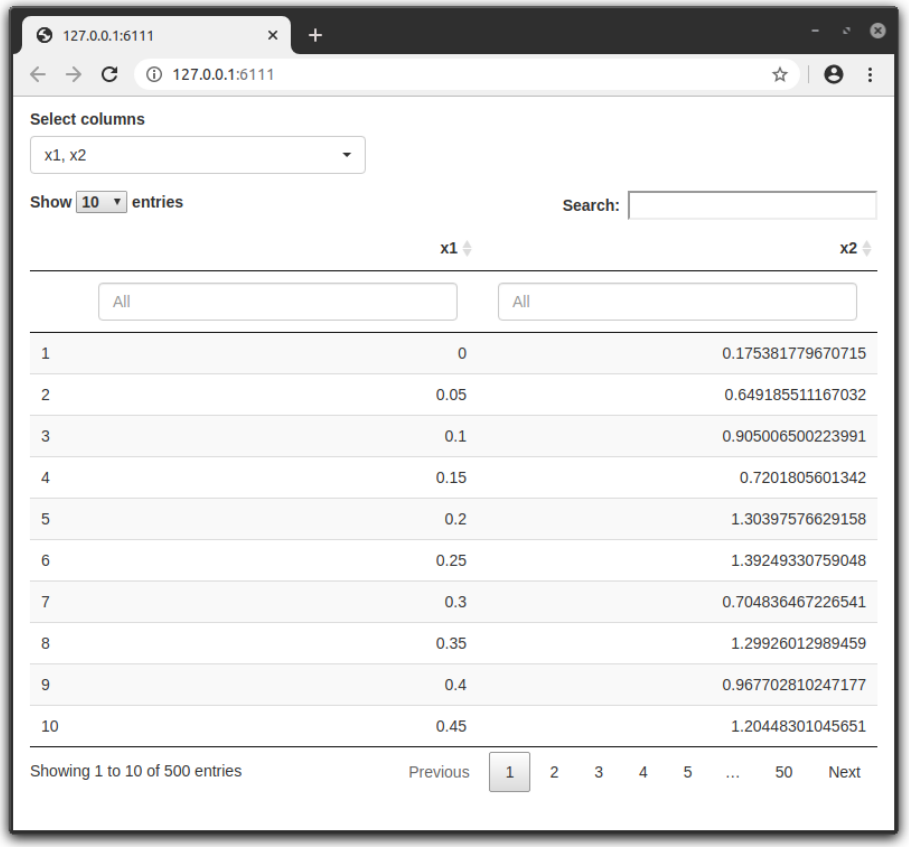

Figure 7: Data inspection - shiny app

2.3.8 Plot editing

In order to be able to interactively edit plots a dialog where the user can add or edit plot commands was implemented. To open the dialog the menu entry "edit" under the menu "plot" can be found. On the left side of the dialog a list of the single commands that constitute the plot properties are located. On the top buttons for adding or deleting single commands can be seen. On the right side single properties can be edited when they are applicable. An example can be found in figure 8.

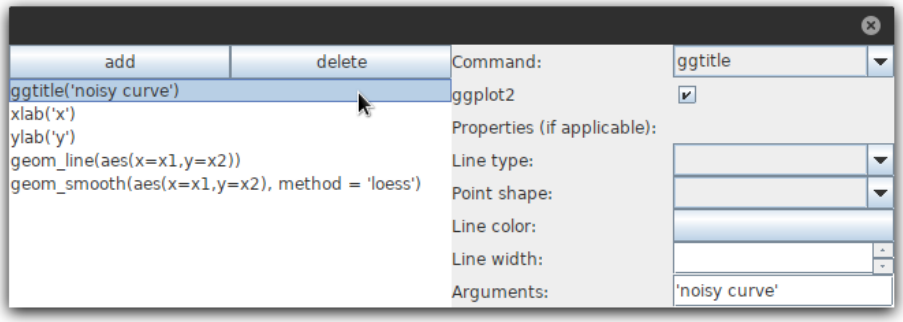

Figure 8: Plot editing dialog

2.3.9 GROGRA analysis functions - shiny version

Facing the advantages of the shiny data inspection browser application it is advantageous to have such application for the GROGRA analysis functions as well. Because of that in the corresponding dialog - opened by the menu entry "GROGRA functions" - the menu entry "Analysis (shiny)" was added. Figure 9 shows the menu.

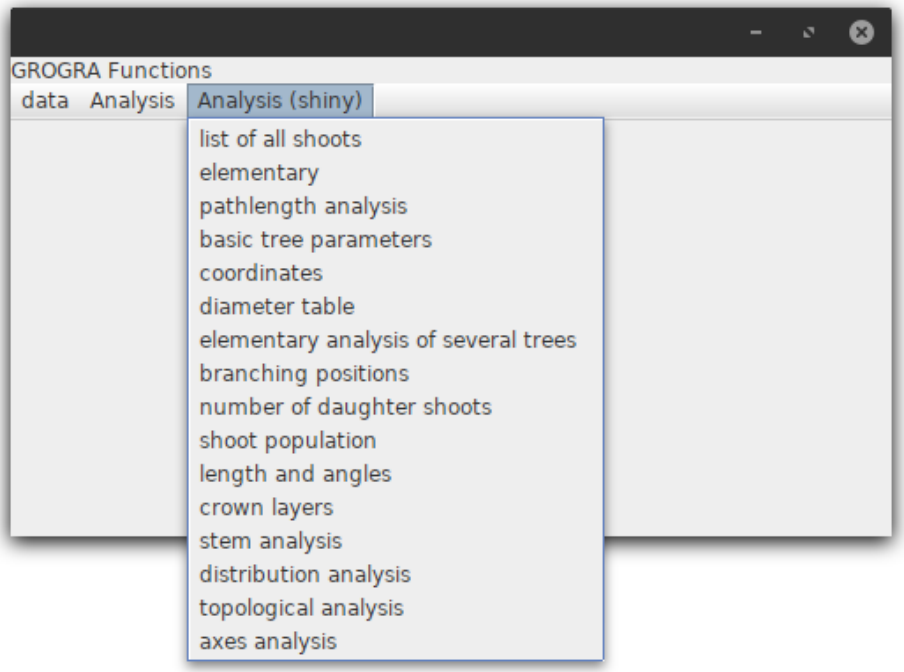

Figure 9: GROGRA analysis functions - shiny table app version

2.3.10 Example section supplementation

The example section was supplemented by 5 examples that show use-cases of Rchart. In figure 10 the examples can be seen.

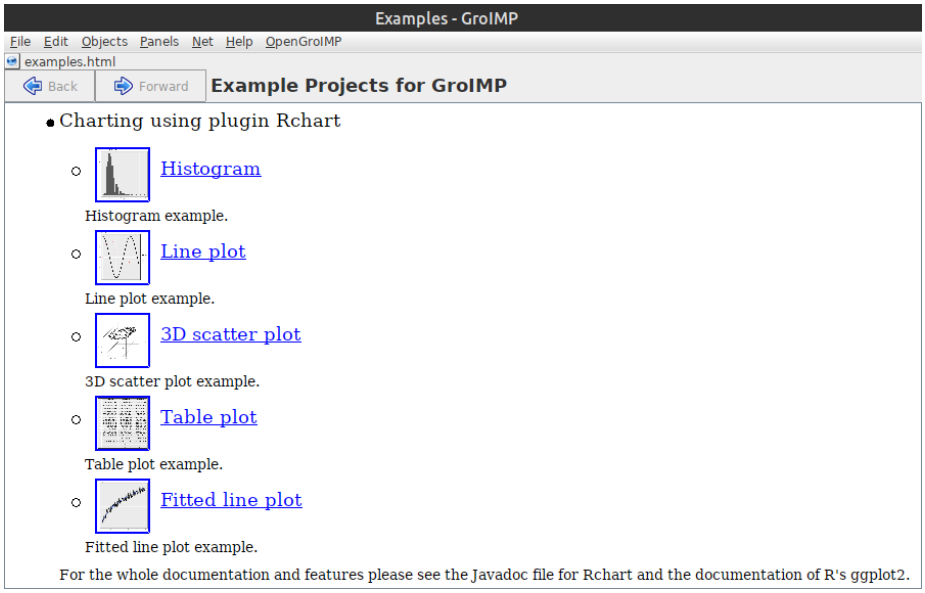

Figure 10: Rchart examples section

2.3.11 R installation/plugin check

When GroIMP runs on Linux or Mac there is the prerequisite of the installation of R and the presence of certain plugins. In order to provide user-friendliness in case of non-compliance error messages are provided. Theses messages give the precise way of proceeding to overcome this deficiency. Figures 11 and 12 show these dialogs.

Figure 11: R installation missing error message

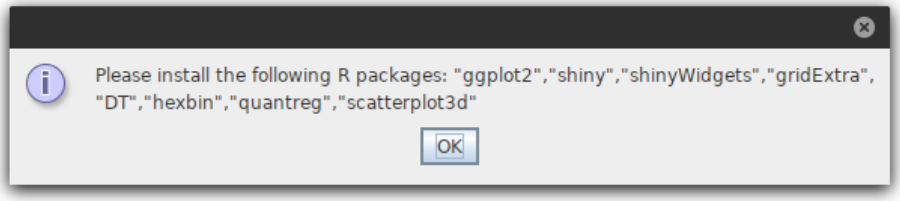

Figure 12: R packages missing error message