# GrapheR: a Multiplatform GUI for Drawing Customizable Graphs in R

*by Maxime Hervé*

**Abstract** This article presents **GrapheR**, a Graphical User Interface allowing the user to draw customizable and high-quality graphs without knowing any R commands. Six kinds of graph are available: histograms, box-and-whisker plots, bar plots, pie charts, curves and scatter plots. The complete process is described with the examples of a bar plot and a scatter plot illustrating the legendary puzzle of African and European swallows' migrations.

## Observation and aims

Unlike statistical software based on a Graphical User Interface (*GUI*), such as Minitab® or Stata®, the R language allows you to control exactly how your data will be displayed. This comes at a cost: you must know the exact arguments of the commands to get the result you want.

In this context, beginners find that drawing elaborate graphs is often a long and difficult process made up of lots of trials and slight code modifications. Some well-known R GUIs already exist (e.g. R Commander (Fox, 2005), **JGR** (Helbig et al., 2005), Sci-Views R (Grosjean, 2010), Rattle (Williams, 2009) or **playwith** (Andrews, 2010)), but the graphs they produce have mainly an explorative function in data analysis, and are not customizable to get publication-ready material.

Therefore, many R users, especially beginners (but not only), go back to Excel®-like software and their clickable interfaces to quickly draw the graphs they want to publish. In the compromise between speed and simplicity vs. graph quality, they must sacrifice graph quality in the process. This is regrettable, because R can do lots of things Excel® cannot, by combining high and low-level graphical functions. The goal of the **GrapheR** (Herve, 2011) package is therefore to combine the simplicity of a GUI with the powerful capabilities and the graphical quality of R.

To be immediately accessible to beginners, **GrapheR** does not require any knowledge of the R language. Indeed, the loading of the dataset, the choice of the variables to be represented and the configuration of all graphical options are made with menus, checkboxes and other clickable tools.

The visual structure and functioning of the GUI are entirely based on the **tcltk** package developed by Peter Dalgaard (Dalgaard, 2001, 2002), which adapts the Tcl/Tk language to R.

## Lauching the interface

As with any other package, **GrapheR** is loaded via `library(GrapheR)`, `require(GrapheR)` or the '`Packages`' menu of the Windows R GUI.

Launching the interface is the only step that requires the user to enter a command: `run.GrapheR()`. The interface opens and the console can now be reduced.

## Description of the interface

The interface is divided into three blocks (Figure 1):

1. The navigation bar: it contains seven groups of buttons, each corresponding to one (obligatory or optional) step of the process. From left to right:
   A. Loading and modifying the dataset.
   B. Setting up a graph. From left to right: histogram, box-and-whisker plot, bar plot, pie chart, curve and scatter plot.
   C. Opening a new graphics device: when a graph is drawn, it is in the active window or in a new device if none is open. This button allows the user to open a new graphics device in which several graphs can be drawn.
   D. Draw a graph.
   E. Adding elements to the graph. From left to right: add a horizontal line, add a vertical line, add any other kind of line, add a theoretical distribution curve, add text, and add *p*-values.
   F. Saving graph(s).
   G. Divers options: user language and help.
2. The messages frame: when the mouse is over some specific elements of the GUI or when some specific actions are performed, messages are displayed in this frame. Three kinds of message can be displayed:
   - in blue: informative messages,
   - in green: warnings, for when particular attention should be paid to a specific point (for example: if this option is chosen, then this one can not be),
   - in red: error messages, for when the action requested can not be completed. The message indicates the origin of the error.

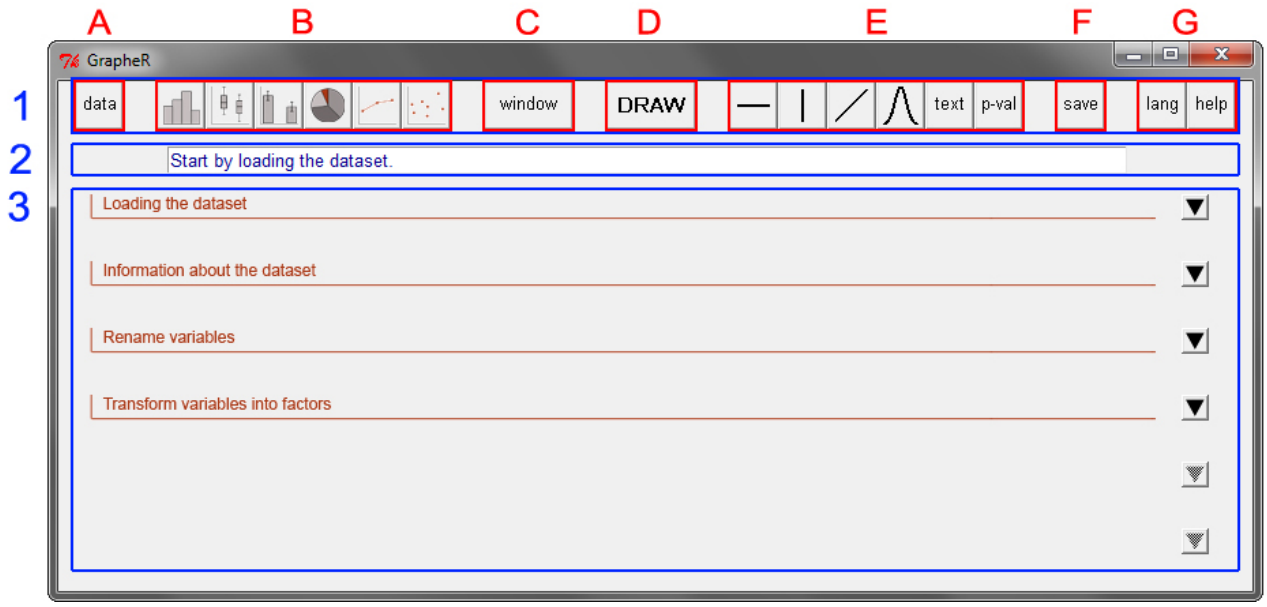


Figure 1: User interface – global view (display under Windows 7)

3. The settings block: it is divided into four to six sub-blocks, each containing options relating to a given theme: general parameters of the graph, title of the graph, legend... Each sub-block can be opened or closed with the corresponding arrow situated on the right of the interface. Of course, defined settings are not lost when a sub-block is closed.

## Use

The examples shown here are based on the dataset provided in the package, called '`Swallows`'. To load it, use the command `data(Swallows)`. This (fictional!) dataset exemplifies the legendary puzzle of African and European swallows' migrations (Gilliam and Jones, 1975).

### Loading and modifying the dataset

The first sub-block (Figure 2) allows you to load the dataset. Data can be imported from an external file ('txt' and 'csv' extensions are available so far) or can be an already existing R object of class `"data.frame"` (i.e. a table).

The next sub-block allows you to get information on the dataset structure. When a variable is selected in the list on the left, its type (`numeric`, `factor`, `logical`...) and its summary are displayed in the frame on the right.

The next two sub-blocks allow the dataset to be modified if needed:

- by renaming variables (for example if the dataset does not contain their names),
- by converting variables into factors. The conversion can be applied to variables of class `"character"` or to numeric variables (in this case values can be grouped into classes). The latter case is necessary when a factor is numerically coded, e.g. a binary factor (0/1), otherwise R would treat it as a numeric variable.

### Setting up a graph

Once the dataset is ready, click on the button corresponding to the type of graph to be drawn (histogram, box-and-whisker plot, bar plot, pie chart, curve or scatter plot).

Whatever the type of graph chosen, all parameters have a default value except general parameters – which correspond to variable(s) to be represented. Hence, to quickly draw a graph, only general parameters must be defined. If they are not (or not all) defined, an error occurs when trying to draw the graph.

Here the aim is to draw two graphs: a barplot displaying a size comparison of African and European swallows, and a scatter plot displaying the relationship between weight and size in the two species. Let's begin with the barplot. We start by clicking on the '`Bar plot`' button in the navigation bar.

We want to show mean sizes of swallows depending on their species. In the general parameters, we hence choose the '`Means`' type (bar plots can also display proportions), '`Size`' as the variable to be represented and '`Species`' as the obligatory factor (Figure 3). A second, optional, factor could be added, if for example we would like to display mean sizes of swallows depending on their species and their sex (you can try to do it, the dataset contains a '`Sex`' factor).

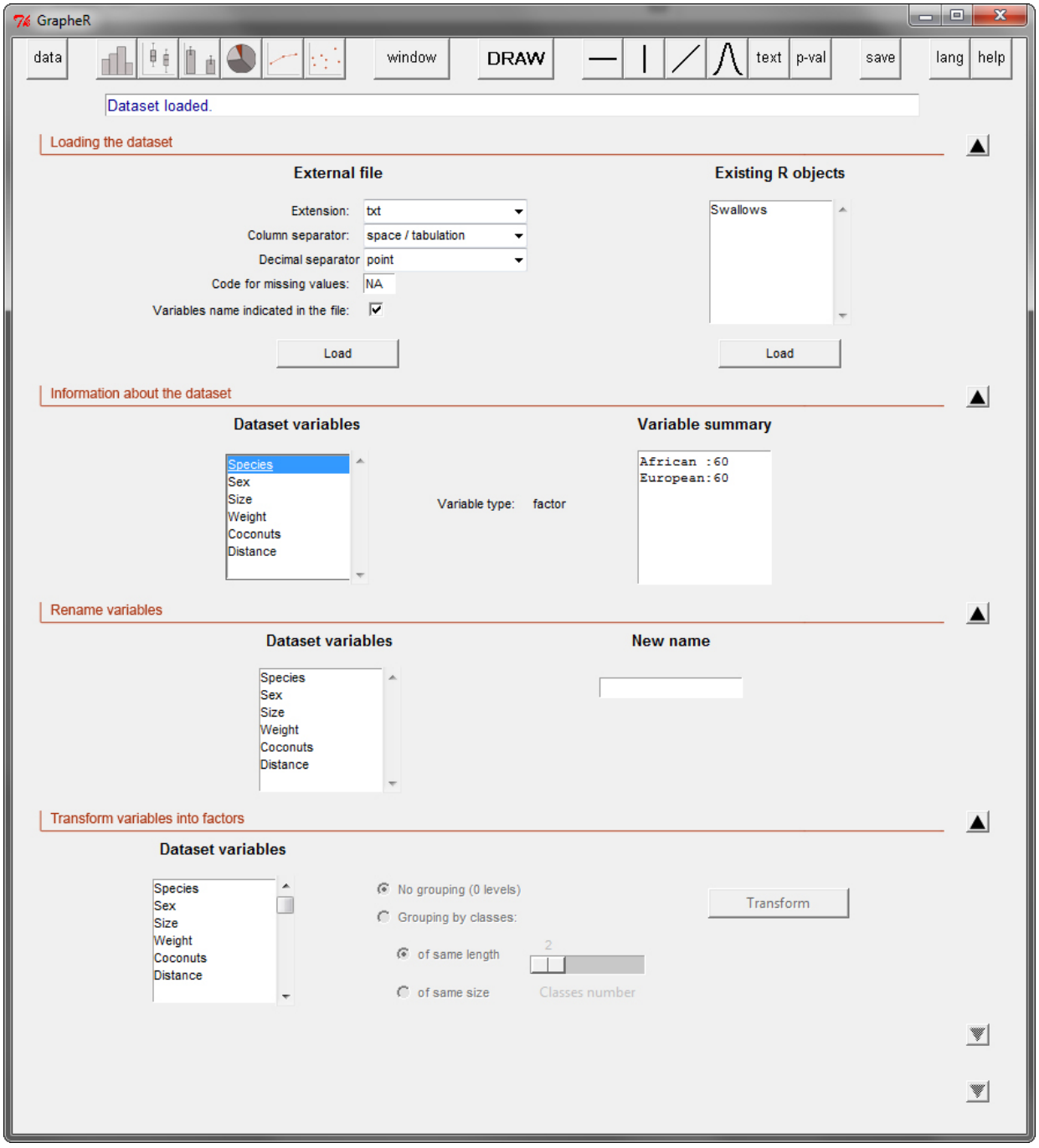


Figure 2: Loading the dataset

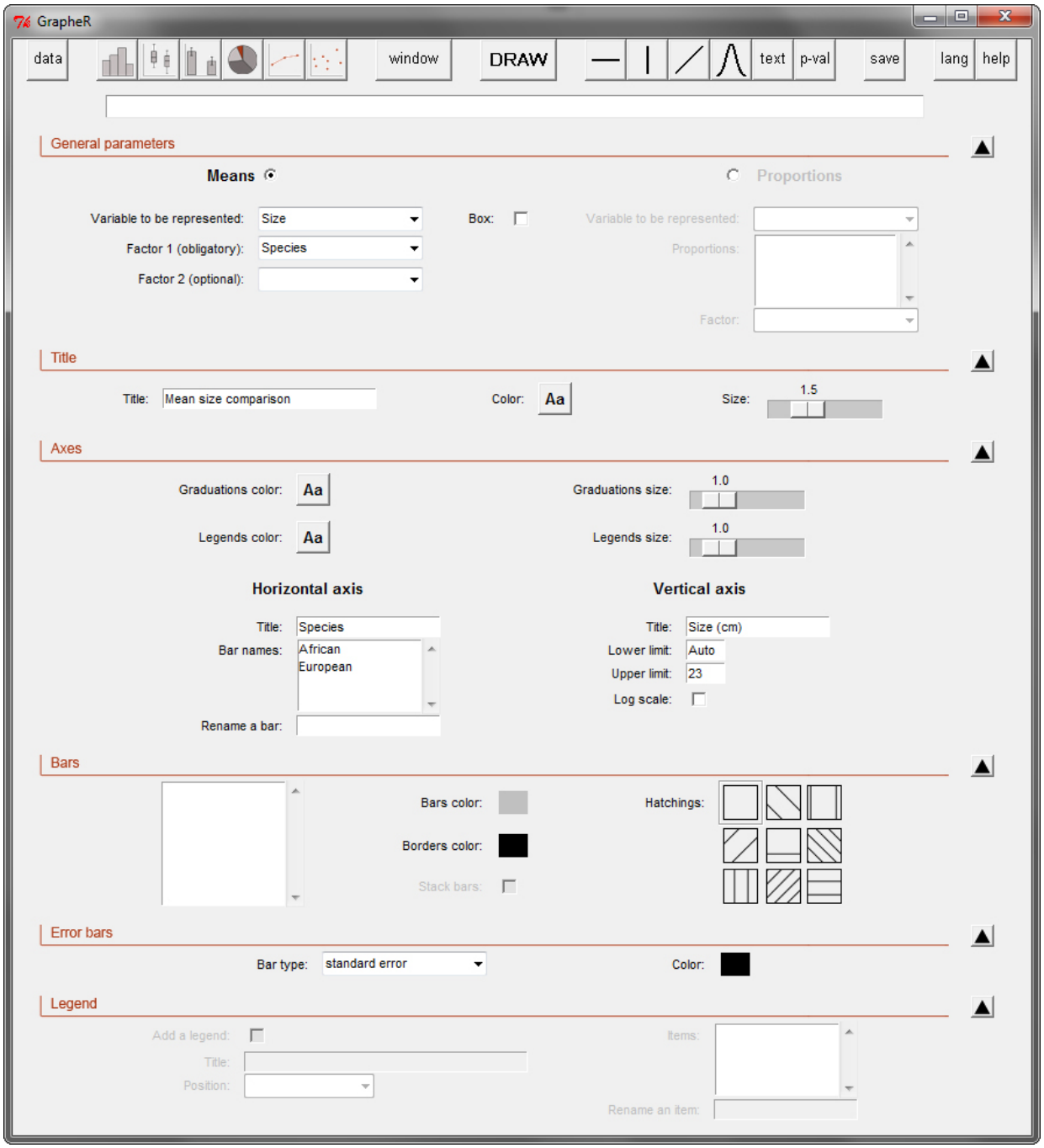


Figure 3: Illustrative bar plot

Graph title and axis legends are optional (no text is written by default). In our example we call our graph '`Mean size comparison`' and name the axes '`Species`' (horizontal axis) and '`Size (cm)`' (vertical axis). Bar names are by default the names of the corresponding levels of factor 1. However we could change this, which would only change the names *displayed* on the graph (and not the ones in the dataset). This allows you to use names containing several words, spaces or accents on your graph (which is either impossible or strongly discouraged for object names).

Lower and upper limits of the vertical axis can be changed, and a logarithmic scale can be used. If limit values are left on '`Auto`', **GrapheR** will adjust them automatically. Here the lower limit is left on '`Auto`', whilst the upper limit is set to '`23`', for reasons explained later.

If no second factor is defined, all bars are by default solid white with a black border and no hatchings. We can set these three parameters (bar color, border color and hatchings), but in this example we simply change bar color to a classic grey. If a second factor is defined, one color/hatching pattern can be attributed by level of this factor. In this case colors are by default set to different shades of grey (borders are always black). Bars can be stacked in this situation, but if you choose this option, no error bar can be drawn.

Means and percentages in scientific bar plots are nearly always supplied with error bars, but getting those bars right using the command line is notoriously challenging for R beginners. **GrapheR** makes it easy to draw error bars to represent either the standard deviation, the standard error of the mean or the 95% confidence interval of the mean. Here we choose for example '`standard errors`'.

When a second factor is defined, a legend box can be added to the graph. Its items are by default the levels of this factor. As for bar names, legend items can be renamed, which will not change actual level names in the dataset. A title can also be added to the legend. Finally, its position on the graph can be set.

Once all options have been thus configured, just click on the '`DRAW`' button of the navigation bar... *et voilà!* By default the graph is produced in the active graphics device, but here we want to display two graphs in the same window. Hence we click on the '`window`' button of the navigation bar before drawing our bar plot.

## Opening a new graphics device

A dialog box opens on the right of the interface, allowing specification of the number of graphs to be drawn in the device to be created, and the background color of this device (Figure 4).

It is possible to draw up to 16 graphs in the same device, shared between four rows and four columns. However, the larger the number of graphs to be drawn, the smaller the space allocated to each.

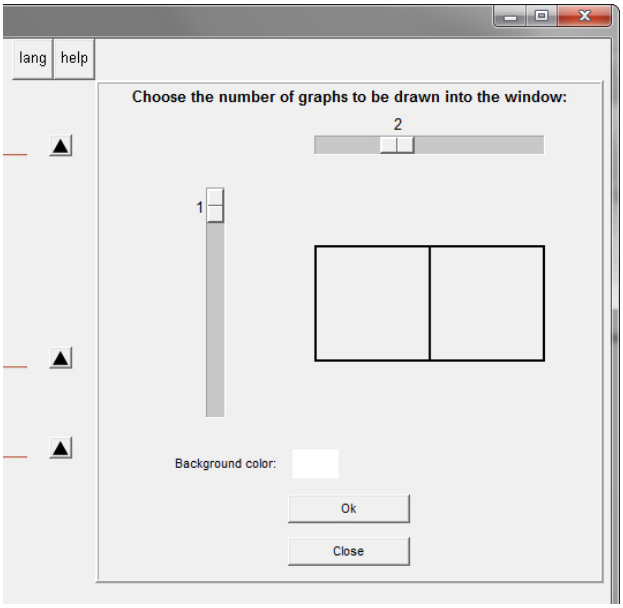


Figure 4: Opening a new graphics device

After creating the new graphics device, we can draw our graph by clicking on the '`DRAW`' button.

## Adding elements to the graph

Once the graph is drawn, elements can be added to complete it:

- one or several horizontal line(s)
- one or several vertical line(s)
- any other kind of line(s)
- one or several theoretical distribution curve(s): only on density histograms
- text
- *p*-values: only on bar plots

These elements are always added to the last graph drawn.

For each element, clicking on the corresponding button in the navigation bar opens a dialog box on the right of the interface.

In our bar plot example, let's assume that we had used R beforehand to perform a statistical test to compare the sizes of our two species. We can now use the *p*-values tool to add the result of this test on the graph: just click on the '`p-val`' button.

Two designs are available, depending on the number of bars to be compared (Figure 5). Whatever the design, the process is the same: enter the text you want, customize its size and color if needed, and finally click on the '`Select`' button. Then just click on the bars to be compared on the graph.

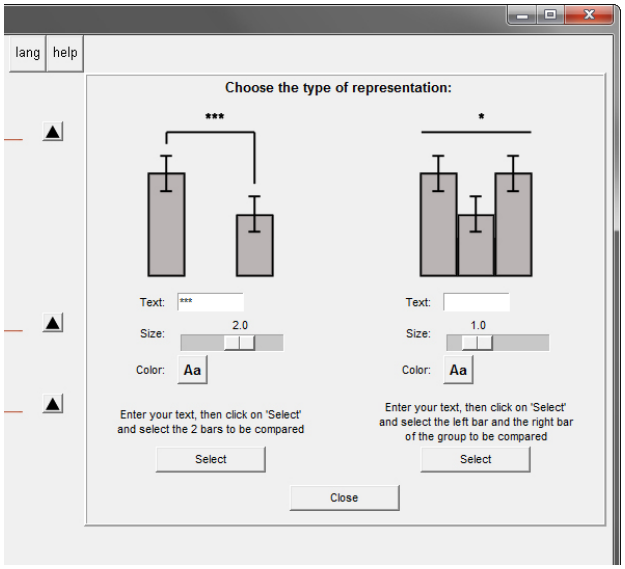


Figure 5: Adding *p*-values to the bar plot

The result, sometimes, will not look good the first time. It is because there was not enough space above the bars to add text. If you want to add text above your graph, remember to allow some space by setting a higher upper limit for the vertical axis (this is the origin of the value '23' in Figure 3). If you omitted this, no problem, drawing the graph again will be a matter of seconds in **GrapheR**. The first graph is now ready (left part of Figure 6).

## Second graph

The second graph we want to draw is a scatter plot displaying the relationship between weight and size in the two species. We start by clicking on the 'Scatter plot' button of the navigation bar.

In the general parameters, we define 'Weight' as the X variable and 'Size' as the Y variable (Figure 7).

To draw a scatter plot by species, we add an optional factor, 'Species', and select its two levels from the list. Finally, for aesthetic reasons we choose to draw a box around the graph by ticking the 'Box' checkbox.

We add a title for the graph ('Weight - size relationship'), for the horizontal axis ('Weight (g)') and for the vertical axis ('Size (cm)').

For each scatter plot, a different point symbol (as well as its color and size) can be defined. By default all symbols are empty circles. When only one scatter plot is drawn the default color is black, whereas when several are drawn colors are set to different shades of grey. Here we choose full circles for the two scatter plots and change only the second color (corresponding to the 'European' level) to a dark red.

A line/curve can be added by scatter plot, chosen among different types:

- a linear regression line (performed with the least squares method)
- a linear regression line (performed with the least rectangles method)
- a quadratic regression line
- a simple tendency curve

Here no variable is independent or dependent, but both are interdependent. We hence choose to add a linear regression line performed with the least rectangles method to each scatter plot. Among the three available line types (full, large dashed or fine dashed), we select 'full'.

We add a legend box by ticking the 'Add a legend' checkbox. We give it a title ('Species') and set its position to 'top - left'. We leave the default items, which correspond to levels of the factor defined in the general parameters.

Finally, we click on the 'DRAW' button of the navigation bar... the second graph is born (Figure 6)!

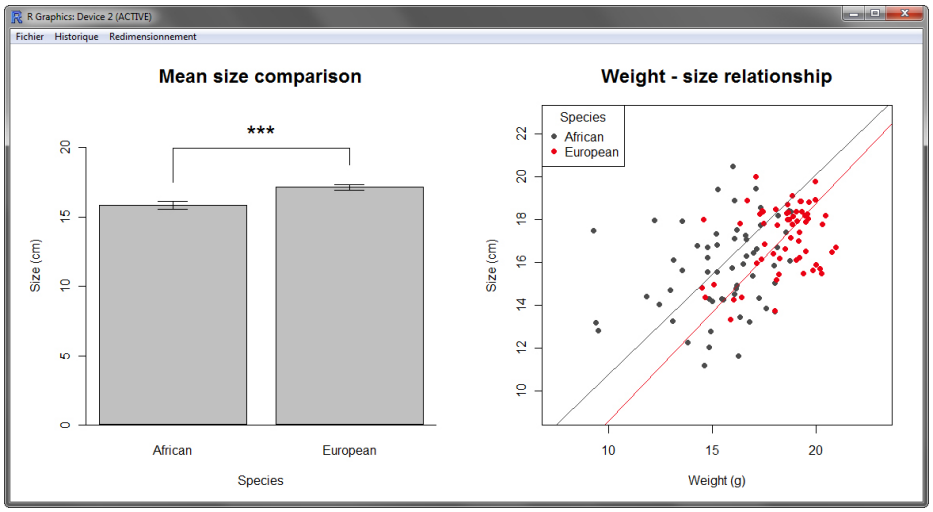


Figure 6: Final plots

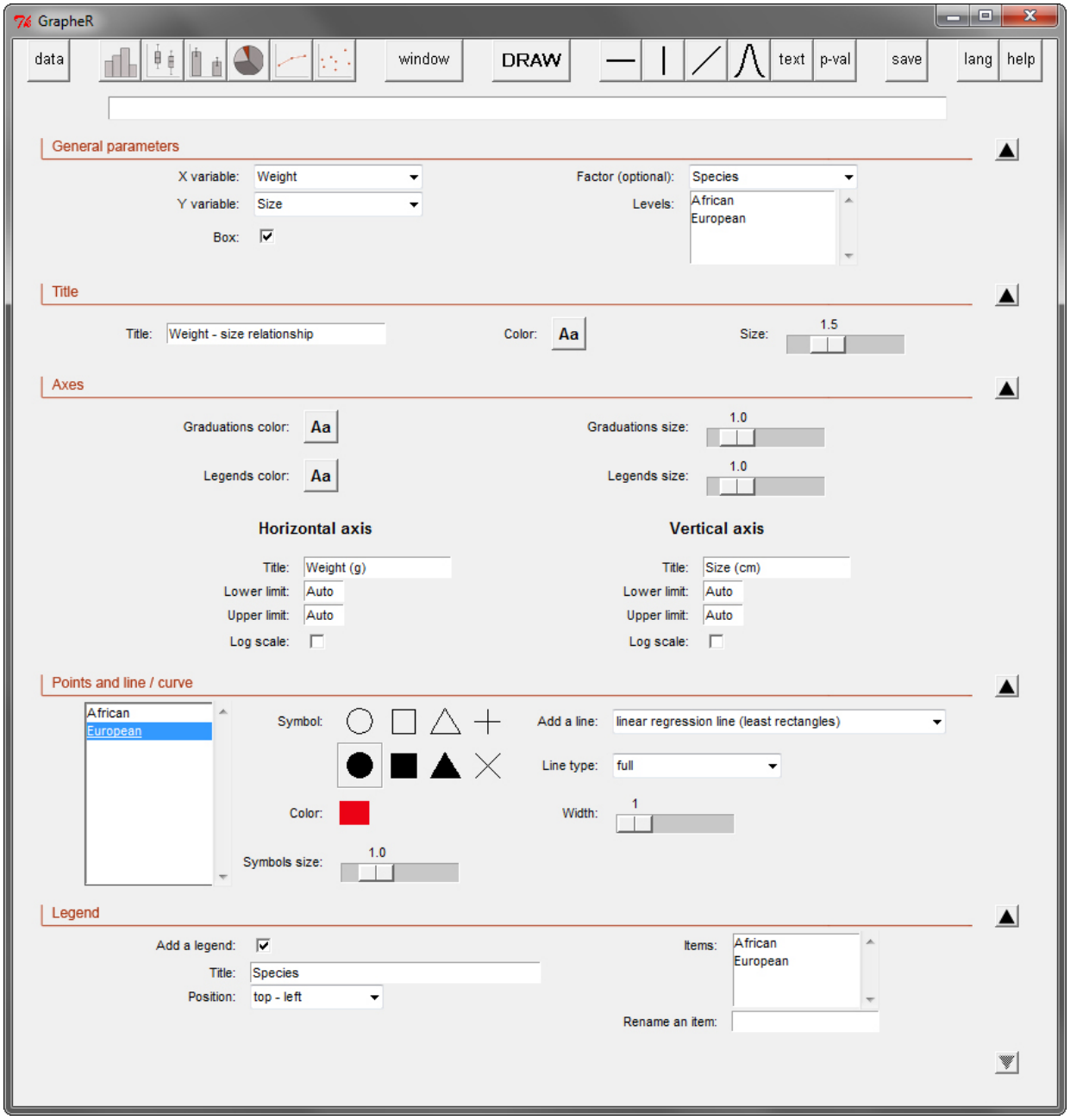


Figure 7: Illustrative scatter plot

### Saving graph(s)

Once all graphs are drawn, they can be saved by clicking on the 'Save' button of the navigation bar. In the dialog box opening on the right of the interface, select the device containing the graph to be saved (Figure 8). Then choose the extension of the file to be created ('jpg', 'png' and 'pdf' are available) and the image length. Image height is automatically calculated from the length value. Finally click on the 'Save' button.

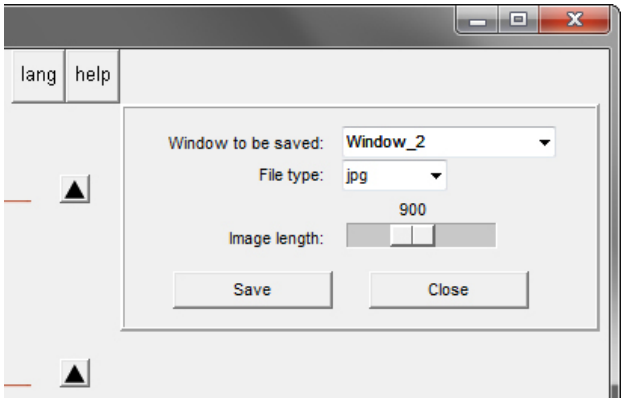


Figure 8: Saving graph(s)

### Changing the user language

To change the user language, click on the 'lang' button of the navigation bar. A dialog box opens on the right of the interface (Figure 9). Choose the desired language in the drop-down menu. To fix your preference in the future, tick the 'Save the preference' checkbox. Click on the 'Ok' button to validate. The interface is closed and re-opened in the chosen language (but note that any dataset loaded in the previous language session is lost and has to be re-loaded).

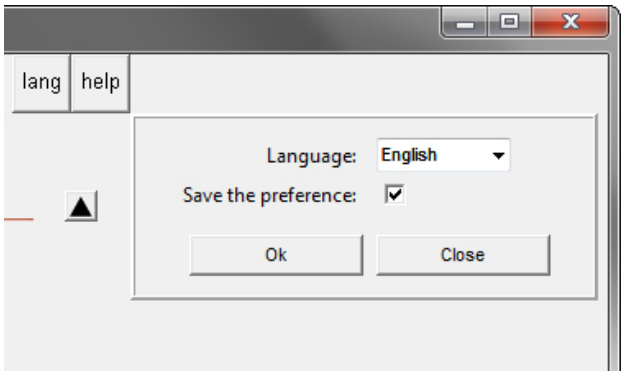


Figure 9: Changing the user language

## Implementing GrapheR in another language

Implementing **GrapheR** in another language is very easy, because no word appearing in the interface is written in the code. The button names, etc. come from an external file which is loaded depending on the language setting. In the current version (1.9-66), only English and French are available (files 'Language_en.csv' and 'Language_fr.csv' in the 'lang' directory).

Therefore, adding a new language just requires a (strict) translation of each line of one of the existing files (including spaces before and/or after words). The new file must then be saved as 'Language_XX.csv'.

If you want to implement **GrapheR** in your language, you are most welcome. In that case, remember that it would be a good idea (but a tougher job) to also translate the user manual (contained in the 'doc' directory). If you want to participate, do not hesitate to contact me. I will take care of all screenshots to be included in the manual, and then add the link to the new language file in the code.

## Acknowledgments

I am very grateful to Denis Poinsot, Dennis Webb, Clément Goubert and two anonymous referees for their precious advice on GUI ergonomics, **GrapheR** English translation and previous versions of this manuscript.

## Bibliography

F. Andrews. playwith: A GUI for interactive plots using GTK+. R package version 0.9-53. URL http://code.google.com/p/playwith/.

P. Dalgaard. A Primer on the R-Tcl/Tk Package. *R News*, 1(3):27–31, 2001. URL http://journal.r-project.org/.

P. Dalgaard. Changes to the R-Tcl/Tk Package. *R News*, 2(3):25–71, 2002. URL http://journal.r-project.org/.

J. Fox. The R Commander: A Basic-Statistics Graphical User Interface to R. *Journal of Statistical Software*, 14(9), 2005. URL http://www.jstatsoft.org/.

T. Gilliam and T. Jones. Monty Python Holy Grail. EMI Films, UK.

P. Grosjean. *Sci-Views-R: A GUI API for R*. UMONS, Mons, Belgium, 2010. URL http://www.sciviews.org/SciViews-R/.

M. Helbig, M. Theus, and S. Urbanek. JGR: Java GUI for R. *Statistical Computing and Graphics Newsletter*, 16(2):9–12, 2005. URL http://stat-computing.org/newsletter/.

M. Hervé. GrapheR: A multiplatform GUI for drawing customizable graphs in R. R package version 1.9-66. URL http://cran.r-project.org/package=GrapheR.

R. Ihaka and R. Gentleman. R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. URL http://www.amstat.org/publications/jcgs/.

G. J. Williams. Rattle: A Data Mining GUI for R. *The R Journal*, 1(2):45–55, 2009. URL http://journal.r-project.org/.

*Maxime Hervé*
*UMR 1099 BiO3P (Biology of Organisms and Populations applied to Plant Protection)*
*University of Rennes 1 - Agrocampus Ouest - INRA*
*263 avenue du Général Leclerc, 35042 Rennes Cedex*
*France*
mx.herve@gmail.com