An introduction to R (demo lesson)

Jan Verbesselt

September 23, 2013

Abstract

This is an example of a lesson and tutorial for the R course. ... Below a short introduction to R can be found.

1 Getting started with R

When you open R you will see Fig. 1. The window in the top left corner is the R console (e.g. statistical and spatial analysis tools). Go to the menu, click on File > new script and a script window will appear. The interface should now look something like Fig. 2.

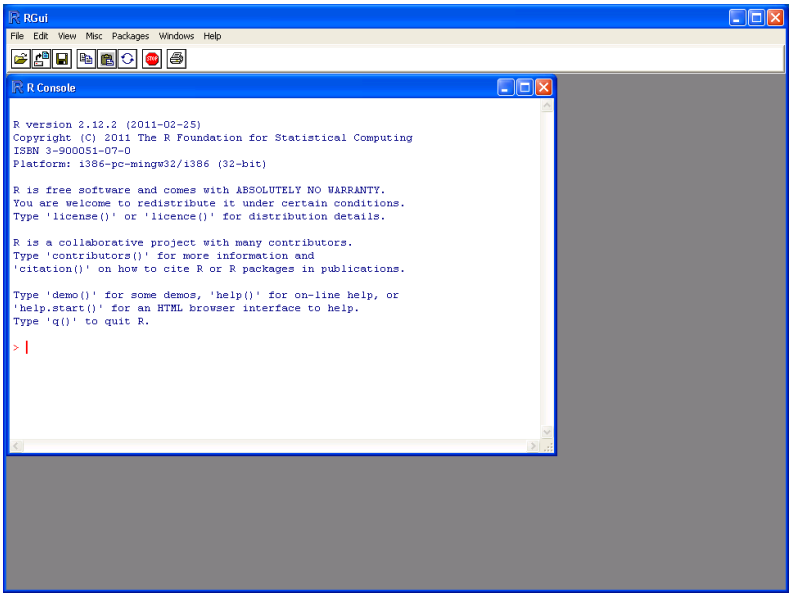


Figure 1: The graphical user interface to R

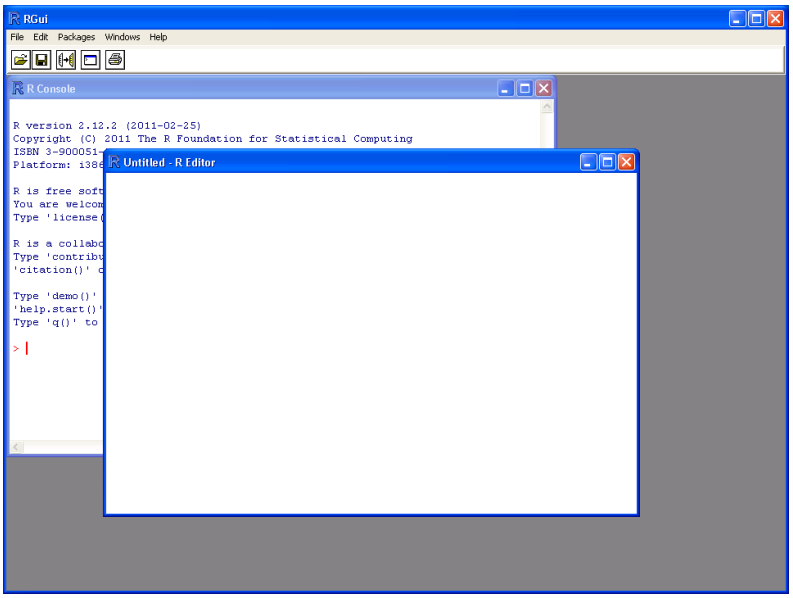


Figure 2: The graphical user interface with an empty script

Now type the following script in the script window:
a <- 1 a

You are now going to pass what you have written in your script to the console line by line and we will discuss what R is doing with your code. Select the first two lines with your mouse and then type **Ctrl-r**. The selected lines will be passed to the R console and your console should now look like something like this:

```
R> a <- 1
R> a

[1] 1
```

The first line you passed to the console created a new object named *a* in memory. The symbol '<-' is somewhat equivalent to an equal sign. In the second line you printed *a* to the console by simply typing it's name.

Now try to obtain he following output in the R console by writing the commands in the script window and running the via **Crtl-r** (make sure you remove > and +). R commands shown in the following section should be written in your Script file without > and + in front of it, so that you can run them by using **Crtl-r** and the result in the R console will look like this:

```
R> class(a)

[1] "numeric"

R> b <- 2
R> a + b

[1] 3
```

```
R> newfunc <- function(x, y) {
+          2*x + y
+  }
R> a2b <- newfunc(2, 4)
R> a2b
```

```
[1] 8
```

```
R> rm(a, b, newfunc, a2b)
```

You now have requested the **class** attribute of a and the console has returned the attribute: **numeric**. R possesses a simple mechanism to support an object-oriented style of programming. All objects (a in this case) have a class attribute assigned to them. **R** is quite forgiving and will assign a class to an object even if you haven't specified one (as you didn't in this case). Classes are a very important feature of the **R** environment. Any function or method that is applied to an object takes into account its class and uses this information to determine the correct course of action. A simple example should suffice to explain further. Select the next two lines using your mouse and pass these to the console using **Crtl-r**. The first line passed declares a new object b. The second line passed adds a and b together and prints the solution to the console. **R** has assessed the class attribute of a and b; determined they are both **numeric** objects, and; carried out the arithmetic calculation as requested.

The 4th line passed declares a new object **newfunc** (this is just a name and if you like you can give this function another name). It is a new function. Appearing in the first set of brackets is an argument list that specifies (in this case) two names. The value of the function appears within the second set of brackets where the process applied to the named objects from the argument list is defined.

Next, a new object $a2b$ is created which contains the result of applying **newfunc** to the two objects you have defined earlier. The second last R command prints this new object to the console. Finally, you can now remove the objects you have created to make room for the next exercise by selecting and running the last line of the code.

R is supported by a very comprehensive help system. Help on any function can be accessed by entering the name of the function into the console preceded with a ?. The easiest way to access the system is to open a web-browser. This help system can be started by entering **help.start()** in the R console. Try it and see what happens.

```
R> ?class
```

2 Making a graph in R

Below a simple example is show of how you create a graph in R. First we load the data.

```
R> attach(mtcars) ## add it to the search path in R
R> # you can find more info about the data set via ?mtcars
R> # the data itself looks like
R> wt # weigth
```

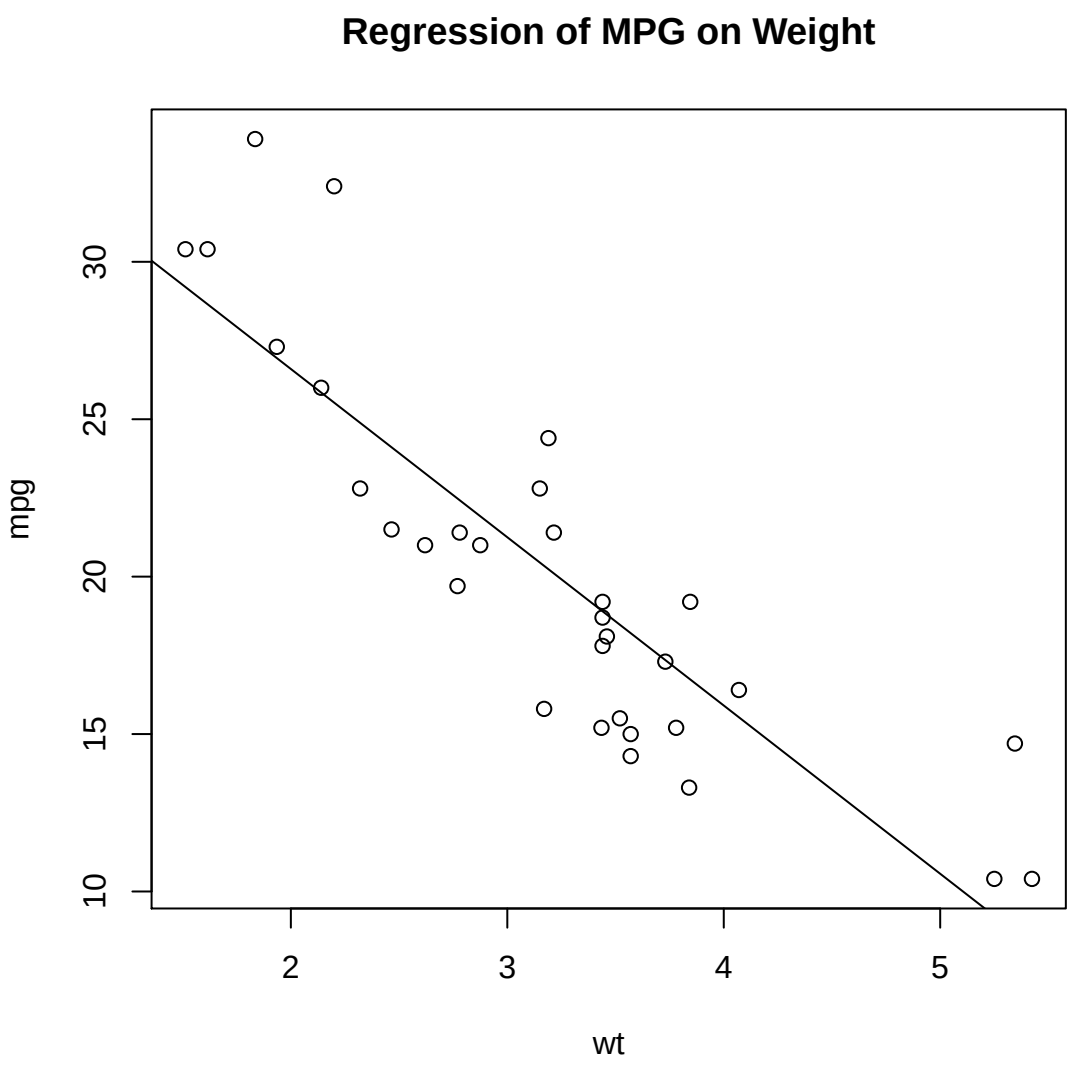
```
 [1] 2.620 2.875 2.320 3.215 3.440 3.460 3.570 3.190 3.150 3.440 3.440 4.070
[13] 3.730 3.780 5.250 5.424 5.345 2.200 1.615 1.835 2.465 3.520 3.435 3.840
[25] 3.845 1.935 2.140 1.513 3.170 2.770 3.570 2.780
```

```
R> mpg # miles per gallon

 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
[31] 15.0 21.4
```

Now we make a graph where the plot( ) function opens a graph widow and plots weight vs. miles per gallon. The next line of code adds a regression line to this graph. The final line adds a title.

```
R> # Creating a Graph
R> plot(wt, mpg)
R> abline(lm(mpg~wt))
R> title("Regression of MPG on Weight")
R> detach(mtcars) # remove the data from the search path
```



3 Defining a function in R

4 More information

For more information about R please refer to the following links http://www.statmethods.net/index.html. This is a great website for learning R function, graphs, and stats. Also visit http://www.r-project.org/ and check out the Manuals i.e an introductions to R. Welcome the Rrrrrr world! See also the book on Applied spatial Data analysis with R http://www.asdar-book.org/ (Bivand et al., 2013).

References

Bivand, R. S., Pebesma, E. J., & Rubio, V. G. (2013). Applied spatial data analysis with R, .