Nuevas Perspectivas – Master de Agrobiologia Ambiental

Introduction to R: Statistics basics and data visualization

Adrian Gracia Romero, PhD Student. ✉ a.graciaromero@ub.edu
Integrative Crop Ecophysiology Group, University of Barcelona

Content

1. Configuration of R and RStudio

To download R, go to CRAN, the comprehensive R archive network. CRAN is composed of a set of mirror servers distributed around the world and is used to distribute R and R packages.

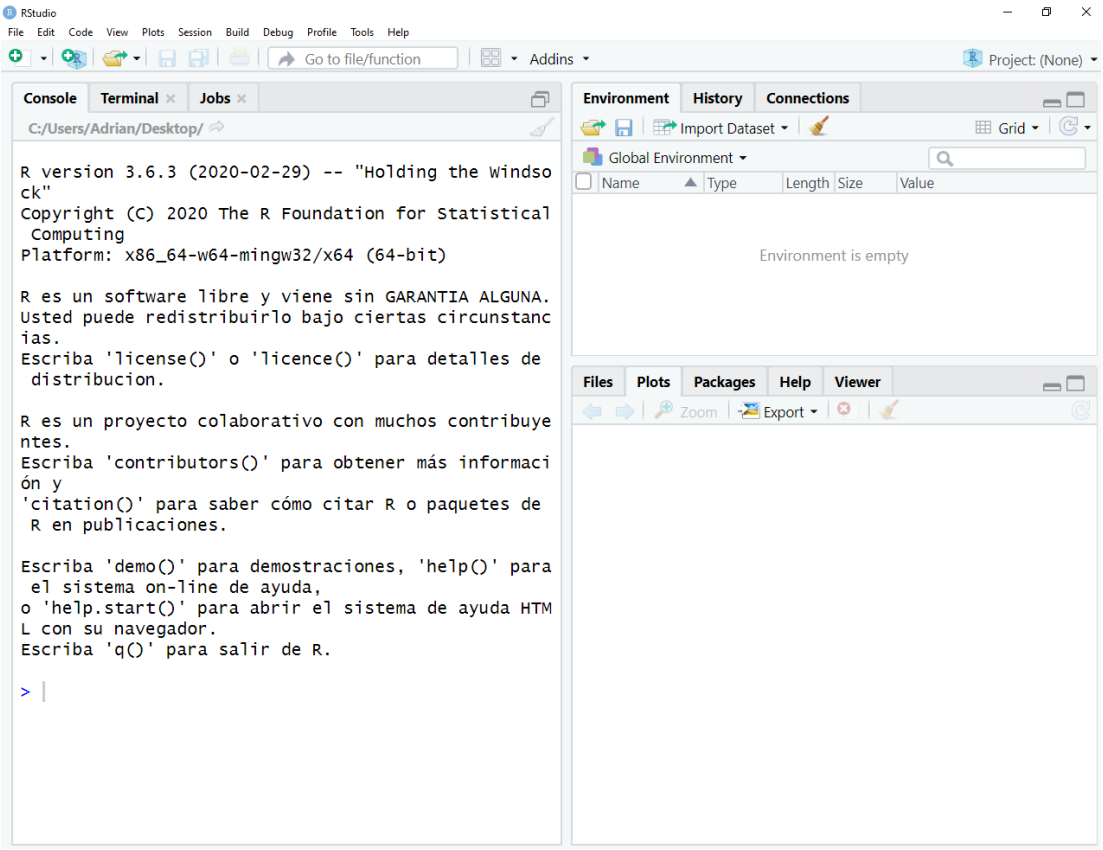
Windows: https://cran.r-project.org/bin/windows/base/
Mac OS: https://cran.r-project.org/bin/macosx/

RStudio is an integrated development environment, or IDE, for R programming.

Windows: https://www.rstudio.com/products/rstudio/download/#download
Mac OS : https://www.rstudio.com/products/rstudio/download/#download

2. Basic concepts

When you start RStudio, you'll see two key regions in the interface:



The RStudio interface is simple. You type R code into the bottom line of the RStudio console pane and then click Enter to run it. The code you type is called a command, because it will command your computer to do something for you. The line you type it into is called the command line.

When you type a command at the prompt and hit Enter, your computer executes the command and shows you the results. Then RStudio displays a fresh prompt for your next command. For example, if you type 1 + 1 and hit Enter, RStudio will display:

```
> 1 + 1
[1] 2
```

You'll notice that a [1] appears next to your result. R is just letting you know that this line begins with the first value in your result. Some commands return more than one value, and their results may fill up multiple lines. For example, the command 100:130 returns 31 values; it creates a sequence of integers from 100 to 130. Notice that new bracketed numbers appear at the start of the second and third lines of output. These numbers just mean that the second line begins with the 14th value in the result, and the third line begins with the 25th value. You can mostly ignore the numbers that appear in brackets:

```
> 100:130
[1] 100 101 102 103 104 105 106 107 108 109 110 111 112
[14] 113 114 115 116 117 118 119 120 121 122 123 124 125
[25] 126 127 128 129 130
```

If you type an incomplete command and press Enter, R will display a + prompt, which means it is waiting for you to type the rest of your command. Either finish the command or hit Escape to start over:

```
> 5 -
+
+ 1
[1] 4
```

R lets you save data by storing it inside an R object. What's an object? Just a name that you can use to call up stored data. For example, you can save data into an object like a or b. When you create an object, the object will appear in the environment pane of RStudio. Wherever R encounters the object, it will replace it with the data saved inside, like so:

```
a <- 1
a
[1] 1

a + 2
[1] 3
```

To create an R object, choose a name and then use the less-than symbol, <, followed by a minus sign, -, to save data into it. This combination looks like an arrow, <-. R will make an object, give it your name, and store in it whatever follows the arrow. When you ask R what's in a, it tells you on the next line. You can use your object in new R commands, too. Since a previously stored the value of 1, you're now adding 1 to 2. So, for another example, the following code would create an object named die that contains the numbers one through six. To see what is stored in an object, just type the object's name by itself:

```
a <- 1:6
a
[1] 1 2 3 4 5 6

a <- c(1,3,4,7)
a
[1] 1 3 4 7
```

3. Packages and functions

An R package is a collection of functions, data, and documentation that extends the capabilities of base R. You can install the complete package with a single line of code:

```
install.packages("NAME OF THE PACKAGE")
```

On your own computer, type that line of code in the console, and then press enter to run it. R will download the packages from CRAN and install them on to your computer. If you have problems installing, make sure that you are connected to the internet, and that https://cloud.r-project.org/ isn't blocked by your firewall or proxy. You will not be able to use the functions, objects, and help files in a package until you load it with library():su

```
library(NAME OF THE PACKAGE) #Without “”
```

Every function in R has three basic parts: a name, a body of code, and a set of arguments.

There are some functions already in R. One example is matrix(), that is a function to create matrix of data. The common sintaxis of this function is matrix(data, nrow, ncol, byrow=F). Data is the values that will integrate the matrix. Dimension of the matrix can be defined by passing appropriate value for arguments nrow and ncol. We can see that the matrix is filled column-wise. This can be reversed to row-wise filling by passing TRUE to the argument byrow. hese names can be accessed or changed with two helpful functions colnames() and rownames().

```
> matrix(1:6)
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
[5,]    5
[6,]    6
```

```
> matrix(1:6,nrow=2)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> matrix(1:6,nrow=2,byrow=T)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

4. Create a new script

A script is simply a text file containing a set of commands and comments. The script can be saved and used later to re-execute the saved commands. The script can also be edited so you can execute a modified version of the commands. It is easy to create a new script in RStudio. You can open a new empty script by clicking the New File icon in the upper left of the main RStudio toolbar. This icon looks like a white square with a white plus sign in a green circle. Clicking the icon opens the New File Menu. Click the R Script menu option and the script editor will open with an empty script.

5. Before starting: Working directory

The working directory is the folder where the data will be saved. We can choose our working directory using the function setwd(). For checking if the working directory set is the correct use the function getwd(). If you want to check what is inside the working directory, use the function list.files().

```
setwd("C:/Users/Adrian/Documents/Projectes")
getwd()
list.files()
```

6. Import the data from an excel document

Load excel file with the package readxl and the function read_excel().
https://cran.project.org/web/packages/readxl/readxl.pdf

read_excel() calls excel_format() to determine if path is xls or xlsx, based on the file extension and the file itself, in that order. Path to the xls/xlsx file. Sheet to read. Either a string (the name

of a sheet), or an integer (the position of the sheet). Ignored if the sheet is specified via range. If neither argument specifies the sheet, defaults to the first sheet. col_names TRUE to use the first row as column names, FALSE to get default names, or a character vector giving a name for each column. If user provides col_types as a vector, col_names can have one entry per column, i.e. have the same length as col_types, or one entry per unskipped column.

```
install.packages("readxl")
library(readxl)
mydata <- read_excel(path, sheet = NULL, col_names=T)
View(mydata)
```

Other useful functions

dim(): tells the dimensions of a matrix.
colnames(): names of columns of a matrix.
rownames(): names of the rows of a matrix.
str(): display the internal structure of an R object
length(): length of a matrix
apply(): Aplies a function obre the columns or rows of a matrix
cbind(): Add a new column
rbind(): Add a new row

The interpretation of a factor depends on both the codes and the "levels" attribute. Be careful only to compare factors with the same set of levels (in the same order). In particular, as.numeric() applied to a factor is meaningless, and may happen by implicit coercion. To transform a column to categorical values use as.factor().

7. Data manipulation

The packages dplyr and tidyr are useful to manipulate easily the data. Use filter() to filter the data into groups, select() to select columns, subset() to create a new data using only some columns…

```
## Data manipulation packages "dplyr" and "tidyr"
## https://cran.r-project.org/web/packages/dplyr/dplyr.pdf
install.packages("dplyr")
library(dplyr)
## https://cran.r-project.org/web/packages/tidyr/tidyr.pdf
install.packages("tidyr")
library(tidyr)
mydata.A <- filter(mydata, Locality == "Aranjuez")
    mydata.A.I <- filter(mydata.A, Treatment == "Irrigation")
    mydata.A.R <- filter(mydata.A, Treatment == "Rainfed")
mydata.V <- filter(mydata, Locality == "Valladolid")
    mydata.V.I <- filter(mydata.A, Treatment == "Irrigation")
    mydata.V.R <- filter(mydata.A, Treatment == "Rainfed")


mydata.factors <- subset(mydata, , -c(NDVI.avg_field:harvest.index))
```

8. Statistics basics

For the calculation of the main statistics basics we can use the function summarySE(), and we will need the packages lattice, plyr and Rmisc. Using the argument mesurevar we will indicate the parameter that we want to study and using groupvars we will indicate the grouping variable.

```
## Gives count, mean, standard deviation, standard error of the mean, and confidence interval
## https://cran.r-project.org/web/packages/Rmisc/Rmisc.pdf
install.packages("Rmisc")
install.packages("lattice")
## if it doesnt work, close and open the programe
library(lattice)
library(plyr)
library(Rmisc)

summarySE(mydata, measurevar="GY", groupvars=c("Locality", "Treatment"))
```

Another option is using the function describeBy() inside the package "psych".

```
## Report basic summary statistics by a grouping variable
## https://cran.r-project.org/web/packages/psych/psych.pdf
install.packages("psych")
library(psych)
describeBy(mydata.A, group = mydata.A$Treatment)
```

9. ANOVA

The ANOVA anlysis can be made using the function anova().

```
anova(lm(GY ~ Genotype, mydata.A.I))
```

10. Correlations

Para el cálculo de una correlación entre dos parámetros podemos utilizar la función cor.test().

```
cor.test(mydata.A.I$GA.g, mydata.A.I$GY)
```

11. Data visualization with "ggplot2"

One of the most useful packages for data visualization is ggplot2. All ggplot2 plots begin with a call to ggplot(), supplying default data and aesthethic mappings, specified by aes(). You then add layers, scales, coords and facets with +. To save a plot to disk, use ggsave().

Following the webs below, you will succed doing almost any graphic.

https://ggplot2.tidyverse.org/reference/
https://www.r-graph-gallery.com/index.html