R for life sciences. Chapter 1: Easy R start

2020-10-13

# FIRST STEPS WITH R

## Installing R and RStudio

In other tutorials you have a lot of information to install R and RStudio. I am not going to extend with this.

To install R and RStudio, the best way is to follow the instructions from the official sites:

* <https://cran.r-project.org/>
* <https://www.rstudio.com/products/RStudio/#Desktop> Just make sure you download and install the free RStudio desktop, not a pay version.

If you need more help, a well detailed tutorial on how to install and start with R is <http://a-little-book-of-r-for-bioinformatics.readthedocs.io/en/latest/src/installr.html>.

There is a lot of information about R. Some good and free books are in the official page <https://cran.r-project.org/other-docs.html> Look a bit through them and may be you find one you would like to read more thoroughly.

The official R documentation is in <https://cran.r-project.org/manuals.html> but it feels a bit difficult to read, at least for a beginner, so just take a quick look.

Before start is a good idea to have a ref card near:

* <https://cran.r-project.org/doc/contrib/Short-refcard.pdf>

## Important facts before starting with R

* R has core packages and other packages. Core packages are always installed, but they may have to be activated using library(). If the package is not installed but it is in the repositories, it can be installed using install.packages().
* R is case sensitive. LM() is not the same that lm().
* Use " # " to comment text into your R script. Anything after " # " will not be run. It is important to explain all you do, commented, so any person (or you a month latter) can understand what is in your code.
* R works with commands and objects. The commands run always with () and may have different parameters inside. Objects can be functions, data objects, and results objects (commands are really function objects).
* To assign or create an object is possible to use either " = " or " <- “. Most people use” <- “, and save” = " only for equations and equalities. **To write " <- " easily use ALT + " - "**.
* It is important that objects are not named as commands or other previously existing objects unless you want to overwrite them. This is one of the main sources of trouble at the beginning.

## R objects

There are many possible types of data (and results) objects, but the most important ones are:

|  |  |
| --- | --- |
| Object | Description |
| Simple objects | a number, a string of text, etc |
| Vectors | a list of simple objects. Can be numbers, text, etc |
| Factors | Categorical data. includes a vector with numbers for each category and another of levels |
| Matrices | Can only contain numbers |
| Data frames | Columns are vectors (or factors) and may have a variable name |
| Lists | Lists of any other objects |
|  |  |

Vectors may be *numeric* (numbers), *character* (text) or *logical* (TRUE, FALSE). A vector with mixed data will be treated as *character*.

Data frames and lists may have different types of data into the same object.

### Creating simple objects and vectors

[***c()****, print(), mode(),* ***str()****,* ***length()****,* ***as.character()****,* ***as.numeric()****,* ***[ ]****, ls(), rm()*](#26in1rg)

To create an object you only need to assign it as in the next examples. For vectors the command **c()** is used. To see what is inside the objects we can run the object name, or run print(object) if you are inside a loop or a function.

To see which type of object you can use mode() or better **str()**. The use of str() must be very frequent to become familiar to the structure of the different objects. The main characteristics of a vector are its mode() and its length().

We can convert a vector from numeric to character and vice versa, using as.character() and as.numeric().

With [] we can point to parts of a vector to extract or replace them.

You may use ls() to see a list of all your objects, as in the “Environment” tab in RStudio. Also it is possible to remove an object with rm().

### Factors

[***as.vector(), as.factor(), levels(), factor()***](#26in1rg)

Categorical data can be in two different formats: *character vectors* or factors. To change between them use as.vector() or as.factor(). Factors structure is a numeric vector with the order and a character vector called *names* with the list of categories.

### Matrices

[***matrix()****, dim()*](#26in1rg)

Matrices are numeric data organized by rows and columns. In R, a matrix is actually a vector with an additional attribute (dim), which is itself a numeric vector with length 2, and defines the numbers of rows and columns of the matrix. A matrix can be created in different ways. The most used is with the function matrix().

The option byrow indicates whether the values given by data must fill successively the columns (the default) or the rows (if TRUE). The option dimnames allows to give names to the rows and columns.

Another way to create a matrix is with the vector of values and giving values to the dim attribute:

### Data frames

[*seq(), rep(),* ***data.frame()****,* ***names()****,* ***paste()****, row.names()*](#26in1rg)

Data frames are the way we will usually have the data.

A data frame is a rectangular set of data that, as matrices, has columns (variables) and rows (observations), but in this case, columns (variables) are vectors of the same length. Each column (variable) is a vectorthat can only have one type of data (mode).

### Lists

[***list()***](#26in1rg)

Into a data frame it is possible to put vectors and factors of the same length. Into a list it is possible to put almost anything, even other lists. There is no constraint on the objects that can be included. Several results objects exited from analyses will be lists.

Into the structure of L1 there is a data frame, a matrix, a factor, a numeric vector and a character vector. Names of the original variables are not kept, and usually are not needed.

To put names into a list, if needed, use names() same way that for data frames

To point to data or objects into a list use [[ ]] to get each object into the list and then it is the same than for each object type.

## Getting help

[*help(), ?, help.start()*](#26in1rg)

To see the help page of a command the easiest way is using the Help tab in RStudio. If not using it, you can do the same with **help()** or **?**.

Using help.start() opens the R internal HTML complete help.

## Before the exercises

Before start the exercises, lets do some steps to start with the right workflow.

* Create a new RStudio project. File > new project > choose folder (a git repo will not harm). Now everything from this project will be into the new folder.
  + Better if the folder is in a cloud repository. Use the Intranet disk, for example, so you can access the folder from any computer.
* Create a R script. File > new file > R script. Save with informative name.
* All the code to solve the exercises should be into this script.
* Never save RStudio workspace, nor history. You can change default options in Tools > global options > General.
* When working in a R script, be sure to save it regularly and at the end.

Doing allways these steps at the begining of a project, you will have a project folder with everything inside: data, scripts and results. It makes easy to share your work and to keep it in order.

When writting your first script, be sure you copy-paste the statements, always commented, and comment everything you do, so anybody that sees your script knows what are you doing (e.g., you next week or next year).

# Exercises

1. Create a vector with numbers from 1 to 31. With this vector and the command *paste()* create a vector named “tree.name” with 31 tree names, from “Tree\_1” to “Tree\_31”.
2. Make and object called “d0” with the data frame “trees” in R datasets. Look into the help to see what is into this data frame. ¿How many variables and observations there are in d0?
3. Add your variable tree.name to d0
4. Make the code to extract the name in tree.name of the larger (volume), the highest and the widest tree.
5. Using the function *mean()* calculate the mean Diameter, Height and Volume.
6. Make a new factor variable into d0 with two levels: “Large” for trees with volume larger or equal the mean and “Small” with trees with volume smaller than the mean. ¿How many “Large” trees are there?
7. Make a new factor variable into d0 with “Tall” for trees taller or equal the mean and “Short” with trees with Height shorter than the mean.
8. Make a subset with the trees that are both short and large and calculate the mean diameter of these Short-Large trees in meters. How many “Short\_Large” trees are there?

# Important commands

[**library()**](#1fob9te)

Load a package before using it.

[**install.packages()**](#1fob9te)

Download and install packages from CRAN repositories or local files.

[**c()**](#2et92p0)

Combine values into a Vector (default) or list.

[**str()**](#2et92p0)

Display the internal nested structure of an R object.

[**length()**](#2et92p0)

Get or set the length of vectors, lists, factors and of any other R object.

[**as.character()**](#2et92p0)

convert vectors to character.

[**as.numeric()**](#2et92p0)

convert vectors to numeric.

[**[]**](#2et92p0)

Extract or replace parts of vectors, matrices, arrays, data frames and lists.

[**as.factor() and factor()**](#tyjcwt)

Encodes a vector as a factor.

[**as.vector()**](#tyjcwt) **and vector()**

Attempts to coerce its argument into a vector of most convenient mode if it is not specified.

[**levels()**](#tyjcwt)

Provides access to the levels of a factor. Also to replace them.

[**matrix()**](#3dy6vkm)

Creates a matrix from the given set of values.

[**data.frame()**](#1t3h5sf)

Creates data frames.

[**names()**](#1t3h5sf)

Get or set the names of an object.

[**paste()**](#1t3h5sf)

Concatenate vectors after converting to character.

[**list()**](#4d34og8)

Get or set the names of an object.

## Other commands

[**print()**](#2et92p0)

Prints its argument at console with some options.

[**mode()**](#2et92p0)

Get or set the type or storage mode of an object.

[**ls() and objects()**](#2et92p0)

return a vector of character strings giving the names of the objects in the specified environment.

[**rm() and remove()**](#2et92p0)

Remove objects specified successively as character strings, or in the character vector list.

[**dim()**](#3dy6vkm)

Retrieve or set the dimension of an object.

[**seq()**](#1t3h5sf)

Generate regular sequences.

[**rep()**](#1t3h5sf)

Replicates the values in x.

[**row.names()**](#1t3h5sf)

All data frames have a row names attribute, a character vector of length the number of rows with no duplicates nor missing values.

[**help()**](#2s8eyo1)

help is the primary interface to the help systems. Provide access to documentation on a topic.

[**?**](#2s8eyo1)

same than help()

[**help.start()**](#2s8eyo1)

Start the hypertext version of R’s online documentation.

# About this tutorial

Cite as: Alfonso Garmendia (2020) R for life sciences. Chapter 1: Easy R start. <http://personales.upv.es/algarsal/R-tutorials/01_Tutorial-1_R-easy-start.html>.

Available also in other formats (pdf, docx, …): <https://drive.google.com/drive/folders/19w914WCg8BVTVBE_zpgShmg2vpjguV1e?usp=sharing>.

Other simmilar tutorials: <https://garmendia.blogs.upv.es/r-lecture-notes/>

Originals are in bitbucket repository: <https://bitbucket.org/alfonsogar/tea_daa_tutorials>.

Document written in [Rmarkdown](http://rmarkdown.rstudio.com/), using [Rstudio](https://www.rstudio.com/).

* System: R version 3.6.3 (2020-02-29) x86\_64-pc-linux-gnu (64-bit) Ubuntu 18.04.5 LTS.
* Base packages: stats 3.6.3, graphics 3.6.3, grDevices 3.6.3, utils 3.6.3, datasets 3.6.3, methods 3.6.3, base 3.6.3.
* Other loaded packages: writexl 1.2, rmarkdown 1.17, googledrive 0.1.3.



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](http://creativecommons.org/licenses/by-nc-sa/4.0/).