R for life sciences. Chapter 1: Easy R start

By Alfonso Garmendia (Universitat Politècnica de València)

Contents

Cite as: Alfonso Garmendia (2016) R for life sciences. Chapter 1: Easy R start. http://personales.upv.es/algarsal/Documentation/Garmendia-R-Tutorial-01_Easy_R_Start.html

available in PDF and EPUB

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Written in Rmarkdown, using Rstudio and pandoc.

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 follow instructions from the official sites:

- https://cran.r-project.org/
- https://www.rstudio.com/products/RStudio/#Desktop

If you need more help, you can look here:

- http://a-little-book-of-r-for-bioinformatics.readthedocs.io/en/latest/src/installr.html
- https://cran.r-project.org/manuals.html
- https://cran.r-project.org/other-docs.html
- http://rseek.org/

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

library(), install.packages()

- 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 comment all you do, so you (or other person) can understand what are you doing, next time you see your script.
- 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.
- 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 substitute 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()*

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.

```
#·····························································
#       Creating simple objects and vectors
#·····························································
#       Simple Objects
N5 <- 5                             # Numeric
Te <- "Five"                        # Text
#
#       Vectors
VN <- c(1,3,4,N5)                   # Numeric vector
VT <- c("One", "Three", "Four", Te) # Text vector
#
####   We created four objects: N5, Te, VN and VT.
####   We used the first two objects as part of the vectors.
#
####   To see the content of the objects
N5   # or print(N5)
Te
VN
VT
```

```
#
####   To see the  mode of the objects
mode(N5)
mode(Te)
mode(VN)
mode(VT)
#
####   To see the  structure of the objects
str(N5)
str(Te)
str(VN)
str(VT)
####   To see the  length of the objects
length(VT)
#### If we mix character and numeric objects into a vector,
#### it will became a character vector:
UN <- c(VN,VT)
UN
str(UN)
length(UN)
#
#### Convert numeric vector to character and vice versa
VNC <- as.character(VN)
str(VN)
str(VNC)                # Notice that character data are in quotations
as.numeric(VNC)         # Back to previous numeric vector
#
#### To extract part of a vector
VT [1:2]                # Extracts fist two positions of VN
VT [c(1, 3)]            # Extracts positions 1 and 3 of VN
#
#### To replace parts of a vector
VT                                    # See initial VT
VT [2:4] <- c("two", "three", "four") # Replace
VT                                    # See the result
#
#### Logical Vectors
VN                                    # This is a numeric vector
VN <= 3                               # This is a logical vector
VN [VN <= 3]                          # Crop the values <= 3
#
```

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().

```
ls()                                  # to see the actual objects
rm (Te)                               # Remove Te
ls()                                  # Check Te disappeared
```

Factors

as.vector(), as.factor(), levels(), factor()

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.

```
#.........................
#   Factors
#.........................
Vec <- c("Red","Blue","Red","Blue","Red") # This is a character vector
str(Vec)                                  # Notice its structure
Fac <- as.factor(Vec)                     # Create factor
as.vector(Fac)                            # Vector
str(Fac)        # Notice structure of a factor with levels and numbers
Fac[3]          # Position 3
levels(Fac)     # To see the levels.
                # Notice that levels are in alphabetical order
levels(Fac)[2]  # The second level of the factor
#
######## Reorder Factors #####
levels(Fac)[c(2, 1)]                  # The levels ordered as wanted
FacR <- factor(Fac, levels = levels(Fac)[c(2, 1)]) # Redo the factor
str(FacR)                             # Factor ordered
#
######## Other way ############
FacR2 <- factor(Vec, levels = c("Red", "Blue"))
str(FacR2)                            # Same result
#
```

Matrices

matrix(), *dim()*

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()

matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)

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.

```
######## Create a matrix of 3 rows and 4 columns filled with 0s
matrix(data=0, nrow = 3, ncol = 4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
## [3,]    0    0    0    0
```

```
######## Matrix of 3 rows and 4 columns filled with 1:12 by columns
matrix(1:12, 3, 4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
matrix(1:12, 3, 4, byrow = TRUE) # Same, but filled by rows
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

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

```
######## Create a matrix from a vector
Mdata <- 1:12              # Numeric vector
Mdata
dim(Mdata)          # Null. Vectors do not have dimensions, only length
dim(Mdata) <- c(3,4)       # assign dimensions
Mdata                      # The matrix
#
######## To convert the matrix into the original vector
as.vector(Mdata)
#
######## Characteristics of a matrix
length(Mdata)
dim(Mdata)
str(Mdata)             # int means integers (numeric without decimals)
######## To point into a place in the matrix
Mdata[3,4]             # 12
Mdata[2,2]             #  5
```

Data frames

seq(), rep(), ***data.frame()****,* ***names()****,* ***paste()****, row.names()*

Data frames are the way we usually will have the original data. A data frame is a rectangular set of data that, as matrices, has columns (variables) and rows (observations). Also it could be considered as a set of vectors of the same length (variables) and mode. Each column (variable) is a vector (or factor for categorical data) that can only have one type of data (mode).

```
######## Creating different vectors or variables
V1 <- c(1:5)
V2 <- seq(1,10,2)                  # Sequence from 1 to 10 by 2
V3 <- c("one","two","three","four","five")
                                   # Replicate the vector to length 5
V4 <- rep(c("odd", "even"),length=5)
#
######## Creating the data frame with four variables
D15 <- data.frame(V1,V2,V3,V4)
D15
```

```
##   V1 V2    V3   V4
## 1  1  1   one  odd
## 2  2  3   two even
## 3  3  5 three  odd
## 4  4  7  four even
## 5  5  9  five  odd
```

```
str(D15)                         # two numeric vectors and two factors
```

```
## 'data.frame':    5 obs. of  4 variables:
##  $ V1: int  1 2 3 4 5
##  $ V2: num  1 3 5 7 9
##  $ V3: Factor w/ 5 levels "five","four",..: 3 5 4 2 1
##  $ V4: Factor w/ 2 levels "even","odd": 2 1 2 1 2
```

```
dim(D15)                          # same dimensions than matrices
```

```
## [1] 5 4
```

```
######## Different ways of accessing one variable from the data frame
D15$V1              # Fist variable
str(D15$V1)         # Vector
str(D15$V3)         # Factor
D15[3]              # Still a data frame but with only one variable
D15[[3]]            # Same factor than D15$V3
#
####### Accessing data into a data frame (several ways to do the same)
D15$V3[4]                # Rows 4 of third variable (Still factor)
D15[[3]][4]              # Same than before
D15[4,3]                 # Same than matrices
as.character(D15$V3[4])  # Same, but as character
#
######## Variable names
names(D15)                   # Show variable names
PrevNames <- names(D15)      # Save variable names to use them latter
NewNames  <- c("Num","OddNum","Text","OddEven")
    # Change variable names using paste
names(D15) <- paste(PrevNames,NewNames,sep="_")
names(D15)[1] <- "V1_Num15"          # Change only one variable name
names(D15)
#
######### Row names
row.names(D15) <- D15$Var3
D15
#
######### Subset a data frame
D15[D15$Var4=="odd",]       # Subset of  all rows with Var4="odd"
    # Subset of  all rows with Var4="odd" in variables 1 to 2.
D15[D15$Var4=="odd",1:2]
#
```

Lists

list()

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.

```
# ls()                            # All objects already active in R
        # List of some of the previous objects of this chapter
L1 <- list(D15,Mdata,Fac,VN,UN)
L1                                # A list with different objects
```

```
## [[1]]
```

```
##   V1_Num15 V2_OddNum V3_Text V4_OddEven
## 1        1         1     one        odd
## 2        2         3     two       even
## 3        3         5   three        odd
## 4        4         7    four       even
## 5        5         9    five        odd
##
## [[2]]
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## [[3]]
## [1] Red  Blue Red  Blue Red
## Levels: Blue Red
##
## [[4]]
## [1] 1 3 4 5
##
## [[5]]
## [1] "1"     "3"     "4"     "5"     "One"   "Three" "Four"  "Five"
```

```
str(L1)
```

```
## List of 5
##  $ :'data.frame':    5 obs. of  4 variables:
##   ..$ V1_Num15  : int [1:5] 1 2 3 4 5
##   ..$ V2_OddNum : num [1:5] 1 3 5 7 9
##   ..$ V3_Text   : Factor w/ 5 levels "five","four",..: 3 5 4 2 1
##   ..$ V4_OddEven: Factor w/ 2 levels "even","odd": 2 1 2 1 2
##  $ : int [1:3, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
##  $ : Factor w/ 2 levels "Blue","Red": 2 1 2 1 2
##  $ : num [1:4] 1 3 4 5
##  $ : chr [1:8] "1" "3" "4" "5" ...
```

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.

```
####     Name objects into a list
names(L1) <- c("D15","Mdata","Fac","VN","UN")
L1
str(L1)
#
####    Point to objects into a list.
L1[[1]]          # The first object
L1$D15           # Same than previous, using the name
#
####    Point to data into objects into a list
L1[[1]]$Var1     # First variable of the data frame
L1$D15$Var1      # Same than previous using the name
L1[[1]][[1]]     # Same than previous using only numbers.
```

```
# This notation is very useful when using iterations.
```

Getting help

help(), ?, help.start()

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.

```
help.start()      # General R help
help(list)        # Help of command list()
?list             # Same
```

Exercises

1. Create a vector with numbers from 1 to 31. With this vector and the command **paste()** create a vector named "TreeName" with 31 tree names, from "Tree_1" to "Tree_31".
2. Make and object called "Tre" 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 Tre?
3. Add your variable TreeName to your data frame Tre
4. Make the code to extract the name in TreeName of the larger (volume), the highest and the widest tree.
5. Convert all the three variables to meters or cubic meters adding new variables "Diamet_m" "Height_m" and "Volume_m"
6. Calculate the volume using Diamet_m and Height_m into a variable "Volum2_m". ¿Is it the same value that the one into Volume_m? ¿is it higher or lower?
7. Using the function **mean()** calculate the mean Diameter, Height and Volume in m.
8. Make a new factor variable into Tre with "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?
9. Make a new factor variable into Tre with "Tall" for trees taller or equal the mean and "Short" with trees with Height shorter than the mean.
10. Make a subset with the trees that are both short and large and calculate the mean diameter of these Short-Large trees in m. ¿How many are there?

Important commands

library() Load a package before using it.
install.packages() Download and install packages from CRAN repositories or local files.
c() Combine values into a Vector (default) or list.
str() Display the internal nested structure of an R object.
length() Get or set the length of vectors, lists, factors and of any other R object.
as.character() convert vectors to character.
as.numeric() convert vectors to numeric.
[] Extract or replace parts of vectors, matrices, arrays, data frames and lists.
as.factor() and factor() Encodes a vector as a factor.
as.vector() and vector() Attempts to coerce its argument into a vector of most convenient mode if it is not specified.
levels() Provides access to the levels of a factor. Also to replace them.
matrix() Creates a matrix from the given set of values.
data.frame() Creates data frames.
names() Get or set the names of an object.
paste() Concatenate vectors after converting to character.
list() Get or set the names of an object.

Other commands

print() Prints its argument at console with some options.
mode() Get or set the type or storage mode of an object.
ls() and objects() return a vector of character strings giving the names of the objects in the specified environment.
rm() and remove() Remove objects specified successively as character strings, or in the character vector list.
dim() Retrieve or set the dimension of an object.
seq() Generate regular sequences.
rep() Replicates the values in x.
row.names() All data frames have a row names attribute, a character vector of length the number of rows with no duplicates nor missing values.
help() help is the primary interface to the help systems. Provide access to documentation on a topic.
? same than help()
help.start() Start the hypertext version of R's online documentation.