

# Introduction

Marcos Lopez-Sanchez

ISGlobal Barcelona Institute for Global Health  
<http://www.isglobal.org>

Introduction to R  
2nd Edition  
May 29th 2017

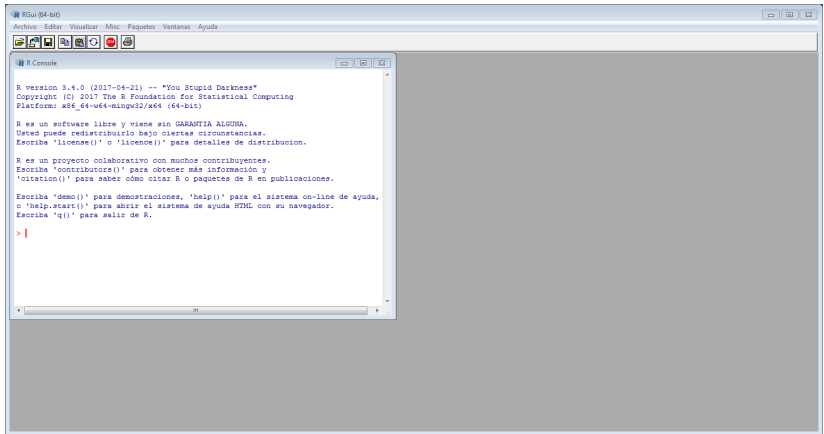
**ISGlobal** **Barcelona**  
**Institute for**  
**Global Health**

- 1 Introduction
- 2 First steps
- 3 Survival guide
- 4 Saving and loading objects and Sessions
- 5 Packages and Repositories
- 6 Session Info

# Outline

- 1 Introduction
- 2 First steps
- 3 Survival guide
- 4 Saving and loading objects and Sessions
- 5 Packages and Repositories
- 6 Session Info

# What is “R” ?



The screenshot shows the RGui (64-bit) window. The menu bar includes Archivo, Editar, Visualizar, Misc., Paquetes, Ventanas, and Ayuda. The toolbar contains icons for file operations and running code. The R Console window is open, displaying the following text:

```
R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

> |
```

# What is “R” ?

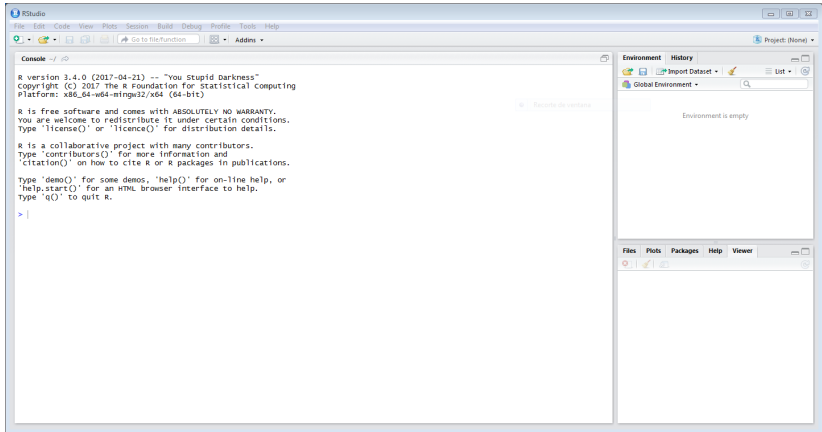
“R”: integrated suite of software for data manipulation, calculation and graphical display. Characteristics:

- Effective data handling and storage facility.
- Suite of operators for calculations on arrays.
- Large, coherent, integrated collection of intermediate tools for data analysis.
- Graphical facilities for data analysis and display.
- Simple and effective programming language (called “S”).

# Installing R

- Distributed as an installer with the name R-X.Y.Z-win.exe (Most updated 3.4.0)
- Maintained by the Comprehensive R Archive Network (CRAN)
- CRAN page (Spanish mirror) –  
`http://cran.es.r-project.org/`
- **Installer** – `https://cran.r-project.org/bin/windows/base/R-3.4.0-win.exe`

# What is “RStudio” ?



# What is “RStudio” ?

“RStudio”: Integrated development environment (IDE) for R. With several features:

- Code editor.
- Syntax highlighting, code completion and smart indentation.
- Visualization tools.
- Integrated help and documentation.



# Installing RStudio

- The RStudio software under Windows OS is distributed as an installer with the name RStudio-X.Y.Z.exe (Most updated 1.0.143)
- RStudio page – <https://www.rstudio.com>
- Installer – <https://download1.rstudio.org/RStudio-1.0.143.exe>

# Outline

- 1 Introduction
- 2 First steps**
- 3 Survival guide
- 4 Saving and loading objects and Sessions
- 5 Packages and Repositories
- 6 Session Info

## Working directory

Folder where R will search for files by default.

It's important to know where are we working. There's a default path when starting R that can be shown with the function:

```
> getwd()
```

```
[1] "/home/marcos/Documentos/Shared/Introduction_R/Second_Edition_2017"
```

It can be changed by:

- Changing the “Start in” field in the R shortcut in windows before opening it.
- Going to “File”→“Change dir” inside R.
- Use `setwd()` function:

```
> setwd("/home/mlopez3/Documentos")
```

# Basic usage

The first things to do once R is started is use it as a basic math solver

```
> 2+2
```

```
[1] 4
```

```
> 1+2*3-4/5
```

```
[1] 6.2
```

```
> (19465*0.25)^23
```

```
[1] 6.389569e+84
```

It follows the usual precedence for operators, but can be forced with the use of parentheses ().

So the first usual thing in programming is to print the “hello world”

```
> print("hello world")
```

```
[1] "hello world"
```

Hello world is a string, and because that we must pass it between single or double quotes.

## Closing R and saving Workspace

When we have to close the session, we can click the close button of the window, go to the “File” menu and select quit, or use the `q()` function.

```
> q()
```

R *always* will ask us if we want to save the workspace. Saving the workspace we save all the variables and the commands used in the session, but if we are working with large datasets, it's recommended instead to save only the objects we are interested in or the results, as we will see later. I recommend choosing “No” and only save the image if it's really necessary.

# Outline

- 1 Introduction
- 2 First steps
- 3 Survival guide**
- 4 Saving and loading objects and Sessions
- 5 Packages and Repositories
- 6 Session Info

# Assignment and Vectors

R operates on data structures. The simplest structure is the numeric vector.

In order to assign values to a variable, we use an arrow “<-”.

```
> a <- 1  
> b <- 10
```

If we want to assign more than one element, we use the concatenate function “c”.

If there are more than one element assigned to the variable, then it's a vector of elements, in this case, numeric values.

```
> c <- c(a, 5)  
> d <- c(c, b, 50, 100)  
> d  
[1] 1 5 10 50 100
```

## Assignment and Vectors II

Numbers can also be generated easily within R with several functions:

```
> 1:10 ## With Colons forward and backward
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
> seq(10,1) ## With seq function forward and backward too
```

```
[1] 10 9 8 7 6 5 4 3 2 1
```

```
> seq(from=1,to=10,by=2) ## With seq can be also done 2 by 2
```

```
[1] 1 3 5 7 9
```

```
> rep(c(1,2),times=5) ## repeat a element 5 times
```

```
[1] 1 2 1 2 1 2 1 2 1 2
```



# Operators and Vectors

In addition to the elementary arithmetic operators (+, -, \*, / and ^), R have available common arithmetic functions and useful function for vectors. Except some, those operators are applied to all elements of the vector, and are:

```
> log(d) ## Logarithm
> exp(d) ## Exponential
> sin(d); cos(d); tan(d) ## Trigonometric functions
> sqrt(d) ## Squared Root
> max(d) ## Returns the maximum
> min(d) ## Returns the minimum
> length(d) ## Returns the number of elements in the vector
> sum(d) ## Returns the sum
> prod(d) ## Returns the product
> mean(d) ## Returns the mean
> var(d) ## Returns the variance
```

## Logical vectors

R also allows manipulation of logical quantities. The elements of a logical vector can be `TRUE`, `FALSE` and `NA`. These are generated by conditions using logical operators. The logicals operators are `<`, `<=`, `>`, `>=`, `==` and `!=`.

Different conditions can be evaluated together with the AND (`&`) and OR (`|`) operators

```
> d1 <- d < 50
> d1
[1] TRUE TRUE TRUE FALSE FALSE
> d2 <- d < 75 & d > 7
> d2
[1] FALSE FALSE TRUE TRUE FALSE
> !d2
[1] TRUE TRUE FALSE FALSE TRUE
```

Take care with the missing values `NA` or other types of “missing” values like `NaN` values (Not a Number) and `Inf` and `-Inf`

# Character vectors

There are also vectors of character or “words” and are mainly used as variable labels. These character elements or words are delimited by the double quote character, and can be concatenated with the `c()` function as can be seen:

```
> e <- "PFC"  
> f <- "PM"  
> c(e, f)  
[1] "PFC" "PM"
```

Also there are functions that allows us to manipulate these character vectors, and with a bit of skill name variables easily:

```
> paste(e, f) # allows to concatenate strings  
[1] "PFC PM"  
> ef <- c(paste(e, 1:3, sep=""), paste(f, c(2.5, 10), sep=""))  
> ef  
[1] "PFC1" "PFC2" "PFC3" "PM2.5" "PM10"
```

# Subsetting vectors I

Vectors can be subsetting by using the name of the vector and a index vector in square brackets. There are 4 different types depending on the index vector:

- A Logical vector: If we have a index vector of the same length as the vector from which the elements are to be selected.

```
> d2 <- d < 75 & d > 7  
> d3 <- d[d2]
```

- A vector of positive integral quantities: The values in the index vector must lie in the set. The corresponding elements of the vector are selected and concatenated in that order.

```
> d[3]  
[1] 10  
> d[c(1,4,3,2,5)]  
[1] 1 50 10 5 100  
> d[length(d):1]  
[1] 100 50 10 5 1
```

## Subsetting vectors II

- A vector of negative integral quantities: The index vector specifies the values to be excluded.

```
> d[-3]
```

```
[1] 1 5 50 100
```

```
> d[-(2:4)]
```

```
[1] 1 100
```

- A vector of character strings: This requires that the object has a names attribute to identify its components.

```
> exposures <- d
```

```
> names(exposures) <- ef
```

```
> exposures[c("PFC3", "PM2.5")]
```

```
PFC3 PM2.5
```

```
10 50
```

# Classes I

So, normally we can have continual data (numerical) or categorical (character). If we have a numerical vector, the function `summary` returns some of the statistics of the variable passed. If the variable is character, we can see that the summary function dreturns the length and the type of the data. So, in order to process categorical data, we must change the “character” type to “factor” in order to process it with the function `as.factor()`. We can check the class of the object using the function `class()`.

NOTE: To change a factor object to a numeric object we must first change it to character using the functions `as.character()` and `as.numeric()`

# Classes II

```
> summary(d)
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1.0     5.0    10.0    33.2   50.0   100.0

> class(d)
[1] "numeric"

> summary(ef)
      Length      Class      Mode
      5 character character

> class(ef)
[1] "character"

> casecontrol <- c(rep("case", 4), "control")
> casecontrol <- as.factor(casecontrol)
> class(casecontrol)
[1] "factor"

> summary(casecontrol)
  case control
    4        1
```

# Getting help

R has a help function to get more information on any specific named function. It's useful to know what the function queried does and the input parameters.

```
> help(solve)
> ?solve
```

For words with syntactic meaning like “if”, “for” and “function” or special characters, the argument must be enclosed in double or single quotes

If we don't know what's the name of the function but we know the method to apply or the action we want to do we can search the function.

```
> help.search("random")
> ??random
```



# Outline

- 1 Introduction
- 2 First steps
- 3 Survival guide
- 4 Saving and loading objects and Sessions
- 5 Packages and Repositories
- 6 Session Info

# Saving and Loading I

As said at the beginning, when we quit R we are asked to save the workspace. Furthermore, we can save the workspace where we want with the function `save.image()`. Moreover, if we want to save only an object from the workspace, we can save it with the function `save()` by stating the name of the variable and the destination file, ended with the `.RData` extension. The files will be saved in the working directory if no path is specified.

```
> save.image("Rcourse.RData")  
> save(d, file="d.RData")
```

## Saving and Loading II

Remember that THE NAME OF THE FILE IS NOT THE NAME OF THE VARIABLE INSIDE THE FILE.

To load the objects or sessions, we use the function `load()` specifying the file on the working directory or the path to the file otherwise. In order to check which objects do we have in the workspace, we can use the function `ls()`

```
> load("d.RData")  
> ls()  
> load("Rcourse.RData")  
> ls()
```

# Outline

- 1 Introduction
- 2 First steps
- 3 Survival guide
- 4 Saving and loading objects and Sessions
- 5 Packages and Repositories**
- 6 Session Info

# Packages and Repositories

Sometimes will be necessary to use methods not implemented in these basic operations. All R functions and datasets are stored in packages, in order to be efficient in terms of memory usage. To see which packages are installed, we can see it with the function `library()`.

```
> library() ## Show all packages installed  
> library(boot) ## Load a package
```

In order to install packages, the function `install.packages()` and `update.packages()` are used, the first for installing from scratch and the second for updating packages.

```
> install.packages("zoo")  
> update.packages("base")
```

Moreover, there are third-party repositories with packages like Bioconductor, specialized in biological and bioinformatics data analysis. (<http://www.bioconductor.org/>)

# Outline

- 1 Introduction
- 2 First steps
- 3 Survival guide
- 4 Saving and loading objects and Sessions
- 5 Packages and Repositories
- 6 Session Info**

# Session Info

Finally, to see what R version was used to make these slides, the function `SessionInfo` shows the R version and the packages loaded.

```
> sessionInfo()
```

```
R version 3.3.3 (2017-03-06)
```

```
Platform: x86_64-redhat-linux-gnu (64-bit)
```

```
Running under: Red Hat Enterprise Linux
```

```
locale:
```

```
[1] LC_CTYPE=es_ES.UTF-8      LC_NUMERIC=C
[3] LC_TIME=es_ES.UTF-8      LC_COLLATE=es_ES.UTF-8
[5] LC_MONETARY=es_ES.UTF-8  LC_MESSAGES=es_ES.UTF-8
[7] LC_PAPER=es_ES.UTF-8     LC_NAME=C
[9] LC_ADDRESS=C             LC_TELEPHONE=C
[11] LC_MEASUREMENT=es_ES.UTF-8 LC_IDENTIFICATION=C
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets
[6] methods    base
```

```
loaded via a namespace (and not attached):
```