INTRODUCTION TO R

Introducción a la Ciencia de Datos

Coral del Val Muñoz

Dept. Ciencias de la Computación e Inteligencia Artificial,
Universidad de Granada
Dept. Molecular Biophysics, German Cancer Research Center Heidelberg, Alemania

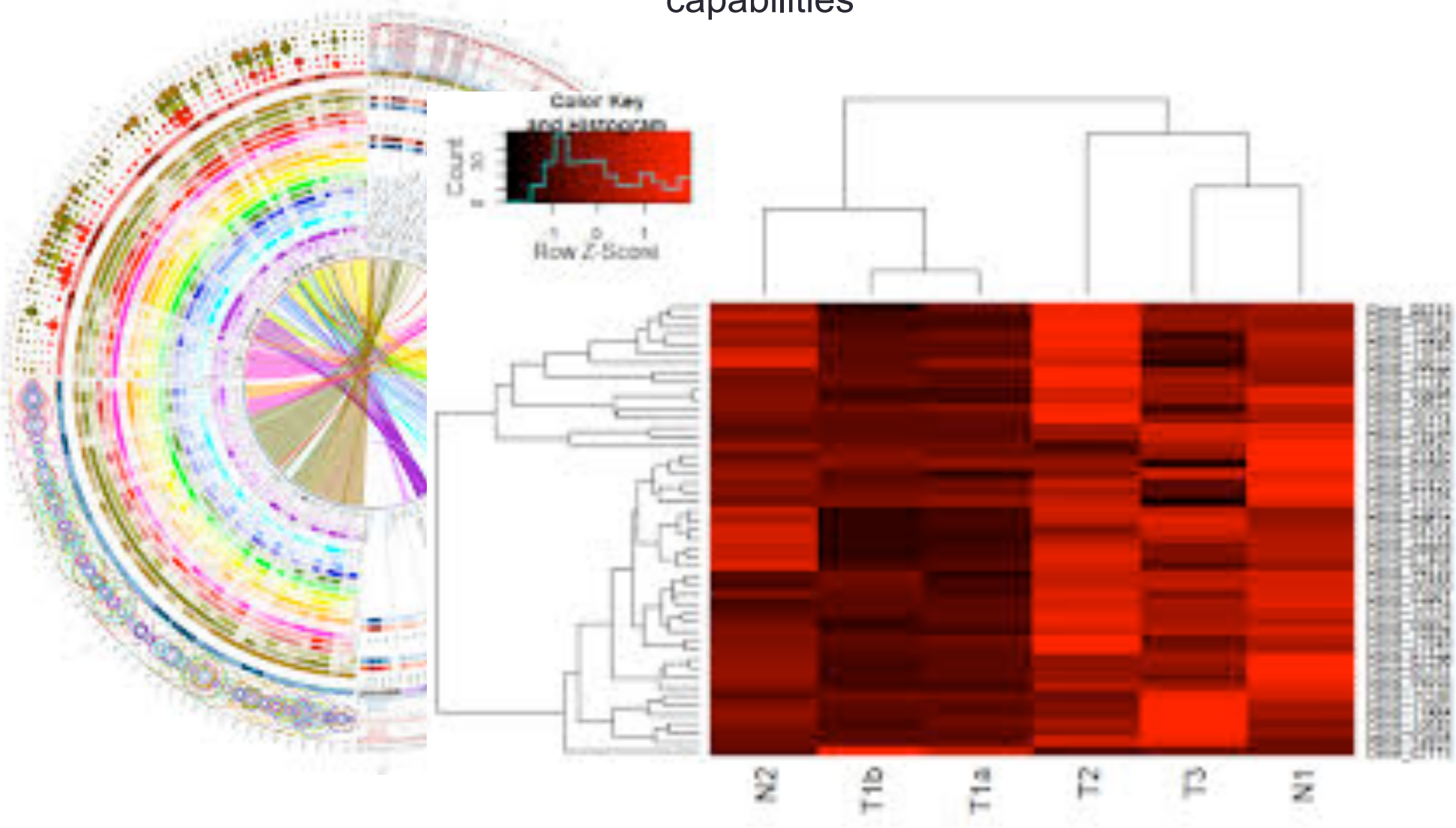
Index

Statistic Softwares– why R ?

- it is free version of *S*+ (http://cran.R-project.org)
- R is a *statistical language*
  - can perform any common statistical functions
  - interactive
  - It contains advanced statistical routines
- It runs on a variety of platforms including Windows, Unix and MacOS.
- It provides an unparalleled platform for programming new statistical methods in an easy and straightforward manner

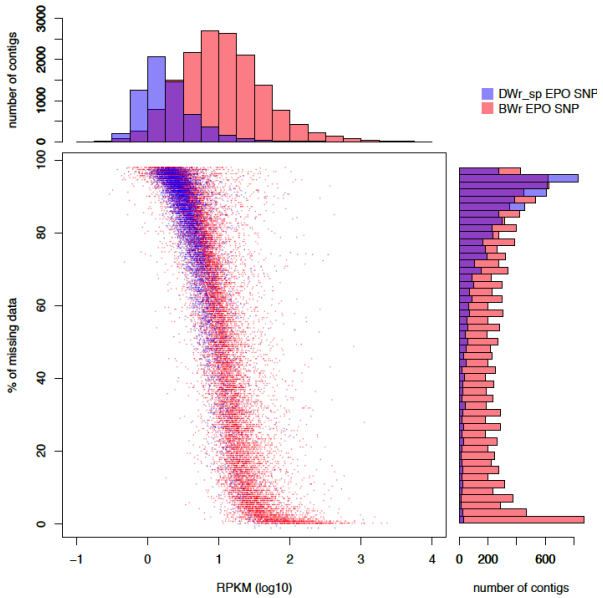
Why R?

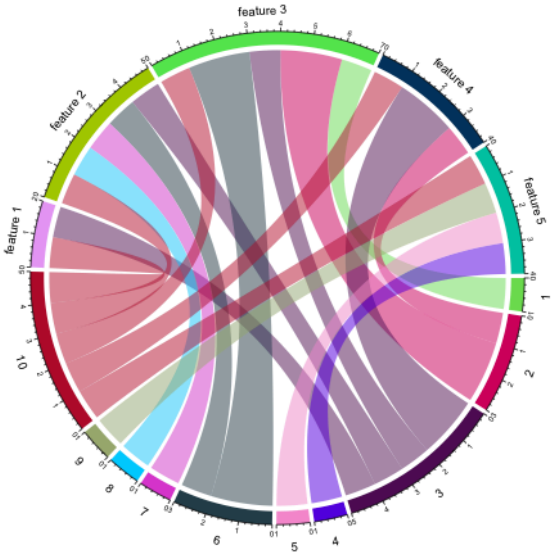
- It has state-of-the-art graphics capabilities

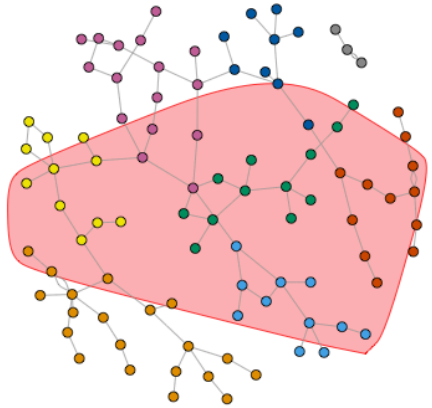


R-graph Gallery

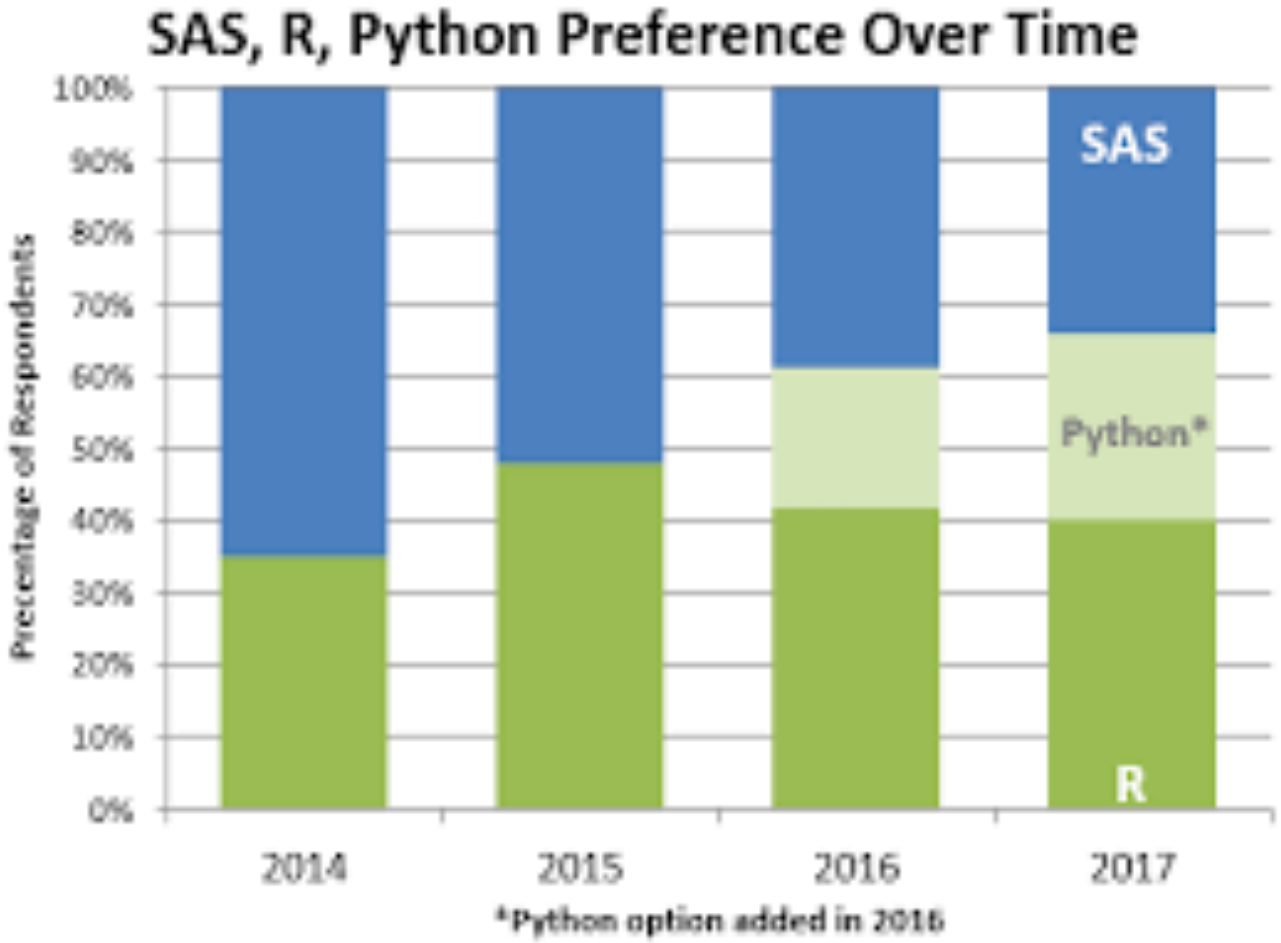
- http://www.r-graph-gallery.com/portfolio/ggplot2-package/







Use of R

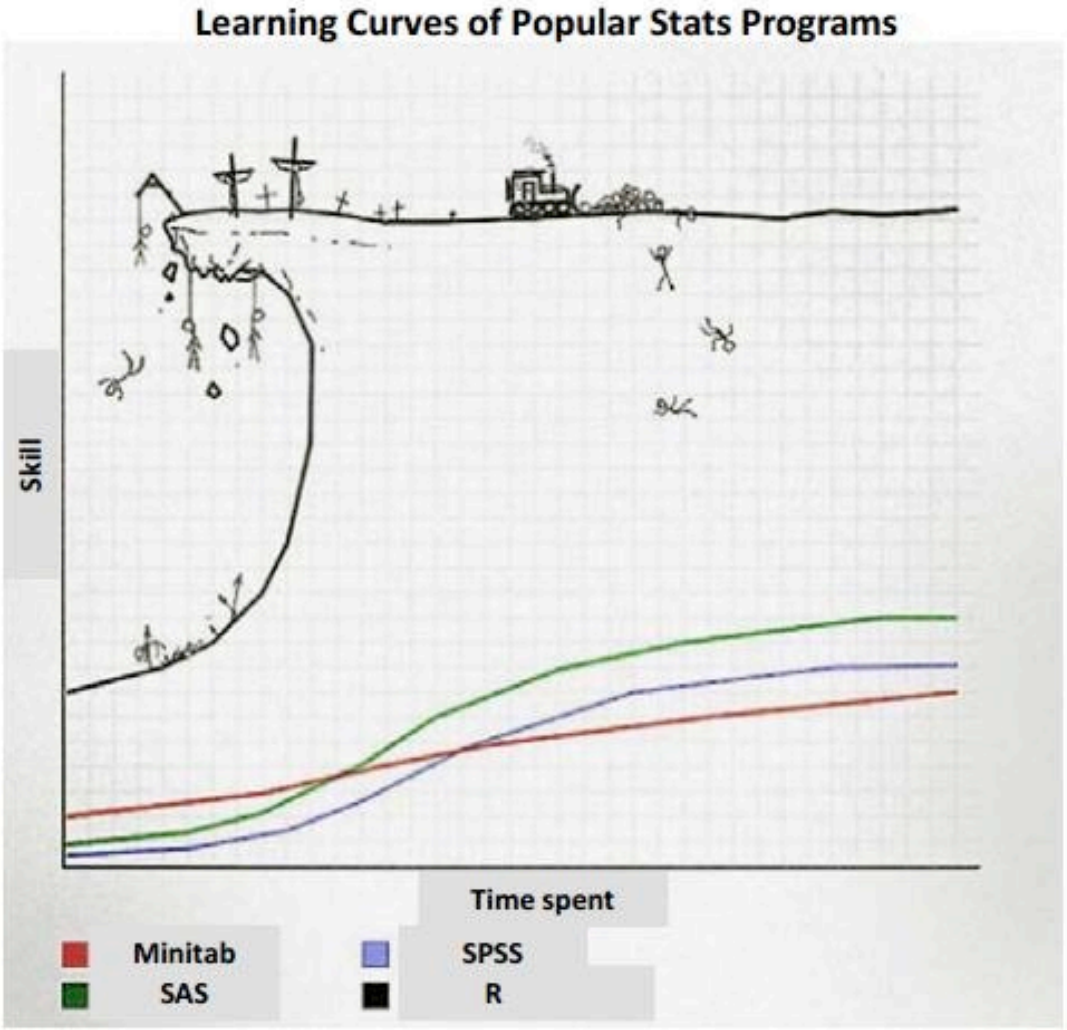


http://www.burtchworks.com/2016/07/13/sas-r-python-survey-2016-tool-analytics-pros-prefer/

Puntos débiles

- No es intuitivo ni fácil de usar al principio
- No tiene asistencia comercial
- Bastante más lento que otros lenguajes de programación (e.g. Perl, Java, C++)

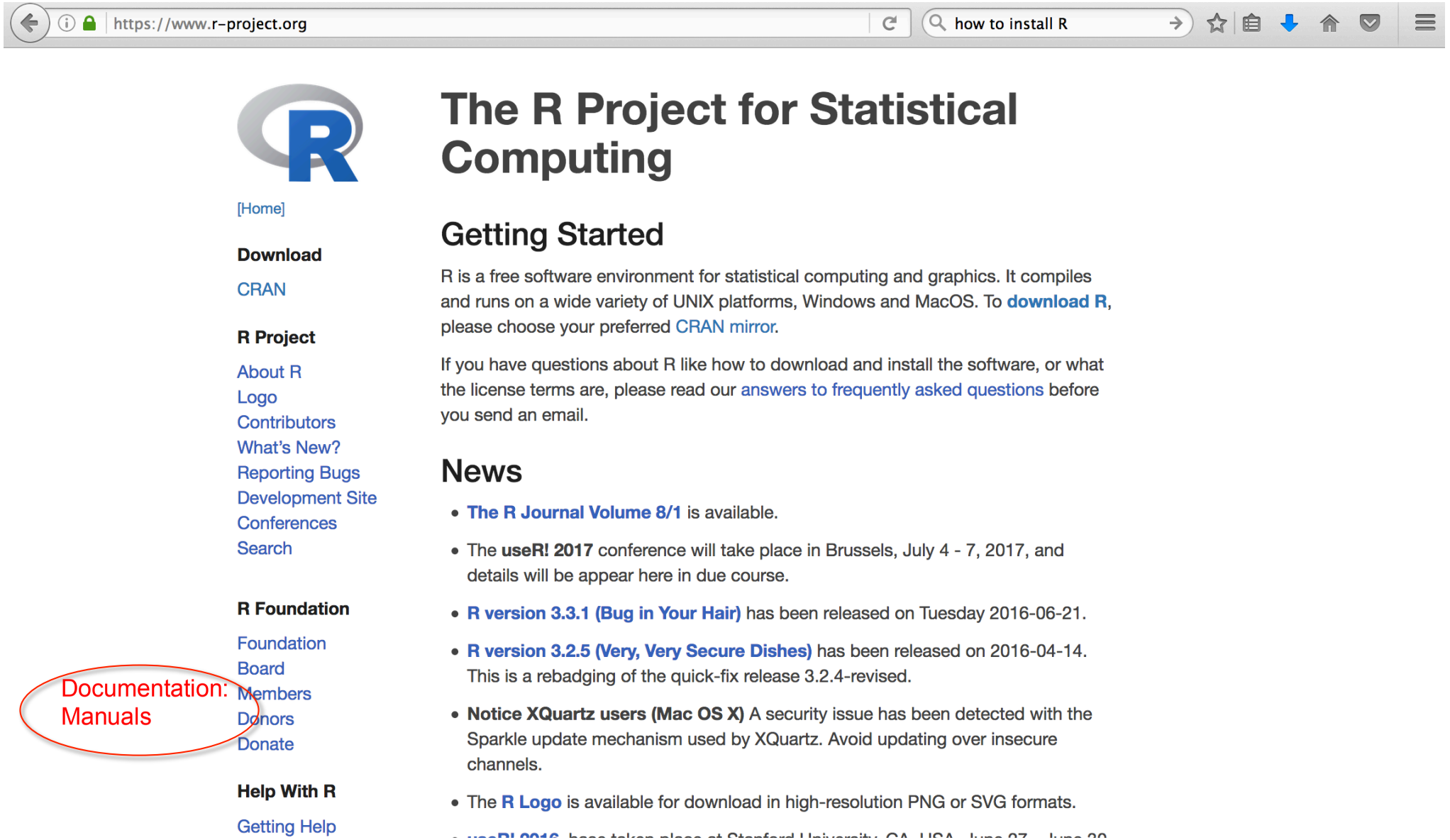
R Learning Curve



https://mobile.twitter.com/rogierK/status/730863729420701697/photo/1

http://www.vayapotra.es/wordpress/wp-content/uploads/2009/02/2rmqi6o.gif

R https://www.r-project.org/



The R manuals

The R Manuals

edited by the R Development Core Team.

The following manuals for R were created on Debian Linux and may differ from the manuals for Mac or Windows on platform-specific pages, but most parts will be identical for all platforms. The correct version of the manuals for each platform are part of the respective R installations. The manuals change with R, hence we provide versions for the most recent released R version (R-release), a very current version for the patched release version (R-patched) and finally a version for the forthcoming R version that is still in development (R-devel).

Here they can be downloaded as PDF files, EPUB files, or directly browsed as HTML:

Manual	R-release	R-patched	R-devel
An Introduction to R is based on the former "Notes on R", gives an introduction to the language and how to use R for doing statistical analysis and graphics.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Data Import/Export describes the import and export facilities available either in R itself or via packages which are available from CRAN.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Installation and Administration	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
Writing R Extensions covers how to create your own packages, write R help files, and the foreign language (C, C++, Fortran, ...) interfaces.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
A draft of **The R language definition** documents the language *per se*. That is, the objects that it works on, and the details of the expression evaluation process, which are useful to know when programming R functions.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Internals: a guide to the internal structures of R and coding standards for the core team working on R itself.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB

How to install R? Unix sources, binaries and documentation for R can be obtained via CRAN,

About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

Misc
Bioconductor
Related Projects
User Groups
Links

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
http://r.mirror.mendoza-conicet.gob.ar/	CONICET Mendoza
Australia	
http://cran.csiro.au/	CSIRO
http://cran.ms.u	
Austria	
http://cran.at.r	
Belgium	
http://www.fr	
Brazil	
http://cran-r.c3sl.ufpr.br/	Universidade Federal do Parana
http://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
http://www.vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
http://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba
Canada	
http://cran.stat.sfu.ca/	Simon Fraser University, Burnaby
http://mirror.its.dal.ca/cran/	Dalhousie University, Halifax
http://probability.ca/cran/	University of Toronto
http://cran.skazkaforyou.com/	iWeb, Montreal
http://cran.parentingamerica.com/	iWeb, Montreal
Chile	
http://dirichlet.mat.puc.cl/	Pontificia Universidad Catolica de Chile, Santiago
China	
http://ftp.ctex.org/mirrors/CRAN/	CTEX.ORG
http://cran.csdb.cn/	Computer Network Information Center, CAS, Beijing
http://mirrors.ustc.edu.cn/CRAN/	University of Science and Technology of China
http://mirrors.geoexpat.com/cran/	GeoExpat.Com
http://mirrors.xmu.edu.cn/CRAN/	Xiamen University
Colombia	
http://www.laqee.unal.edu.co/CRAN/	National University of Colombia
Denmark	
http://mirrors.dotsrc.org/cran/	dotsrc.org, Aalborg
France	

This server is hosted by the Institute for Statistics and Mathematics of the WU Wien.

Elige un servidor

Windows and MacOS. To

s to frequently asked

RAN. Binaries will arrive in

How to install R for windows

https://cran.r-project.org/doc/manuals/R-admin.html#Installing-R-under-Windows

3 Installing R under Windows

The `bin/windows` directory of a CRAN site contains binaries for a base distribution and a large number of add-on packages from CRAN to run on 32- or 64-bit Windows (XP or later) on '`ix86`' and '`x86_64`' CPUs.

Your file system must allow long file names (as is likely except perhaps for some network-mounted systems). If it doesn't also support conversion to short name equivalents (a.k.a. DOS 8.3 names), then R *must* be installed in a path that does not contain spaces.

Installation is *via* the installer `R-3.3.1-win.exe`. Just double-click on the icon and follow the instructions. When installing on a 64-bit version of Windows the options will include 32- or 64-bit versions of R (and the default is to install both). You can uninstall R from the Control Panel.

Note that you will be asked to choose a language for installation, and that choice applies to both installation and un-installation but not to running R itself.

See the R Windows FAQ for more details on the binary installer.

Next: Testing a Windows Installation, Previous: Installing R under Windows, Up: Installing R under Windows [Contents][Index]

3.1 Building from source

R can be built as either a 32-bit or 64-bit application on Windows: to build the 64-bit application you need a 64-bit edition of Windows: such an OS can also be used to build 32-bit R.

The standard installer combines 32-bit and 64-bit builds into a single executable which can then be installed into the same location and share all the files except the `.exe` and `.dll` files and some configuration files in the `etc` directory.

Building is only tested in a 8-bit locale: using a multi-byte locale (as used for CJK languages) is unsupported and may not work (the scripts do try to select a 'C' locale; Windows may not honour this).

NB: The build process is currently being changed to require external binary distributions of third-party software. Their location is set using macro `EXT_LIBS` with default setting `$(LOCAL_SOFT)`; the $(LOCAL_SOFT) macro defaults to `$(R_HOME)/extsoft`. This directory can be populated using `make rsync-extsoft`. The location can be overridden by setting `EXT_LIBS` to a different path in `src/gnuwin32/MkRules.local`. A suitable collection of files can also be obtained from https://CRAN.R-project.org/bin/windows/extsoft or https://www.stats.ox.ac.uk/pub/Rtools/libs.html.

How to install R? MacOS X

- https://cran.r-project.org/doc/manuals/R-admin.html#Installing-R-under-OS-X

4 Installing R under OS X

The front page of a CRAN site has a link 'Download R for OS X'. Click on that, then download the file `R-3.3.1.pkg` and install it. This runs on OS X 10.9 and later (Mavericks, Yosemite, El Capitan, …).

Installers for R-patched and R-devel are usually available from https://r.research.att.com.

For some older versions of the OS you can in principle (it is little tested) install R from the sources.

It is important that if you use a binary installer package that your OS is fully updated: look at 'Updates' from the 'App Store' to be sure. (If using XQuartz, check that is current.)

To install, just double-click on the icon of the file you downloaded. At the 'Installation Type' stage, note the option to 'Customize'. This currently shows four components: everyone will need the 'R Framework' component: the remaining components are optional. (The 'Tcl/Tk' component is needed to use package **tcltk**. The 'Texinfo' component is only needed by those installing source packages.)

This is an Apple Installer package. If you encounter any problem during the installation, please check the Installer log by clicking on the "Window" menu and item "Installer Log". The full output (select "Show All Log") is useful for tracking down problems. Note the the installer is clever enough to try to upgrade the last-installed version of the application where you installed it (which may not be where you want this time …).

Various parts of the build require XQuartz to be installed: : see https://xquartz.macosforge.org/. These include the **tcltk** package and the `X11` device: attempting to use these without XQuartz will remind you.

If you update your OS X version, you should re-install R (and perhaps XQuartz): the installer tailors the installation to the current version of the OS.

For building R from source, see OS X.

- Running R under OS X:
- Uninstalling under OS X:
- Multiple versions:

Next: Uninstalling under OS X, Previous: Installing R under OS X, Up: Installing R under OS X [Contents][Index]

4.1 Running R under OS X

There are two ways to run R on OS X from a CRAN binary distribution.

Tools

R
www.r-project.org

The engine*

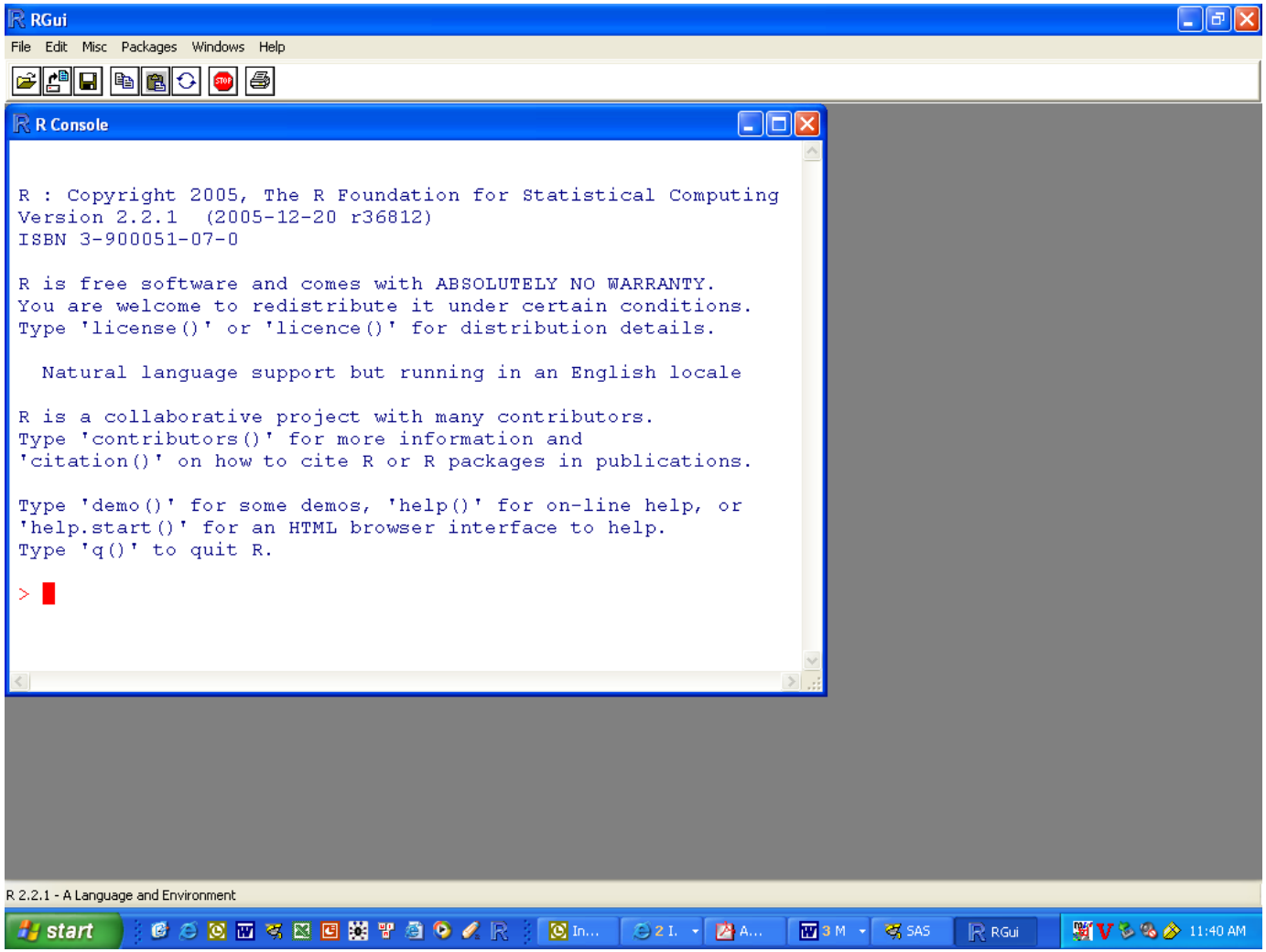
Rstudio
www.rstudio.org

The pretty face**

R Interfaz

- Arranca el sistema de R, y la ventana principal (RGui) aparecerá con una subventana (R Console)
- En la ventana `Console' el cursor estará esperando para que introduzcas comandos de R

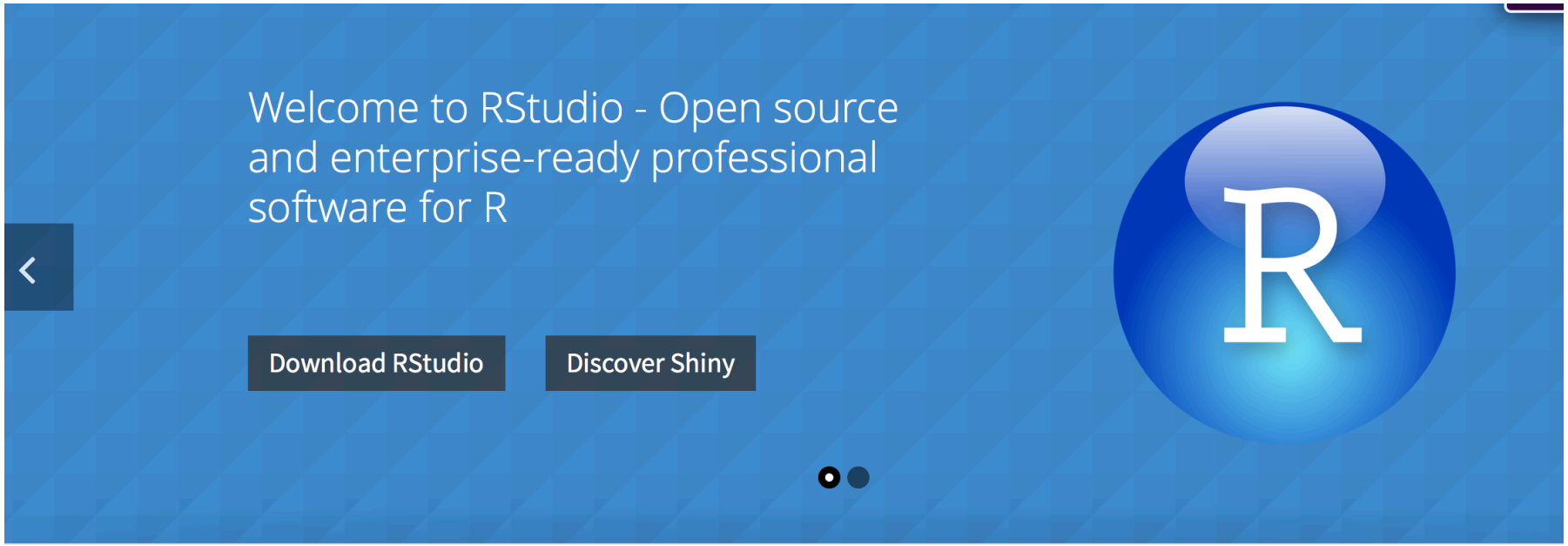
R console screenshot



IDE for R: RStudio

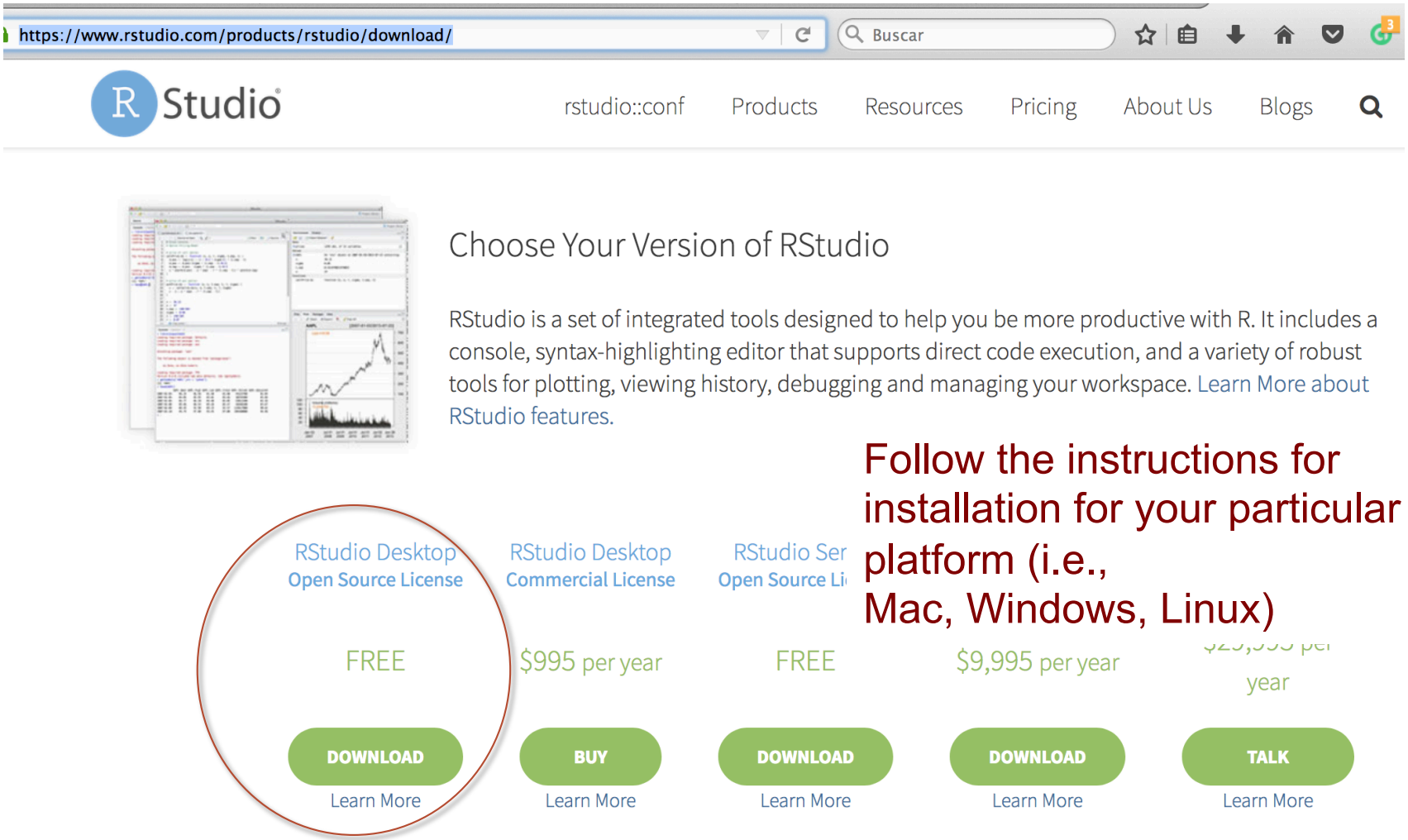
- "RStudio es un nuevo entorno de desarrollo (IDE) para R"

http://www.rstudio.org/download/



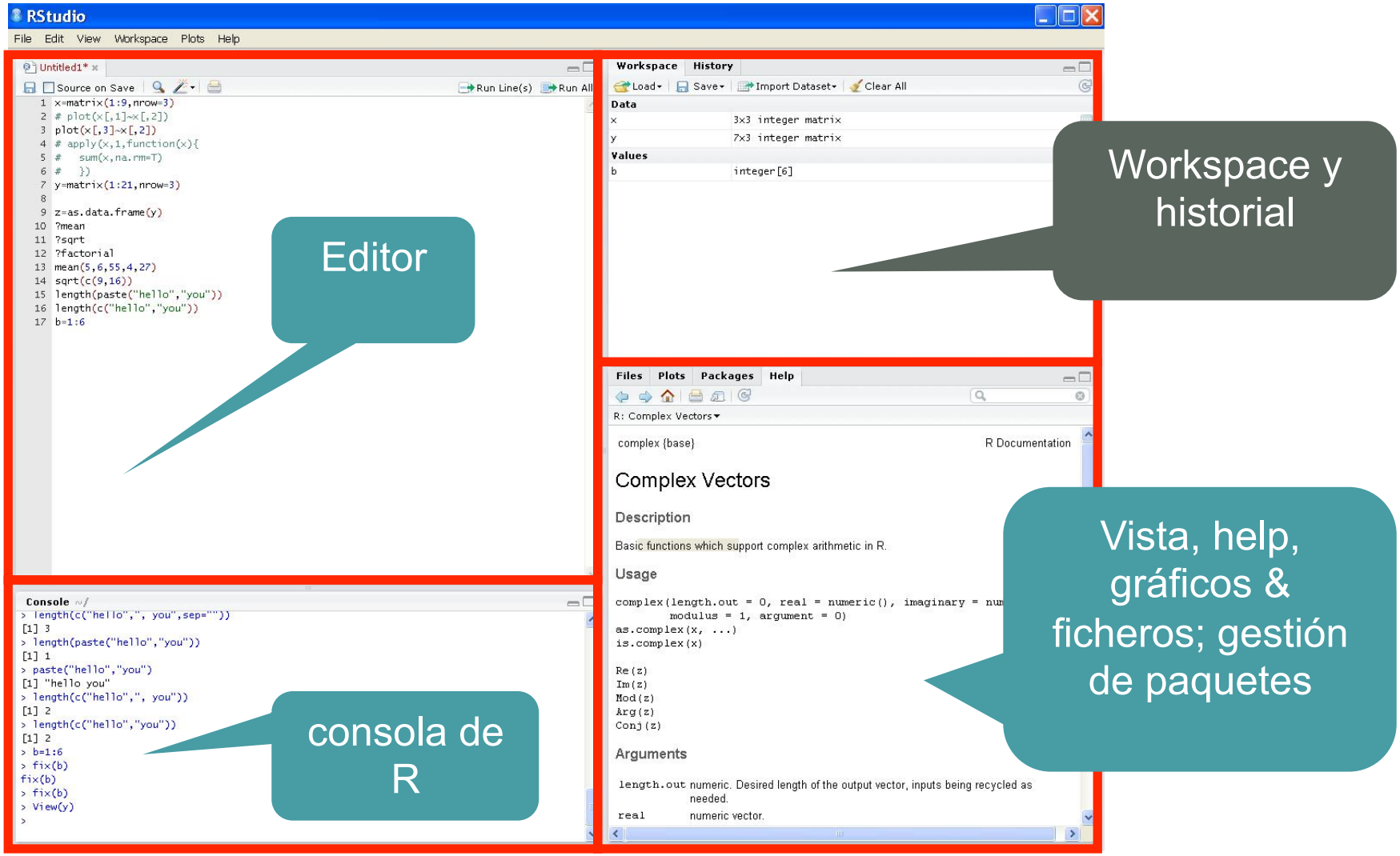
Install R-studio

Go to: https://www.rstudio.com/products/rstudio/download/



Follow the instructions for installation for your particular platform (i.e., Mac, Windows, Linux)

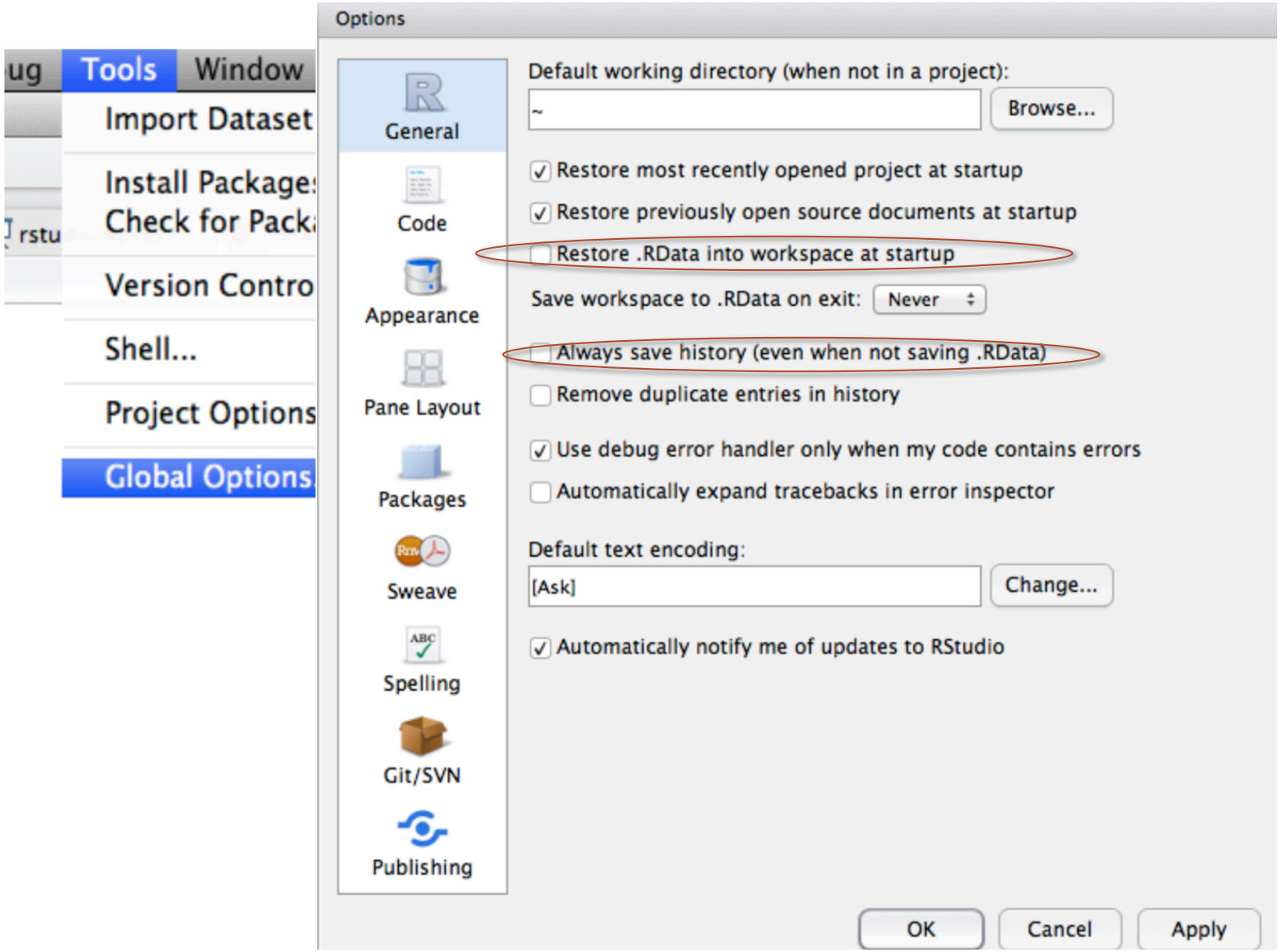
RStudio


RStudio
File Edit View Workspace Plots Help
Untitled1*
Source on Save
Run Line(s)
Run All
x=matrix(1:9,nrow=3)
# plot(x[,1]~x[,2])
plot(x[,3]~x[,2])
# apply(x,1,function(x){
# sum(x,na.rm=T)
# })
y=matrix(1:21,nrow=3)
z=as.data.frame(y)
?mean
?sqrt
?factorial
mean(5,6,55,4,27)
sqrt(c(9,16))
length(paste("hello","you"))
length(c("hello","you"))
b=1:6
Editor
Workspace
History
Load
Save
Import Dataset
Clear All
Data
x 3x3 integer matrix
y 7x3 integer matrix
Values
b integer[6]
Workspace y historial
Files
Plots
Packages
Help
R: Complex Vectors
complex {base}
R Documentation
Complex Vectors
Description
Basic functions which support complex arithmetic in R.
Usage
complex(length.out = 0, real = numeric(), imaginary = nu
modulus = 1, argument = 0)
as.complex(x, ...)
is.complex(x)
Re(z)
Im(z)
Mod(z)
Arg(z)
Conj(z)
Arguments
length.out numeric. Desired length of the output vector, inputs being recycled as needed.
real numeric vector.
Vista, help, gráficos & ficheros; gestión de paquetes
Console
> length(c("hello","", you",sep=""))
[1] 3
> length(paste("hello","you"))
[1] 1
> paste("hello","you")
[1] "hello you"
> length(c("hello","", you"))
[1] 2
> length(c("hello","you"))
[1] 2
> b=1:6
> fix(b)
fix(b)
> fix(b)
> View(y)
>
consola de R

R Tips

- R is case-sensitive
- Comment your code so you remember what it does; comments are preceded with **#**
- R scripts are simply text files with a **.R extension**
- Use Ctrl + R to submit code
- Use the Tab key to let R/R Studio finish typing commands for you
- Use Shift + down arrow to highlight lines or blocks of code
- In R Studio: Ctrl + 1 and Ctrl + 2 switches between script and console
- Use up and down arrows to cycle through previous commands in console
- Don’t be afraid of errors; you won’t break R
- If you get stuck, Google is your friend

Rstudio: basic settings recomendations

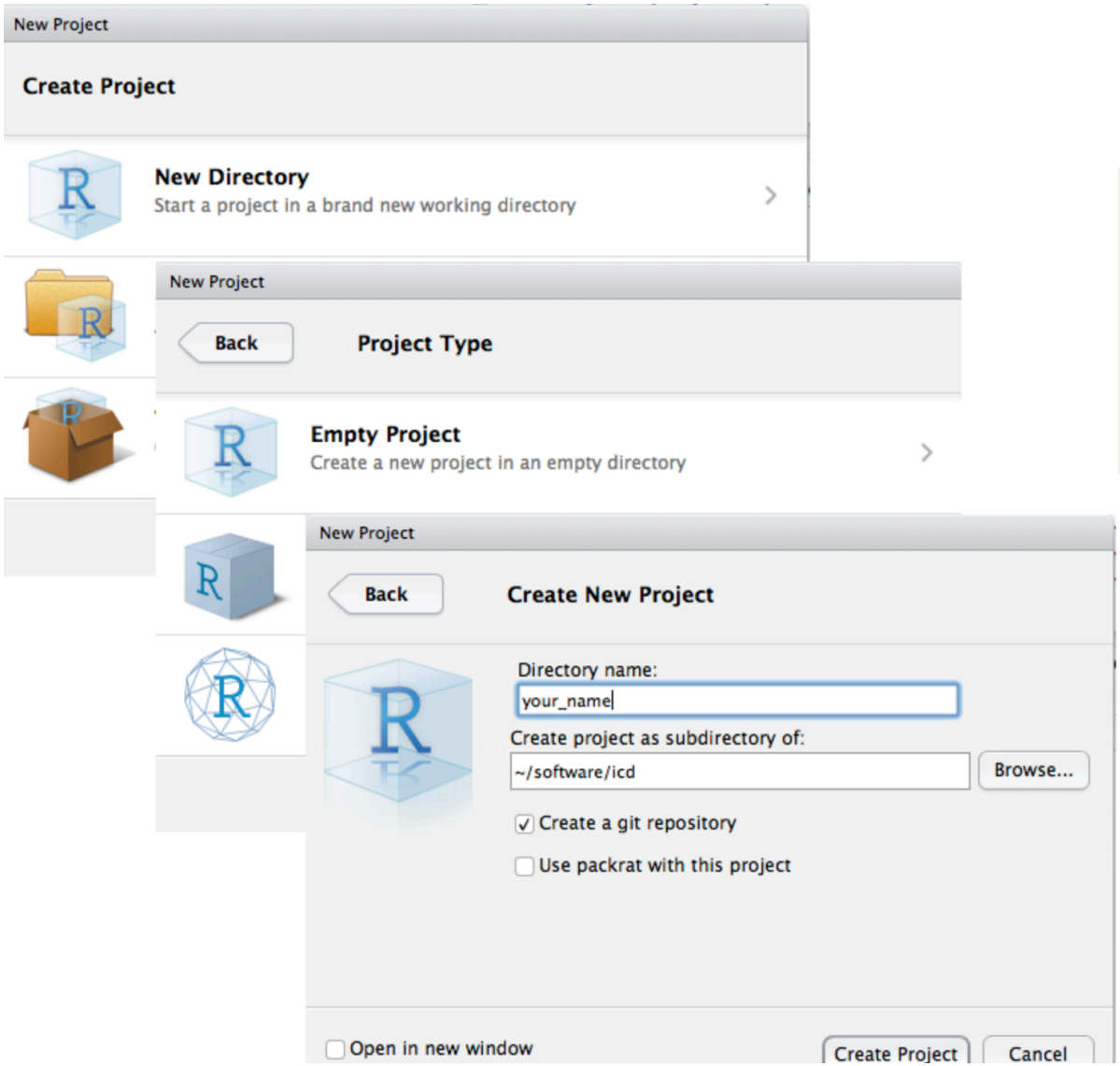


Project management in RStudio

- Using projects in RStudio is a powerful way of working on several projects at the same time. Each of the projects has its own:
  - Working directory
  - history
  - R source code files
  - Version control repository

Create a RStudio project for each of your projects.

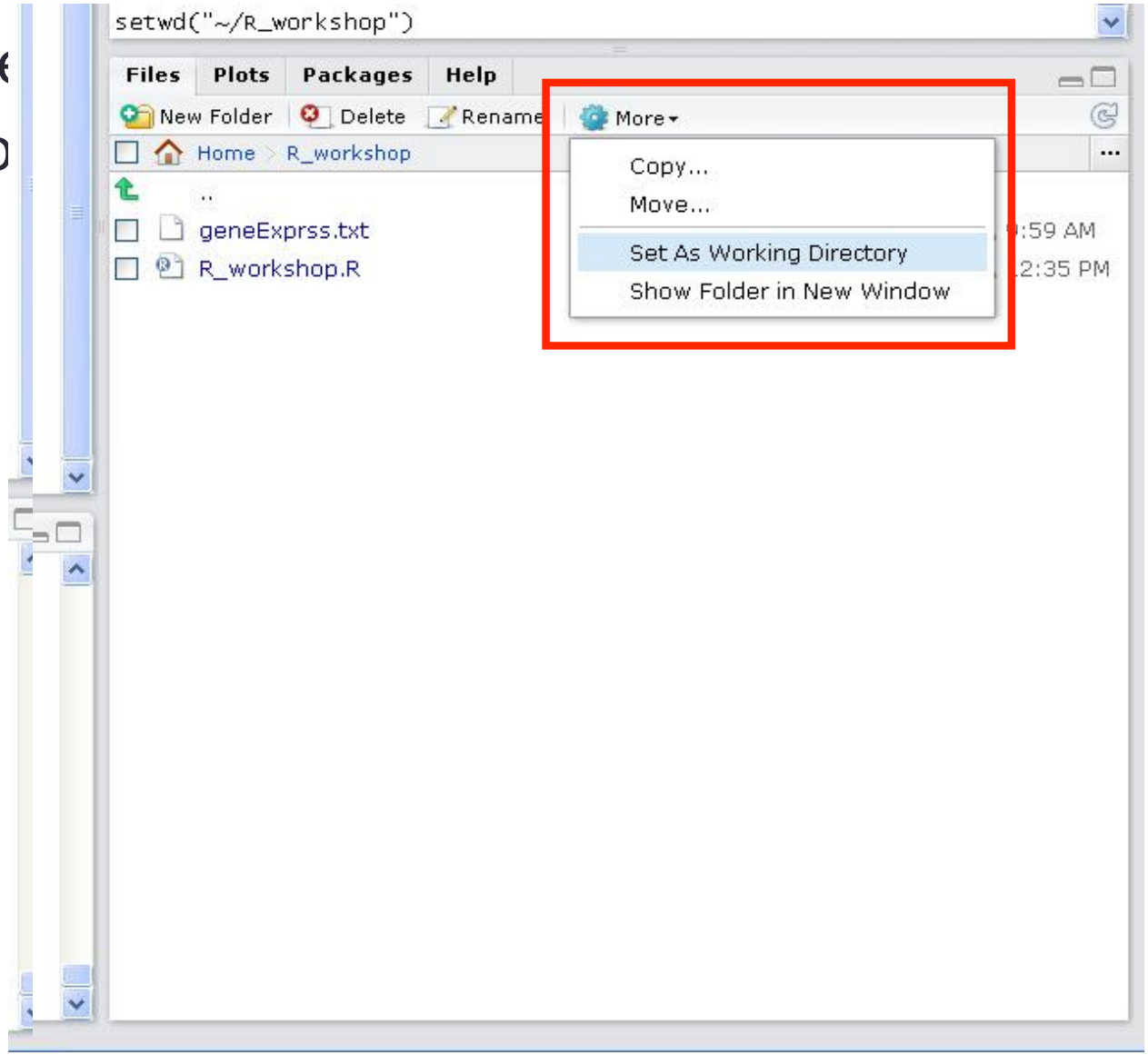
Project management in RStudio



- Select File > New project
- Select New Directory
- Select Empty project
- Select a name and sub directory of your project.

Como empezar con

- Cre
- Op



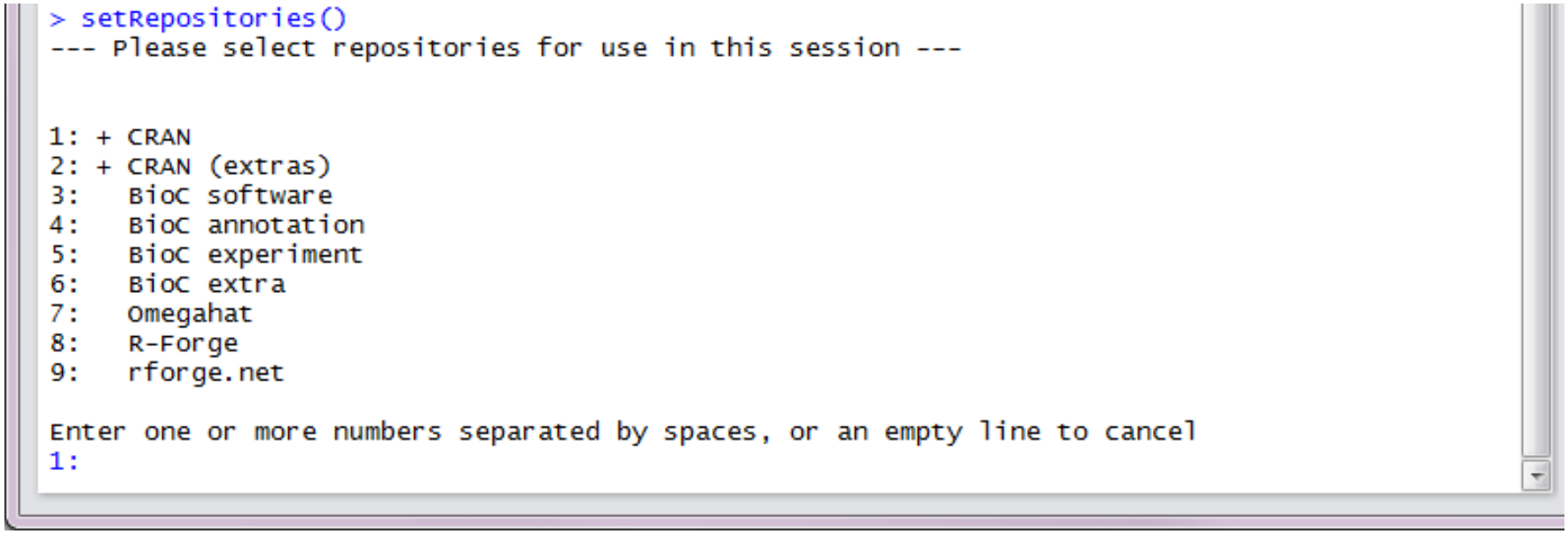
How to choose repositories

- Repositories host the packages
  - CRAN, CRANextra, BioCsoft, BioCann, BioCexp, BioCext, Omegahat, R-Forge and rforge.net

setRepositories()

- Use this code to set your repositories

```
setRepositories()
```



Paquetes en R

- Many contributed functionalities of R are available in R packages/libraries.
- Seven packages are distributed with R . The others need to be downloaded and installed separately.

  - use the function search to see a list of packages
  - **`> search()`**

```
[1] ".GlobalEnv" "package:stats" "package:graphics"
[4] "package:grDevices" "package:datasets"
  "package:utils"
```

- use the function install for installing packages

```
> install.packages("samr")
```

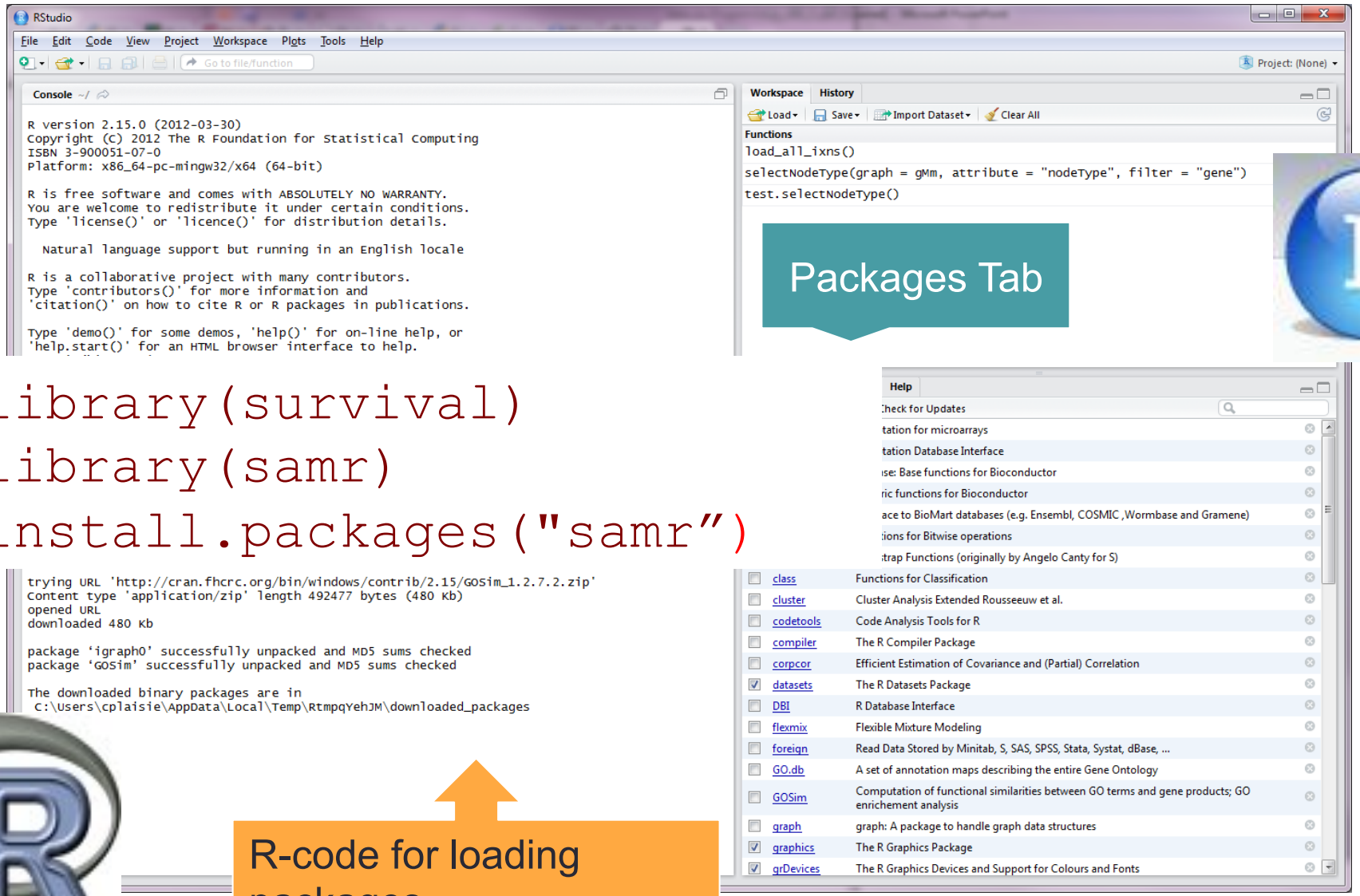
Librerías en R

- The function library can also be used to list all the available libraries on your system with a short description. Run the function without any arguments

```
> library()
Packages in library 'C:/PROGRA~1/R/R-25~1.0/
  library':
base                The R Base Package
Boot                 Bootstrap R (S-Plus) Functions
  (Canty)
class               Functions for Classification
cluster             Cluster Analysis Extended

> library(samr)
```

Como instalar librerias y paquetes



R Workspace

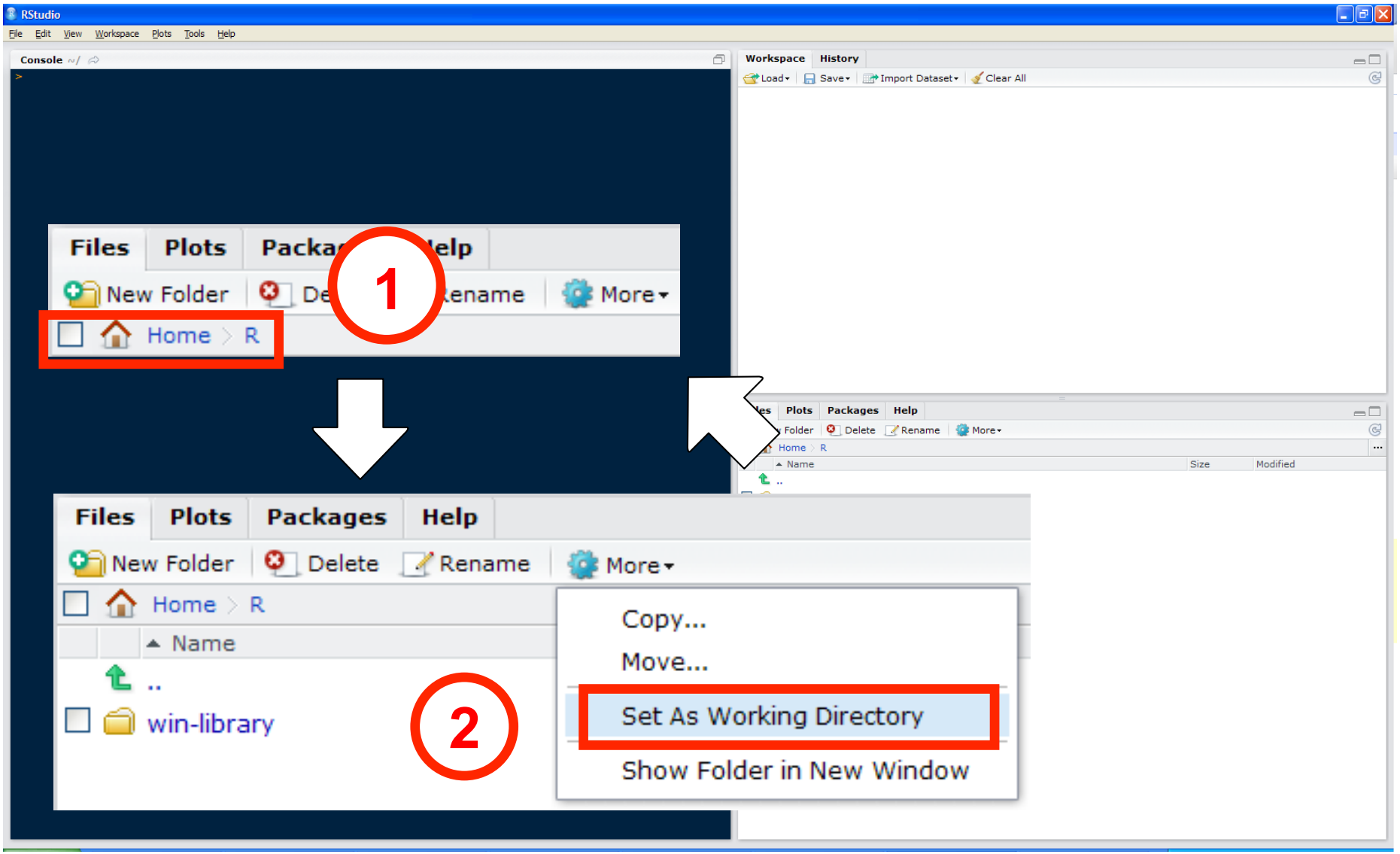
- Objects that you create during an R session are hold in memory
- The collection of objects that you currently have is called the workspace.
- This workspace is not saved on disk unless R is told to do so.
- Objects are lost when R is closed

R Workspace

- To save the workspace image. Go to the `File ‘ menu and then select `Save Workspace...', or use the `save.image` function.

```
## save to the current working directory
save.image()
## just checking what the current working
directory is
getwd()
## save to a specific file and location
save.image("C:\\Program Files\\R\\R-2.5.0\\bin\
\.RData")
```

Working Directory



R Workspace

```
getwd()
# print the current working directory

ls()
# list the objects in the current
  workspace
setwd(mydirectory)
# change to mydirectory
setwd("c:/docs/mydir")
```

R Workspace

- Commands are entered interactively at the R user prompt.
- **Up** and **down** arrow keys scroll through your command history.
- In R you can save your entire workspace, variables and all in its current state
- Then you can reload this at a later time or can provide this to collaborators
- Workspace data files are saved with the extension '.Rdata'

R Workspace

```
# save your command history
  savehistory(file="myfile")
# default is ".Rhistory"
# recall your command history
  loadhistory(file="myfile")
# default is ".Rhistory“
```

R Help

Once **R** is installed, there is a comprehensive built-in help system.

- Inside of R

```
help.start()    # general help
help(sqrt)       # help about function foo
?foo             # same thing
apropos("str") # list all function containing string foo
example(foo)    # show an example of function foo
# search for foo in help manuals and archived mailing lists
RSiteSearch("foo")
# get vignettes on using installed packages
vignette("foo") # show specific vignette
```

- On the web
  - www.rseek.org and R only search engine
  - CRAN
  - Google topic with CRAN

R Help

Examples

Many functions and data sets in R include example code demonstrating typical uses.

```
>example(hist)
```

will generate a number of example plots (and provide you with the commands used to create them).

Demos

Demos are bits of R code that can be executed using the demo() command with the name of the demo. They are intended to illustrate a concept, a method, or some such thing, and are independent of any particular function or data set.

You can get a list of available demos using

- `demo() # all demos`
- `Demo(graphics)`

```
> demo(package='mosaic') # just demos from mosaic package
```

Vignettes

- Some packages have vignettes
  - A vignettes is an example of how to run the code and a lot of additional text explaining a lot more that you may want to know
- List all available vignettes:

```
vignette()
```

- Display the vignette as a pdf by executing. In Rstudio they are available through the help panel

```
vignette('<topic>')
vignette("grid")
```

- To play with the vignette code

```
vig = vignette('<topic>')
edit(vig)
```

Ready to Code!

- The working directory is where Rstudio will look first for scripts
- Keeping everything in a self contained directory helps organize code and analyses
- Check you current working directory with

```
getwd()
```

R as a calculator

- R evaluates expressions.
- Allowing its use as a calculator.

Tip:

Predefined symbols:

`pi`, `letters`, `month.name`

Special symbols:

`NA`, `NaN`, `Inf`, `NULL`, `TRUE`, `FALSE`

```
> -27*12/21
[1] -15.42857

> sqrt(10)
[1] 3.162278

> log(10)
[1] 2.302585

> log10(2+3*pi)
[1] 1.057848

> exp(2.7689)
[1] 15.94109

> (25 - 5)^3
[1] 8000

> cos(pi)
[1] -1
```

“atomic” classes of objects

- character
- numeric (real numbers)
- integer
- complex
- logical (True/False)

Entering inputs: assignments

We need to be able to store intermediate results.

In R, we can assign data to variables using the symbol **<-**.

```
msg <- "hello”
y <- sin(pi/6)

The # character indicates a comment.
Anything to the right of the # (including the # itself)
is ignored.
```

Deleting variables

- rm() function

```
> x <- 2*pi
> x
[1] 6.283185
> rm(x)
> x
Error: object "x" not found
```

To erase the entire workspace at once. The rm function has a list argument

consisting of a vector of names of variables to remove.

```
> ls()
[1] "f" "x" "y" "z"
> rm(list=ls())
> ls()
character(0)
```

Strings

Although you can accomplish all of your string-handling needs in **R**, other programming languages may be more suitable.

```
> "Hello"
[1] "Hello"
> x <- paste("Hello", "World")
> x
[1] "Hello World”
```

By default, paste() puts a space between the different elements,

Task:
Assign a first and a last name to two variables.
Create a third variable that contains the initials.

Numbers

- Numbers in R a generally treated as numeric objects (i.e. double precision real numbers)
- There is also a special number `Inf` which represents infinity;

```
1 / 0; Inf can be used in ordinary calculations;
1 / Inf is 0
```

- The value `NaN` represents an undefined value (\not a number");

```
0 / 0; NaN
can also be thought of as a missing value
```

Using NA

- R's statistical functions, can instruct the function to skip over any missing values, or NAs:

```
> x <- c(88,NA,12,168,13)
> x
[1] 88 NA 12 168 13
> mean(x)
[1] NA
> mean(x,na.rm=T)
[1] 70.25
> x <- c(88,NULL,12,168,13)
> mean(x)
[1] 70.25
```

- In the first call, mean() refused to calculate, as one value in x was NA. But by setting the optional argument na.rm (NA remove) to true (T), we calculated the mean of the remaining elements.

Arithmetic Operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Numeric Functions

Function	Description
abs(*x*)	absolute value
sqrt(*x*)	square root
ceiling(*x*)	ceiling(3.475) is 4
floor(*x*)	floor(3.475) is 3
trunc(*x*)	trunc(5.99) is 5
round(*x*, digits=*n*)	round(3.475, digits=2) is 3.48
signif(*x*, digits=*n*)	signif(3.475, digits=2) is 3.5
cos(*x*), sin(*x*), tan(*x*)	also acos(*x*), cosh(*x*), acosh(*x*), etc.
log(*x*)	natural logarithm
log10(*x*)	common logarithm
exp(*x*)	e^*x*

Logical Operators

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if x is TRUE

DATA TYPES:

- Vectors
- Matrices
- Factors
- Lists
- Dataframes
- Names

Data structures

"Vector"

"Matrix"

”Array"

”Dataframe”. Columns can be of different modes

“Listas” Can contain:
vectors, arrays, matrices, Dataframes, lists

Vectors

- The c() function can be used to create vectors of objects.

```
> x <- c(0.5, 0.6)       ## numeric
> x <- c(TRUE, FALSE)    ## logical
> x <- c(T, F)           ## logical
> x <- c("a", "b", "c") ## character
> x <- 9:29              ## integer
> x <- c(1+0i, 2+4i)     ## complex
```

- Using the vector() function

```
> x <- vector("numeric", length = 10)
> x
[1] 0 0 0 0 0 0 0 0 0 0
```

Indexing

Indexing allows to directly extract elements

```
> # Indexing a vector
> Num <- c(0,3,2,2,1)
> Num[1]
[1] 0
> Num[2:3]
[1] 3 2
> Num[c(1,3)]
[1] 0 2
```

Manipulation of vectors

- Our vector: `x=c(100,101,102,103)`
- `[]` are used to access elements in **x**
- Extract 2nd element in **x**

```
> x[2]
[1] 101
```

- Extract 3rd and 4th elements in **x**

```
> x[3:4] # or x[c(3,4)]
[1] 102 103
```

Manipulating vectors

```
> x
[1] 100 101 102 103
```

- Add 1 to all elements in **x**:

```
> x+1
[1] 101 102 103 104
```

- Multiply all elements in **x** by 2:

```
> x*2
[1] 200 202 204 206
```

Vectors comparisons

Operator	Result
x == y	Returns TRUE if x exactly equals y
x != y	Returns TRUE if x differs from y
x > y	Returns TRUE if x is larger than y
x >= y	Returns TRUE if x is larger than or exactly equal to y
x < y	Returns TRUE if x is smaller than y
x <= y	Returns TRUE if x is smaller than or exactly equal to y
x & y	Returns the result of x and y
x | y	Returns the result of x or y
! x	Returns not x
xor(x,y)	Returns the result of x xor y (x or y but not x and y)

Recycling Rule in R

- When doing vector arithmetic, R performs element-by-element operations. That works well when both vectors are of the same length
- For vectors with unequal lengths R invokes the Recycling Rule.
  - When the shorter vector is exhausted while the longer vector still has unprocessed elements, R returns to the beginning of the shorter vector, “recycling” its elements.

```
> x=c(2:5)
> y=3
> x
[1] 2 3 4 5
> y
[1] 3 8
x>y
[1] FALSE FALSE  TRUE FALSE
```

Mixing Objects

- What about the following?

```
> y <- c(1.7, "a")  ## character
> y <- c(TRUE, 2)   ## numeric
> y <- c("a", TRUE) ## character
```

- When different objects are mixed in a vector, *coercion* occurs so that every element in the vector is of the same class.

Coercing types

Tipo	*Comprobación*	*Coerción*
array	is.array()	as.array()
character	is.character()	as.character()
complex	is.complex()	as.complex()
double	is.double()	as.double()
factor	is.factor()	as.factor()
integer	is.integer()	as.integer()
list	is.list()	as.list()
logical	is.logical()	as.logical()
matrix	is.matrix()	as.matrix()
NA	is.na()	-
NaN	is.nan()	-
NULL	is.null()	as.null()
numeric	is.numeric()	as.numeric()
ts	is.ts()	as.ts()
vector	is.vector()	as.vector()

*Tabla de David Conesa, GE**ei** **tE**ma (UV)*

Mixing Objects explicitly

- Objects can be explicitly coerced from one class to another using the as.* functions, if available.

```
> x <- 0:6
> class(x)
[1] "integer"
> as.numeric(x)
[1] 0 1 2 3 4 5 6
> as.logical(x)
[1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE
> as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
> as.complex(x)
[1] 0+0i 1+0i 2+0i 3+0i 4+0i 5+0i 6+0i
10
```

Generating Vector Sequences with seq()

- the seq() (or sequence) function, which generates a sequence in arithmetic progression

```
> seq(from=12,to=30,by=3)
[1] 12 15 18 21 24 27 30
```

The spacing can be a non-integer value, too, say 0.1.

```
> seq(from=1.1,to=2,length=10)
[1] 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

Repeating Vector Constants with rep()

- The rep() (or repeat) function allows us to conveniently put the same constant into long vectors. The call form is rep(x,times) , which creates a vector of times*length(x) elements—that is, times copies of x . Here is an example:

```
> x <- rep(8,4)
> x
[1] 8 8 8 8
> rep(c(5,12,13),3)
[1] 5 12 13 5 12 13 5 12 13
> rep(1:3,2)
[1] 1 2 3 1 2 3
```

- There is also a named argument each, with very different behavior, which interleaves the copies of x .

```
> rep(c(5,12,13),each=2)
[1] 5 5 12 12 13 13
```

Using all() and any()

- The any() and all() functions are handy shortcuts. They report whether any or all of their arguments are TRUE .

```
> x <- 1:10
> any(x > 8)
[1] TRUE
> any(x > 88)
[1] FALSE
> all(x > 88)
[1] FALSE
```

Manipulating vectors

Our vector: `x=100:150`

Elements of **x** higher than 145

```
> x[x>145]
[1] 146 147 148 149 150
```

Elements of **x** higher than 135 and lower than 140

```
> x[ x>135 & x<140 ]
[1] 136 137 138 139
```

Vector In, Vector Out

- You saw examples of vectorized functions earlier in the chapter, with the + and * operators. Another example is > .

```
> u <- c(5,2,8)
> v <- c(1,3,9)
> u > v
[1] TRUE FALSE FALSE
```

Using arithmetic vectors operations

Function	What it does
sum(x)	Calculates the sum of all values in x
prod(x)	Calculates the product of all values in x
min(x)	Gives the minimum of all values in x
max(x)	Gives the maximum of all values in x
cumsum(x)	Gives the cumulative sum of all values in x
cumprod(x)	Gives the cumulative product of all values in x
cummin(x)	Gives the minimum for all values in x from the start of the vector until the position of that value
cummax(x)	Gives the maximum for all values in x from the start of the vector until the position of that value
diff(x)	Gives for every value the difference between that value and the next value in the vector