

# Introduction to



**BIST Biostatistics course**

May 6<sup>th</sup>, 2016

Sarah Bonnin

CRG Bioinformatics Core Facility



Barcelona Institute of  
Science and Technology

06/05/16



Barcelona  
Biomedical  
Research  
Park

# Outline

- R: what, when, why?
- Getting started
- Data structures
- Libraries / Packages
- Basic commands
- Functions
- R Studio

**R: what, when, why?**

# What is R?

- Integrated suite of software facilities for **data manipulation, calculation** and **graphical display**.
- Simple and effective programming language which includes conditionals, loops etc.
- Implementation of the **S** programming language (Bell laboratories)

# What is R?

- Created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand
- Now developed by the “**R development core team**”
- GNU project (free software, pass collaboration)

→ **Open source !**

# R: what, when, why?



[\[Home\]](#)

## Download

[CRAN](#)

## R Project

[About R](#)

[Logo](#)

[Contributors](#)

[What's New?](#)

[Mailing Lists](#)

[Bug Tracking](#)

[Development Site](#)

[Conferences](#)

[Search](#)

## R Foundation

[Foundation](#)

[Board](#)

[Members](#)

[Donors](#)

[Donate](#)

## Documentation

[Manuals](#)

[FAQs](#)

[The R Journal](#)

[Books](#)

[Certification](#)

[Other](#)

## Links

[Bioconductor](#)

[Related Projects](#)

# The R Project for Statistical Computing

## Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).

If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

## News

- Beta test period for version 3.3.0 has been extended to accommodate new Windows toolchain for CRAN. Final release rescheduled for Tuesday 2016-05-03.
- **Notice XQuartz users (Mac OS X)** A security issue has been detected with the Sparkle update mechanism used by XQuartz. Avoid updating over insecure channels.
- **R version 3.2.4 (Very Secure Dishes)** has been released on Thursday 2016-03-10.
- **R version 3.3.0 (Supposedly Educational) prerelease versions** will appear starting Monday 2016-03-14. Final release is scheduled for Thursday 2016-04-14.
- The **R Logo** is available for download in high-resolution PNG or SVG formats.
- **useR! 2016**, will take place at Stanford University, CA, USA, June 27 - June 30, 2016.
- **The R Journal Volume 7/2** is available.
- **R version 3.2.3 (Wooden Christmas-Tree)** has been released on 2015-12-10.
- **R version 3.1.3 (Smooth Sidewalk)** has been released on 2015-03-09.

<https://www.r-project.org/>

# When to use R?

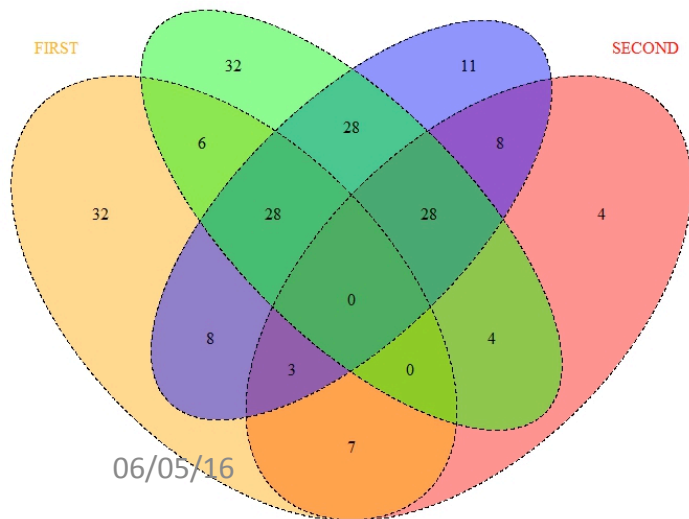
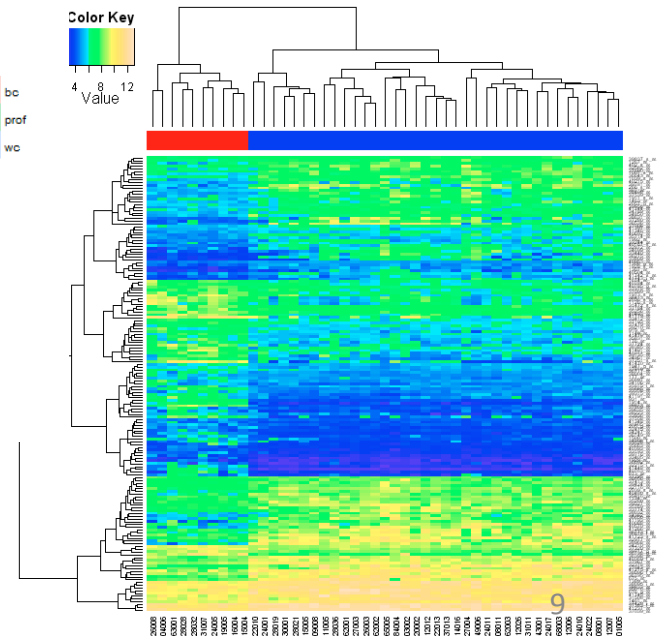
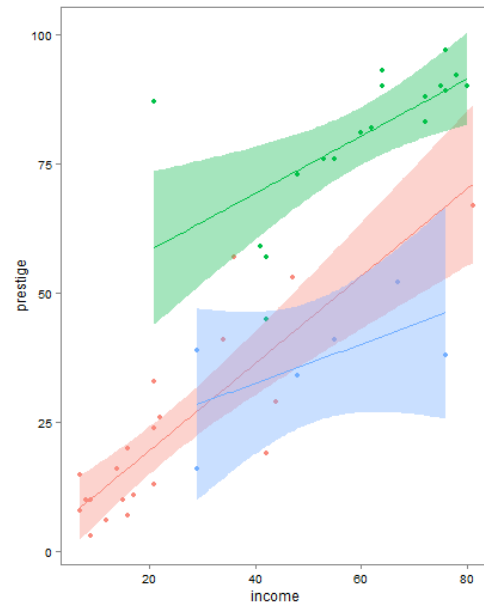
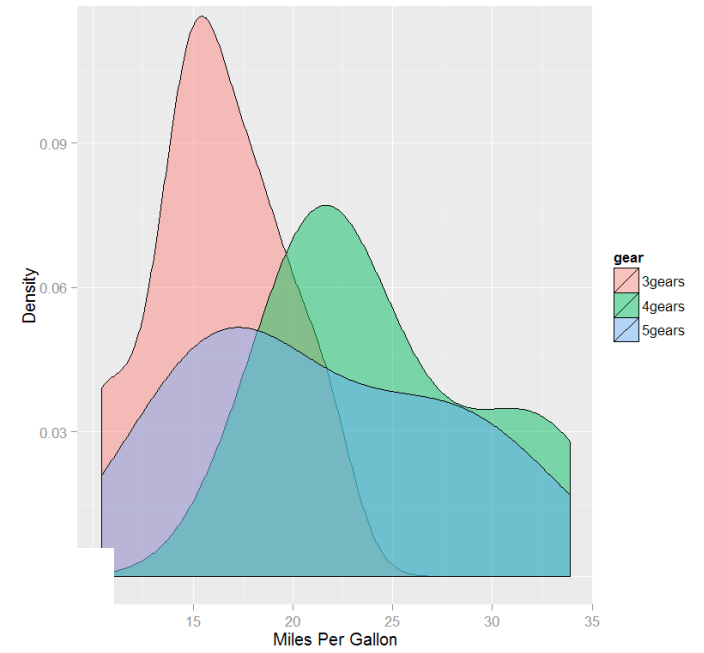
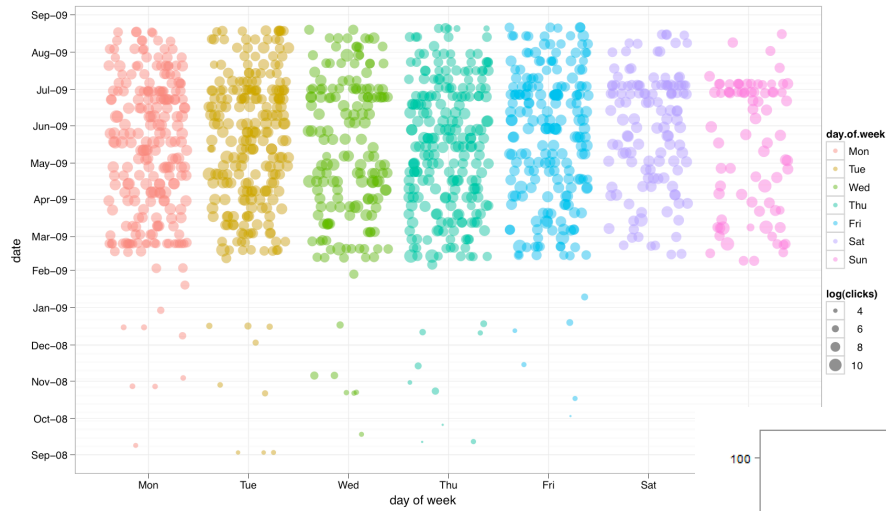
- Data analysis
- Statistical modeling
- Simulation
- Graphics

# Why to use R?

- Flexible
- Powerful
- Interactive
- **Very active community of developers and users!**



# R: what, when, why?

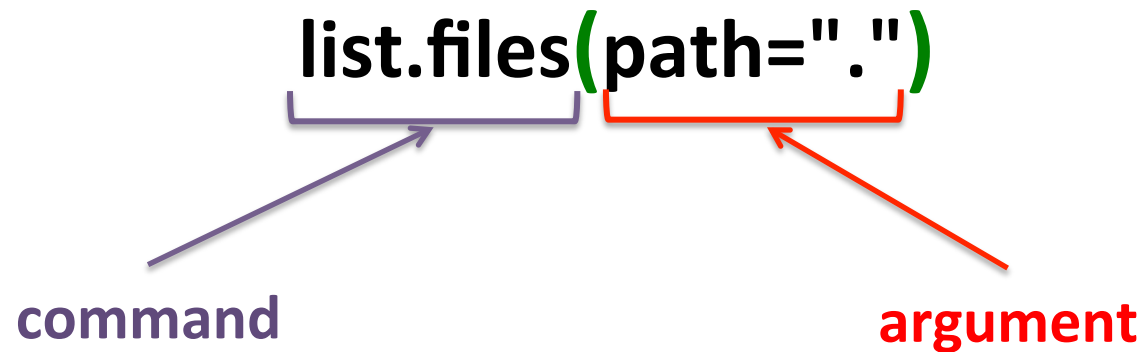


# Getting started

# R syntax

**function(arguments)**

example:



# R syntax

**a <- function(arguments)**

example:

**myobject <- list.files(path=".")**

**object**

**assignment operator**

# R syntax

- Case sensitive:

List.files  list.files

- Comment lines start with #
- Commands separated by ; or a new line
- Arguments in a function separated by ,

# Getting help

- **help(function)**
- **?function**
  
- Google!

## *Getting started*

# Starting R interactively from a terminal

- Starting session:

**R**

- Ending session:

**q()**

# Data Structures

**Vectors**

**Factors**

**Matrices**

**Data frames**

**List**



# Data Types

- Every value has a **data type** that tells what sort of value it is.
- Most common data types:
  - Numeric (Numbers)
  - Character (Text)
  - Logical (True / False)
- Checking object type with `mode()`:
  - `mode("a") / [1] "character"`
  - `mode(10) / [1] "numeric"`
  - `mode(FALSE) / [1] "logical"`

# Vectors

- Sequence of data elements of the same type
- Elements of an atomic vector are of **one type only**, either:
  - Numeric (1, 2, 5.3, 6, -2, 4)
  - Character ("one", "two", "three")
  - Logical (TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)

# Vectors

- Assignment of values to vector using the **c** command (combining elements)

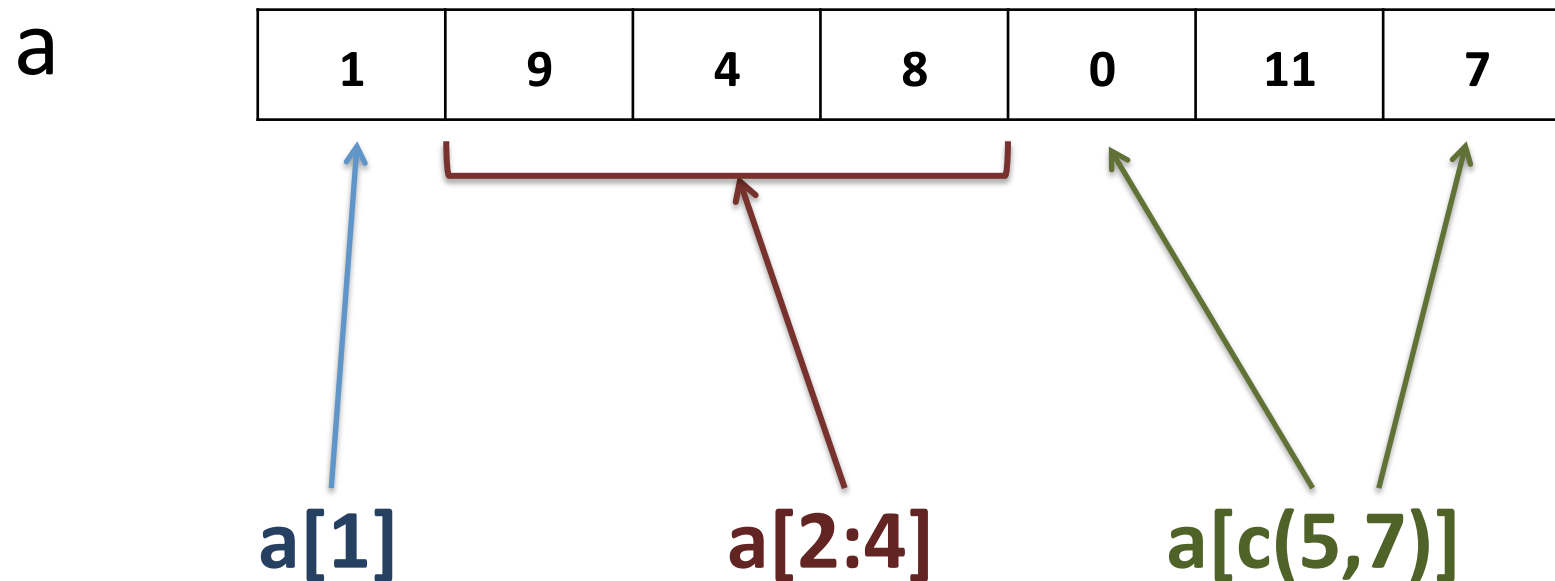
```
myvector <- c(0.5, 2, 10, 3, 8)
```

```
myvector <- 1:5  same as c(1, 2, 3, 4, 5)
```

- Checking if an object is a vector with **is.vector(myvector)**

# Vectors

- Fetch elements of a vector: subscripts



# Factors

- Vector **object** used to specify a **discrete classification** of the components in a data set:  
→ **Categorical variables**
- Used mainly in **statistical modeling**, but also in some graphical functions.
- Similar to vectors, but their values are limited to a **fixed set of possible values**.

# Factors

- Both numeric and character variables can be made into factors.

```
myfactor <- factor(c("a","a","b","c",1,2))
```

```
myfactor
```

```
[1] a a b c 1 2
```

```
Levels: 1 2 a b c ] ← Levels are character vectors
```

# Factors

- Checking if an object is a factor:
  - **is.factor**(myfactor)
- Transforming a vector into a factor:
  - **as.factor**(myvector)

# Matrices

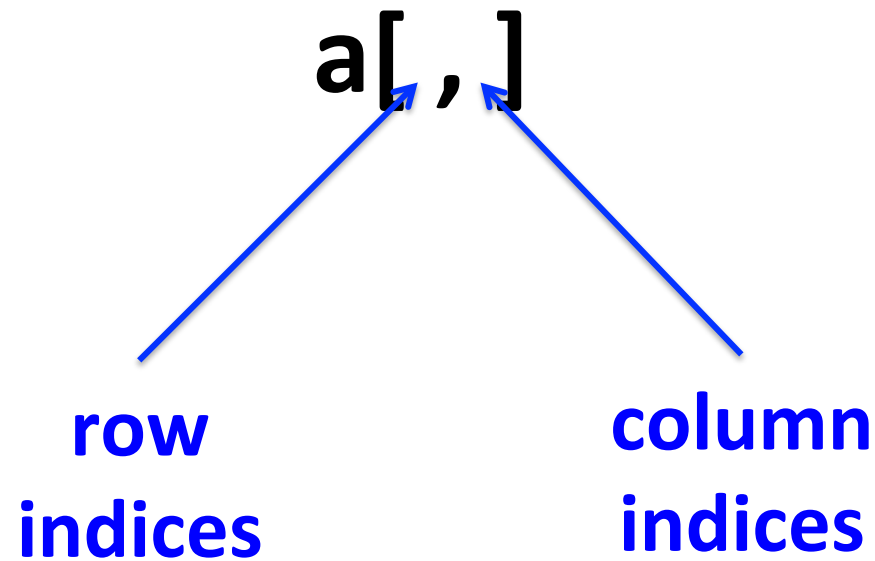
- A matrix is a vector of **2 dimensions**
- All columns in a matrix must have :
  - the same **type** (numeric, character, logical)
  - the same **length**

```
a <- matrix(c(1, 0, 34, 5, 13, 44, 12, 4, 3, 8, 6, 9, 22, 7, 76),  
           nrow=5,  
           ncol=3)
```



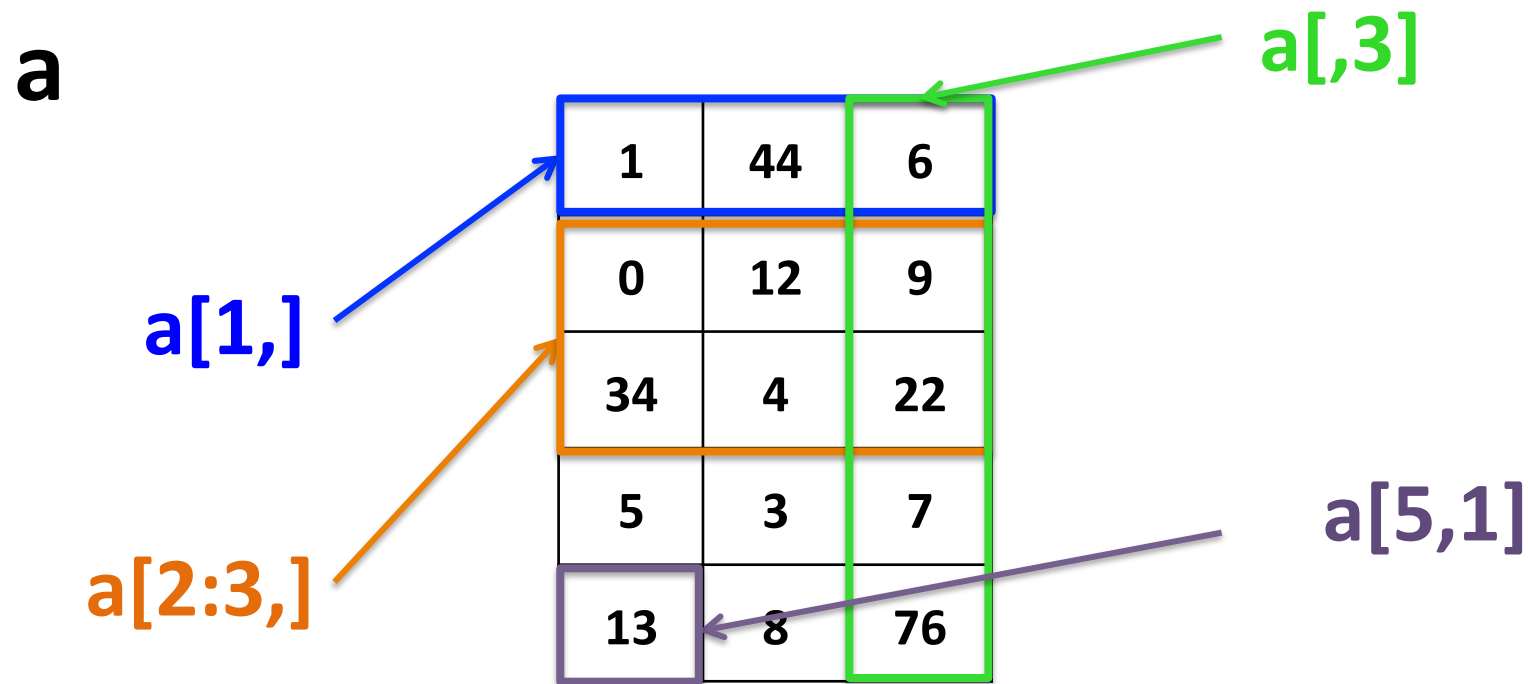
# Matrices

- Fetch rows, columns or single elements of a matrix using subscript:



# Matrices

- Fetch rows, columns or single elements of a matrix using subscript:



# Data frames

- Structures of 2 dimensions.
- More general than matrices.
- Different columns can have **different types** but must have the **same length**.

```
a <- data.frame(c(1, 3, 34, 5, 13),  
               c("etc", "ok", "yes", "no", "well"),  
               c(TRUE, TRUE, FALSE, FALSE, TRUE))
```

# Data frames

- Fetch rows, columns or single elements of a data frame using subscript:

a

The diagram shows a data frame 'a' with 5 rows and 3 columns. The cells contain the following values:

1	etc	TRUE
3	ok	TRUE
34	yes	FALSE
5	not	FALSE
13	well	TRUE

Annotations:

- A blue arrow points to the first row, labeled `a[1,]`.
- An orange arrow points to the second and third rows, labeled `a[2:3,]`.
- A green arrow points to the third column, labeled `a[,3]`.
- A purple arrow points to the first element of the fifth row, labeled `a[5,1]`.

# Matrices and data frames dimension names

```
a <- matrix(c(1, 0, 34, 5, 13, 44),  
           nrow=3,  
           ncol=2,  
           dimnames=list(c("row1", "row2", "row3"), c("col1", "col2")))
```

```
b <- data.frame(c(1, 0, 34),  
              c(5, 13),  
              row.names=c("row1", "row2", "row3"),  
              col.names=, c("col1", "col2"))
```

# Matrices and data frames

## dimension names

- Changing column and / or row names:  
`colnames(x) <- c("a", "b", "c")`  
`rownames(x) <- 1:5`
- Changing column and row names at once:  
`dimnames(x) <- list(c("a", "b", "c"), 1:5)`

# Matrices and data frames dimension names

## Matrix

`a["col1"]` ✓

`a["col1"]` ✗

`a$col1` ✗

`a["row1",]` ✓

`a["row1"]` ✗

`a["row1","col1"]` ✓

	col1	col2
row1	1	5
row2	0	13
row3	34	44

## Data frame

`b["col1"]` ✓

`b["col1"]` ✓

`b$col1` ✓

`b["row1",]` ✓

`b["row1"]` ✗

`b["row1","col1"]` ✓

# Matrices and data frames

## Checks and conversions

- Checking if the object is a matrix...  
`is.matrix(mymatrix)`
- ... or a data frame:  
`is.data.frame(mydataframe)`
- Converting a matrix into a data frame...  
`as.matrix(mydataframe)`
- ... or vice versa:  
`as.data.frame(mymatrix)`



# Lists

- Linear structures.
- A component of a list can be **any data structure**  
(matrix, vector, data frame, another list)

# Lists

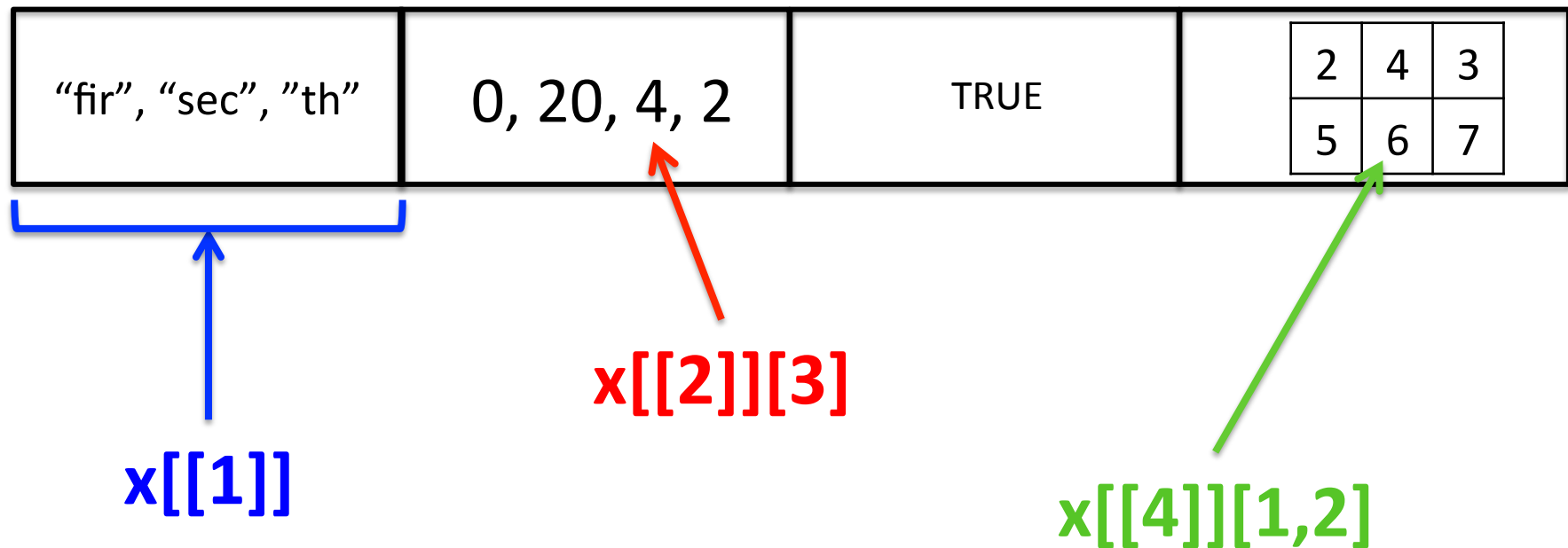
- Create a list:

```
x<- list()      # Empty list that can be filled later on
```

```
x<- list(c("fir", "sec", "th"),      # 1st element of the list  
         c(0, 20, 4, 2),             # 2d element of the list  
         TRUE,                       # 3d element of the list  
         matrix(c(2, 5, 4, 6, 3, 7), # 4th element of the list  
               nrow=2, ncol=3))
```

# Lists

- Accessing elements of a list x:



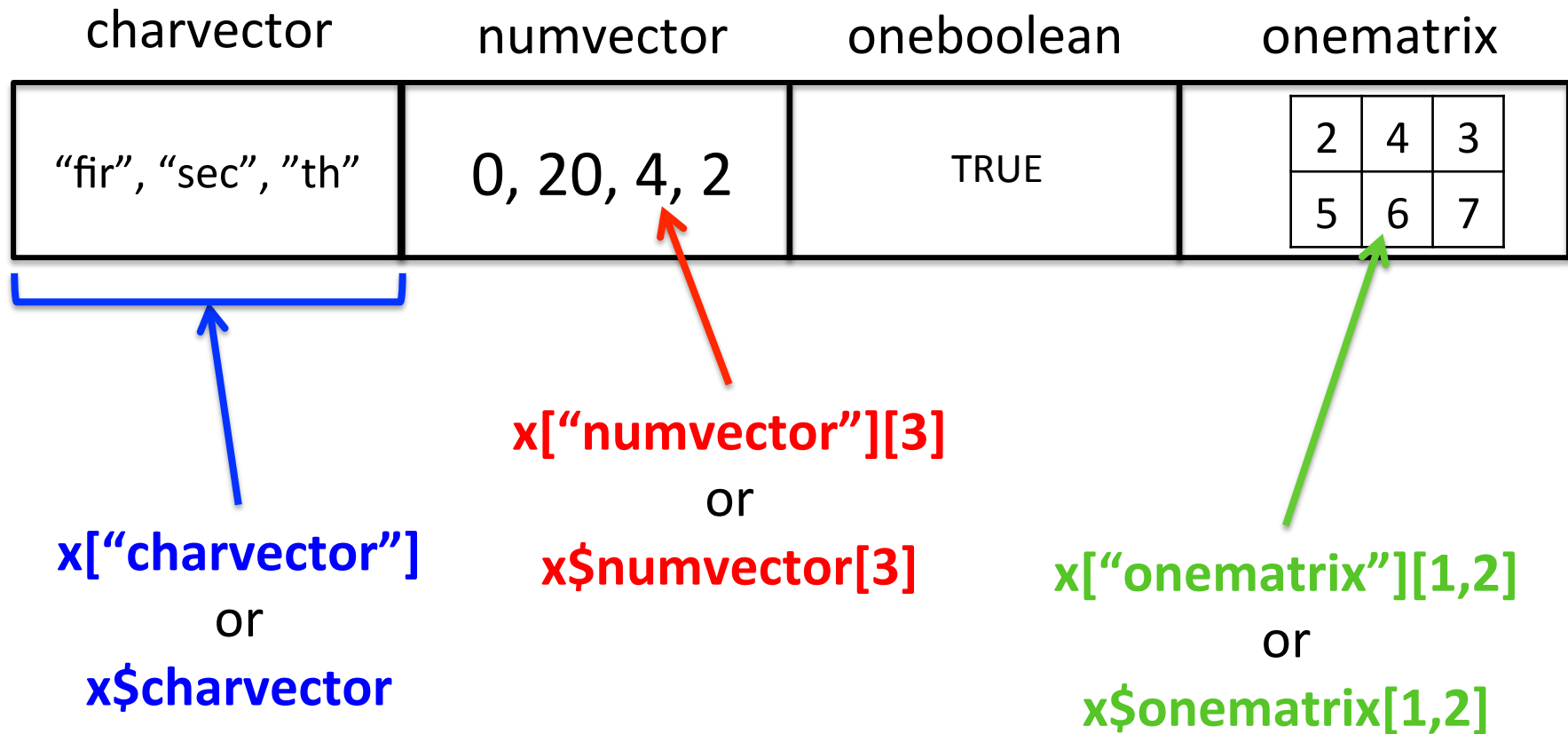
## Naming elements of a list

```
x <- list(charvector=c("fir", "sec", "th"),  
         numvector=c(0, 20, 4, 2),  
         oneboolean=TRUE,  
         onematrix=matrix(c(2, 5, 4, 6, 3, 7),  
                           nrow=2, ncol=3))
```

or

```
names(x) <- c("charvector", "numvector",  
              "oneboolean", "onematrix")
```

# Accessing elements of list x per name



# Length and dimension of lists

- Lists have a length but only **one dimension**

**length(x)**

**4**

- Elements within a list can have lengths and dimensions, depending on their data structure.

"fir", "sec", "th"	0, 20, 4, 2	TRUE	<table border="1"><tr><td>2</td><td>4</td><td>3</td></tr><tr><td>5</td><td>6</td><td>7</td></tr></table>	2	4	3	5	6	7
2	4	3							
5	6	7							

**length(x[[2]])**

**4**

**dim(x[[4]])**

**2 3**

# Checking and converting data types and structures

<b>Checking</b>	<b>Converting</b>
<code>is.matrix(x)</code>	<code>as.matrix()</code>
<code>is.data.frame(x)</code>	<code>as.data.frame()</code>
<code>is.vector()</code>	<code>as.vector()</code>
<code>is.list()</code>	<code>as.list()</code>
<code>is.factor()</code>	<code>as.factor()</code>
<code>is.character()</code>	<code>as.character()</code>
<code>is.numeric()</code>	<code>as.numeric()</code>

# Library / packages



# Library/packages

- **Packages** are collections of R functions, data, and compiled code in a well-defined format.
- The directory where packages are stored is called the **library**.

*Definitions from <http://www.statmethods.net/interface/packages.html>*

# Standard packages

- About 25 standard packages are supplied with R by default (example: base, stats, graphics).
- On May 4th 2016, **8340 packages** were available!
- Anyone can contribute to the R package repository

# Installing and loading packages

- Packages can be installed from:
  - the interactive session:  
`install.packages("ggplot2")`  
`install.packages("ggplot2", repos=  
http://cran.r-project.org/web/packages/)`
  - the terminal:  
`R CMD INSTALL ggplot2_version.tar.gz`
- And loaded:  
`library("ggplot2")`

# Listing packages

- Installed packages: **library()**

acepack	ace() and avas() for selecting regression transformations
affy	Methods for Affymetrix Oligonucleotide Arrays
affyio	Tools for parsing Affymetrix data files
AnnotationDbi	Annotation Database Interface
Biobase	Biobase: Base functions for Bioconductor
BiocGenerics	S4 generic functions for Bioconductor
...	

- Loaded packages: **search()**

```
[1] ".GlobalEnv" "package:stats" "package:graphics"  
[4] "package:grDevices" "package:utils" "package:datasets"  
[7] "package:methods" "Autoloads" "package:base"
```

## Listing functions from packages

- **ls("package:yourpackage")**

- **ls("package:VennDiagram")**

```
[1] "add.title"          "adjust.venn"  
[3] "calculate.overlap" "decide.special.case"  
[5] "draw.pairwise.venn" "draw.quad.venn"  
[7] "draw.quintuple.venn" "draw.single.venn"  
...."
```

# Accessing package's functions

- Usually, calling the function by its name is enough:  
**add.title()**
- If 2 packages have a function with the same name, make sure you are using the right one:  
**VennDiagram::add.title ()**

# Basic commands

## Getting and changing working directory

- **getwd()**
  - Returns current working directory
- **setwd("/home/mydir/")**
  - Changes current working directory to "/home/mydir"



# Information about objects

- **summary(x)**
  - for numerical data: min, max, median etc.
  - for character data: count of items occurrences

col1	col2	col3
1	ok	TRUE
3	yes	FALSE
5	no	FALSE
6	maybe	FALSE

```
col1      col2      col3
Min.   :1.00   maybe:1   Mode :logical
1st Qu.:2.50   no  :1    FALSE:3
Median :4.00   ok  :1    TRUE :1
Mean   :3.75   yes :1    NA's :0
3rd Qu.:5.25
Max.   :6.00
```

# Information about objects

- **str(x):**

→ Internal structure of an R object

col1	col2	col3
1	ok	TRUE
3	yes	FALSE
5	no	FALSE
6	maybe	FALSE

'data.frame': 4 obs. of 3 variables:

\$ col1: num 1 3 5 6

\$ col2: Factor w/ 4 levels "maybe","no","ok",...: 3 4 2 1

\$ col3: logi TRUE FALSE FALSE FALSE

# Size/dimensions of objects

number of elements  
in vector, factor or list

**length(x)**

---

dimensions  
of matrix or data frame

**dim(x)**

---

number of rows  
of matrix or data frame

**nrow(x)**

---

number of columns  
of matrix or data frame

**ncol(x)**

# Elementary arithmetic operators

addition                    **+**

---

subtraction                    **-**

---

division                    **/**

---

multiplication                    **\***

---

exponentiation                    **^** or **\*\***

# Logical operators

inferior	<
inferior or equal	<=
superior	>
superior or equal	>=
equality	==
inequality	!=
intersection (“and”)	&
union (“or”)	

# Obtaining summary statistics

average/mean	<b>mean(x)</b>
median	<b>median(x)</b>
minimum	<b>min(x)</b>
maximum	<b>max(x)</b>
variance	<b>var(x)</b>
correlation	<b>cor(x)</b>

# Some common arithmetic functions

natural logarithm	<b>log(x)</b>
exponential function $e^x$	<b>exp(x)</b>
sine	<b>sin(x)</b>
cosine	<b>cos(x)</b>
tangent	<b>tan(x)</b>
absolute value	<b>abs(x)</b>
square root	<b>sqrt(x)</b>

## Objects stored in the global environment

- Listing:  
**ls()** or **objects()**
- Removing one object from environment:  
**rm(x)**
- Removing several objects:  
**rm(x, y)**
- Removing all object from environment:  
**rm(list=ls())**



# Reading and writing files

Reading file into object

---

```
a <- read.table("file.txt")
```

---

Writing object to file

---

```
write.table(a, "file.txt")
```

---

## Saving objects or session

- Save objects x and y into “myobjects.RData” file
  - **save(x, y, file=“myobjects.RData”)**
- Load objects x and y into current directory:
  - **load(“myobjects.RData”)**
- Save the current workspace (all objects):
  - **save.image(file=“.RData”)**

# Command history

- Last 25 commands:
  - **history()**
- All previous commands
  - **history(max.show=Inf)**
- Save command history:
  - **savehistory()**
- Load command history:
  - **loadhistory()**

## Basic commands

# Information about the current session

## sessionInfo()

```
R version 3.2.4 (2016-03-10)  
Platform: x86_64-apple-darwin13.4.0 (64-bit)  
Running under: OS X 10.9.5 (Mavericks)
```

R version

Platform and  
OS version

```
locale:
```

```
[1] C/UTF-8/C/C/C/C
```

```
attached base packages:
```

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

```
other attached packages:
```

```
[1] ggplot2_2.1.0
```

Packages attached

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

```
[1] colorspace_1.2-6 scales_0.4.0      plyr_1.8.3      gtable_0.2.0  
[5] Rcpp_0.12.4      grid_3.2.4      munsell_0.4.3
```

# Functions in R

# Knowing the source code of a function

- Name of the function without ()

For example:

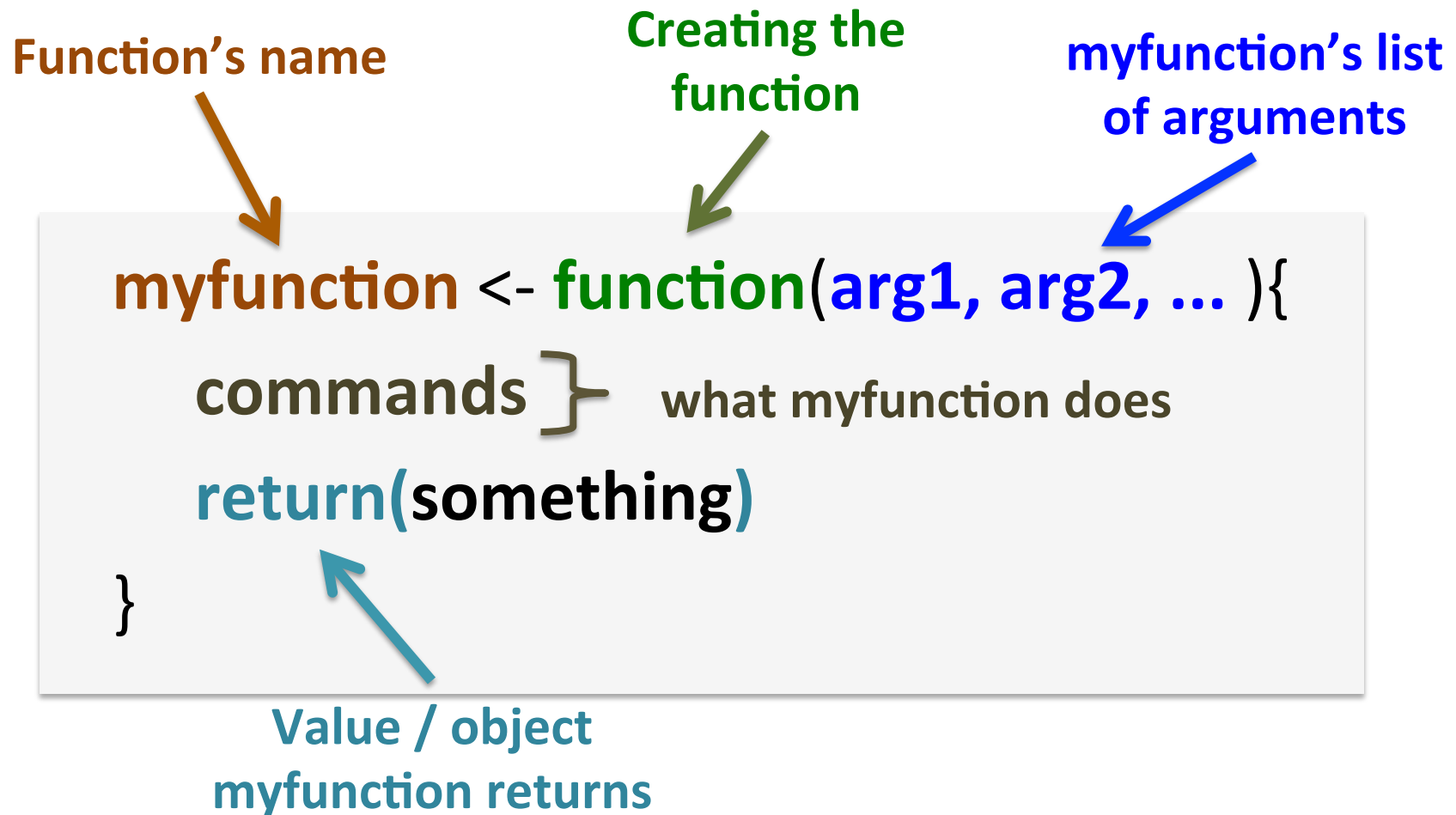
### **sort**

```
function (x, decreasing = FALSE, ...)  
{  
  if (!is.logical(decreasing) || length(decreasing) != 1L)  
    stop("'decreasing' must be a length-1 logical vector.\nDid  
you intend to set 'partial'?")  
  UseMethod("sort")  
}
```

# User-written functions

- Functions are pieces of code written **to carry out (a) specified task(s)** and be able to repeat it(them) easily.
- R allows you to create your own functions.

# Functions' structure





# Functions

**Objects in a function are local to that function!**

from  
<https://www.datacamp.com/community/tutorials/functions-in-r-a-tutorial>  
06/05/16

R environment

Script environment

```
# some other code

....

# define a simple function
myFirstFun<-function(n)

{
  n*n # compute the square of n
}

# define a value
k<-10

# call the function with that value
m<-myFirstFun(k)

...
```

Function environment

Global Environment

Packages

...

# Objects in functions

```
myfunction <- function(arg1){  
  a <- arg1  
  return(a+1)  
}
```

```
> myfunction(10)  
[1] 11  
> a  
Error: object 'a' not found
```

```
> a <- 12  
> myfunction(10)  
[1] 11  
> a  
[1] 12
```

## Example of a function

### Function “sometstats”:

**takes as an argument:** one numeric **vector**.

**computes:** **median** and **mean** of that vector.

**returns:** a **vector** containing these 2 values.

# Example of a function

```
somestats <- function(vector_input){  
  my_mean <- mean(vector_input)  
  my_median <- median(vector_input)  
  return(c(my_mean, my_median))  
}
```

```
x <- c(0, 2, 1, 6.3, 2.2, 10, 8, 5.4)
```

```
somestats(x)  
4.3625 3.8000
```

# Saving and sourcing functions

## myfunctions.R

```
somestats <- function(vector_input){  
  my_mean <- mean(vector_input)  
  my_median <- median(vector_input)  
  vector_output <- c(my_mean, my_median)  
  return(c(vector_output))  
}
```

## source("myfunctions.R")

→ Content of **myfunctions.R** loaded in the user workspace (global environment)

# R Studio

# What is R studio?

- **Free and open source** IDE (Integrated Development Environment) for R
- Available for Windows, Mac OS and Linux
- Written in C++
- First beta version available in 2009
- User friendly environment



# R Studio

## Screen: 4 windows

The screenshot shows the R Studio interface with four windows highlighted by boxes:

- 2. R script:** The main editor window containing R code for getting the working directory, loading packages (ggplot2 and limma), creating a matrix 'a', and creating a list 'b'.
- 3. Environment and history:** The Environment pane showing the current environment with variables 'a' (Named num [1:12]) and 'b' (List of 2).
- 1. Console:** The Console window showing the execution of the R script, including the output of 'getwd()' and 'list.files()'. The output of 'list.files()' is a 3x4 grid of file names.
- 4. Files, plots, packages, help, viewer:** The Files pane showing the file explorer view of the current directory, listing various folders like .R, .Rhistory, Applications, Desktop, Documents, Library, Movies, Music, Pictures, Projects, and Public.

```
1 # get working directory
2 getwd()
3
4 # load packages
5 library("ggplot2")$
6 library("limma")
7
8 # create matrix
9 a <- c(1:10, nrow=2, ncol=5)
10
11 # create list
12 b <- list(a, c("first", "second", "third"))
```

```
> # get working directory
> getwd()
[1] "/Users/sbonnin"
>
> # load packages
> library("ggplot2")
> library("limma")
>
> # create matrix
> a <- c(1:10, nrow=2, ncol=5)
> b <- list(a, c("first", "second", "third"))
>
>
> list.files()
[1] "Applications" "Desktop" "Documents" "Downloads"
[5] "Dropbox" "Dropbox (CRG)" "Dropbox (Personal)" "DropboxCRG"
[9] "Library" "Movies" "Music" "Pictures"
[13] "Projects" "Public"
>
```

Name	Size	Modified
.R		
.Rhistory	815 B	Apr 13, 2016, 6:14 PM
Applications		
Desktop		
Documents		
Library		
Movies		
Music		
Pictures		
Projects		
Public		





## 2. R script

The screenshot displays the R Studio interface with a script editor containing the following code:

```
1 # get working directory
2 getwd()
3
4 # load packages
5 library("ggplot2")
6 library("limma")
7
8 # create matrix
9 a <- c(1:10, nrow=2, ncol=5)
10
11 # create list
12 b <- list(a, c("first", "second", "third"))$
```

Annotations in the image:

- A box labeled "Can be run by line / block" has an arrow pointing to the "Run" button in the toolbar.
- A box labeled "Can be run entirely" has an arrow pointing to the "Source" button in the toolbar.
- A box labeled "Saved script" is positioned below the code editor.

The status bar at the bottom shows the time 12:45, the current environment (Top Level), and the file type (R Script).

### 3. Environment and history

The screenshot shows the R Studio interface with the Environment and History panes. The Environment pane is active, showing the Global Environment with a table of values. The History pane is also visible, showing a list of packages. Annotations include a box around the Global Environment, a box around the Values table, a box around the package list, and a box listing actions like saving and loading workspaces.

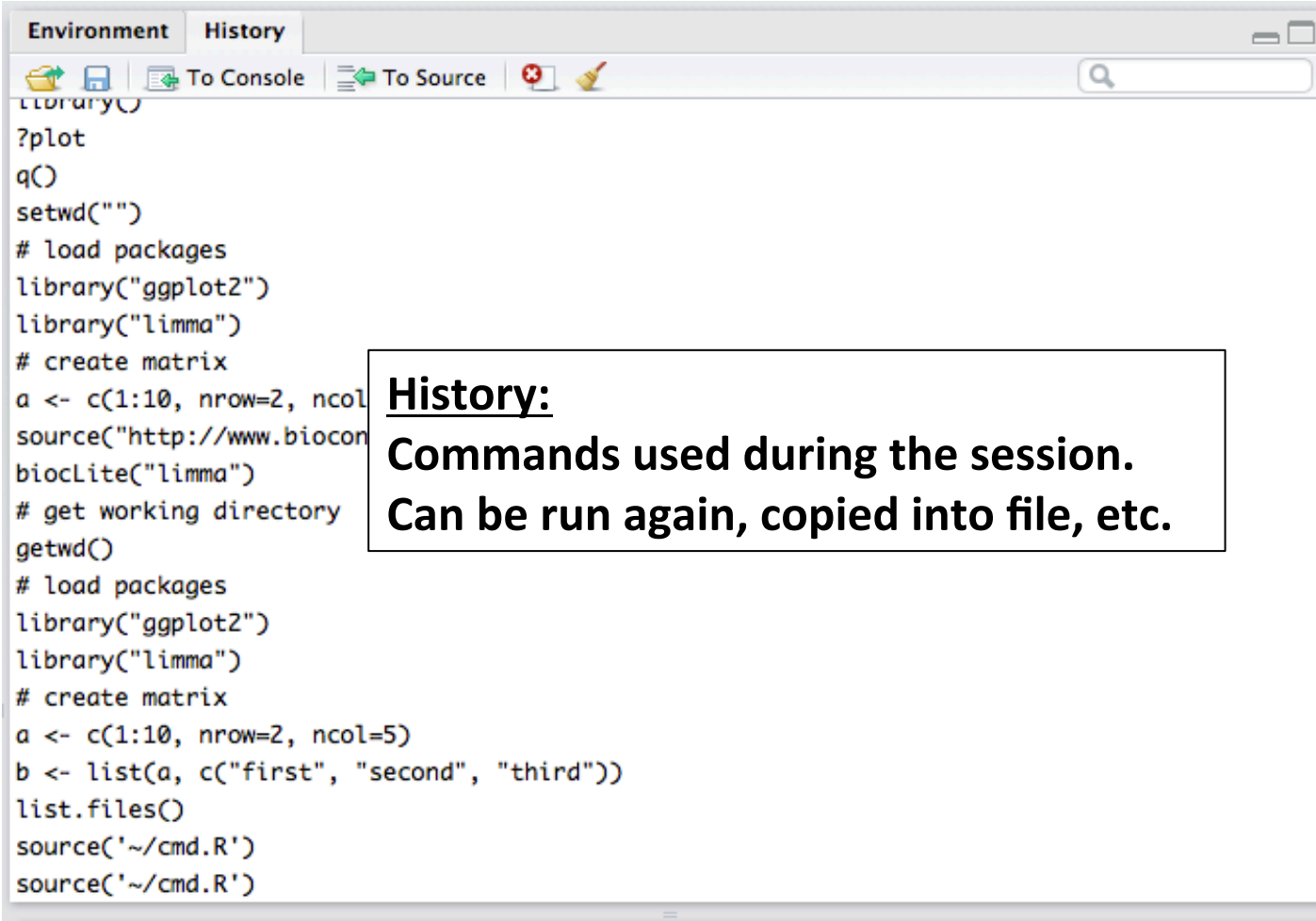
Values	
a	Named num [1:12] 1 2 3 4 5 6 7 8 9 10 ...
b	List of 2

**Global environment:**  
all objects present in the session

**List functions available per package**

**You can also:**  
Save the current workspace  
Load a saved workspace  
Import datasets

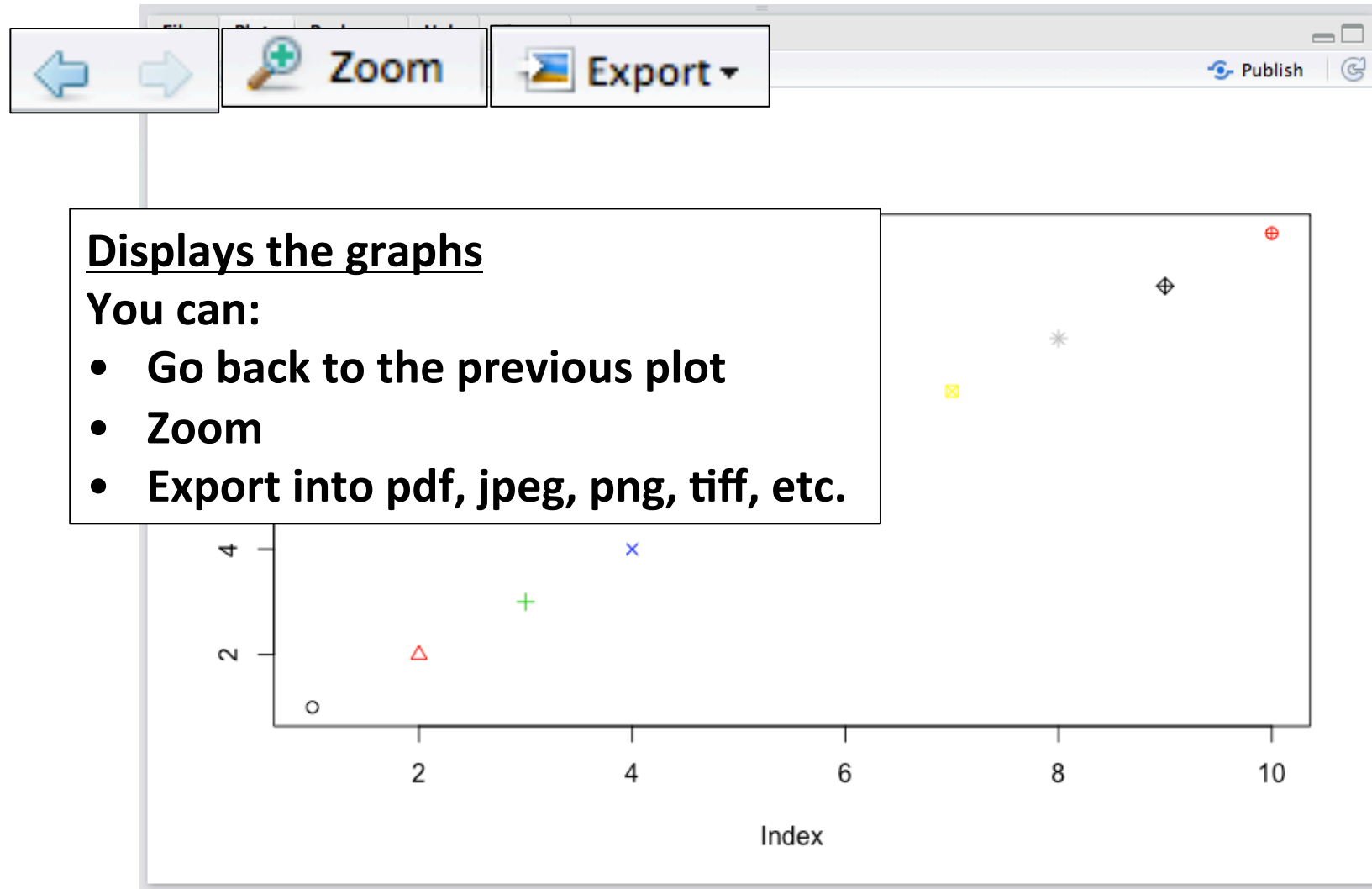
## 3. Environment and history



The screenshot shows the 'History' window in R Studio. The window title is 'Environment History'. It contains a list of R commands that have been executed during the session. The commands are: `?plot`, `q()`, `setwd("")`, `# load packages`, `library("ggplot2")`, `library("limma")`, `# create matrix`, `a <- c(1:10, nrow=2, ncol=5)`, `source("http://www.bioc.org/biocLite.R")`, `biocLite("limma")`, `# get working directory`, `getwd()`, `# load packages`, `library("ggplot2")`, `library("limma")`, `# create matrix`, `a <- c(1:10, nrow=2, ncol=5)`, `b <- list(a, c("first", "second", "third"))`, `list.files()`, `source('~/.cmd.R')`, and `source('~/.cmd.R')`. A text box is overlaid on the right side of the window, containing the text: **History:**  
**Commands used during the session.**  
**Can be run again, copied into file, etc.**

```
Environment History
?plot
q()
setwd("")
# load packages
library("ggplot2")
library("limma")
# create matrix
a <- c(1:10, nrow=2, ncol=5)
source("http://www.bioc.org/biocLite.R")
biocLite("limma")
# get working directory
getwd()
# load packages
library("ggplot2")
library("limma")
# create matrix
a <- c(1:10, nrow=2, ncol=5)
b <- list(a, c("first", "second", "third"))
list.files()
source('~/.cmd.R')
source('~/.cmd.R')
```

## 4. Plot area



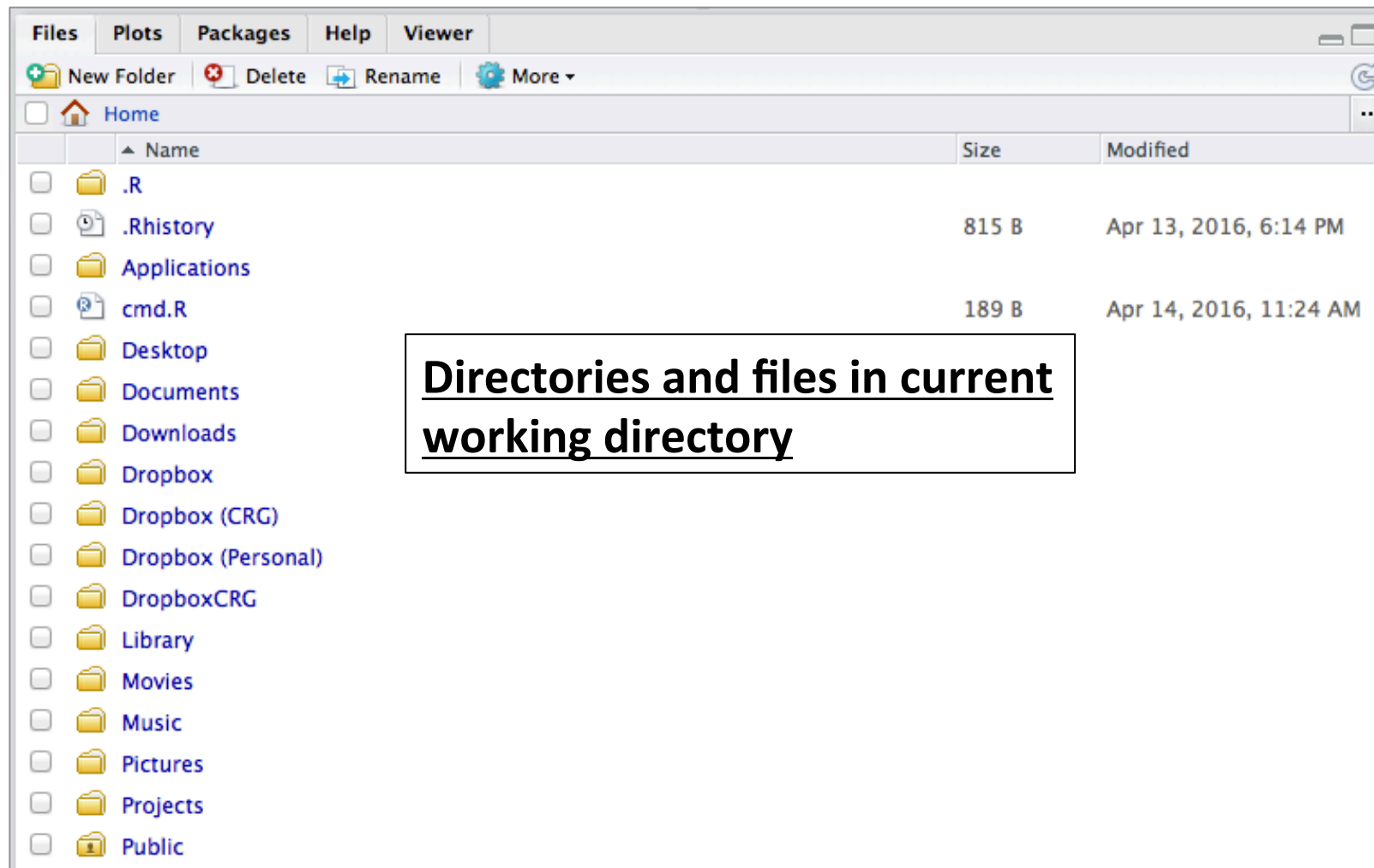
The image shows a screenshot of the R Studio interface. At the top, there is a toolbar with three buttons: a left arrow, a right arrow, and a 'Zoom' button with a magnifying glass icon. To the right of the 'Zoom' button is an 'Export' button with a document icon and a dropdown arrow. Further right, there are 'Publish' and 'Refresh' icons. Below the toolbar is a plot area. A text box on the left side of the plot area contains the following text:

**Displays the graphs**  
**You can:**

- **Go back to the previous plot**
- **Zoom**
- **Export into pdf, jpeg, png, tiff, etc.**

The plot area shows a scatter plot with the x-axis labeled 'Index' and the y-axis labeled '4' and '2'. The x-axis has tick marks at 2, 4, 6, 8, and 10. The y-axis has tick marks at 2 and 4. The plot contains several data points: a small circle at (1, 1), a red triangle at (2, 2), a green plus sign at (3, 3), a blue cross at (4, 4), a yellow square with an 'x' at (7, 3), a grey asterisk at (8, 4), a grey diamond with a cross at (9, 5), and a red circle with a cross at (10, 6).

## 4. Files



## 4. Packages

**You can:**

- Install new packages
- Update packages to newest versions

**List of packages:**

- Available
- Loaded in current session(ticked)

Package Name	Description	Version	Status
<input type="checkbox"/> base64enc	Tools for base64 encoding		
<input checked="" type="checkbox"/> BiocInstaller	Install/Update Bioconductor and CRAN Packages	1.20.1	⊗
<input type="checkbox"/> bitops	Bitwise Operations	1.0-6	⊗
<input type="checkbox"/> boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-18	⊗
<input type="checkbox"/> caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1	⊗
<input type="checkbox"/> class	Functions for Classification	7.3-14	⊗
<input type="checkbox"/> cluster	"Finding Clusters in Data": Cluster Analysis Extended, Reorganized, and	2.0.3	⊗
<input type="checkbox"/> codetools		0.2-14	⊗
<input type="checkbox"/> colorspace		1.2-6	⊗
<input type="checkbox"/> compiler		3.2.4	⊗
<input checked="" type="checkbox"/> datasets		3.2.4	⊗
<input type="checkbox"/> dichromat		2.0-0	⊗
<input type="checkbox"/> digest	Create Compact Hash Digests of R Objects	0.6.9	⊗
<input type="checkbox"/> evaluate	Parsing and Evaluation Tools that Provide More Details than the Default	0.8.3	⊗
<input type="checkbox"/> foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...	0.8-66	⊗
<input type="checkbox"/> formatR	Format R Code Automatically	1.3	⊗
<input type="checkbox"/> gdata	Various R Programming Tools for Data Manipulation	2.17.0	⊗
<input checked="" type="checkbox"/> ggplot2	An Implementation of the Grammar of Graphics	2.1.0	⊗
<input type="checkbox"/> gplots	Various R Programming Tools for Plotting Data	3.0.1	⊗
<input checked="" type="checkbox"/> graphics	The R Graphics Package	3.2.4	⊗

## 4. Help

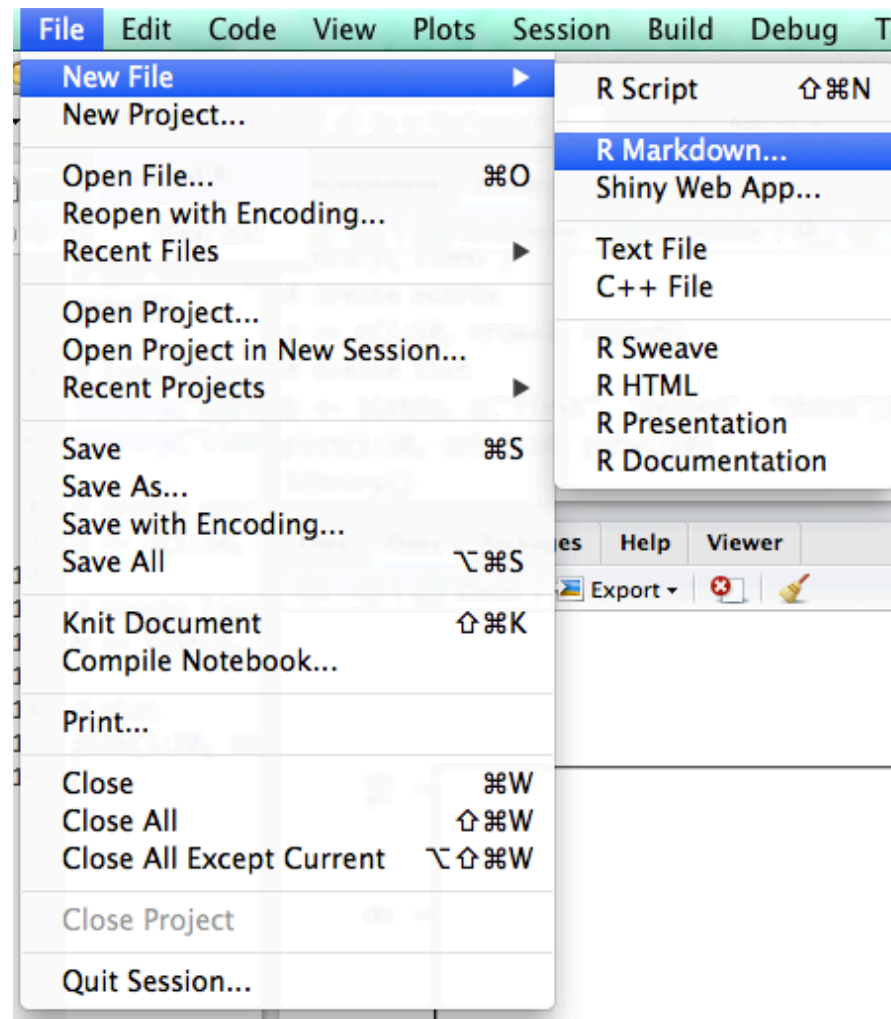
The screenshot shows the R Studio Help interface. The top navigation bar includes 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. Below this is a search bar with 'Home' and 'Find in Topic'. The main content area is split into two panes. The left pane shows 'R Resources' with links to 'Learning R Online', 'CRAN Task Views', 'R on StackOverflow', and 'Getting Help with R'. Below this is a 'Manuals' section with links to 'An Introduction to R', 'Writing R Extensions', and 'R Data Import/Export'. The right pane displays the help page for 'plot {graphics}', titled 'Generic X-Y Plotting'. It includes a 'Description' section, a 'Usage' section with the code `plot(x, y, ...)`, and an 'Arguments' section. The 'Arguments' section lists 'x' as the coordinates of points, 'y' as the y coordinates, and 'type' as the plot type, with a bullet point for 'n' for points. A summary box at the bottom left of the screenshot contains the text: 'Help tab: R and R Studio documentation Packages and functions help page displayed here when requested'. The bottom right of the screenshot shows a 'Line & Keywords' section with links to 'Resources', 'Thanks', and 'Technical papers'.



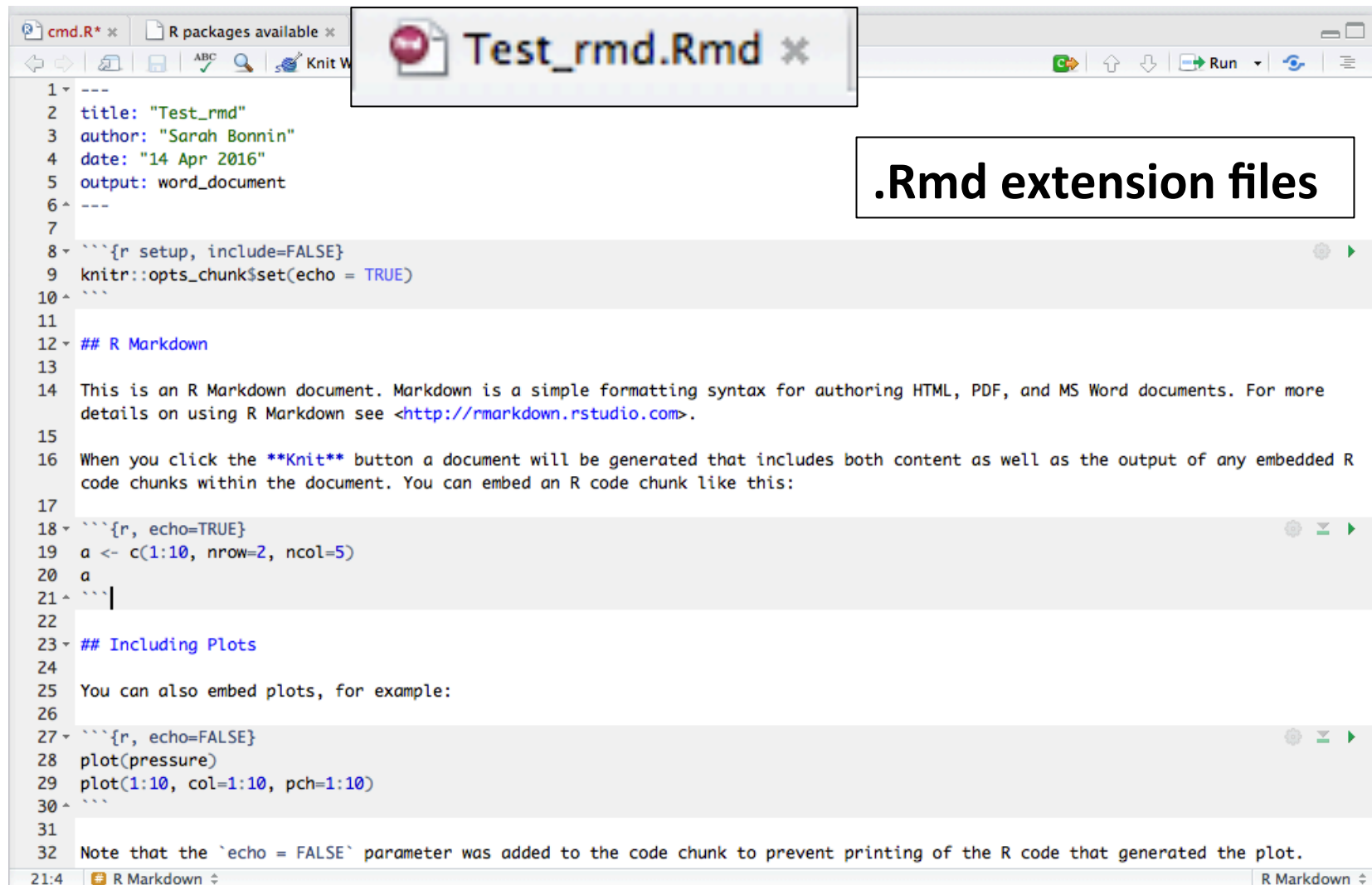
# RMD: R Markdown

- Format for writing **reproducible, dynamic reports** with R.
- Work directly inserted into formatted documents (HTML, PDF and Word)
- Easy to use!

# RMD: R Markdown



## RMD: R Markdown



The screenshot shows the R Studio interface with a file named `Test_rmd.Rmd` open. The code editor displays the following content:

```
1 ---
2 title: "Test_rmd"
3 author: "Sarah Bonnin"
4 date: "14 Apr 2016"
5 output: word_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more
15 details on using R Markdown see <http://rmarkdown.rstudio.com>.
16
17 When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R
18 code chunks within the document. You can embed an R code chunk like this:
19
20 ```{r, echo=TRUE}
21 a <- c(1:10, nrow=2, ncol=5)
22 a
23 ```
24
25 ## Including Plots
26
27 You can also embed plots, for example:
28
29 ```{r, echo=FALSE}
30 plot(pressure)
31 plot(1:10, col=1:10, pch=1:10)
32 ```
33
34 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

**.Rmd extension files**

## RMD: R Markdown

The screenshot displays the R Studio interface with a source editor on the left and a preview window on the right. The source editor shows R Markdown code with line numbers 1 through 32. The code includes a YAML header, a Knit button call, a title, author, and date, followed by a code chunk with `knitr::opts_chunk$set(echo = TRUE)`. Below this is a paragraph of text, another code chunk with `knitr::opts_chunk$set(echo = TRUE)` and `a <- c(1:10, nrow=2, ncol=5)`, a section titled "Including Plots", and a final code chunk with `knitr::opts_chunk$set(echo = FALSE)` and `plot(pressure)`. The preview window shows the rendered output, including the YAML header, the text paragraph, and the R code from the first chunk. The status bar at the bottom indicates the current line is 21:4 in the R Markdown file.

```
1 ---
2 title: "Test_rmd"
3 author: "Sarah Bonnin"
4 date: "14 Apr 2016"
5 output: word_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple
15 details on using R Markdown see <http://rmarkdown.rstudio.com>
16
17 When you click the Knit button a document will be generated
18 containing the contents of the document and the output of all
19 code chunks within the document. You can embed this
20 document in a web page.
21
22
23 ## Including Plots
24
25 You can also embed plots, for example:
26
27 ```{r, echo=FALSE}
28 plot(pressure)
29 plot(1:10, col=1:10, pch=1:10)
30 ```
31
32 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

## RMD: R Markdown

The screenshot shows the R Studio interface with a file named 'Test\_rmd.Rmd' open. The editor contains the following R Markdown code:

```
1 ---
2 title: "Test_rmd"
3 author: "Sarah Bonnin"
4 date: "14 Apr 2016"
5 output: word_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple
15 text format for writing documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>
16
17 When you click the Knit button a document will be generated containing all the code chunks within the document. You can embed an
18
19 ```{r, echo=TRUE}
20 a <- c(1:10, nrow=2, ncol=5)
21 a
22 ```
23
24 ## Including Plots
25
26 You can also embed plots, for example:
27
28 ```{r, echo=FALSE}
29 plot(pressure)
30 plot(1:10, col=1:10, pch=1:10)
31 ```
32
33 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

Two callout boxes provide additional information:

- The first callout points to the opening of a code chunk and contains the text: `## R Markdown`, ````{r}`, and ***YOUR CODE***.
- The second callout points to the opening of a code chunk with `echo=TRUE` and contains the text: ````{r, echo=TRUE}`, `a <- c(1:10, nrow=2, ncol=5)`, and `a`.

## RMD: R Markdown

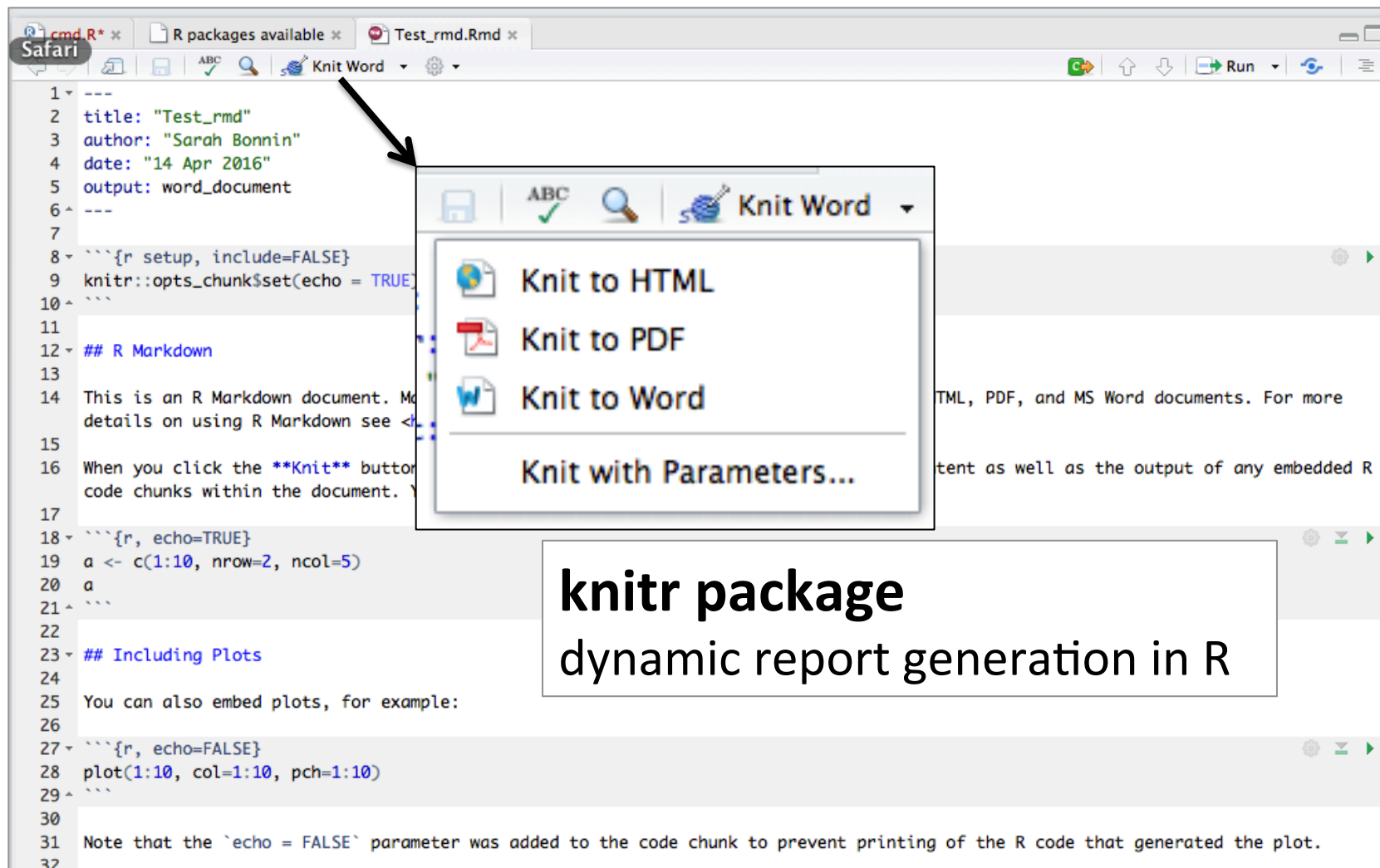
The screenshot shows the R Studio interface with a file named 'Test\_rmd.Rmd' open. The code editor displays the following R Markdown content:

```
1 ---
2 title: "Test_rmd"
3 author: "Sarah Bonnin"
4 date: "14 Apr 2016"
5 output: word_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting
15 details on using R Markdown see <http://rmarkdown.rstudio.com>.
16
17 When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R
18 code chunks within the document. You can embed an R code chunk like this:
19
20 ```{r, echo=TRUE}
21 a <- c(1:10, nrow=2, ncol=5)
22 a
23 ```
24
25 ## Including Plots
26
27 You can also embed plots, for example:
28
29 ```{r, echo=FALSE}
30 plot(pressure)
31 plot(1:10, col=1:10, pch=1:10)
32 ```
33
34 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

Annotations in the image include:

- A box on the right side of the code editor containing the text: **echo=TRUE eval=TRUE** code + output in report and **echo=FALSE eval=TRUE** only output in report.
- A box at the bottom right containing the text: **## Including Plots** and You can also embed plots, for example: followed by the code chunk ````{r, echo=FALSE} plot(1:10, col=1:10, pch=1:10) ````.
- Arrows pointing from these boxes to the corresponding code chunks in the editor.

## RMD: R Markdown



The screenshot shows the R Studio interface with an R Markdown document open. The document content is as follows:

```
1 ---
2 title: "Test_rmd"
3 author: "Sarah Bonnin"
4 date: "14 Apr 2016"
5 output: word_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. More details on using R Markdown see <http://rmarkdown.rstudio.com>.
15
16 When you click the Knit button in the top right of the document editor, R will generate a new document of the format specified in the output field of the YAML header. Supported formats include HTML, PDF, and MS Word documents. For more details on using R Markdown to generate reports, see the R Markdown website.
17
18 ```{r, echo=TRUE}
19 a <- c(1:10, nrow=2, ncol=5)
20 a
21 ```
22
23 ## Including Plots
24
25 You can also embed plots, for example:
26
27 ```{r, echo=FALSE}
28 plot(1:10, col=1:10, pch=1:10)
29 ```
30
31 Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.
32
```

The Knit menu is open, showing options: Knit to HTML, Knit to PDF, Knit to Word, and Knit with Parameters... An arrow points from the Knit button in the top right of the document editor to the menu.

**knitr package**  
dynamic report generation in R

# RMD: R Markdown

**Knit to HTML**  
→ **Creates a .html file**

Test\_rmd

Sarah Bonnin  
14 Apr 2016

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
a <- c(1:10, nrow=2, ncol=5)
a
```

```
##
##   1  2  3  4  5  6  7  8  9 10  nrow ncol
## [1] 0.4195168
```

## Including Plots

You can also embed plots, for example:

Index	Value
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10



# Useful resources

- Quick R: <http://www.statmethods.net/>
- R bloggers: <http://www.r-bloggers.com/>
- Cookbook for R: <http://www.cookbook-r.com/>
- R forge forum:  
[https://r-forge.r-project.org/forum/forum.php?forum\\_id=78](https://r-forge.r-project.org/forum/forum.php?forum_id=78)
- R help:  
<http://r.789695.n4.nabble.com/R-help-f789696.html>