Introduction to

BIST Biostatistics course

May 6th, 2016

Sarah Bonnin

CRG Bioinformatics Core Facility

BIST Barcelona Institute of Science and Technology

CRG Centre for Genomic Regulation

prbb 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/

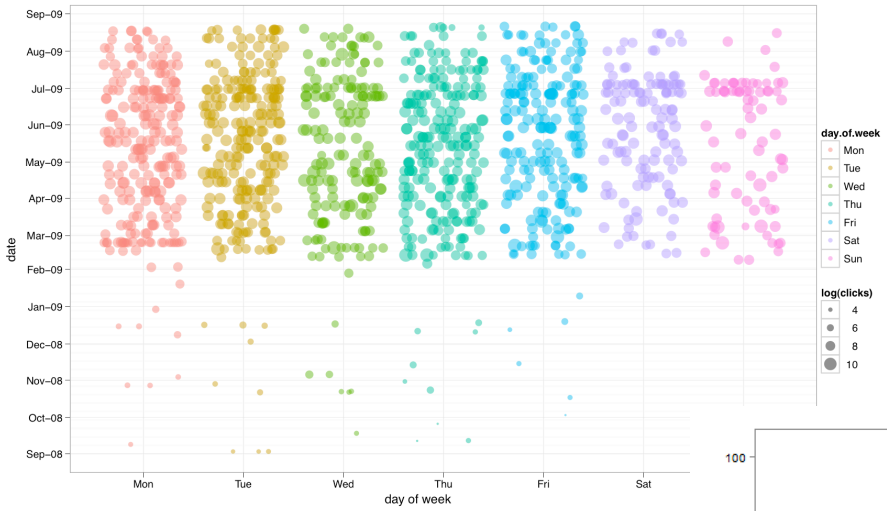
When to use R?

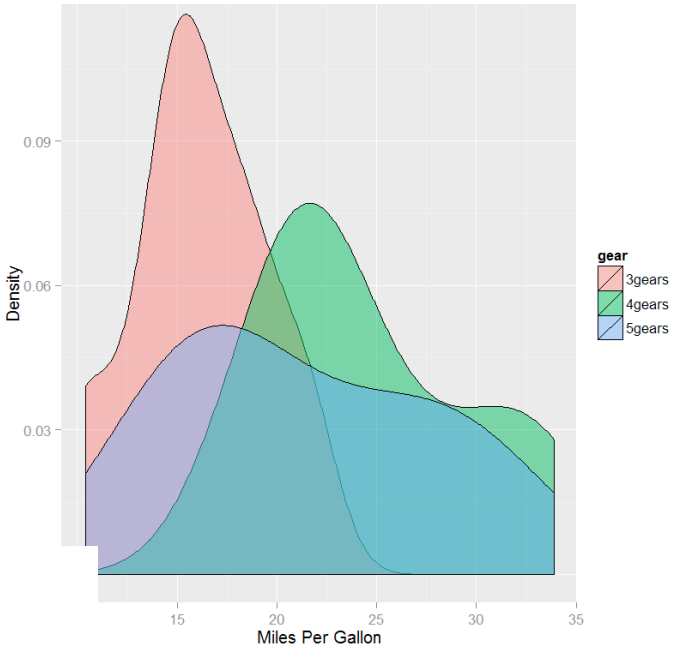
- Data analysis
- Statistical modeling
- Simulation
- Graphics

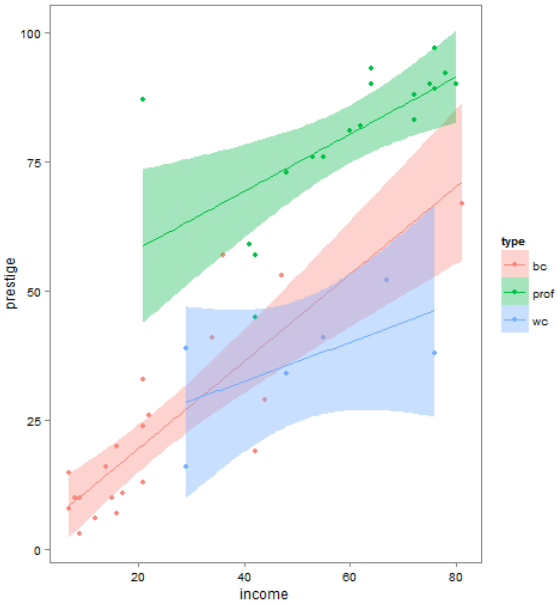
Why to use R?

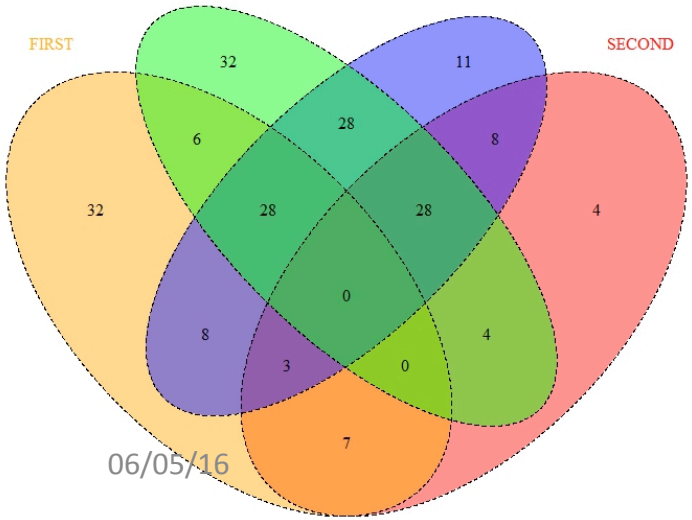
- Flexible
- Powerful
- Interactive
- **Very active community of developers and users!**

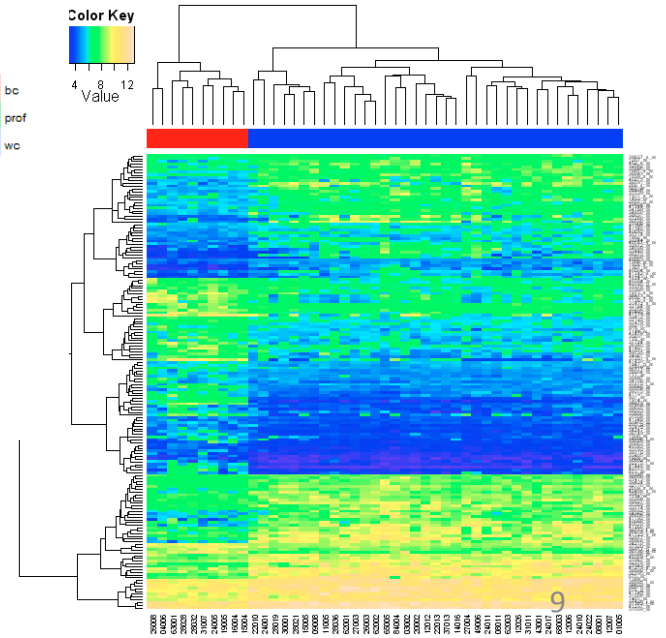
R: what, when, why?









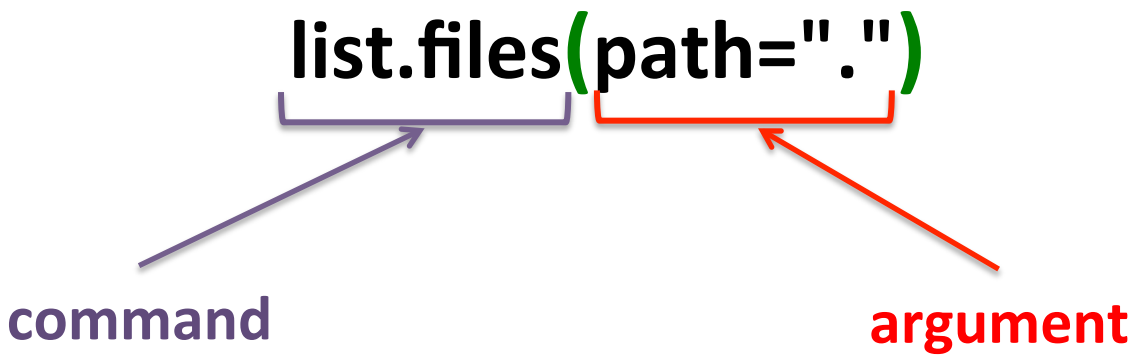


Getting started

R syntax

function(arguments)

example:

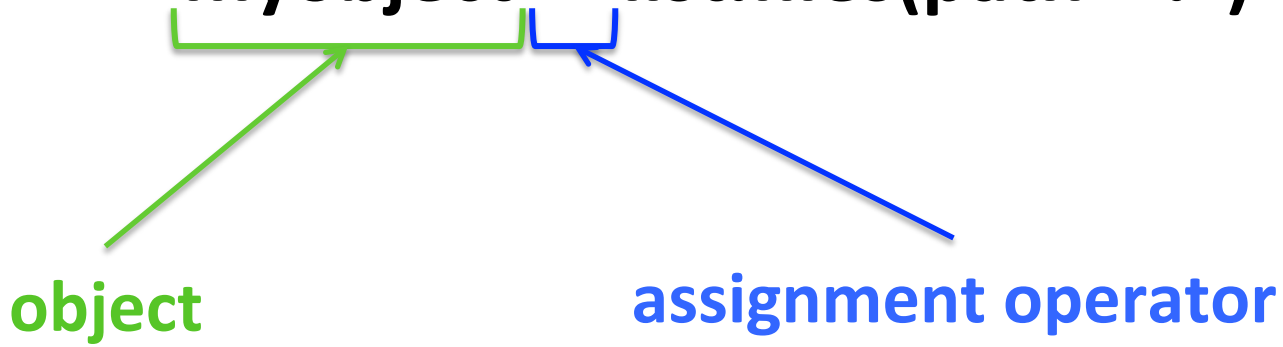


R syntax

a <- function(arguments)

example:

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



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!

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)

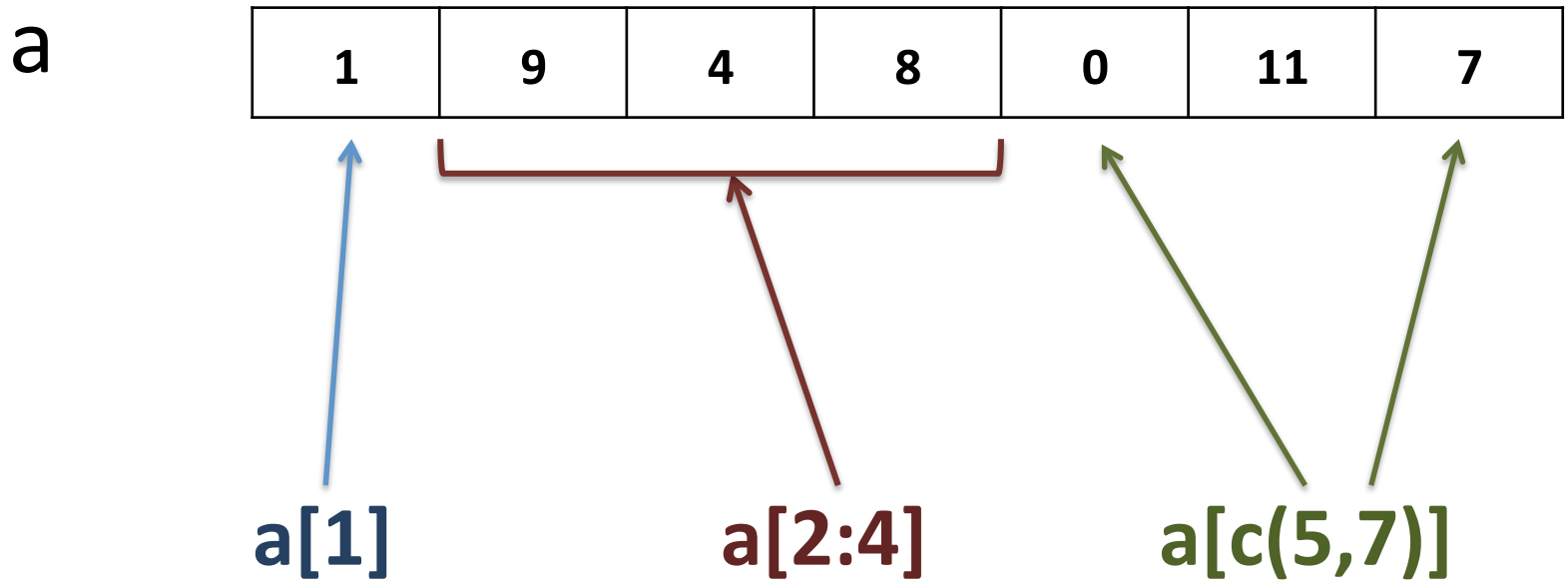
myvector <- **c**(0.5, 2, 10, 3, 8)

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.

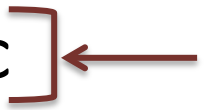
- Similar to vectors, but their values are limited to a **fixed set of possible values.**

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

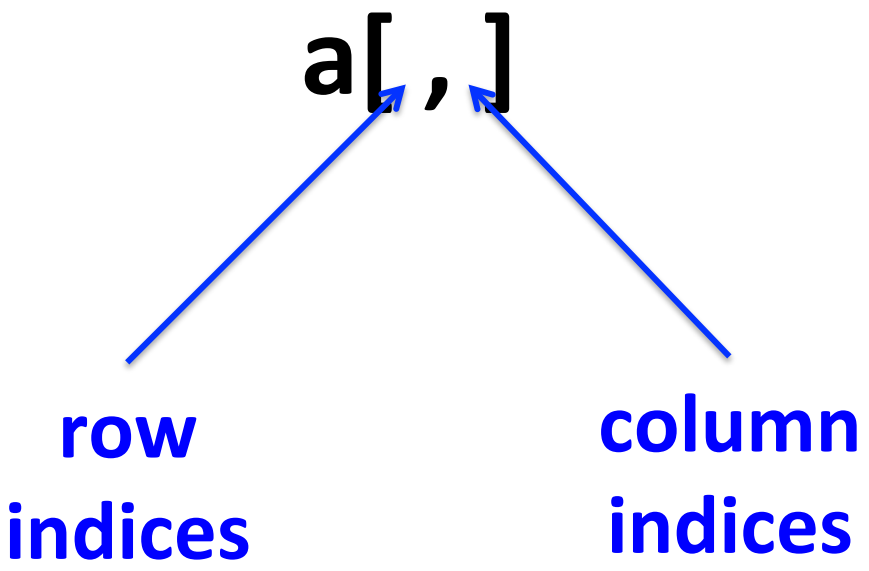
- A matrix is a vector of **2 dimensions**

- 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

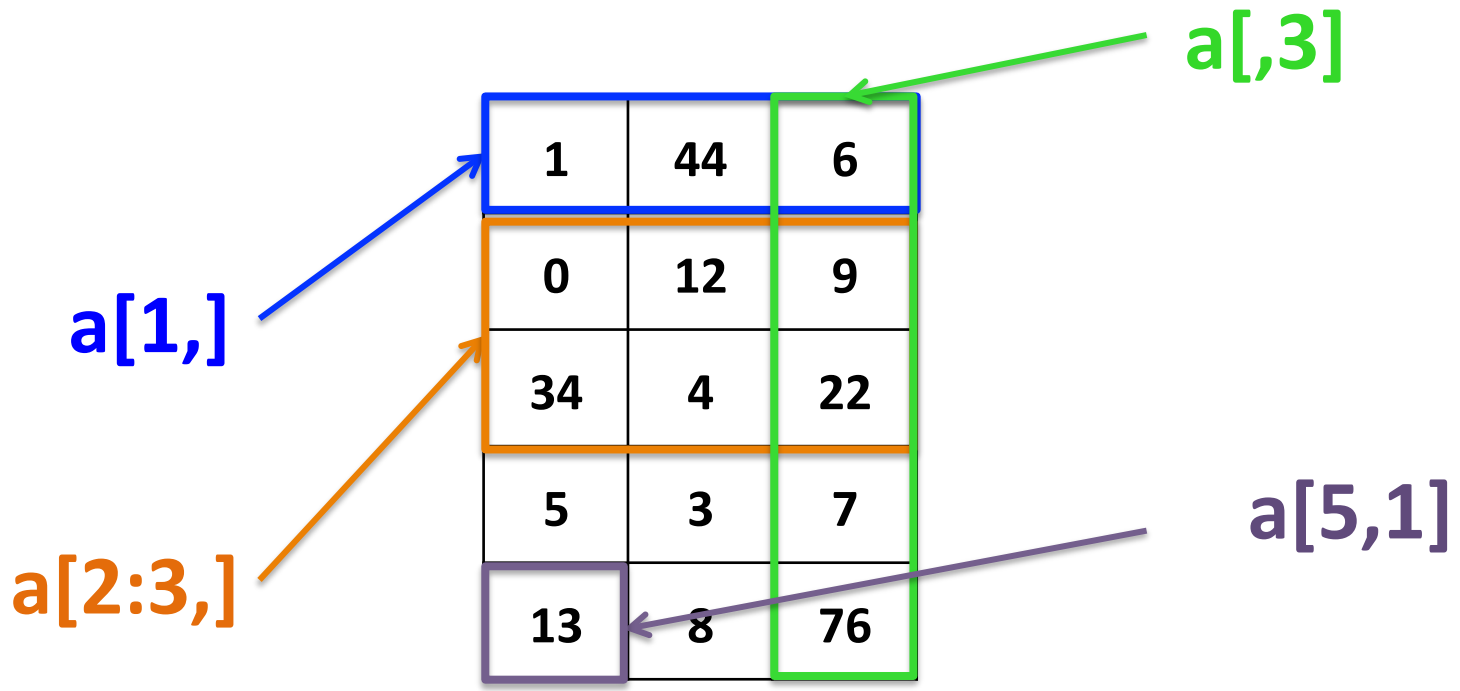
- Fetch rows, columns or single elements of a matrix using subscript:



Matrices

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

a



Data frames

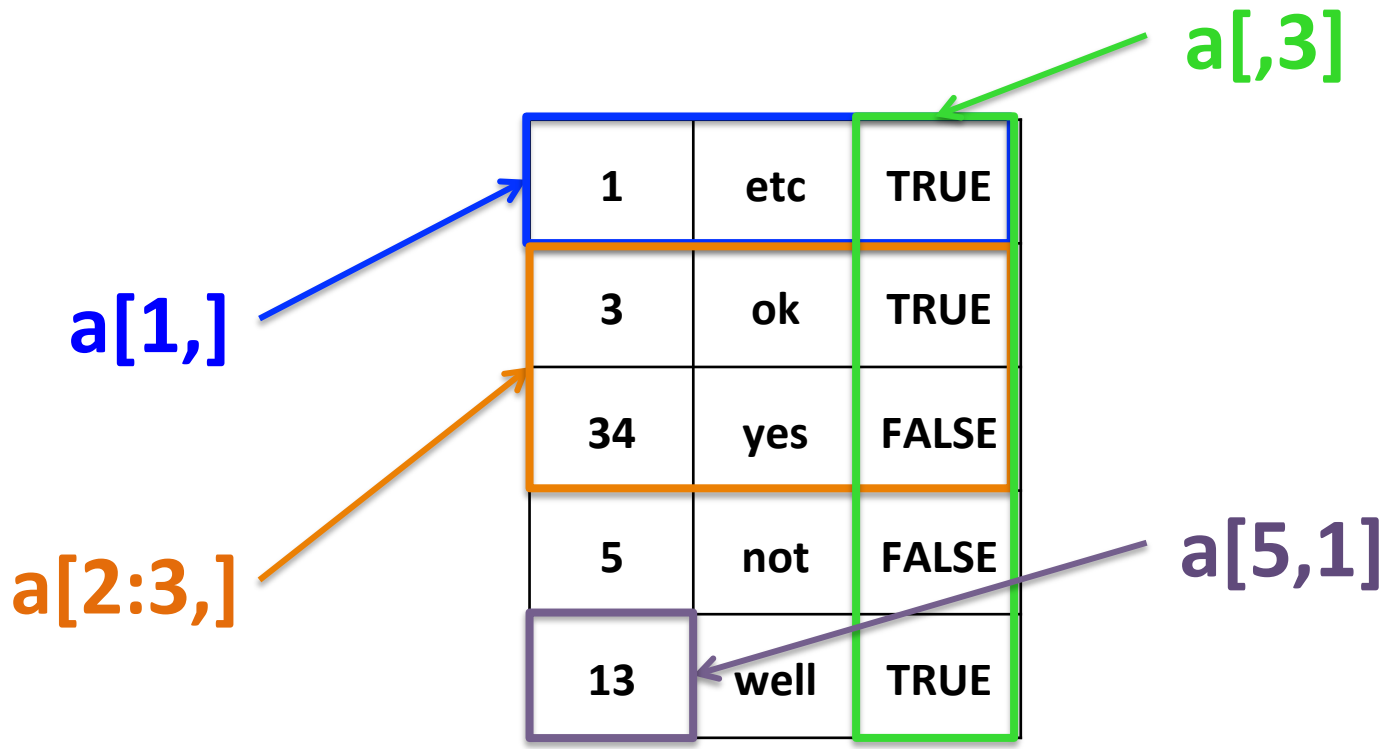
- Structures of 2 dimensions.
- More general than matrices.
- Different columns can have **different types** but must have the **same length.**

```
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



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

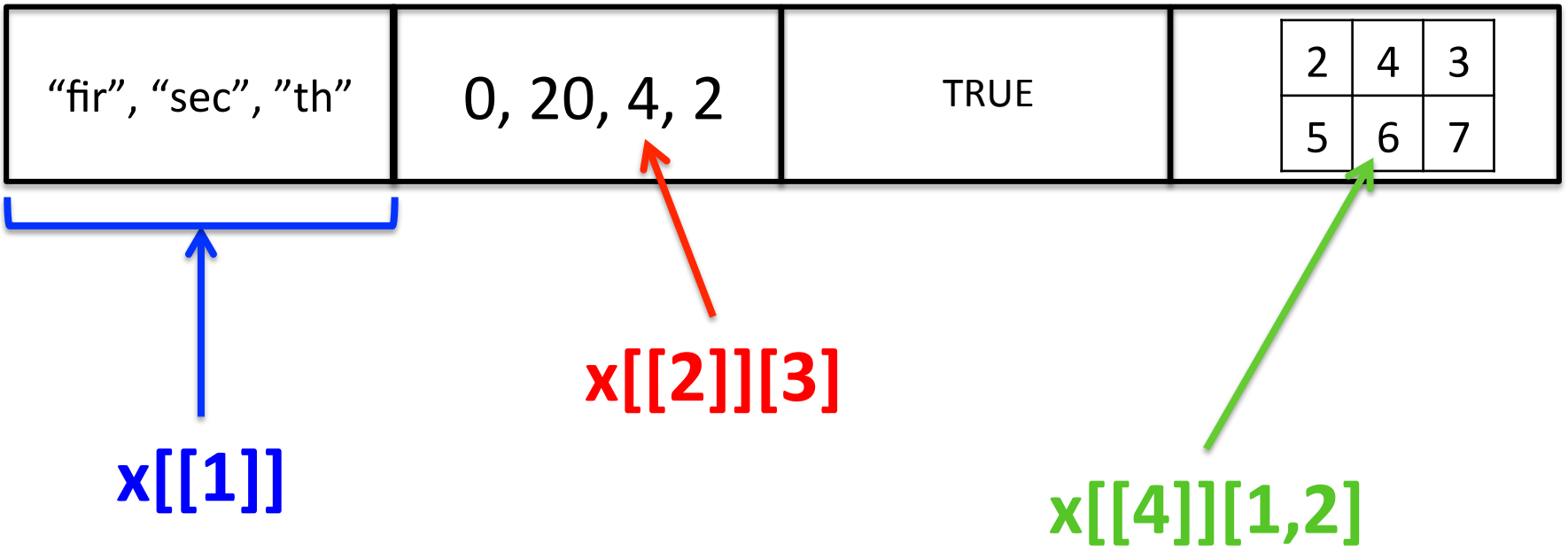
- Create a list:

```
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:



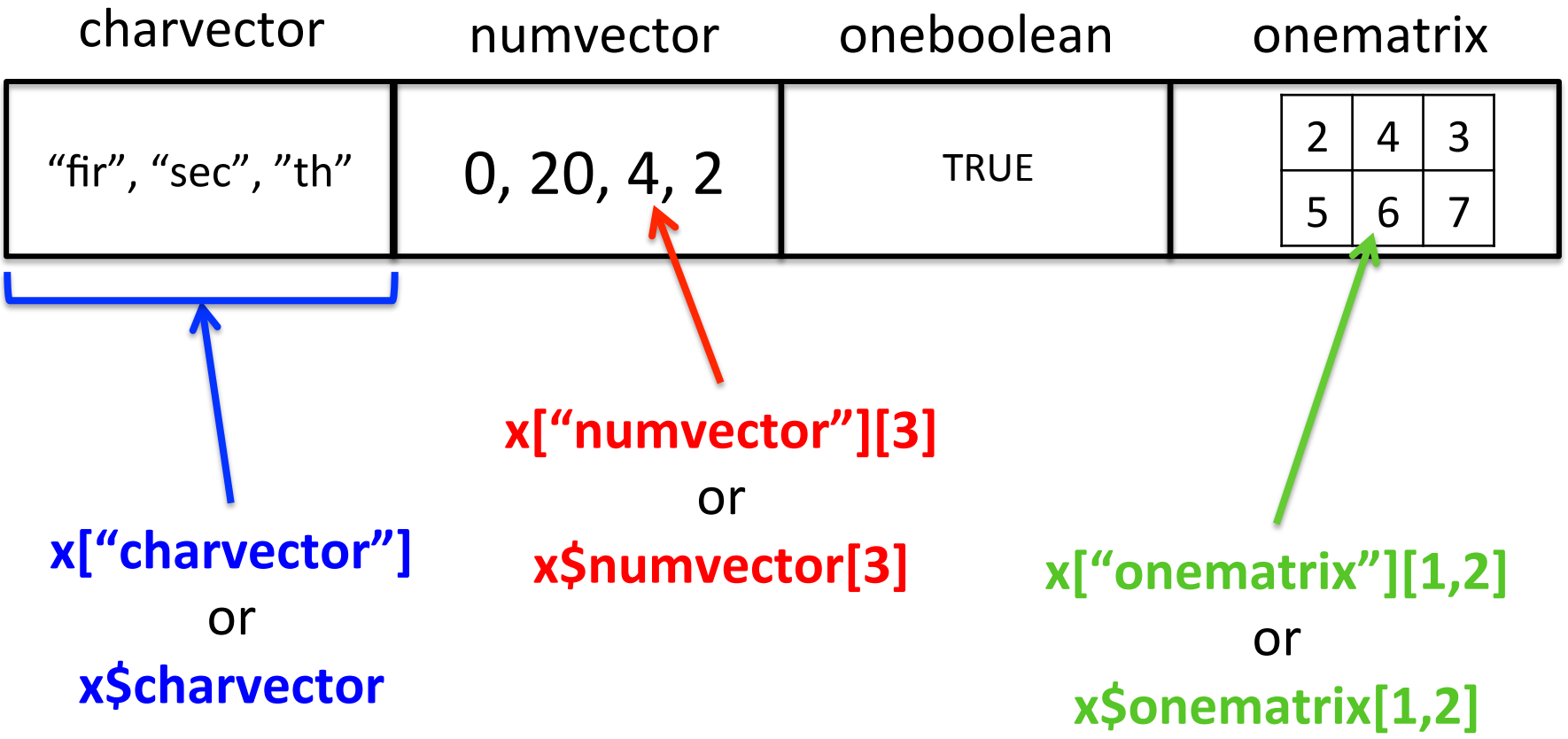
Naming elements of a list

```
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



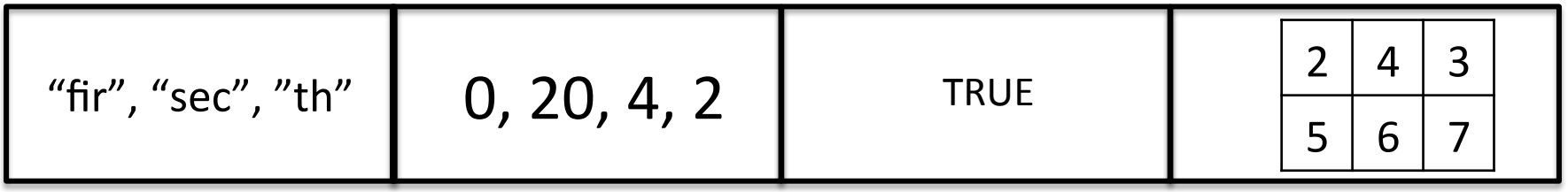
Length and dimension of lists

- 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.



length(x[[2]])

4

dim(x[[4]])

2 3

Checking and converting data types and structures

Checking	Converting
is.matrix(x)	as.matrix()
is.data.frame(x)	as.data.frame()
is.vector()	as.vector()
is.list()	as.list()
is.factor()	as.factor()
is.character()	as.character()
is.numeric()	as.numeric()

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	**mean(x)**
median	**median(x)**
minimum	**min(x)**
maximum	**max(x)**
variance	**var(x)**
correlation	**cor(x)**

Some common arithmetic functions

natural logarithm	**log(x)**
exponential function e^x	**exp(x)**
sine	**sin(x)**
cosine	**cos(x)**
tangent	**tan(x)**
absolute value	**abs(x)**
square root	**sqrt(x)**

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

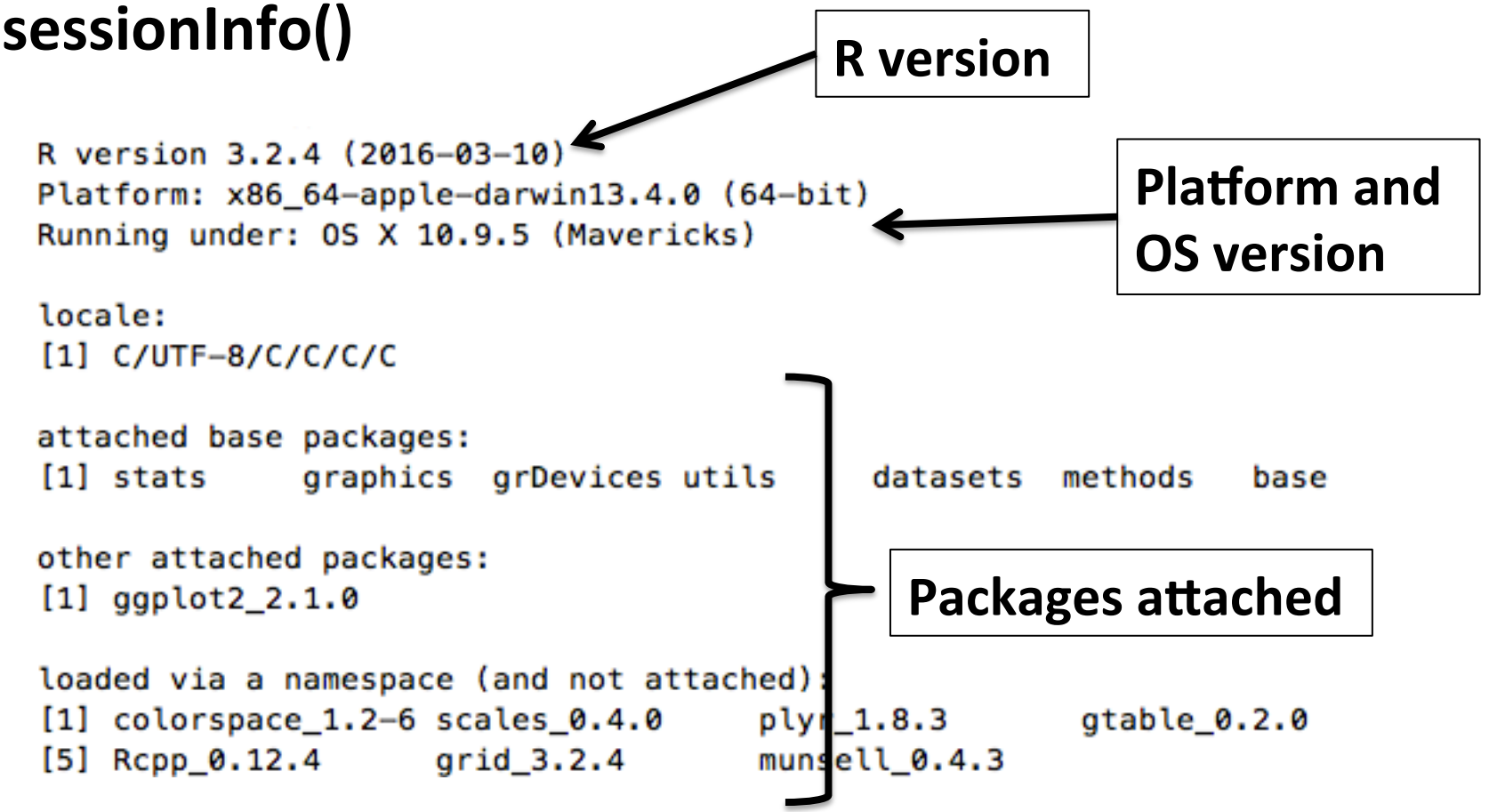
- Save objects x and y into “myobjects.RData” file
  - **save(x, y, file=“myobjects.RData”)**
- Load objects x and y into current directory:
  - **load(“myobjects.RData”)**
- Save the current workspace (all objects):
  - **save.image(file=“.RData”)**

Command history

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

Information about the current session

sessionInfo()



Functions in R

Knowing the source code of a function

- Name of the function without **()**

For example:

sort

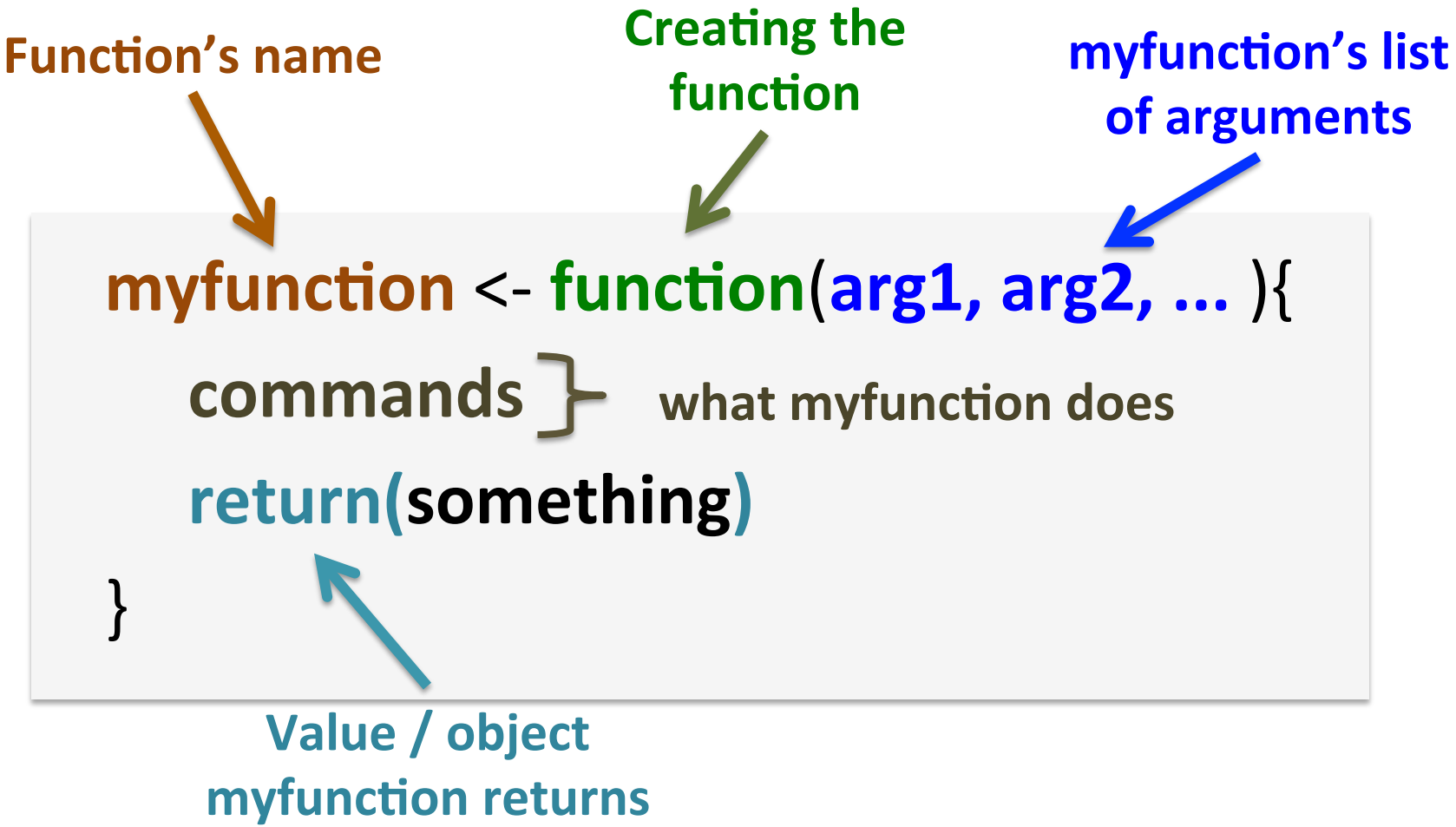
```
function (x, decreasing = FALSE, ...)
{
    if (!is.logical(decreasing) || length(decreasing) != 1L)
        stop("'decreasing' must be a length-1 logical vector.\nDid
you intend to set 'partial'?")
    UseMethod("sort")
}
```

User-written functions

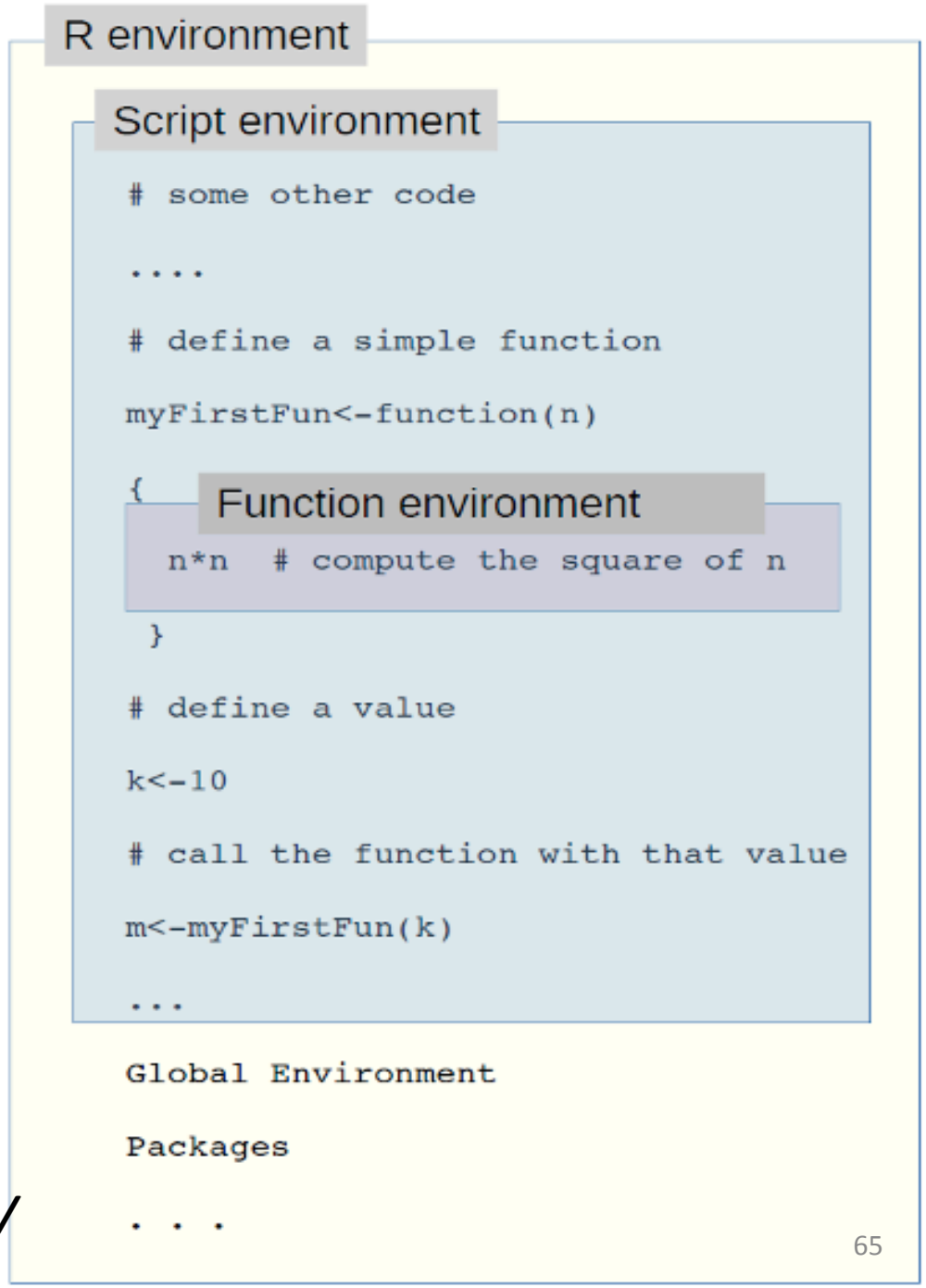
- Functions are pieces of code written **to carry out (a) specified task(s)** and be able to repeat it(them) easily.

- R allows you to create your own functions.

Functions' structure



Objects in a function are local to that function!



from https://www.datacamp.com/community/tutorials/functions-in-r-a-tutorial

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 "somestats":

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)

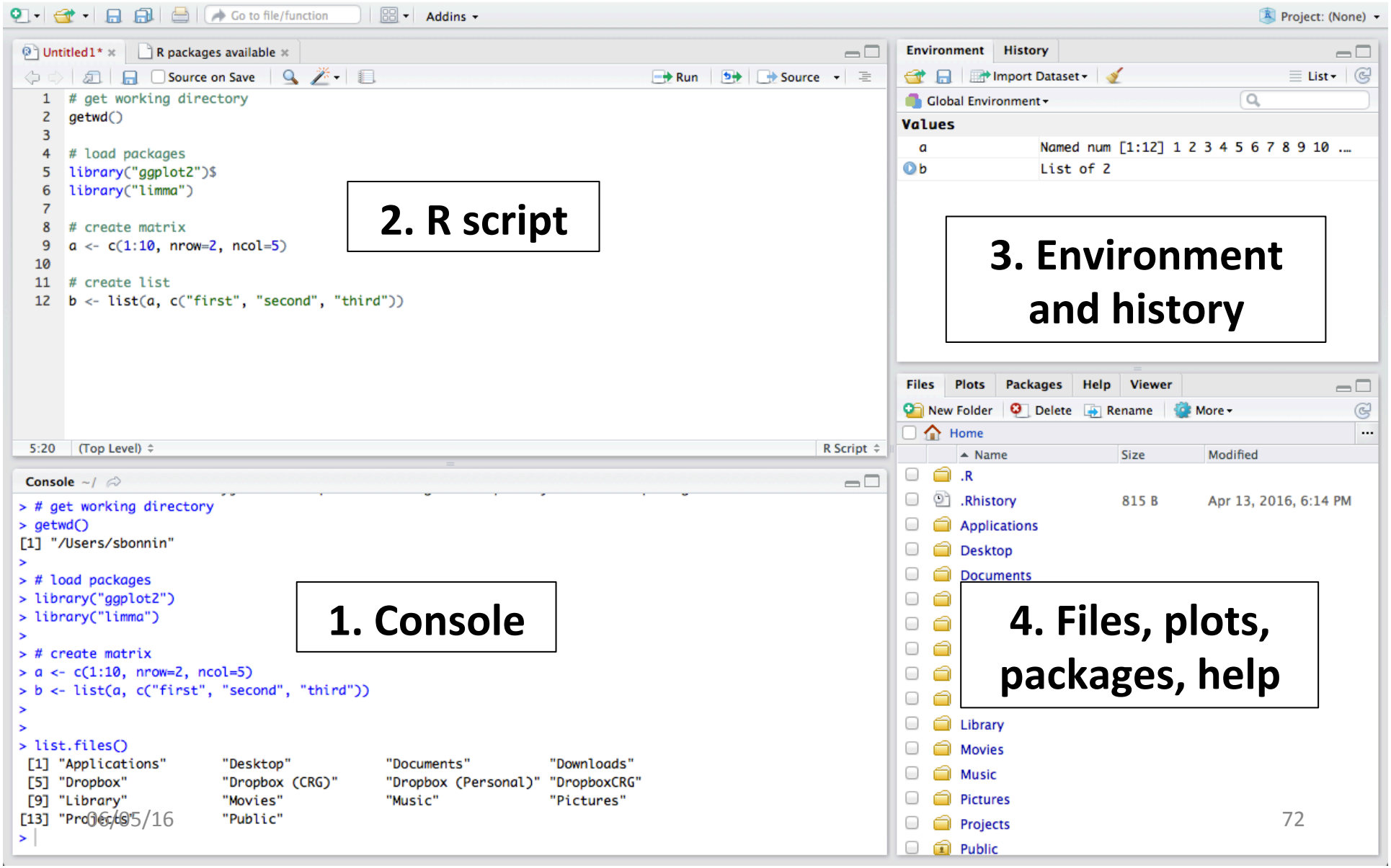
R Studio

What is R studio?

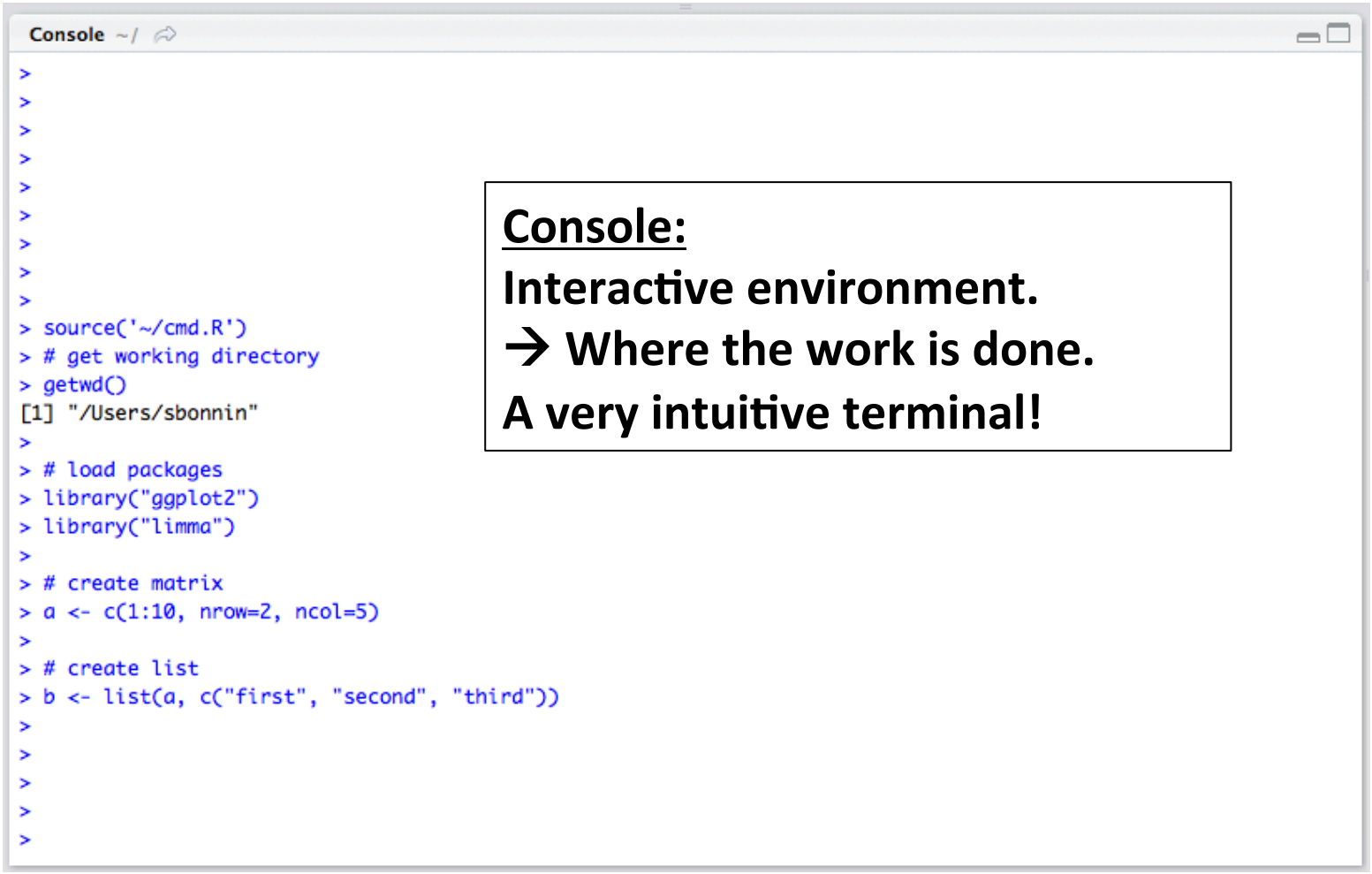
- **Free and open source** IDE (Integrated Development Environment) for R
- Available for Windows, Mac OS and Linux
- Written in C++
- First beta version available in 2009
- User friendly environment

R Studio

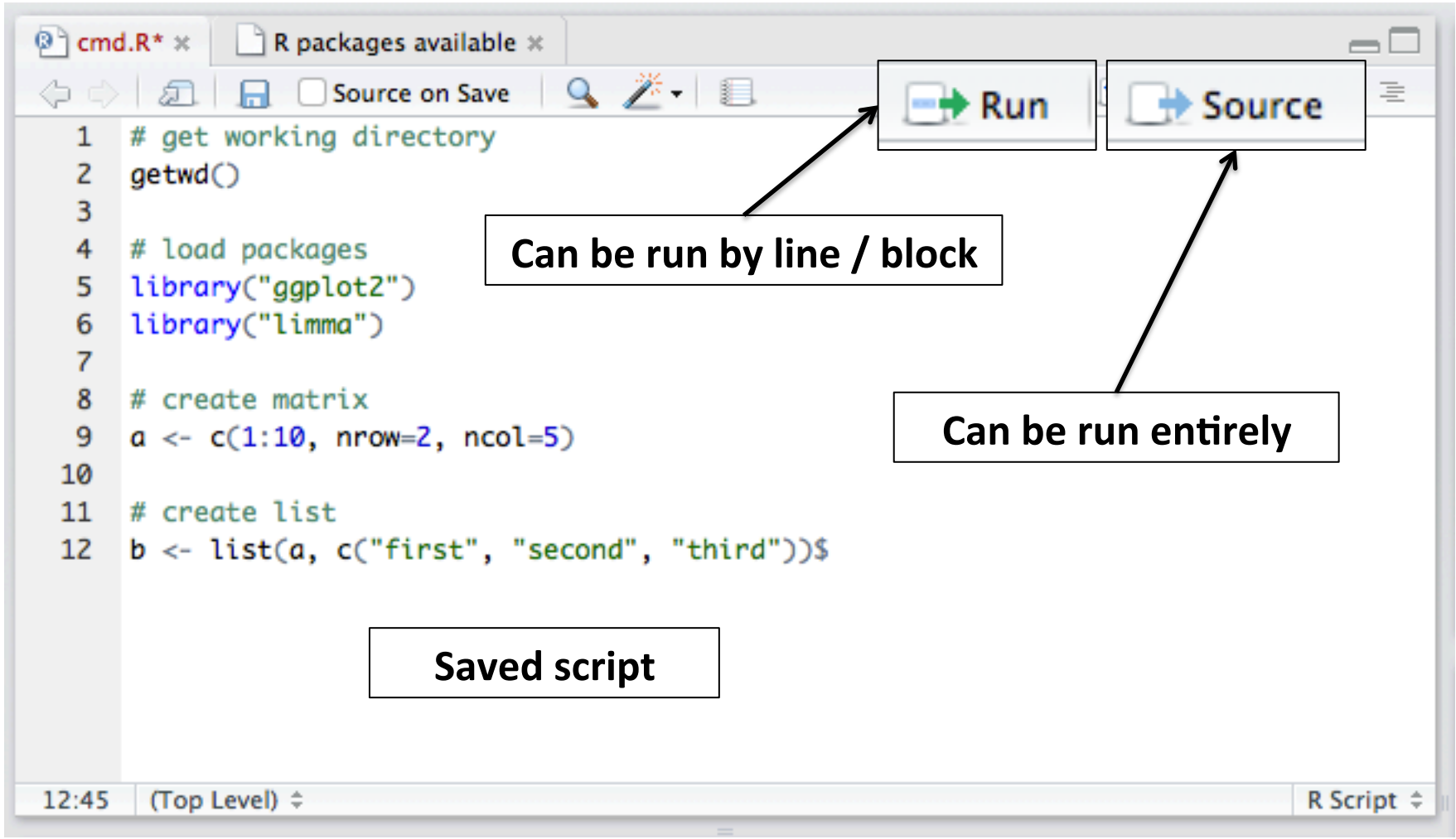
Screen: 4 windows



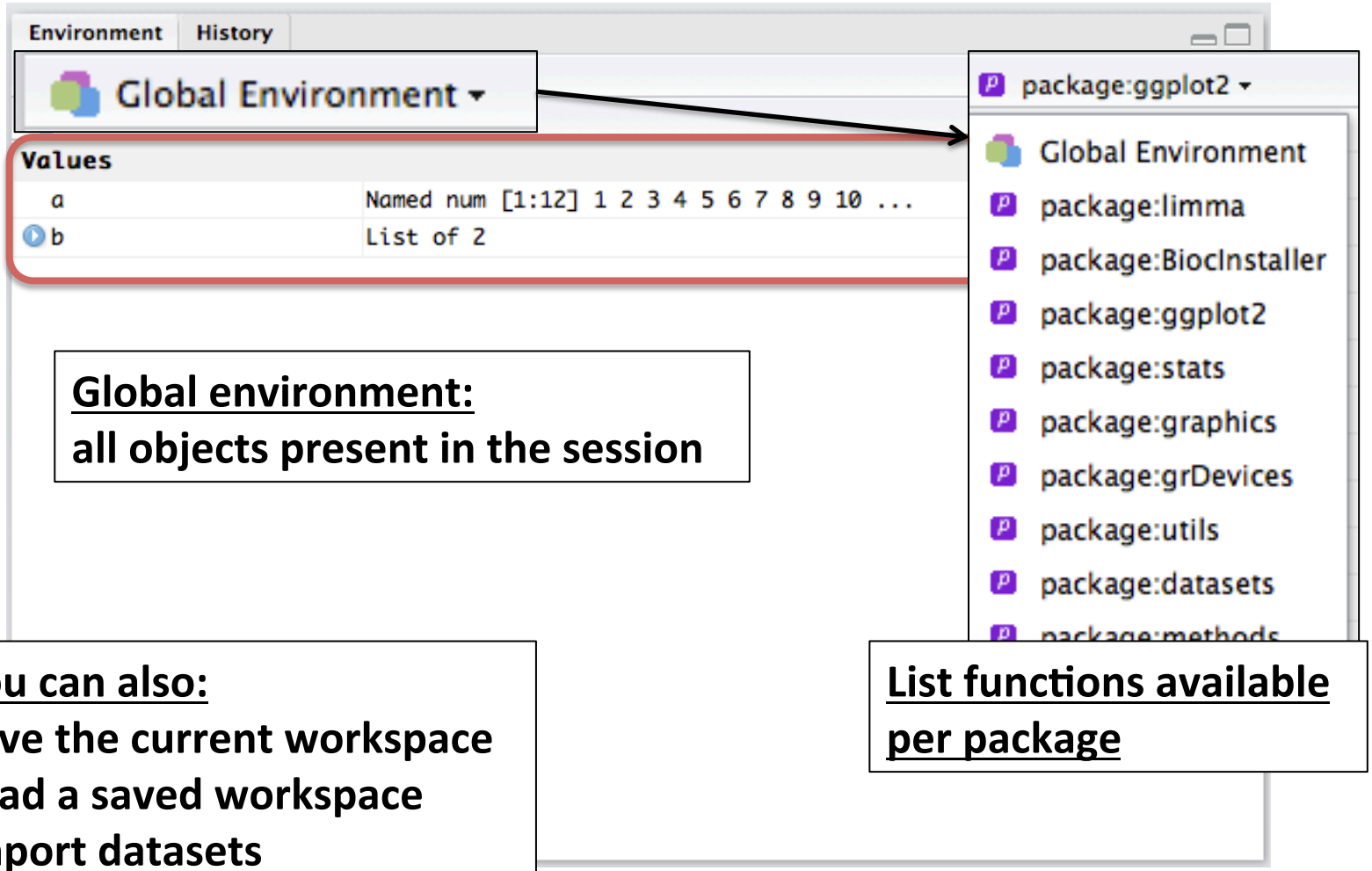
1. Console


Console ~/
>
>
>
>
>
>
>
>
>
>
> source('~/cmd.R')
> # get working directory
> getwd()
[1] "/Users/sbonnin"
>
> # load packages
> library("ggplot2")
> library("limma")
>
> # create matrix
> a <- c(1:10, nrow=2, ncol=5)
>
> # create list
> b <- list(a, c("first", "second", "third"))
>
>
>
>
>
Console:
Interactive environment.
→ Where the work is done.
A very intuitive terminal!

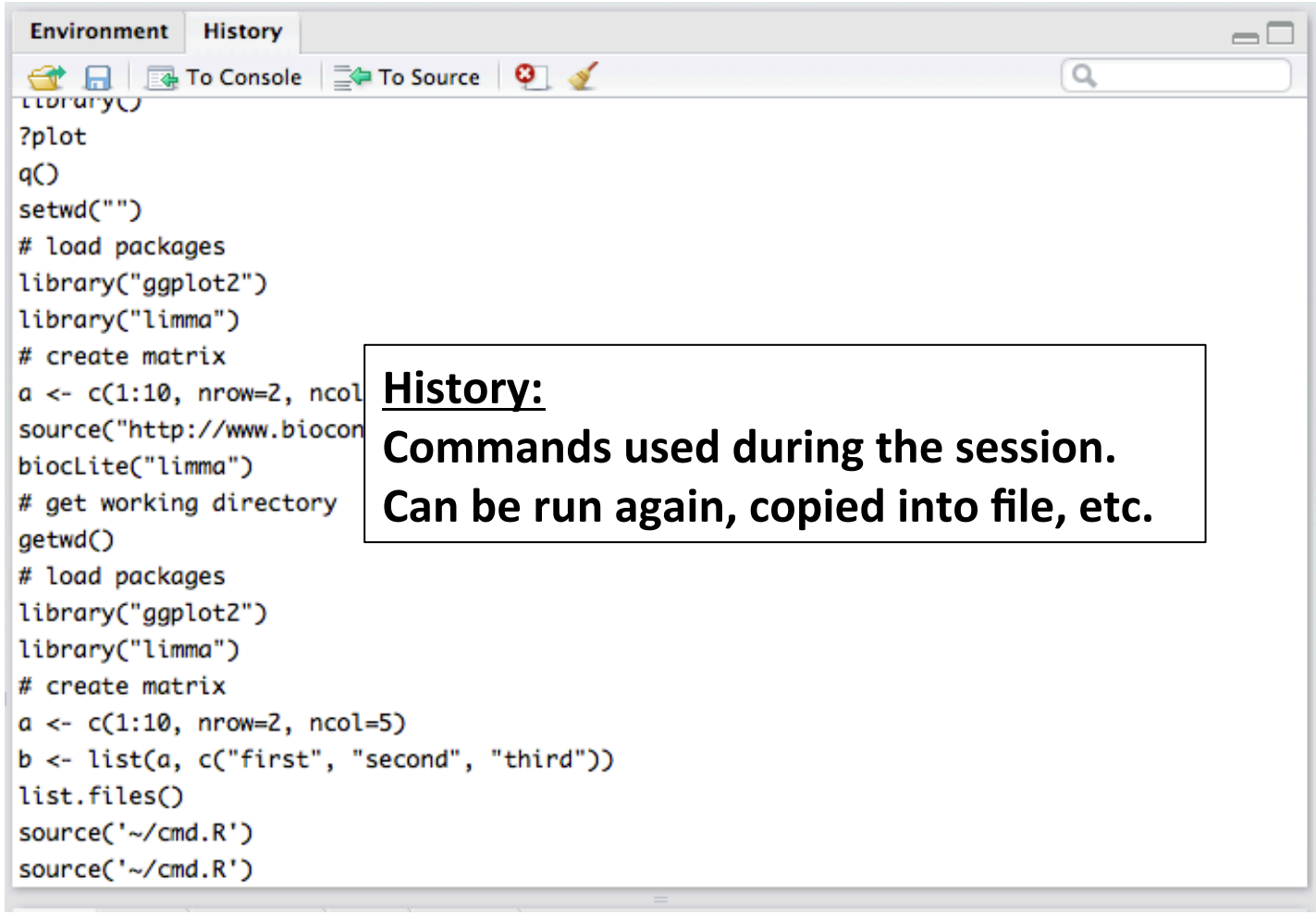
2. R script


cmd.R*
R packages available
Source on Save
Run
Source
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"))$
Can be run by line / block
Can be run entirely
Saved script
12:45
(Top Level)
R Script

3. Environment and history

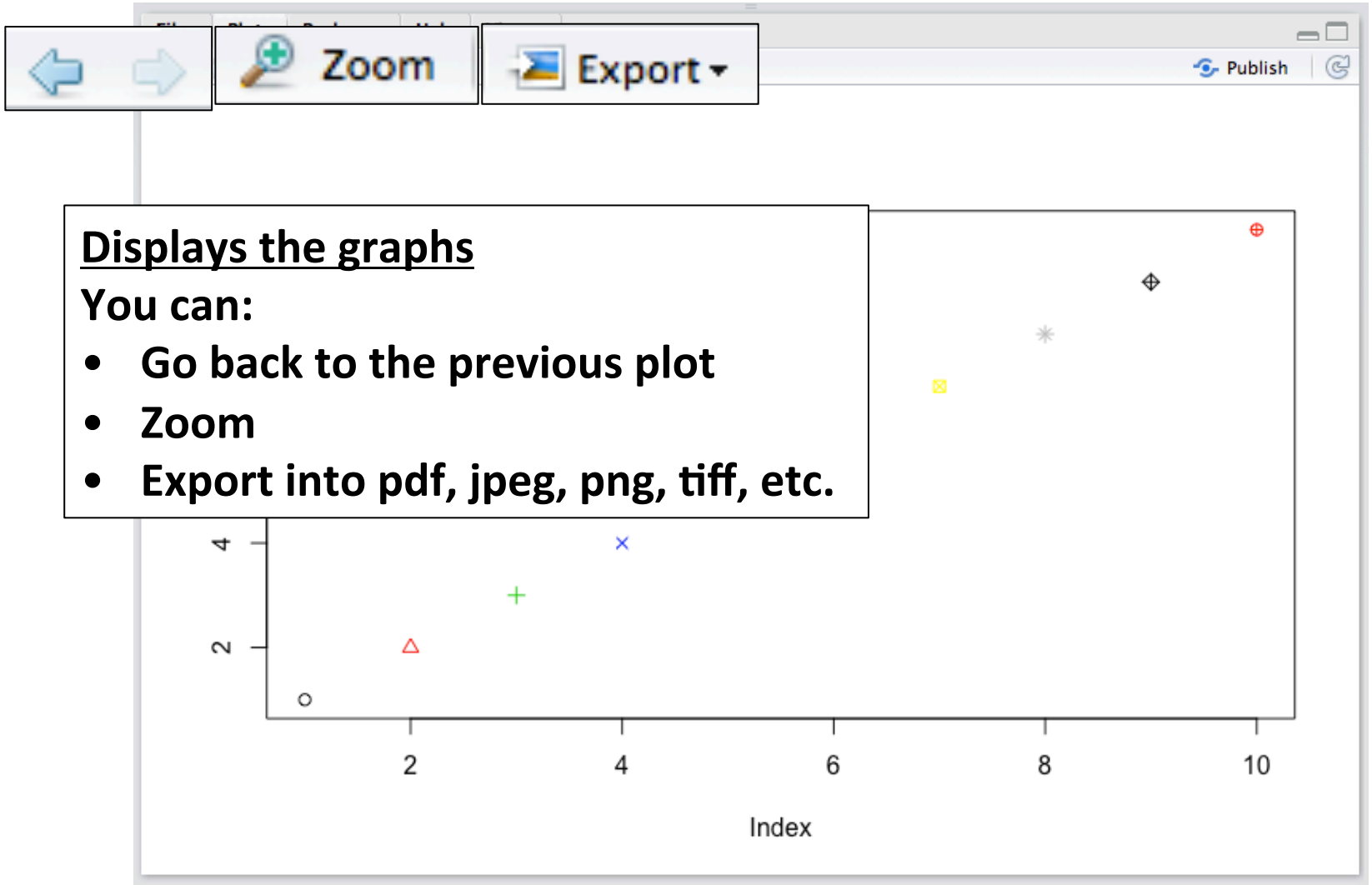


3. Environment and history

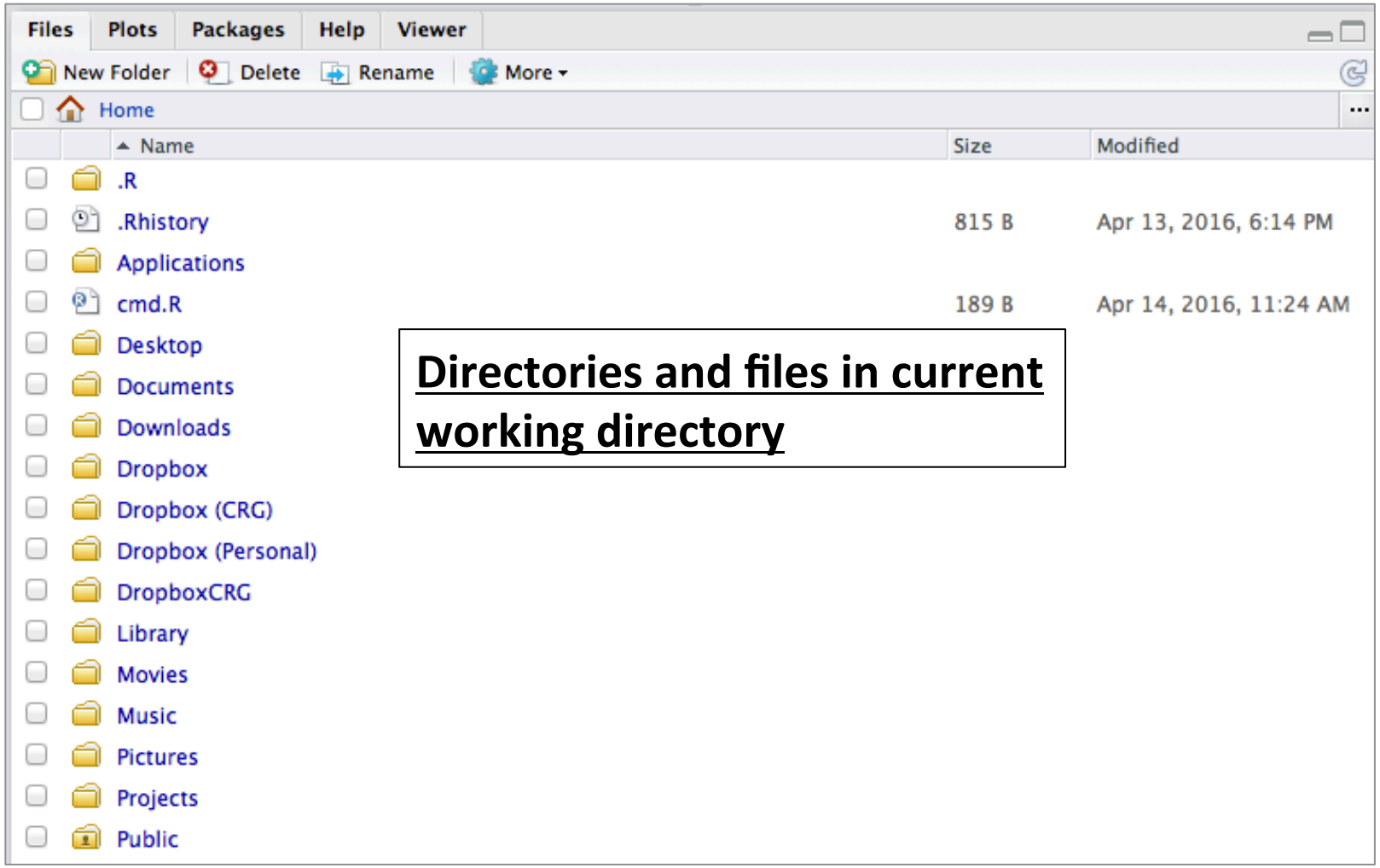


History:
Commands used during the session.
Can be run again, copied into file, etc.

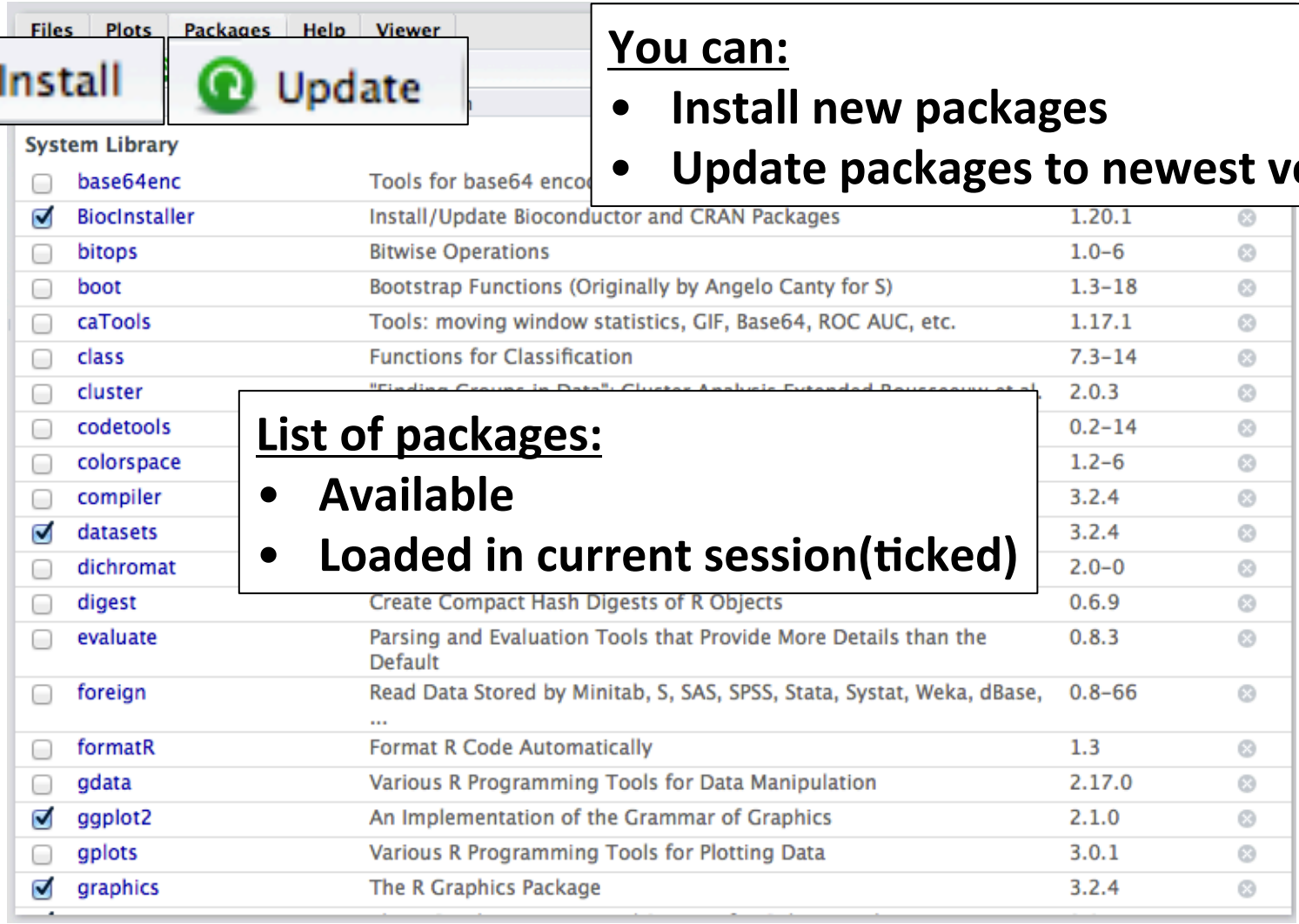
4. Plot area


Zoom
Export
Publish
Displays the graphs
You can:
• Go back to the previous plot
• Zoom
• Export into pdf, jpeg, png, tiff, etc.
2
4
2
4
6
8
10
Index

4. Files


Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.R
.Rhistory
815 B
Apr 13, 2016, 6:14 PM
Applications
cmd.R
189 B
Apr 14, 2016, 11:24 AM
Desktop
Documents
Downloads
Dropbox
Dropbox (CRG)
Dropbox (Personal)
DropboxCRG
Library
Movies
Music
Pictures
Projects
Public
Directories and files in current working directory

4. Packages



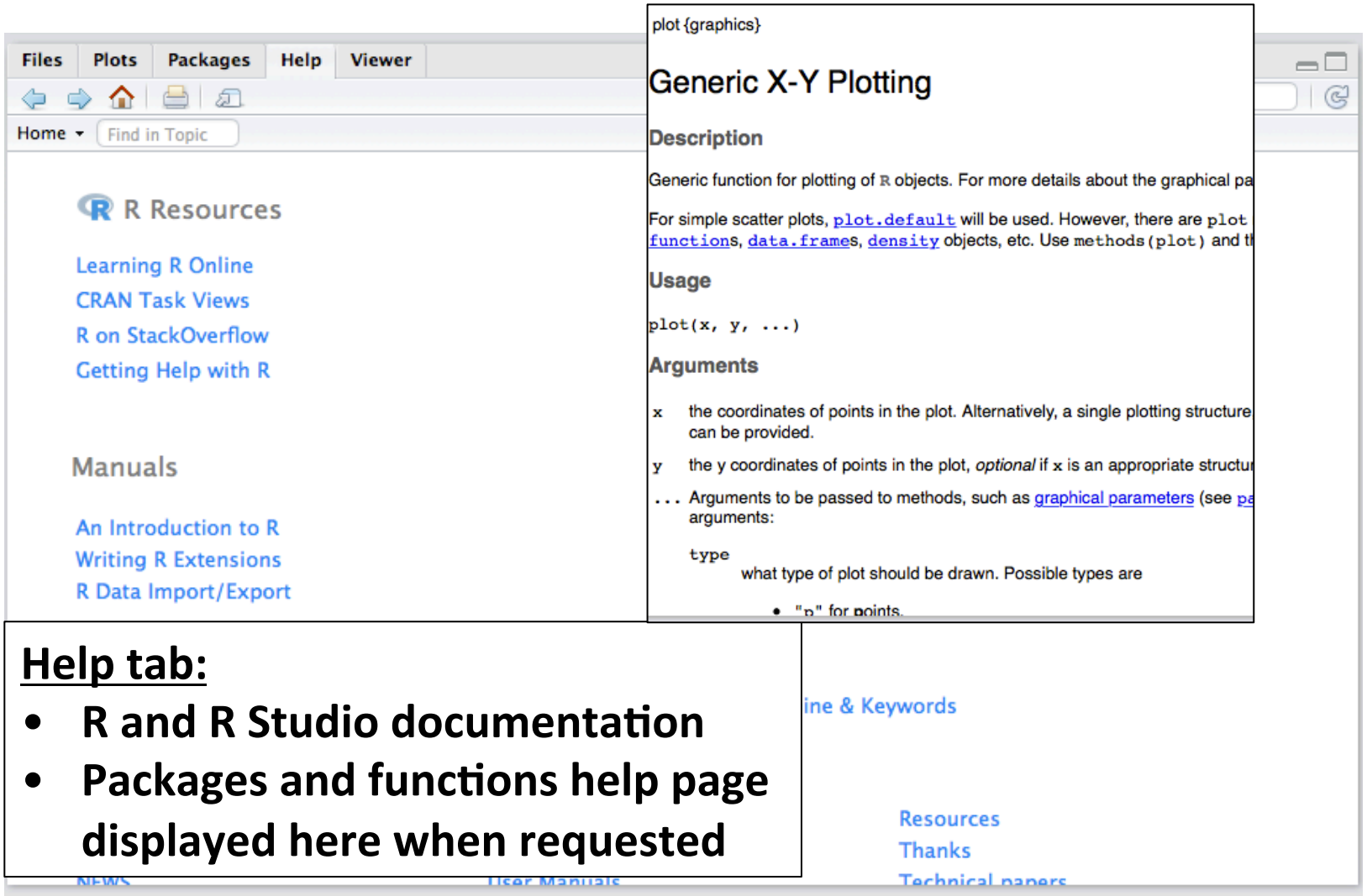
You can:

- **Install new packages**
- **Update packages to newest versions**

List of packages:

- **Available**
- **Loaded in current session(ticked)**

4. Help



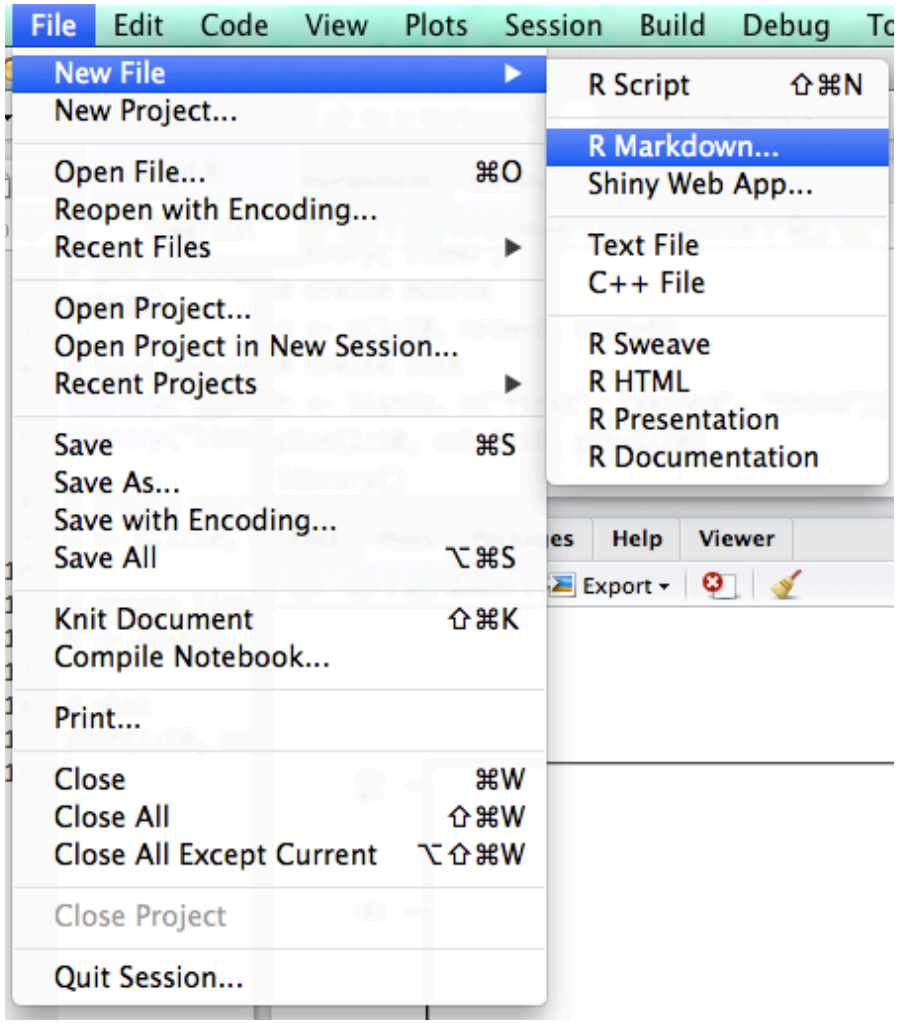
Help tab:

- **R and R Studio documentation**
- **Packages and functions help page displayed here when requested**

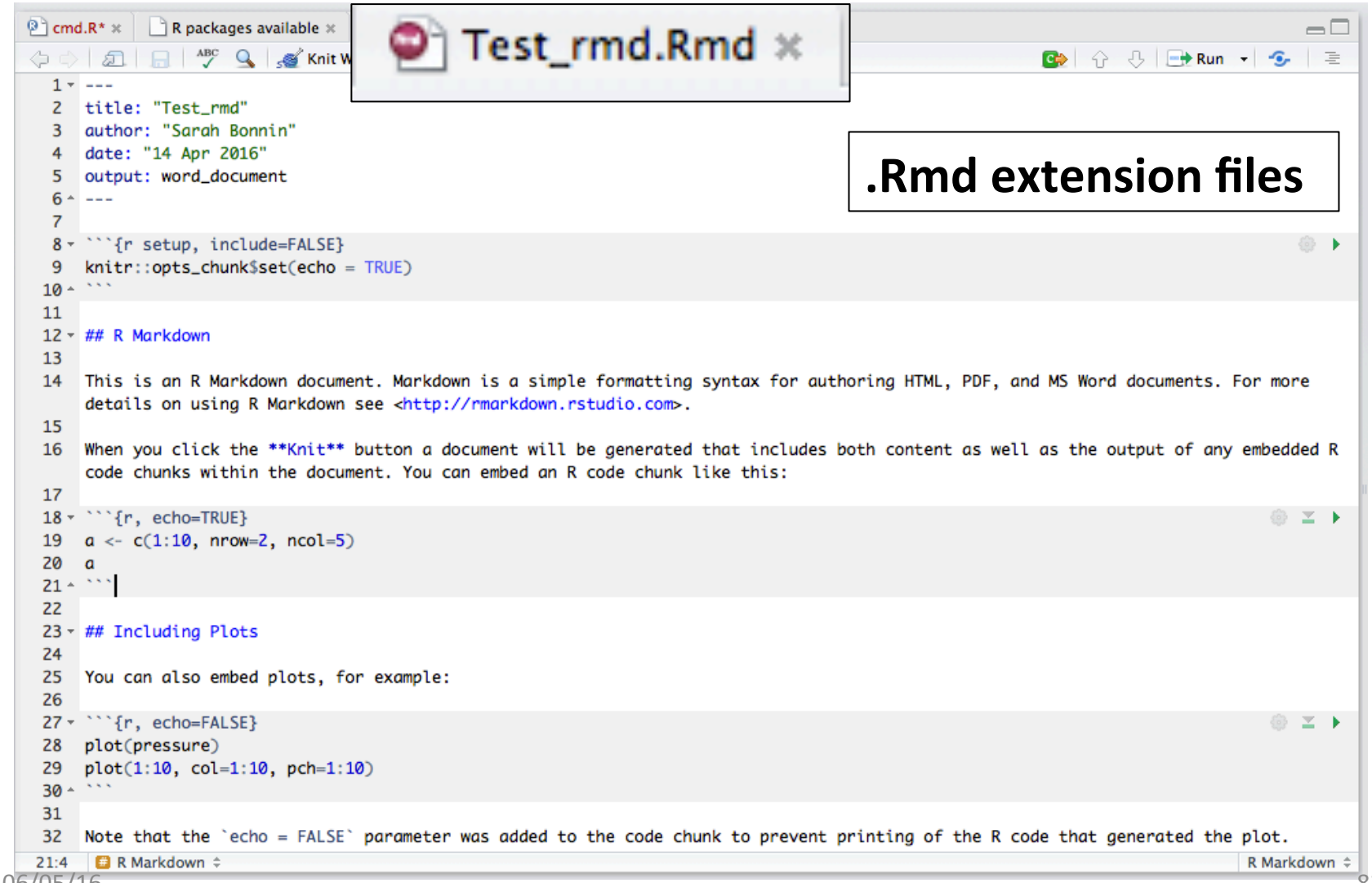
RMD: R Markdown

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

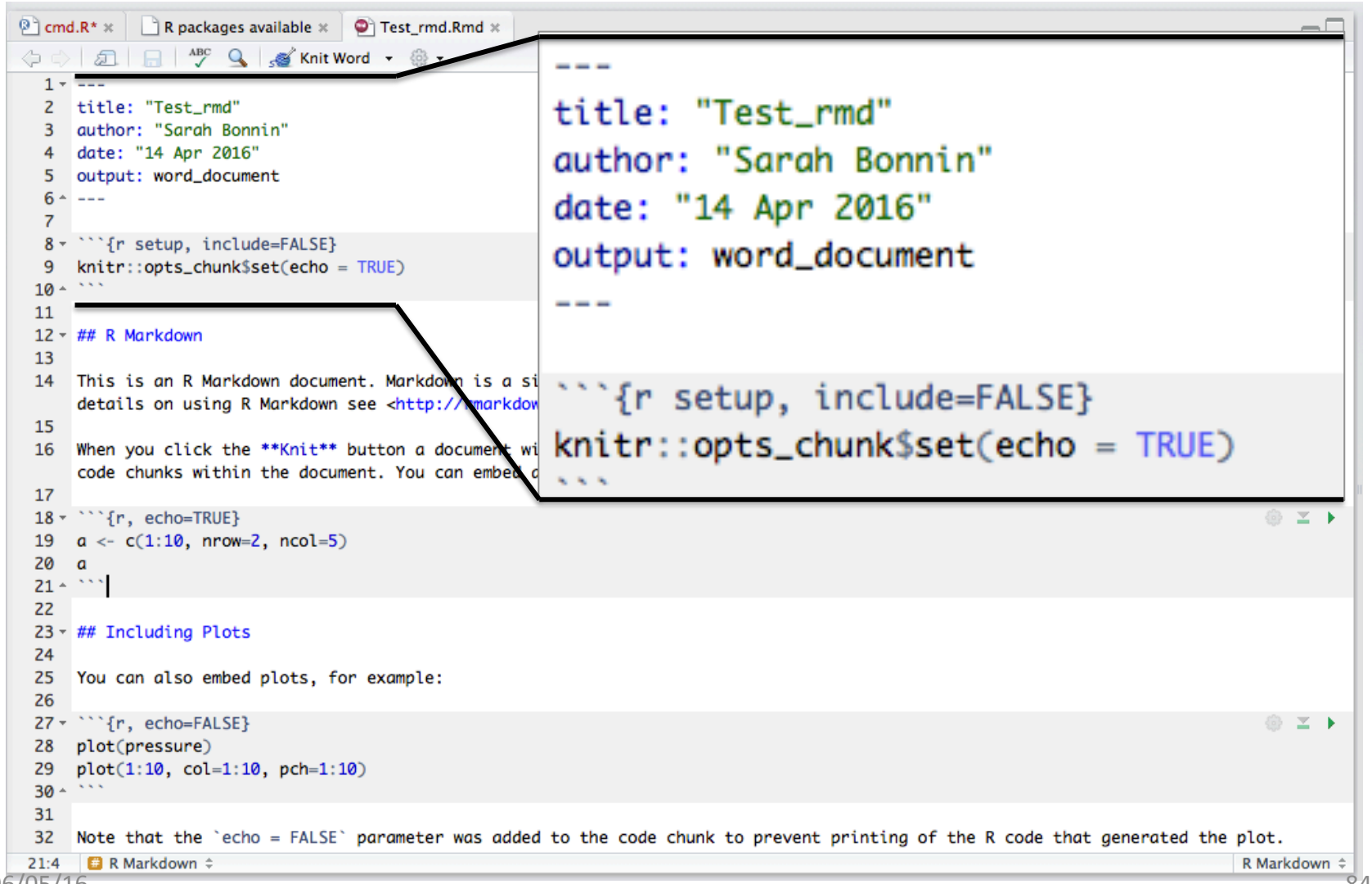
RMD: R Markdown



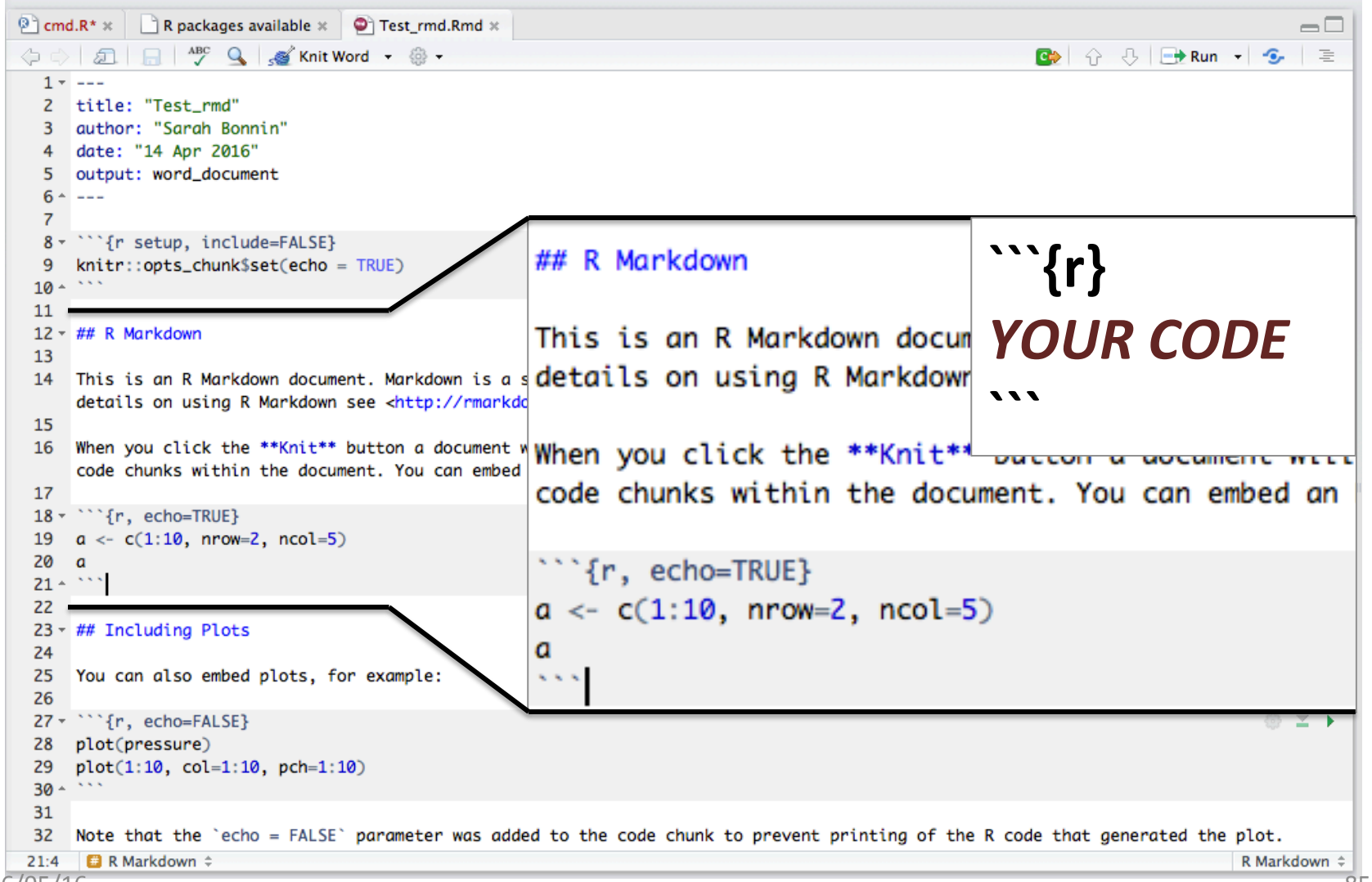
RMD: R Markdown



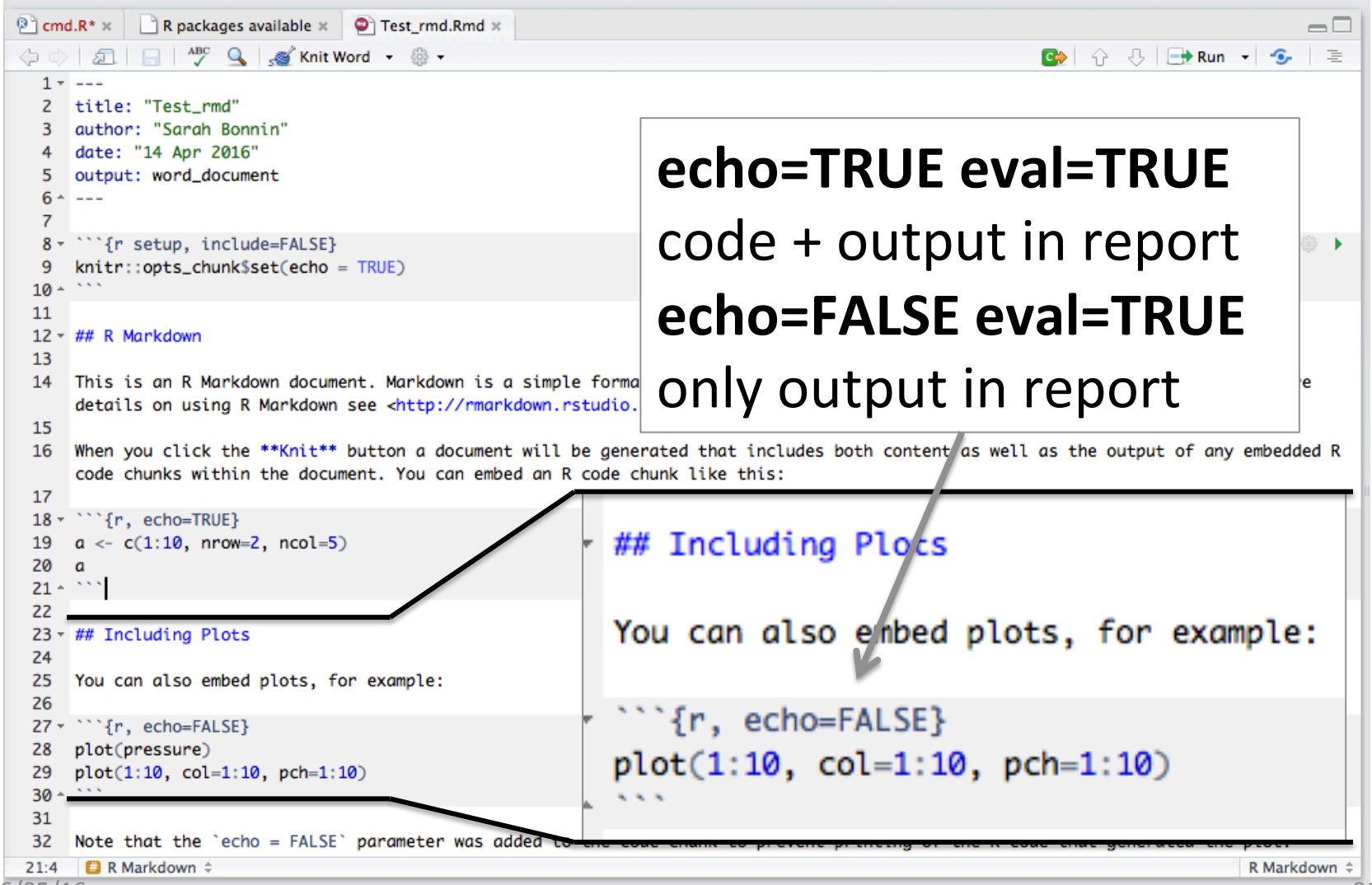
RMD: R Markdown



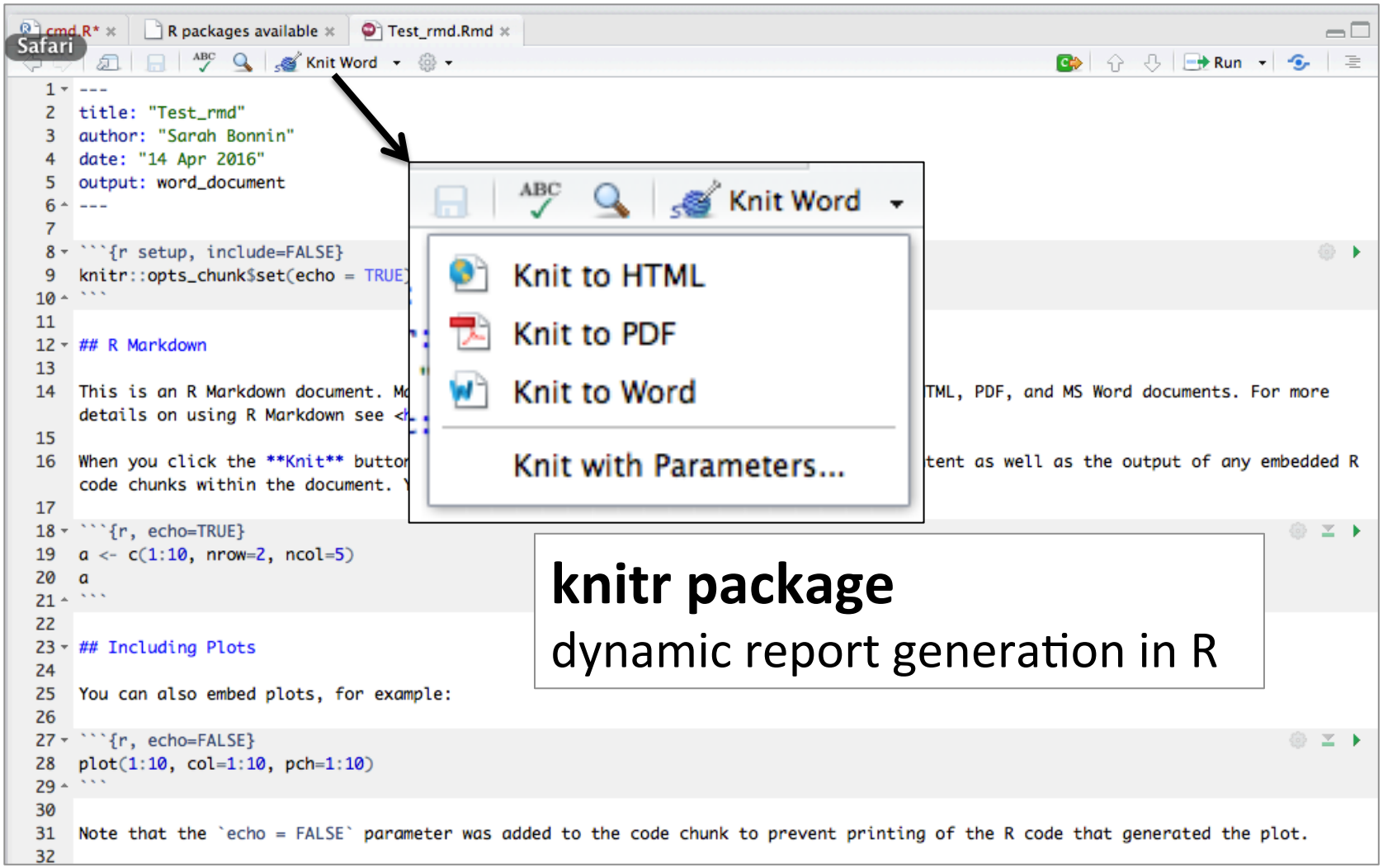
RMD: R Markdown



RMD: R Markdown



RMD: R Markdown



knitr package

dynamic report generation in R

RMD: R Markdown

Knit to HTML
→ Creates a .html file

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
- R help: http://r.789695.n4.nabble.com/R-help-f789696.html