Workshop Introdução à Linguagem de Programação em R

Carina Silva
(*carina.silva@estesl.ipl.pt*)
Escola Superior de Tecnologias da Saúde de Lisboa-IPL e Centro de Estatística e Aplicações, Universidade de Lisboa

Organização: NucM - Núcleo de Matemática da FCT-NOVA
3 dezembro 2021

Contents

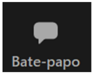
Contents

Contents

Contents

Contents

Symbols



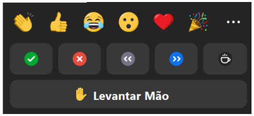


Figure: `https://github.com/CarinaSilva/Workshop-Linguagem-R`

Recomendations

- Relax
- Experiment
- Make mistakes
- Learn
- Enjoy

What R can be used for?

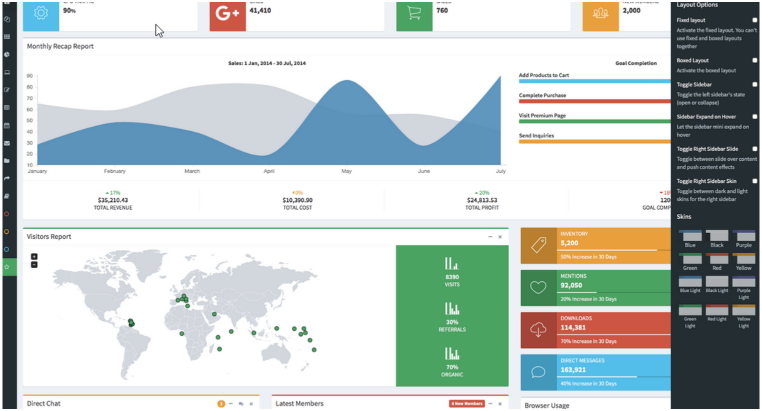
Data Science

Statistical Research

Documentation Production

rmarkdown

Dashboards



Layout Options

Fixed layout
Activate the fixed layout. You can't use fixed and boxed layouts together

Boxed Layout
Activate the boxed layout

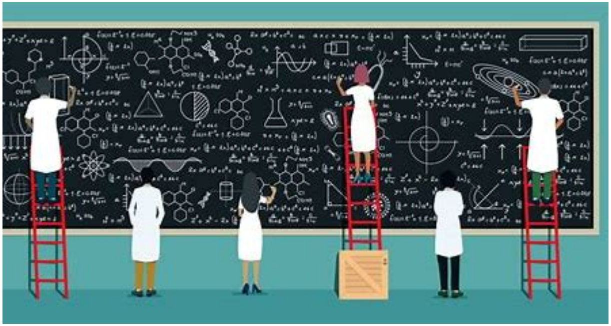
Toggle Sidebar
Toggle the left sidebar's state (open or collapse)

Sidebar Expand on Hover
Let the sidebar mini expand on hover

Toggle Right Sidebar Slide
Toggle between slide over content and push content effects

Toggle Right Sidebar Skin
Toggle between dark and light skins for the right sidebar

Academic Research



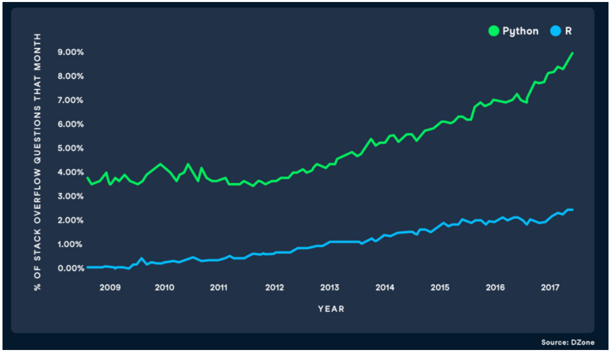
Why R is so popular?

- Specialized language
- Growth of the Data Science field
- Monopoly of Academia
- Interfaces Extremely well

R vs. Python

- R - Data analysis and statistics; Python - Deployment and production
- R is mainly used for statistical analysis
- Python is general purpose

R vs. Python


● Python ● R
% OF STACK OVERFLOW QUESTIONS THAT MONTH
9.00%
8.00%
7.00%
6.00%
5.00%
4.00%
3.00%
2.00%
1.00%
0.00%
2009
2010
2011
2012
2013
2014
2015
2016
2017
YEAR
Source: DZone

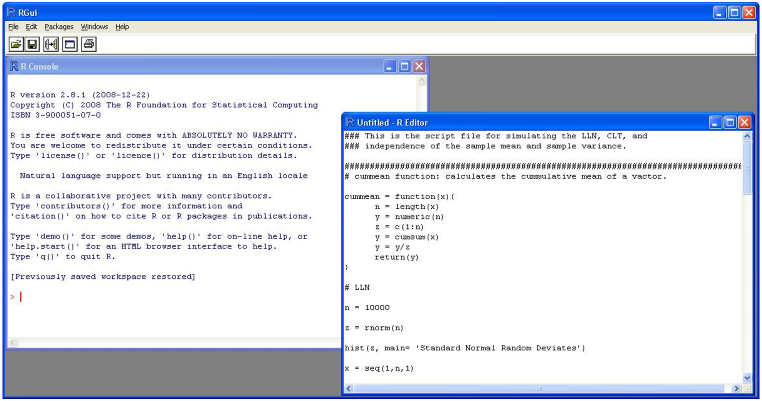
Why R?

- It's free!
- 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.
- It contains advanced statistical routines not yet available in other packages.
- Very good graphics capabilities.

Introduction

- R it is both a commandline environment and a programming language.
- Since R is a programming language, it is flexible but that comes with the price of a somewhat steep learning curve. The best way to learn R is through practice.

R Software



```
R version 2.8.1 (2008-12-22)
Copyright (C) 2008 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

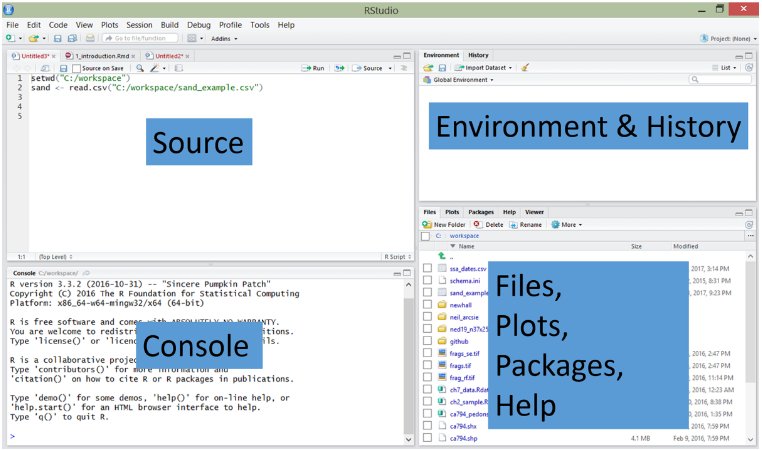
```
### This is the script file for simulating the LLN, CLT, and
### independence of the sample mean and sample variance.

##########################################################################
# cummean function: calculates the cummulative mean of a vactor.

cummean = function(x){
      n = length(x)
      y = numeric(n)
      z = c(1:n)
      y = cumsum(x)
      y = y/z
      return(y)
}

# LLN

n = 10000

z = rnorm(n)

hist(z, main= 'Standard Normal Random Deviates')

x = seq(1,n,1)
```

RStudio: An Integrated Development Environment (IDE) for R

RStudio is similar to the standard RGui, but is considerably more user friendly. It has more drop-down menus, windows with multiple tabs, and many customization options.

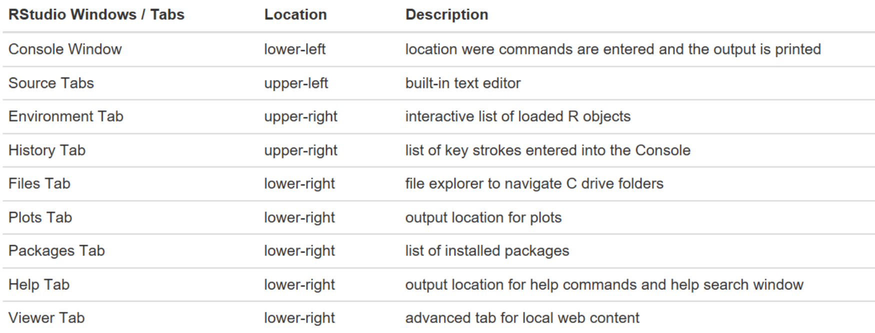
`https://www.rstudio.com/`

RStudio: An Integrated Development Environment (IDE) for R



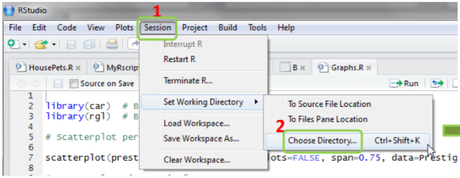
```
setwd("C:/workspace")
sand <- read.csv("C:/workspace/sand_example.csv")
```

```
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistri                    itions.
Type 'license()' or 'licen                     ils.

R is a collaborative projec
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```



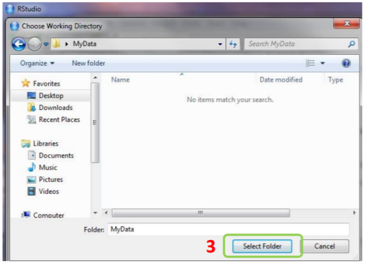
RStudio Windows / Tabs	Location	Description
Console Window	lower-left	location were commands are entered and the output is printed
Source Tabs	upper-left	built-in text editor
Environment Tab	upper-right	interactive list of loaded R objects
History Tab	upper-right	list of key strokes entered into the Console
Files Tab	lower-right	file explorer to navigate C drive folders
Plots Tab	lower-right	output location for plots
Packages Tab	lower-right	list of installed packages
Help Tab	lower-right	output location for help commands and help search window
Viewer Tab	lower-right	advanced tab for local web content

Working Directory



```
library(car)  # B
library(rgl)  # B

# Scatterplot per

scatterplot(prest
```

```
lots=FALSE, span=0.75, data=Prestig
```



If you have different projects you can change the working directory for that session, see above. Or you can type:

```
# Shows the working directory (wd)

getwd()

# Changes the wd

setwd("C:/myfolder/data")
```

Getting Help

- Use the Help tab in the lower-right Window to search commands (such as hist) or topics (such as histogram).
- Help function: `help(function)` or `?function`
- `apropos('distributions')` or `help.search('distributions')`
- `RSiteSearch('neural networks')`
- R page `https://www.r-project.org/help.html`
- Cheat-Sheets
- Stack Overflow `https://stackoverflow.com/`
- Stack Exchange `https://stackexchange.com/`

R Packages

In R, a package is a collection of R functions, data and compiled code.

There are many contributed packages that can be used to extend R.

These are created and maintained by the authors.

The location where the packages are stored is called the library.

R Packages

When you download R, already a number (around 30) of packages are downloaded as well. To use a function in an R package, that package has to be attached to the system. When you start R not all of the downloaded packages are attached, only seven packages are attached to the system by default. You can use the function `search()` to see a list of packages that are currently attached to the system, this list is also called the *search path*.

```
> search()
 [1] ".GlobalEnv"  "CO2"  "package:stats"
 [4] "package:graphics"  "package:grDevices"
"package:utils"
 [7] "package:datasets"  "package:methods"
"Autoloads"
 [10] "package:base"
```

R Packages

To attach another package to the system you can use the menu or the library function. Via the menu:

Select the "Tools" menu and select "Install packages...", and then write the package that you are searching for. Via the `library()` function:

```
> library(MASS)
> shoes
$A
13.2 8.2 10.9 14.3 10.7 6.6 9.5 10.8 8.8 13.3
$B
14.0 8.8 11.2 14.2 11.8 6.4 9.8 11.3 9.3 13.6
```

Library function

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()
```

Basic Tips for using R

- Commands in R are also called functions. The basic format of a function in R is: `function.name(argument1 = , argument2 = )`.
- The up arrow (↑) on your keyboard can be used to bring up previous commands that you have typed in the R console.
- Any text that you do not want R to act on (such as comments, notes, or instructions) needs to be preceded by the ♯ symbol (a.k.a. hash-tag, comment, pound, or number symbol). R ignores the remainder of the script line following ♯.

Basic Tips for using R

- Results of calculations can be stored in objects using the assignment operators:

✓ An arrow (<-) formed by a *smaller than* character and a *hyphen* without a space.

✓ The equal character (=).

Basic Tips for using R

These objects can then be used in other calculations. To print the object just enter the name of the object. There are some restrictions when giving an object a name:

- Object names cannot contain "strange" symbols like , +, -, #.

Basic Tips for using R

These objects can then be used in other calculations. To print the object just enter the name of the object. There are some restrictions when giving an object a name:

- Object names cannot contain "strange" symbols like , +, -, #.
- A dot (.) and an underscore (_) are allowed, also a name starting with a dot.

Basic Tips for using R

These objects can then be used in other calculations. To print the object just enter the name of the object. There are some restrictions when giving an object a name:

- Object names cannot contain "strange" symbols like , +, -, #.
- A dot (.) and an underscore (_) are allowed, also a name starting with a dot.
- Object names can contain a number but cannot start with a number.

Basic Tips for using R

These objects can then be used in other calculations. To print the object just enter the name of the object. There are some restrictions when giving an object a name:

- Object names cannot contain "strange" symbols like , +, -, #.
- A dot (.) and an underscore (_) are allowed, also a name starting with a dot.
- Object names can contain a number but cannot start with a number.
- R is case sensitive, X and x are two different objects, as well as temp and temP.

Basic Tips for using R

- Be consistent with the names of the objects.
- It's usual to use verbs to functions and substantives for objects.

Writing Scripts

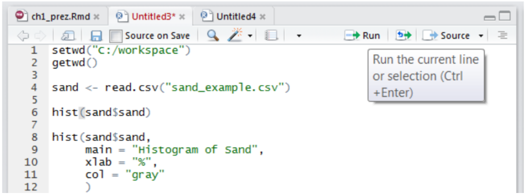
Prior to executing R functions at the Console, commands are typically written down (or scripted). Benefits include:

- allows others to reproduce your work, which is the foundation of science
- serves as instruction/reminder on how to perform a task
- allows rapid iteration, which saves time and allows the evaluation of incremental changes
- reduces the chance of human error

Basic Tips of Scripting

To write a script, simply open a new R script file by clicking **File>New File>R Script**. Within the text editor type out a sequence of functions.

- Place each function (e.g. read.csv()) on a separate line.
- If a function has a long list of arguments, place each argument on a separate line.
- An entire R script file can be executed by clicking the Source button.



```
setwd("C:/workspace")
getwd()

sand <- read.csv("sand_example.csv")

hist(sand$sand)

hist(sand$sand,
     main = "Histogram of Sand",
     xlab = "%",
     col = "gray"
     )
```

Basic Tips of Scripting

- Comment your code ($\sharp$)
- Be consistent with your style within your code
- Put at the beginning of your code all the libraries that you need to run your code
- If your code will read in data from a file, define a variable early in your code that stores the path to that file.

R code example

```
#libraries--------------------------------
library(ggplot2)
library(reshape)
library(vegan)
#-----------------------------------------
#data-------------------------------------
input_file <- "data/data.csv"
output_file <- "data/results.csv"
#------------------------------------------
# read input
input_data <- read.csv(input_file)
# get number of samples in data
sample_number <- nrow(input_data)
# generate results
results <- some_other_function(input_file, sample_num
# write results
write.table(results, output_file)
```

Exercises

- Go to `https://github.com/CarinaSilva/Workshop-Linguagem-R`
- Folder: Exercises: Open the exercises.md file and do exercises 1 and 2.

Figure: 3'

R as a calculator

◇ Calculator +, -, /, *, ^, log, exp, :

```
> (17*0.35)^(1/3)
[1] 1.812059

> log(10)
[1] 2.302585

> exp(1)
[1]2.718282

> 3^ − 1
[1] 0.3333333
```

Assigning Values to variables

Variables are assigned using "<-":

```
> x<-12.6
> x
[1] 12.6
```

Variables that contains many values (vectors), e.g. with the concatenate function:

```
> y<-c(3,7,9,11)
> y
[1] 3 7 9 11
```

Assigning Values to variables

Operator ":" means *a series of integers between*:

```
> x<-1:6
> x
[1] 1 2 3 4 5 6
```

Series in non-integer steps (e.g. 0.1) using the `seq()` function:

```
> b<-seq(0.5,0,-0.1)
> b
[1] 0.5 0.4 0.3 0.2 0.1 0.0
```

- Go to `https://github.com/CarinaSilva/Workshop-Linguagem-R`
- Folder: Exercises: Do exercises 3 and 4.

Figure: 3'

Basic Data Types and Data Structures in R

Everything in R is an object. R has 6 basic data types.

- character: "a", "brands" (string)
- factor: "female", "male" (categorical variable)
- numeric (real or decimal): 2, 12.5
- integer: 2L (the L tells R to store this as an integer)
- logical: TRUE, FALSE
- complex: 1+4i

Basic Data Types and Data Structures in R

R provides many functions to examine features of vectors and other objects, for example.

- `class()` - what kind of object is it?
- `typeof()` - what is the object's data type?
- `length()` - how long is it? What about two dimensional objects?
- `attributes()` - does it have any metadata?
- `str()` - how is data structure?
- `names()` - list the names of the levels of an object

Basic Data Types and Data Structures in R

- A general missing value indicator is `NA`
- `Inf`: Infinity
- `NaN`: means Not a Number. Its an undefined value.

Basic Data Types and Data Structures in R

Data structures

- vector
- list
- factors
- matrix
- data.frame

Vector Functions in R

All elements of a vector should be of the same type.

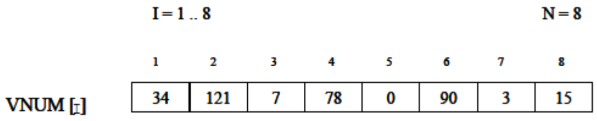
Typical operations on vectors include summary statistics (mean, var, range, max,):

```
> y<-c(5,7,7,8,2,5,6,6,7,5,8,3,4)
> z<-13:1
> mean(y)
[1] 5.615385
> var(z)
[1] 15.16667
```

Elements of vectors by subscripts in []:

```
> y[3]
[1] 7
```

Elements of a vector (array) are represented through sets of variables of a particular data type. A vector contains a name to which a data type is associated depending on the data to be manipulated by the vector, an index of type integer, and a dimension of type integer. A vector is displayed through a name and an index in square brackets.



The third to the seventh elements of y:

```
> y[3:7]
[1] 7 8 2 5 6
```

The third, fifth, sixth and seventh elements:

```
> y[c(3,5,6,7)]
[1] 7 2 5 6
```

To drop an element from the array, use negative subscripts:

```
> y[-1]
[1] 7 7 8 2 5 6 6 7 5 8 3 4
```

To drop the last element of the array without knowing its length:

```
> y[-length(y)]
 [1] 5 7 7 8 2 5 6 6 7 5 8 3
```

Logical condition to find a subset of the values in a vector:

```
> y[y>6]
[1] 7 7 8 7 8
```

To know the values for z for wich y>6:

```
> z[y>6]
[1] 12 11 10 5 3
```

List your objects that you have in your current session:

```
>ls()
```

Remove objects of your current sessions:

```
>rm(y)
```

- Go to `https://github.com/CarinaSilva/Workshop-Linguagem-R`
- Folder: Exercises: Do exercises 5 and 6.

Figure: 3'

Functions

A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions.

An R function is created by using the keyword function. The basic syntax of an R function definition is as follows

```
function_name <- function(arg_1, arg_2, ...) {
   Function body
}
```

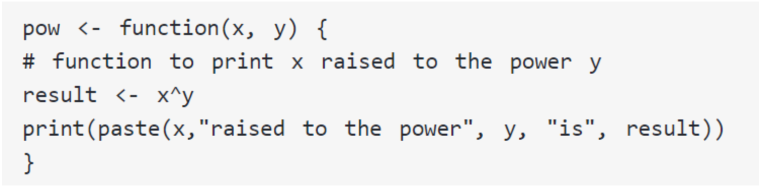
Functions

The different parts of a function are

- **Function Name** - This is the actual name of the function. It is stored in R environment as an object with this name.
- **Arguments** - An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional, that is, a function may contain no arguments. Also arguments can have default values.
- **Function Body** - The function body contains a collection of statements that defines what the function does.
- **Return Value** - The return value of a function is the last expression in the function body to be evaluated.

Functions

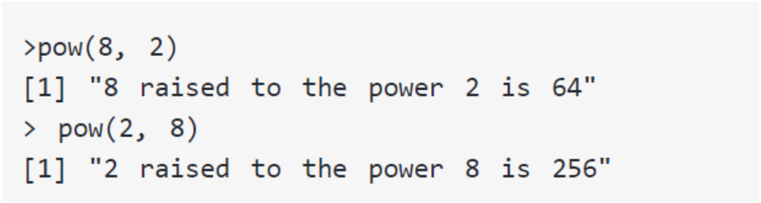
Example:



```
pow <- function(x, y) {
# function to print x raised to the power y
result <- x^y
print(paste(x,"raised to the power", y, "is", result))
}
```

Functions

How to call a function



```
>pow(8, 2)
[1] "8 raised to the power 2 is 64"
> pow(2, 8)
[1] "2 raised to the power 8 is 256"
```

Function body

- The function body appears within {curly brackets}. For functions with just one expression the curly brackets are not required but they may help you read your code.
- Individual commands/operations are separated by new lines
- An object is returned by a function with the `return()` command, where the object to be returned appears inside the parentheses.
- Variables that are created inside the function body exist only for the lifetime of the function. This means they are not accessible outside of the function, in an R session.

Functions: returning a single value

Here's a function for calculating the coefficient of variation (the ratio of the standard deviation to the mean) for a vector:

```
coef.of.var <- function(x){
meanval <- mean(x,na.rm=TRUE) # recall this means
                              # "ignore NAs"
sdval <- sd(x,na.rm=TRUE)
return(sdval/meanval)
}
```

Translated, this function says "if you give me an object, that I will call x, I will store its mean() as meanval, then its sd() as sdval, and then return their ratio sdval/meanval."

Functions: returning multiple values

A function can return multiple objects/values by using `list()` which collects objects of (potentially) different types.

```
popn.mean.sd <- function(x){
n <- length(x)
mean.est <- mean(x,na.rm=TRUE)
var.est <- var(x,na.rm=TRUE)*(n-1)/n
est <- list(mean=mean.est, sd=sqrt(var.est))
return(est)
}
```

- The in-built `var()` applies a bias correction term of n/(n-1), which we don't want here.

Functions: declaring functions within functions

Usually, functions that take arguments, execute R commands, and return output will be enough. But functions can be declared and used inside a function.

```
square.plus.cube <- function(y) {
square <- function(x) { return(x*x) }
cube <- function(x) { return(x^3) }
return(square(y) + cube(y))
}
```

Translated, “if you given me a number, that I will call y, I will define a function I call square that takes a number that it calls x and returns x-squared, then similarly one I call cube that cubes, then I will return the sum of applying square to y and cube to y”.

Functions: some tips

- Avoid rewriting the same code... use functions!
- Modularize as much as possible: write function that call other functions. (Start with the low-level ones)
- Test your functions: use data/arguments for which you know the results to verify that your functions are working properly
- Use meaningful variable and function names
- Later on: provide documentation, including detailed comments describing the procedures being conducted by the functions, especially for large, complex programs

- Go to `https://github.com/CarinaSilva/Workshop-Linguagem-R`
- Folder: Exercises: Do exercise 7.

Figure: 3'

Saving data

To export data from R, use the command `write.table()` function. Since we have already set our working directory, R automatically saves our file into the working directory.

```
write.table(my_data, file = "name_of_the_file")
write.csv(my_data, file = "name_of_the_file.csv")
```

Reading Data from a File

The `read.table` function reads data from a file:

```
>dataframe<-read.table("C:\\ Documents\
+ plotdata.txt",header=T)
```

`header = T`: row 1 of the file contains variable names.

To use the variables you need to use the `attach`:

```
> attach(dataframe)
```

To see the names of the variables:

```
>names(dataframe)
[1] "xvalues"  "yvalues"
```

Workspace (.Rdata)

- The R workspace consists of all the data objects youve created or loaded during your R session.
- When you quit R by either typing q() or exiting out of the application window, R will prompt you to save your workspace. If you choose yes, R saves a file named .RData to your working directory. The next time you open R and reload your .Rdata workspace, all of your data objects will be available in R and all of the commands that you have typed will be accessible by using the up-arrow and down-arrow keys on your keyboard.
- You can also save or load your workspace at any time during your R session from the menu by clicking Session>Save Workspace As.., or the save button on the Environment Tab.

R history (.Rhistory)

- An R history file is a copy of all your key strokes. You can think of it as brute force way of saving your work.
- Like an R file, an Rhistory file is simply a text file that lists all of the commands that you have executed.
- It does not keep a record of the results.
- To load or save your R history from the History Tab click the Open File or Save button.
- If you load an Rhistory file, your previous commands will again become available with the up-arrow and down-arrow keys.

Bibliography

- R web site: `http://www.r-project.org/`
- Seefeld, K. and Linder, E. (2007) Statistics Using R with Biological Examples. `[http://cran.r-project.org/doc/contrib/Seefeld_StatsRBio.pdf]`