Introdução à Linguagem de Programação em R
Dia 1

Carina Silva e Ricardo Ribeiro
(*carina.silva@estesl.ipl.pt ricardo.ribeiro@estesl.ipl.pt*)
Escola Superior de Tecnologias da Saúde de Lisboa

ESCOLA SUPERIOR DE TECNOLOGIA DA SAÚDE DE LISBOA
INSTITUTO POLITÉCNICO DE LISBOA

Contents

R Basics

Introduction

Scripting

Packages

R as calculator

Contents

R Basics

- Introduction
- Scripting
- Packages
- R as calculator

Basic data types and data structures

- Vectors
- Matrices
- Arrays
- Factors
- Lists
- Dataframe

Contents

R Basics

Introduction

Scripting

Packages

R as calculator

Basic data types and data structures

Vectors

Matrices

Arrays

Factors

Lists

Dataframe

Exporting/Importing data

Contents

R Basics

Introduction

Scripting

Packages

R as calculator

Basic data types and data structures

Vectors

Matrices

Arrays

Factors

Lists

Dataframe

Exporting/Importing data

Saving a R session

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.

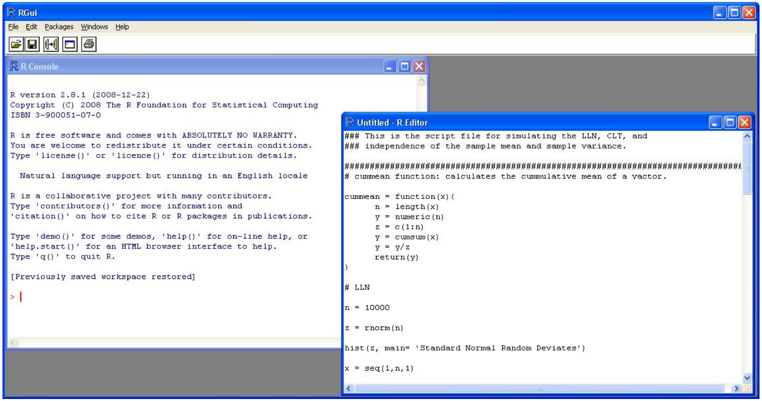
Introduction

- 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.

How to download?

- Google it using `R` or CRAN (Comprehensive R Archive Network)
- `https://www.r-project.org`

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)
```

R Interface

- Start the R system, the main window (RGui) with a sub window (R Console) will appear.
- In the "Console" window the cursor is waiting for you to type in some R commands.
- Or you can open a script file (`R Editor`) and write the R code without the prompt (>) symbol and then run-it through the keys `Ctrl+R` or through the menu `Edit`→ `run all`.

R Overview

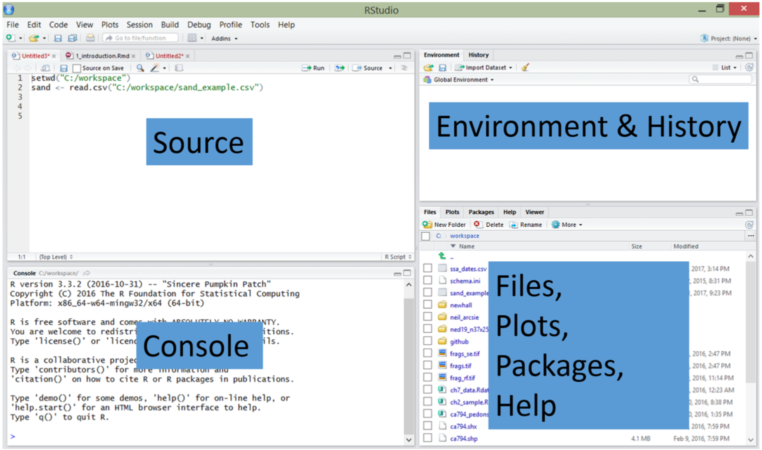
- You can enter commands one at a time at the command `prompt (>)` or run a set of commands from a source file.
- There is a wide variety of data types, including vectors (numerical, character, logical), matrices, dataframes, and lists.
- To quit R, use
  `>q()`

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.

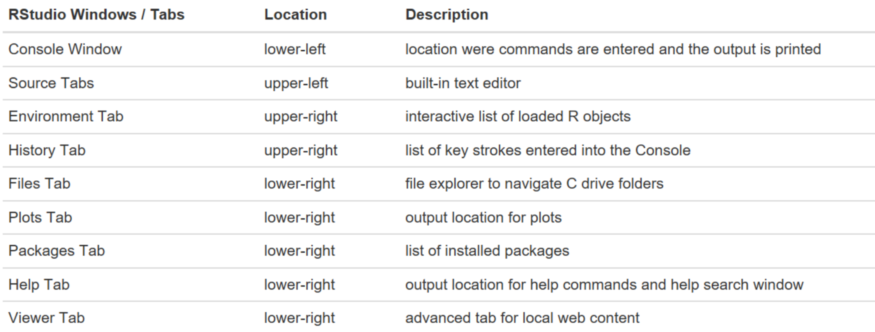
`https://www.rstudio.com/`

RStudio: An Integrated Development Environment (IDE) for R


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled3* 1_introduction.Rmd Untitled2*
Source on Save
Run
Source
Environment History
Import Dataset
Global Environment
List
Files Plots Packages Help Viewer
New Folder Delete Rename More
workspace
Name Size Modified
Source
Environment & History
Console
Files, Plots, Packages, Help

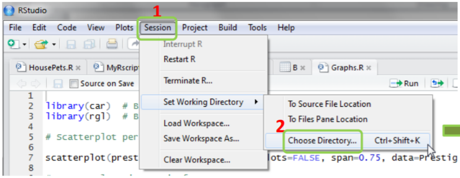
```
setwd("C:/workspace")
sand <- read.csv("C:/workspace/sand_example.csv")
```

```
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 proje
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

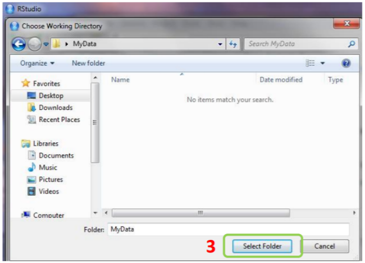
Working Directory



```
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/`

Basic Tips for using R

- R is command-line driven. It requires you to type or copy-and-paste commands after a command prompt ($>$) that appears when you open R. After typing a command in the R console and pressing Enter on your keyboard, the command will run. If your command is not complete, R issues a continuation prompt (signified by a plus sign: $+$).
- Commands in R are also called functions. The basic format of a function in R is: `function.name(argument1 = data, argument2 = TRUE)`.
- The up arrow ($\uparrow$) on your keyboard can be used to bring up previous commands that you have typed in the R console.

Basic Tips for using R

- 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 ♯.
- The $ symbol is used to select a particular column within the table (e.g., table$column).

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

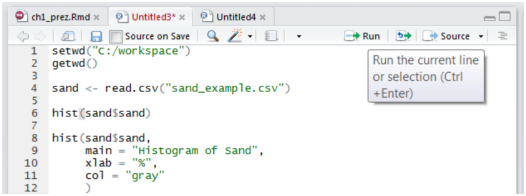
Prior to executing R functions at the Console, commands are typically written down (or scripted). Benefits include:

- 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.



```
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

- A command can be executed from the text editor by placing the cursor on a line and typing **Crtl + Enter**, or by clicking the Run button.
- An entire R script file can be executed by clicking the Source button.

R Packages

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

These libraries are created and maintained by the authors.

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()
```

Exercises

- Go to https://github.com/CarinaSilva/Introducao-Linguagem-de-Programacao-em-R/tree/main/Dia_1
- exercises_day1: Do exercise 1

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
```

Exercises

- Go to https://github.com/CarinaSilva/Introducao-Linguagem-de-Programacao-em-R/tree/main/Dia_1
- exercises_day1: Do exercise 2

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
```

Exercises

- Go to `https://github.com/CarinaSilva/Introducao-Linguagem-de-Programacao-em-R/tree/main/Dia_1`
- exercises_day1: Do exercise 3 and 4

Basic Data Types and Data Structures in R

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

- character: “a”, “brands”
- 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?

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
- matrix
- data.frame
- factors

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
```

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:

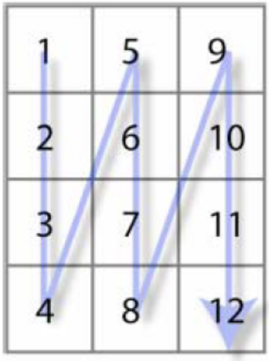
```
>rm(y)
```

Exercises

- Go to `https://github.com/CarinaSilva/Introducao-Linguagem-de-Programacao-em-R/tree/main/Dia_1`
- exercises_day1: Do exercise 5 to 9.

Creating matrices

- `m<-matrix(1:12,4,3)`



Creating Matrices

```
> Y <- matrix(0,  2,  3)
> Y
     [,1]  [,2]  [,3]
[1,]  0     0     0
[2,]  0     0     0
```

Binding columns (rows):

```
> n <- c(1,  3,  5,  7,  8,  9)
> X <- cbind(n, 1:6); X
     X
[1,]  1  1
[2,]  3  2
[3,]  5  3
[4,]  7  4
[5,]  8  5
[6,]  9  6
```

Operations on Matrices

To multiply two matrices A and B together, type:

```
> A %*% B
```

To find Eigen-Values, type:

```
> eigen(A)
```

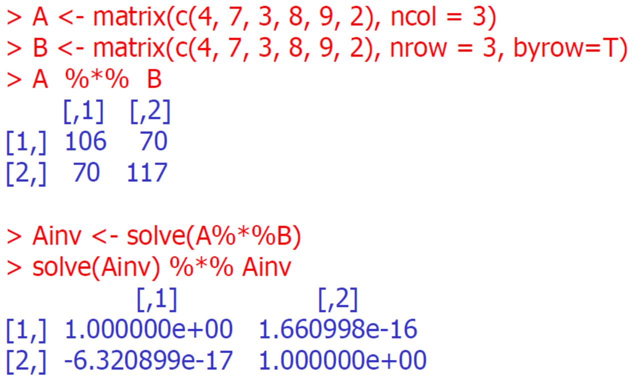
To find the transpose of a matrix, type:

```
> t(A)
```

To find the inverse of a matrix, type:

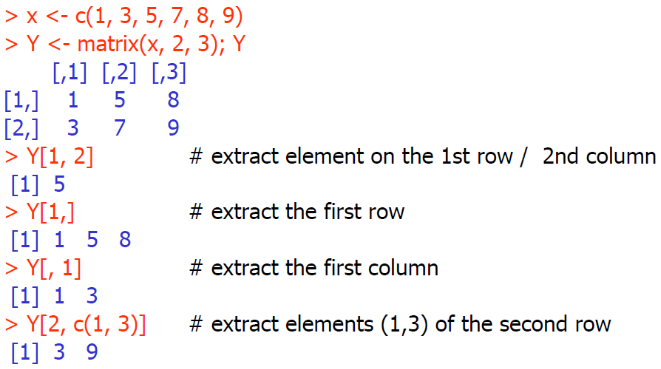
```
> solve(A)
```

Operations on Matrices



```
> A <- matrix(c(4, 7, 3, 8, 9, 2), ncol = 3)
> B <- matrix(c(4, 7, 3, 8, 9, 2), nrow = 3, byrow=T)
> A  %*%  B
     [,1]  [,2]
[1,]  106    70
[2,]   70   117

> Ainv <- solve(A%*%B)
> solve(Ainv) %*% Ainv
              [,1]           [,2]
[1,]  1.000000e+00   1.660998e-16
[2,] -6.320899e-17   1.000000e+00
```

Extraction of Elements



```
> x <- c(1, 3, 5, 7, 8, 9)
> Y <- matrix(x, 2, 3); Y
     [,1] [,2] [,3]
[1,]    1    5    8
[2,]    3    7    9
> Y[1, 2]           # extract element on the 1st row /  2nd column
 [1] 5
> Y[1,]             # extract the first row
 [1] 1  5  8
> Y[, 1]            # extract the first column
 [1] 1  3
> Y[2, c(1, 3)]     # extract elements (1,3) of the second row
 [1] 3  9
```

Contingency Tables

```
> x <- matrix(c(12, 23, 5, 6, 10, 13),
                  nrow=2,  byrow=T)
> dimnames(x) <- list(c("A",  "B"), c("x", "y", "z"))
> x

    x   y   z
A  12  23   5
B   6  10  13
```

Exercises

- Go to `https://github.com/CarinaSilva/Introducao-Linguagem-de-Programacao-em-R/tree/main/Dia_1`
- exercises_day1: Do exercise 11 to 13

Arrays

Three-dimensional array containing the numbers 1 to 30, with five rows and three columns in each two tables:

```
> A<-array(1:30,c(5,3,2))
> A
```

```
, , 1

     [,1] [,2]  [,3]
 [1,]  1    6    11
 [2,]  2    7    12
 [3,]  3    8    13
 [4,]  4    9    14
 [5,]  5    10   15

, , 2


     [,1]  [,2]  [,3]
 [1,]  16    21   26
 [2,]  17    22   27
 [3,]  18    23   28
 [4,]  19    24   29
 [5,]  20    25   30
```

The numbers enter each table column-wise, from left to right (rows, then columns then tables)

Arrays

To select columns of A (e.g. second and third):

```
> A[,2:3,]
```

Columns are the second (middle) subscript

```
, , 1

    [,1]  [,2]
 [1,]  6    11
 [2,]  7    12
 [3,]  8    13
 [4,]  9    14
 [5,]  10   15

, , 2

    [,1]  [,2]
 [1,]  21   26
 [2,]  22   27
 [3,] 23   28
 [4,]  24   29
 [5,]  25   30
```

Arrays

To select columns of A (e.g. second and third) and rows (e.g. two to four), just for the second table:

```
> A[2:4,2:3,2]
```

rows are the first, columns are the second, and table are the third subscript.

```
     [,1] [,2]
[1,]   22   27
[2,]   23   28
[3,]   24   29
```

Factors

Tell R that a variable is nominal by making it a factor.

```
# variable gender with 20 "male" entries and
# 30 "female" entries
>gender <- c(rep("male",20), rep("female", 30))
>gender <- factor(gender)
# R now treats gender as a nominal variable
>summary(gender)
>gender<-as.numeric(gender)
# stores gender as 20 1s and 30 2s and associates
# 1=female, 2=male internally (alphabetically)
>summary(gender)
```

Lists

Lists are subscribed like this `[[3]]`: list called ”cars”, with three elements: ”make”, ”capacity” and ”color”:

```
 >cars<-list(c("Toyota","Nissan","Honda"),
+ c(1500,1800,1750), c("blue","red","black","silver"))
>cars[[1]]
[1] "Toyota"  "Nissan"  "Honda"
```

```
>cars[[2]]
[1] 1500 1800 1750
```

To extract one element of the sub-list:

```
> cars[[3]][2]
[1] "red"
```

Dataframe

`R` handles data in objects known as dataframes;

rows: different observations;
columns: values of the different variables (numbers, text, calendar dates or logical variables (T or F)).

Dataframe

To see the content of the dataframe (object) just type its name.

```
>attach(CO2)
>CO2
```

Subscripts within square brackets: to select part of a dataframe.
[, means "all the rows" and **,]** means "all the columns".
To select the first three column of the dataframe CO2:

```
>CO2[,1:3]
    Plant  Type   Treatment
 1   Qn1   Quebec   nonchilled
 2   Qn1   Quebec   nonchilled
 3   Qn1   Quebec   nonchilled
 4   Qn1   Quebec   nonchilled
 5   Qn1   Quebec   nonchilled
(...)
```

To select certain rows based on logical tests on the values of one or more variables:

```
> CO2[conc>95&uptake<28,]
     Plant  Type Treatment  conc  uptake
 9  Qn2  Quebec  nonchilled  175  27.3
 23  Qc1  Quebec  chilled  175  24.1
 30  Qc2  Quebec  chilled  175  27.3
 37  Qc3  Quebec  chilled  175  21.0
 44  Mn1  Mississippi  nonchilled  175  19.2
 45  Mn1  Mississippi  nonchilled  250  26.2
 51  Mn2  Mississippi  nonchilled  175  22.0
(...)
```

You can sort the rows or the columns in any way you choose but you need to state which column you want to be sorted (i.e. all of them for CO2 1:3)
e.g. Select all rows of the variables "Plant" and "Type" (columns one and two) sorted by "uptake" (this is the variable in column number five [,5]):

```
> CO2[order(CO2[,5]),1:2]
     Plant  Type
 71  Mc2   Mississippi
 29  Qc2   Quebec
 64  Mc1   Mississippi
 43  Mn1   Mississippi
 78  Mc3   Mississippi
(...)
```

Alternatively the dataframe can be sorted in descending order by "conc", with only "Treatment" and "uptake" as output:

```
> CO2[rev(order(CO2[,4])),c(3,5)]
    Treatment  uptake
 84  chilled    19.9
 77  chilled    14.4
 70  chilled    21.9
 63  nonchilled   27.8
 56  nonchilled   31.5
(...)
```

Useful functions

length(object)	number of elements or components
str(object)	structure of an object
class(object)	n class or type of an object
names(object)	names
c(object,object,...)	combine objects into a vector
cbind(object, object, ...)	combine objects as columns
rbind(object, object, ...)	combine objects as rows
ls()	list current objects
rm(object)	delete an object
newobject <- edit(object)	edit copy and save a newobject
fix(object)	edit in place height

Exercises

- Go to `https://github.com/CarinaSilva/Introducao-Linguagem-de-Programacao-em-R/tree/main/Dia_1`
- exercises_day1: Do exercise 14 to 27

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(sand, file = "sand_example1")
- write.csv(sand, file = "sand_example.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"
```

Exercises

- Go to `https://github.com/CarinaSilva/Introducao-Linguagem-de-Programacao-em-R/tree/main/Dia_1`
- exercises_day1: Do exercise 18 to 20

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.

Exercises

- Go to `https://github.com/CarinaSilva/Introducao-Linguagem-de-Programacao-em-R/tree/main/Dia_1`
- Do exercises from file “proposed_exercises_day1”

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]