

Introduction to R

Lecture 1

Louis SIRUGUE

CPES 2 - Fall 2022



Welcome to the course

About me

- PhD student at the Paris School of Economics
- I work primarily on:
 - Intergenerational (income) mobility
 - Residential segregation
 - Discrimination
- I do empirical research, so I use Econometrics and (R) programming on a daily basis
- You can reach me at louis.sirugue@psemail.eu for any question or comment about the course

About the course

- **Objective:**
 - Give you the necessary statistical and data visualization tools to perform data analyses
- **Prerequisites:**
 - None (middle-school maths level)
- **What you'll learn:**
 - Find and manipulate data
 - Summarize data with relevant statistics and compelling graphics
 - Carry out an empirical research project
- **Thus, this course is a mix of:**
 - Programming on R
 - Basic Statistics
 - Introductory Econometrics



First semester

Course format

- The course is divided into **2 parts** that are **structured** the same way:
 -
 -
 -
 -

Part I: Introduction to R programming

Part II: Introduction to Econometrics



First semester

Course format

- The course is divided into **2 parts** that are **structured** the same way:
 - 3 core **lectures with** weekly online **quizzes**
 -
 -
 -

Part I: Introduction to R programming

Lecture 1 Introduction to R*

Lecture 2 Descriptive statistics*

Lecture 3 Basic data manipulation*

Part II: Introduction to Econometrics

Lecture 8 Univariate regressions*

Lecture 9 Multivariate regressions*

Lecture 10 Inference*



First semester

Course format

- The course is divided into **2 parts** that are **structured** the same way:
 - 3 core **lectures with** weekly online **quizzes**
 - 2-3 core **lectures without quizzes**
 -
 -

Part I: Introduction to R programming

Lecture 1 Introduction to R*

Lecture 2 Descriptive statistics*

Lecture 3 Basic data manipulation*

Lecture 4 Data visualization

Lecture 5 R Markdown & LaTeX

Part II: Introduction to Econometrics

Lecture 8 Univariate regressions*

Lecture 9 Multivariate regressions*

Lecture 10 Inference*

Lecture 11 Causality

Lecture 12 Interpretation

Lecture 13 Applications in academic research



First semester

Course format

- The course is divided into **2 parts** that are **structured** the same way:
 - 3 core **lectures with** weekly online **quizzes**
 - 2-3 core **lectures without quizzes**
 - 1 buffer **lecture** that will **not** be **in the exams**
 -

Part I: Introduction to R programming

Lecture 1 Introduction to R*

Lecture 2 Descriptive statistics*

Lecture 3 Basic data manipulation*

Lecture 4 Data visualization

Lecture 5 R Markdown & LaTeX

Lecture 6 Text data & sentiment analysis

Part II: Introduction to Econometrics

Lecture 8 Univariate regressions*

Lecture 9 Multivariate regressions*

Lecture 10 Inference*

Lecture 11 Causality

Lecture 12 Interpretation

Lecture 13 Applications in academic research

Lecture 14 Maps and geolocalized data



First semester

Course format

- The course is divided into **2 parts** that are **structured** the same way:
 - 3 core **lectures with** weekly online **quizzes**
 - 2-3 core **lectures without quizzes**
 - 1 buffer **lecture** that will **not** be **in the exams**
 - 1 **exam** (homework/final exam)

Part I: Introduction to R programming

Lecture 1 Introduction to R*

Lecture 2 Descriptive statistics*

Lecture 3 Basic data manipulation*

Lecture 4 Data visualization

Lecture 5 R Markdown & LaTeX

Lecture 6 Text data & sentiment analysis

Lecture 7 Homework correction

Part II: Introduction to Econometrics

Lecture 8 Univariate regressions*

Lecture 9 Multivariate regressions*

Lecture 10 Inference*

Lecture 11 Causality

Lecture 12 Interpretation

Lecture 13 Applications in academic research

Lecture 14 Maps and geolocalized data

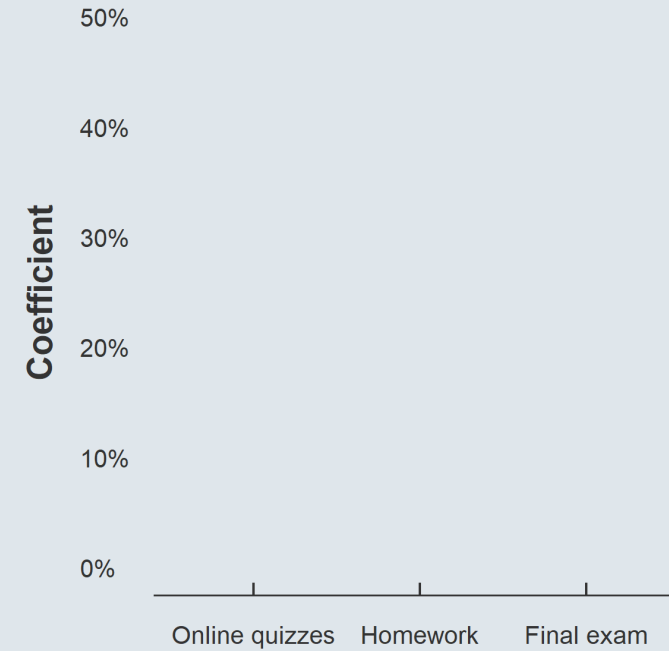
Lecture 15 Final Exam



First semester

Grading:

- The exam **coefficients** are:
 -
 -
 -

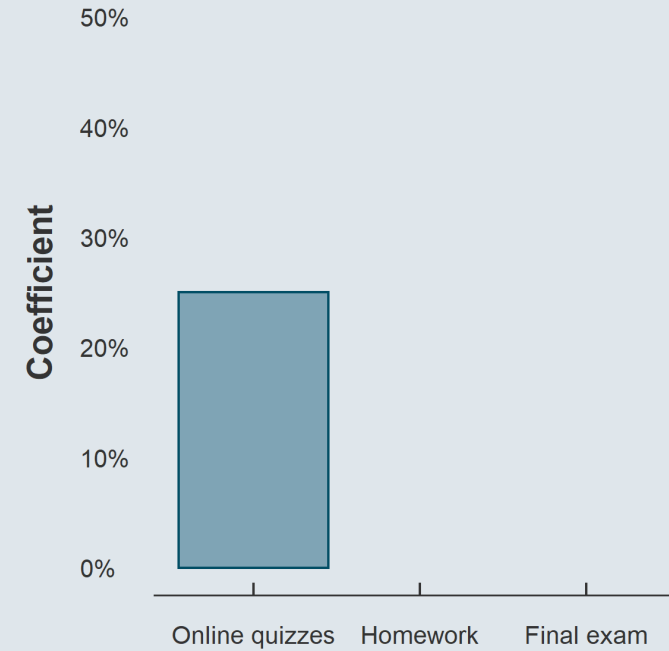




First semester

Grading:

- The exam **coefficients** are:
 - Online quizzes: 25%
 -
 -

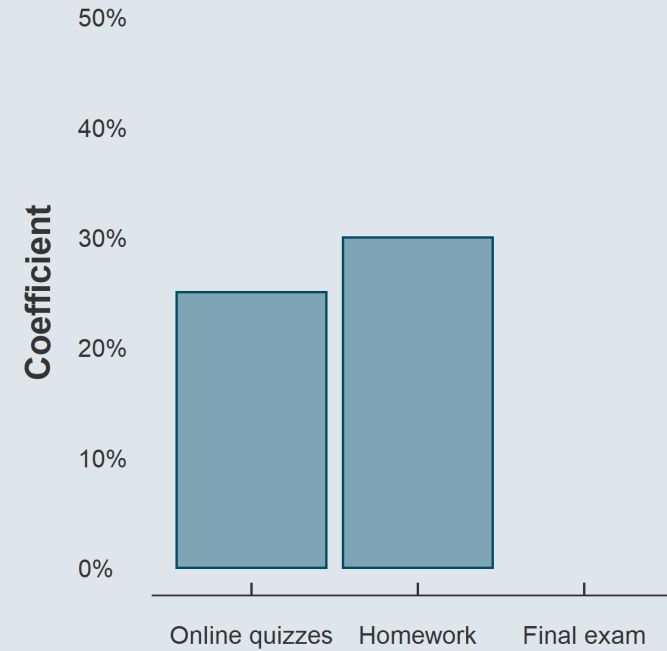




First semester

Grading:

- The exam **coefficients** are:
 - Online quizzes: 25%
 - Homework: 30%
 -

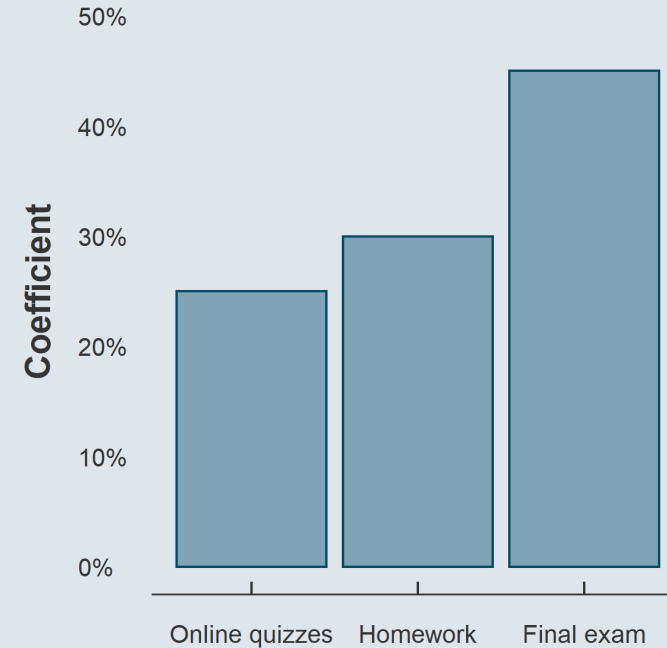




First semester

Grading:

- The exam **coefficients** are:
 - Online quizzes: 25%
 - Homework: 30%
 - Final exam: 45%



- **Online quizzes:**
 - Available in the *Material* section of the course webpage
 - Log in using the verification code I sent you via email
 - **3-5 shorts questions** on to the content of the previous class
 - Quizzes open at the end of the lecture and **close at the beginning of the next one**
 - You can retry the quiz as many times as you want before submitting
 - To submit click on *download results* and **send the downloaded file** to me by email



First semester

Grading:

- **Homework:**

- **Set of exercises** related to the first five lectures of the course
- It can be done **alone or by pairs**
- Must be handed over via email by the end of the sixth week (**16th of October 23:59**)
- **Late submissions will be penalized** by half a point for each 30min beyond the deadline
- I encourage you to **work on it progressively** as we go through the course
- You can help each other but write your answers yourselves (**no copy-pasting**)

- **Final exam:**

- **Paper exam in classroom**
- You can **bring a cheatsheet** with you as long as:
 - It is **handwritten** on paper
 - It stands on a single A4 (21cm x 29.7cm) page, i.e., **recto only**
- Cheatsheets that do not comply with these rules will be confiscated
- Beyond that, standard examination rules apply



Second semester

Course format

- The **second semester** will be all about carrying out your **own research project**:
 -
 -
 -



Second semester

Course format

- The **second semester** will be all about carrying out your **own research project**:
 - **Guidelines and refreshers** will give you time find data and a research question
 -
 -

Lecture 1 How to conduct a research project

Lecture 2 Refresher: R Programming

Lecture 3 Refresher: Econometrics



Second semester

Course format

- The **second semester** will be all about carrying out your **own research project**:
 - **Guidelines and refreshers** will give you time find data and a research question
 - The rest of the semester will be made of **follow-up sessions**
 -

Lecture 1 How to conduct a research project

Lecture 2 Refresher: R Programming

Lecture 3 Refresher: Econometrics

Lecture 4 Presentation of question and data

Lecture 5 Follow-up: Data cleaning I

Lecture 6 Follow-up: Data cleaning II

Lecture 7 Follow-up: Descriptive statistics

Lecture 8 Follow-up: Visualizing the data

Lecture 9 Follow-up: Regression analysis

Lecture 10 Follow-up: Midterm report feedback

Lecture 11 Follow-up: Causality assessment

Lecture 12 Follow-up: Robustness

Lecture 13 Follow-up: Heterogeneity

Lecture 14 Follow-up: Last tips

Lecture 15 Final presentation



Second semester

Course format

- The **second semester** will be all about carrying out your **own research project**:
 - **Guidelines and refreshers** will give you time find data and a research question
 - The rest of the semester will be made of **follow-up sessions**
 - **Exams** will take place at **3 stages** of the research process

Lecture 1 How to conduct a research project

Lecture 2 Refresher: R Programming

Lecture 3 Refresher: Econometrics

Lecture 4 Presentation of question and data

Lecture 5 Follow-up: Data cleaning I

Lecture 6 Follow-up: Data cleaning II

Lecture 7 Follow-up: Descriptive statistics

Lecture 8 Follow-up: Visualizing the data

Lecture 9 Follow-up: Regression analysis

Lecture 10 Follow-up: Midterm report feedback

Lecture 11 Follow-up: Causality assessment

Lecture 12 Follow-up: Robustness

Lecture 13 Follow-up: Heterogeneity

Lecture 14 Follow-up: Last tips

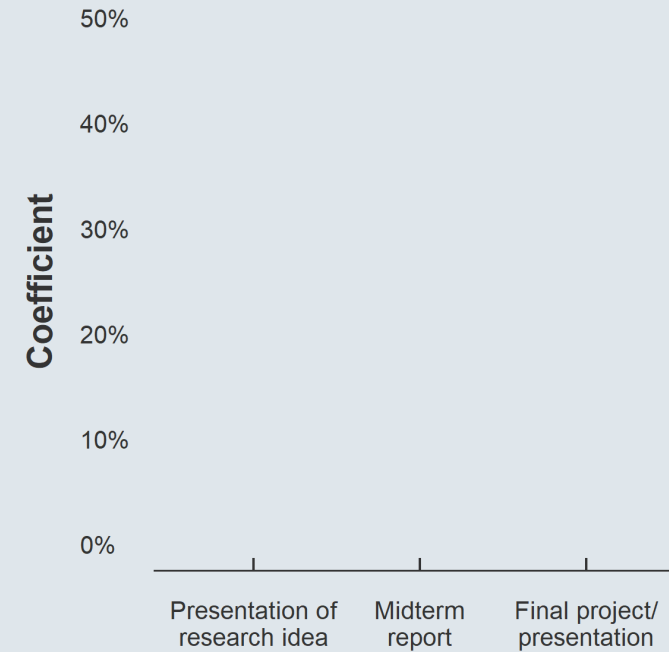
Lecture 15 Final presentation



Second semester

Grading:

- The exam **coefficients** are:
 -
 -
 -

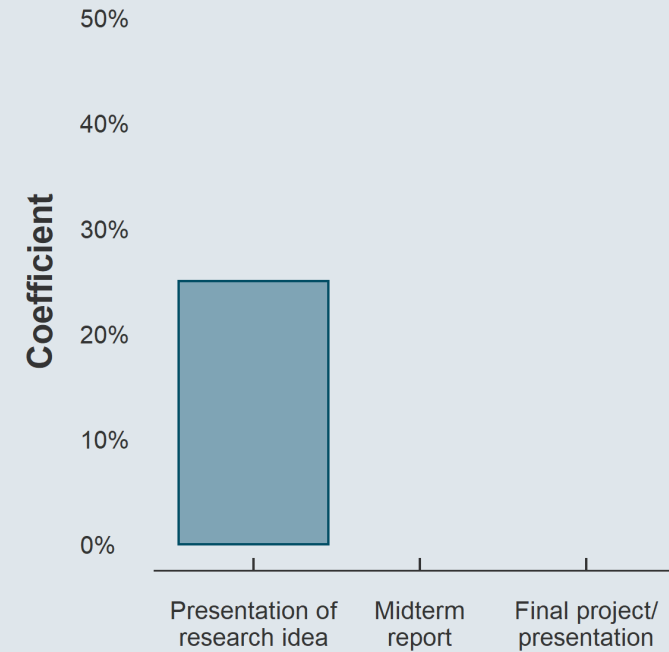




Second semester

Grading:

- The exam **coefficients** are:
 - Presentation of research idea: 25%
 -
 -

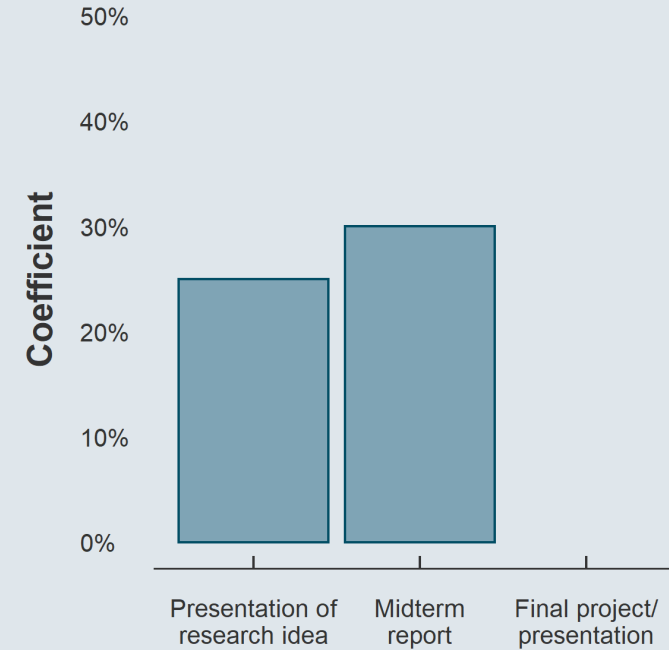




Second semester

Grading:

- The exam **coefficients** are:
 - Presentation of research idea: 25%
 - Midterm report: 30%
 -

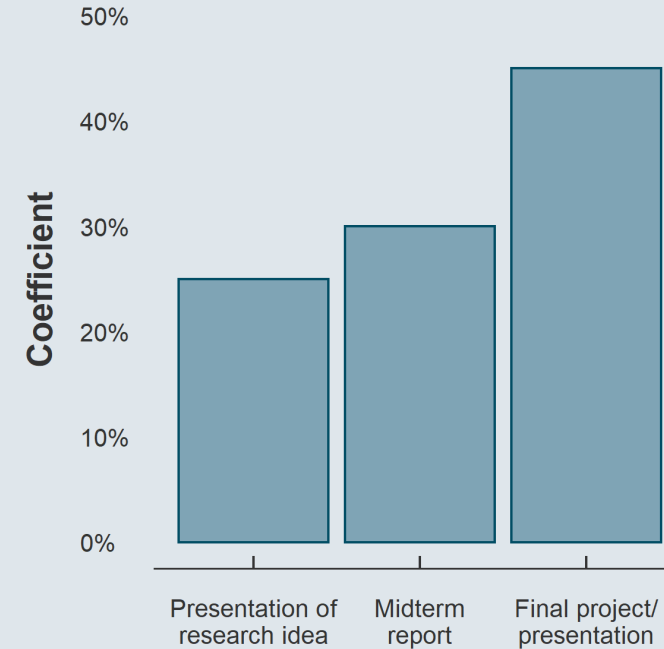




Second semester

Grading:

- The exam **coefficients** are:
 - Presentation of research idea: 25%
 - Midterm report: 30%
 - Final project/presentation: 45%
- **Additional information:**
 - Research projects must be done **by pairs**
 - It can be written in **French or in English**
 - Follow-ups are **group meetings** with a T.A.
 - Find an **example** of what is expected [here](#)
 - We'll come back to that in due time (January)



→ Check out the [course webpage](#) for more details, all the information is there!



Let's delve into it!

1. Getting started

- 1.1. About R
- 1.2. The R Studio IDE
- 1.3. Import and eyeball data
- 1.4. Use functions

2. Anatomy of a data.frame

- 2.1. Data structure
- 2.2. Classes
- 2.3. Vectors
- 2.4. Subsetting

3. Wrap up!



Let's delve into it!

1. Getting started

1.1. About R

1.2. The R Studio IDE

1.3. Import and eyeball data

1.4. Use functions



1. Getting started

1.1. About R

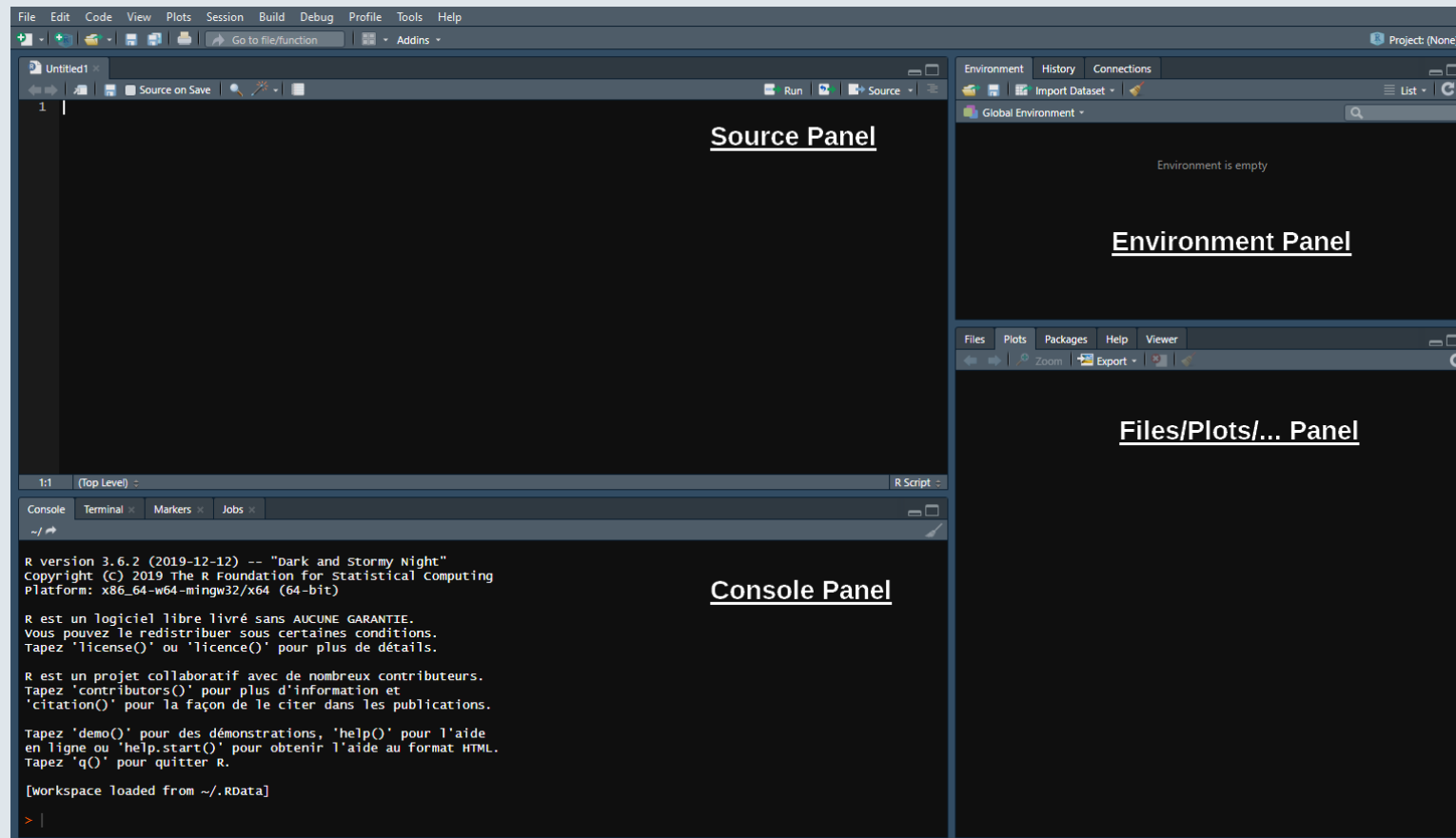
- R is a **programming language** and free software environment for **statistical computing and graphics**
- The R language is widely (and increasingly) used in **academic and non-academic research** in fields like:
 - Economics
 - Statistics
 - Biostats
- Things you can do with R:
 - Reports
 - Nice plots
 - All the material of this course
 - Academic research
 - Win kaggle competitions
 - Interactive data visualization
 - Art





1. Getting started

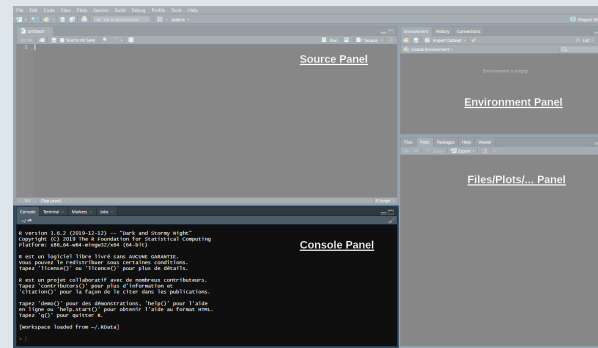
1.2. The R Studio IDE





1. Getting started

1.2. The R Studio IDE



→ The Console panel

- This is where you **communicate with R**
 - You can write instructions after the **>**, **press enter** and R will **execute**
 - Try with **1+1**:

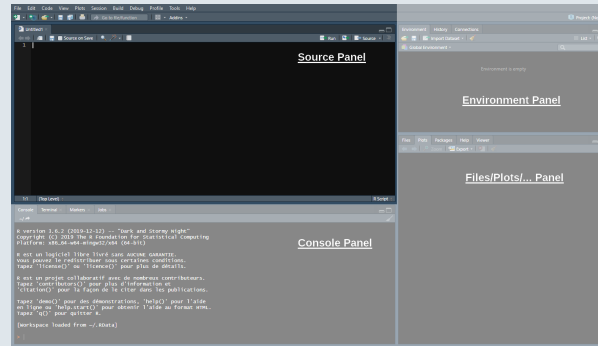
```
1+1
```

```
## [1] 2
```



1. Getting started

1.2. The R Studio IDE



→ The Source panel

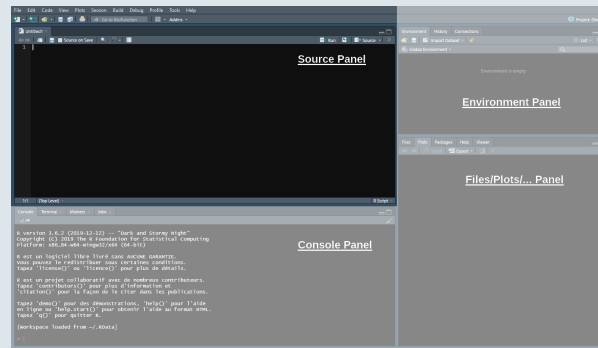
- This is where you **write and save your code** (File > New File > R Script)
 - **Separate** different commands with a **line break**
 - The **#** symbol allows to **comment** your code
 - Everything after **#** will be **ignored** by R until the next line break

```
1+1 # Do not put 2+2 on the same line, press enter to go to next line
2+2
```



1. Getting started

1.2. The R Studio IDE



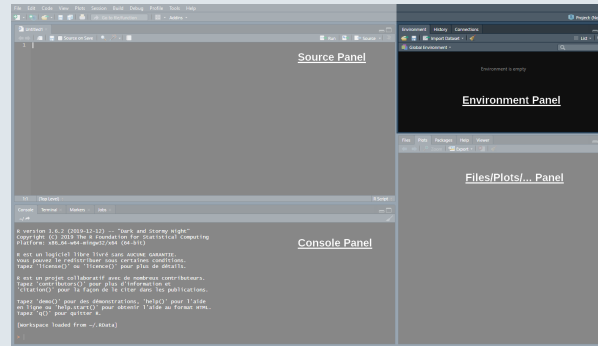
→ The Source panel

- To send the command from the source panel to the console panel:
 1. **Highlight** the lines you want to execute
 2. Press **ctrl + enter**
- If you do not highlight anything the line of code where your cursor stands will be executed
- Check the console to see the output of your code



1. Getting started

1.2. The R Studio IDE



→ The Environment panel

- Data analysis requires manipulating datasets, vectors, functions, etc.
 - These **elements are stored in the environment** panel
- For instance we can assign a value to an object using `<-`

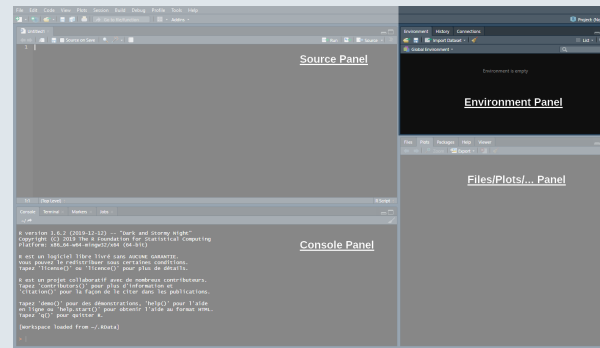
```
x <- 1
```

→ You now have an object called 'x' in your environment, which takes the value 1



1. Getting started

1.2. The R Studio IDE



→ The Environment panel

- Now that the object `x` is stored in your environment, you can use it:
- You can also modify that object at any point:

```
x + 1
```

```
## [1] 2
```

```
x <- x + 1
```

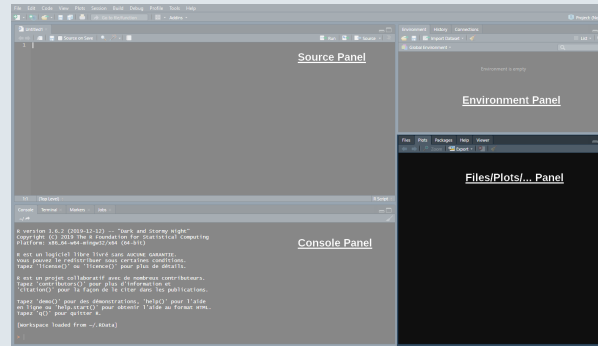
```
x
```

```
## [1] 2
```



1. Getting started

1.2. The R Studio IDE



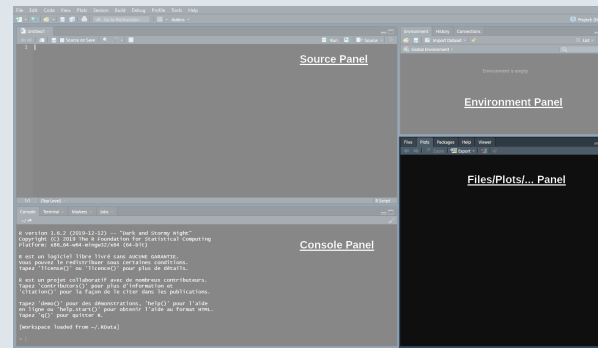
→ The Files/Plots/... panel

- In this panel we'll mainly be interested in the following 4 tabs
 - **Files:** Shows your working directory
 - **Plots:** Where R returns plots
 - **Packages:** A library of tools that we can load if needed
 - **Help:** Where to look for documentation on R functions



1. Getting started

1.2. The R Studio IDE



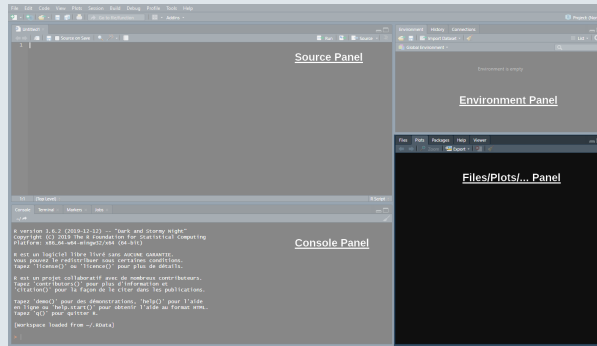
→ The Files/Plots/... panel

- Enter `?getwd()` in the console to see what a **help file** looks like



1. Getting started

1.2. The R Studio IDE



→ The Files/Plots/... panel

- Enter `?getwd()` in the console to see what a **help file** looks like
 - It **describes** what the command does
 - It **explains** the different parameters of the command
 - It **gives examples** of how to use the command

R: Get or Set Working DirectoryFind in Topic

R Documentation

getwd {base}

Get or Set Working Directory

Description

getwd returns an absolute filepath representing the current working directory of the R process; setwd(dir) is used to set the working directory to dir.

Usage

getwd()
setwd(dir)

Arguments

dir A character string: tilde expansion will be done.

Details

See [files](#) for how file paths with marked encodings are interpreted.

Value

getwd returns a character string or NULL if the working directory is not available. On Windows the path returned will use / as the path separator and be encoded in UTF-8. The path will not have a trailing / unless it is the root directory (of a drive or share on Windows).

setwd returns the current directory before the change, invisibly and with the same conventions as getwd. It will give an error if it does not succeed (including if it is not implemented).

Note

Note that the return value is said to be an absolute filepath: there can be more than one representation of the path to a directory and on some OSes the value returned can differ after changing directories and changing back to the same directory (for example if symbolic links have been traversed).

See Also

[list.files](#) for the contents of a directory.
[normalizePath](#) for a 'canonical' path name.

Examples

(WD <- getwd())
if (!is.null(WD)) setwd(WD)

[Package base version 4.0.2 Index]

Practice

03:00

1) Open a new R script (**Ctrl + Shift + N**) and write a code to create these objects:

Objects to create			
Object name:	a	b	c
Assigned value:	2	4	5

2) Run this code and create a new object named **result** that takes the value $\frac{b \times c}{a} + (b - a)^c$

Basic operations in R						
Operation:	Addition	Subtraction	Multiplication	Division	Exponentiation	Parentheses
Symbol in R:	+	-	*	/	^	()

3) Print **result** in your console and save your script somewhere in your computer (**Ctrl+S**)

You've got 3 minutes!

Solution

1) Open a new R script (**Ctrl + Shift + N**) and write a code to create these objects:

Objects to create			
Object name:	a	b	c
Assigned value:	2	4	5

```
a <- 2  
b <- 4  
c <- 5
```

2) Run this code and create a new object named **result** that takes the value $\frac{b \times c}{a} + (b - a)^c$

```
result <- b*c/a + (b-a)^c
```

3) Print **result** in your console and save your script somewhere in your computer (**Ctrl + S**)

```
result
```

```
## [1] 42
```



1. Getting started

1.3. Import and eyeball data

- We now know how to **use R** as a calculator, but our goal is **to analyze data!**

→ Take for instance the statistics from the last season of Ligue 1 available at fbref.com

Scores & Fixtures 2021-2022 Ligue 1													
Share & Export ▼ Glossary													
All Rounds Regular Season French 1/2 Relegation/Promotion Play-offs													
Wk	Day	Date	Time	Home	xG	Score	xG	Away	Attendance	Venue	Referee	Match Report	Notes
1	Fri	2021-08-06	21:00	Monaco	2.0	1-1	0.3	Nantes	7,500	Stade Louis II.	Antony Gautier	Match Report	
	Sat	2021-08-07	17:00	Lyon	1.4	1-1	0.8	Brest	29,018	Groupama Stadium	Mikael Lesage	Match Report	
			21:00	Troyes	0.8	1-2	1.2	Paris S-G	15,248	Stade de l'Aube	Amaury Delerue	Match Report	
	Sun	2021-08-08	13:00	Rennes	0.6	1-1	2.0	Lens	22,567	Roazhon Park	Bastien Dechepy	Match Report	
			15:00	Bordeaux	0.7	0-2	3.3	Clermont Foot	18,748	Stade Matmut-Atlantique	Florent Batta	Match Report	
			15:00	Strasbourg	0.4	0-2	0.9	Angers	23,250	Stade de la Meinau	Jeremy Stinat	Match Report	
			15:00	Nice	0.8	0-0	0.2	Reims	18,030	Stade de Nice	Johan Hamel	Match Report	
			15:00	Saint-Étienne	2.1	1-1	1.3	Lorient	20,461	Stade Geoffroy-Guichard	Thomas Léonard	Match Report	
			17:00	Metz	0.7	3-3	1.4	Lille	15,551	Stade Saint-Symphorien	Eric Wattellier	Match Report	
			20:45	Montpellier	0.5	2-3	2.0	Marseille	13,500	Stade de la Mosson	Jérémie Pignard	Match Report	
2	Fri	2021-08-13	21:00	Lorient	0.8	1-0	0.9	Monaco	12,149	Stade Yves Allainmat - Le Moustoir	Hakim Ben El Hadj Salem	Match Report	
	Sat	2021-08-14	17:00	Lille	1.0	0-4	2.2	Nice	30,144	Stade Pierre-Mauroy	Jérôme Brisard	Match Report	
			21:00	Paris S-G	2.5	4-2	0.6	Strasbourg	46,962	Parc des Princes	Willy Delajod	Match Report	
	Sun	2021-08-15	13:00	Angers	1.4	3-0	0.9	Lyon	6,154	Stade Raymond Kopa	Clément Turpin	Match Report	
			15:00	Clermont Foot	2.3	2-0	0.5	Troyes	11,005	Stade Gabriel Montpied	Romain Lissorgue	Match Report	
			15:00	Brest	1.5	1-1	0.9	Rennes	14,271	Stade Francis-Le Blé	Benoît Bastien	Match Report	



1. Getting started

1.3. Import and eyeball data

- You can **download** this dataset [here](#) or from the course webpage
 - Note that the extension of the file is **.csv** (for **Comma Separated Values**)
 - Let's have a look at **first 5 lines** of the raw csv file

```
Wk,Day,Date,Time,Home,xG,Score,xG,Away,Attendance,Venue,Referee,Match Report,Notes
1,Fri,2021-08-06,21:00,Monaco,2.0,1-1,0.3,Nantes,7500,Stade Louis II.,Antony Gautier,Match Report,
1,Sat,2021-08-07,17:00,Lyon,1.4,1-1,0.8,Brest,29018,Groupama Stadium,Mikael Lesage,Match Report,
1,Sat,2021-08-07,21:00,Troyes,0.8,1-2,1.2,Paris S-G,15248,Stade de l'Aube,Amaury Delerue,Match Report,
1,Sun,2021-08-08,13:00,Rennes,0.6,1-1,2.0,Lens,22567,Roazhon Park,Bastien Dechepy,Match Report,
```

- The **.csv format** is very common and has a very **codified structure**
 - We can see that **each line** corresponds to a **row** (the first row generally contains column names)
 - And for each row the **values** of each column are **separated by commas**

→ But how to get it in our R studio environment?



1. Getting started

1.3. Import and eyeball data

- To **import** stuff in R we use **read functions**
 - They take the **file directory** as an **input**
 - And give the **file content** as an **output**
- The read function dedicated to .csv files is **read.csv()**

```
function(input)
```

```
## [1] "output"
```

```
fb <- read.csv("C:\User\Documents\ligue1.csv")
```

```
## Error: '\U' used without hex digits in character string starting ""C:\U"
```

Oops, slashes must be the other way around!

```
fb <- read.csv("C:/User/Documents/ligue1.csv")
```

→ Let's **inspect** this new object to check that it worked



1. Getting started

1.3. Import and eyeball data

- The first thing we can do is to use `head()` to print the **top rows**

```
head(fb, 4)
```

##	Wk	Day	Date	Time	Home	xG	Score	xG.1	Away	Attendance
## 1	1	Fri	2021-08-06	21:00	Monaco	2.0	1-1	0.3	Nantes	7500
## 2	1	Sat	2021-08-07	17:00	Lyon	1.4	1-1	0.8	Brest	29018
## 3	1	Sat	2021-08-07	21:00	Troyes	0.8	1-2	1.2	Paris S-G	15248
## 4	1	Sun	2021-08-08	13:00	Rennes	0.6	1-1	2.0	Lens	22567
##			Venue		Referee	Match.Report	Notes			
## 1			Stade Louis II.		Antony Gautier	Match Report	NA			
## 2			Groupama Stadium		Mikael Lesage	Match Report	NA			
## 3			Stade de l'Aube		Amaury Delerue	Match Report	NA			
## 4			Roazhon Park		Bastien Dechepy	Match Report	NA			

- `tail()` would print the **bottom rows**
- We can also run `View(fb)` (a new tab will pop-up in your Source panel)



1. Getting started

1.3. Import and eyeball data

	Wk	Day	Date	Time	Home	xG	Score	xG.1	Away	Attendance	Venue	Referee	Match.Report	Notes
1	1	Fri	2021-08-06	21:00	Monaco	2.0	1-1	0.3	Nantes	7500	Stade Louis II.	Antony Gautier	Match Report	NA
2	1	Sat	2021-08-07	17:00	Lyon	1.4	1-1	0.8	Brest	29018	Groupama Stadium	Mikael Lesage	Match Report	NA
3	1	Sat	2021-08-07	21:00	Troyes	0.8	1-2	1.2	Paris S-G	15248	Stade de l'Aube	Amaury Delerue	Match Report	NA
4	1	Sun	2021-08-08	13:00	Rennes	0.6	1-1	2.0	Lens	22567	Roazhon Park	Bastien Dechepy	Match Report	NA
5	1	Sun	2021-08-08	15:00	Bordeaux	0.7	0-2	3.3	Clermont Foot	18748	Stade Matmut-Atlantique	Florent Batta	Match Report	NA
6	1	Sun	2021-08-08	15:00	Strasbourg	0.4	0-2	0.9	Angers	23250	Stade de la Meinau	Jeremy Stinat	Match Report	NA
7	1	Sun	2021-08-08	15:00	Nice	0.8	0-0	0.2	Reims	18030	Stade de Nice	Johan Hamel	Match Report	NA
8	1	Sun	2021-08-08	15:00	Saint-Étienne	2.1	1-1	1.3	Lorient	20461	Stade Geoffroy-Guichard	Thomas L'écuyer	Match Report	NA
9	1	Sun	2021-08-08	17:00	Metz	0.7	3-3	1.4	Lille	15551	Stade Saint-Symphorien	Eric Wattellier	Match Report	NA
10	1	Sun	2021-08-08	20:45	Montpellier	0.5	2-3	2.0	Marseille	13500	Stade de la Mosson	Jérémy Pignard	Match Report	NA
11	2	Fri	2021-08-13	21:00	Lorient	0.8	1-0	0.9	Monaco	12149	Stade Yves Allainmat - Le Moustoir	Hakim Ben El Hadj Salem	Match Report	NA
12	2	Sat	2021-08-14	17:00	Lille	1.0	0-4	2.2	Nice	30144	Stade Pierre-Mauroy	Jérémy Brisard	Match Report	NA
13	2	Sat	2021-08-14	21:00	Paris S-G	2.5	4-2	0.6	Strasbourg	46962	Parc des Princes	Willy Delajod	Match Report	NA
14	2	Sun	2021-08-15	13:00	Angers	1.4	3-0	0.9	Lyon	6154	Stade Raymond Kopa	Clément Turpin	Match Report	NA
15	2	Sun	2021-08-15	15:00	Clermont Foot	2.3	2-0	0.5	Troyes	11005	Stade Gabriel Montpied	Romain Lissorgue	Match Report	NA
16	2	Sun	2021-08-15	15:00	Brest	1.5	1-1	0.9	Rennes	14271	Stade Francis-Le Blé	Benoît Bastien	Match Report	NA
17	2	Sun	2021-08-15	15:00	Nantes	1.4	2-0	1.1	Metz	12054	Stade de la Beaujoire - Louis Fonteneau	Johan Hamel	Match Report	NA

Seems like it worked!



1. Getting started

1.3. Import and eyeball data

	Wk	Day	Date	Time	Home	xG	Score	xG.1	Away	Attendance	Venue	Referee	Match.Report	Notes
1	1	Fri	2021-08-06	21:00	Monaco	2.0	1-1	0.3	Nantes	7500	Stade Louis II.	Antony Gautier	Match Report	NA
2	1	Sat	2021-08-07	17:00	Lyon	1.4	1-1	0.8	Brest	29018	Groupama Stadium	Mikael Lesage	Match Report	NA
3	1	Sat	2021-08-07	21:00	Troyes	0.8	1-2	1.2	Paris S-G	15248	Stade de l'Aube	Amaury Delerue	Match Report	NA
4	1	Sun	2021-08-08	13:00	Rennes	0.6	1-1	2.0	Lens	22567	Roazhon Park	Bastien Dechepy	Match Report	NA
5	1	Sun	2021-08-08	15:00	Bordeaux	0.7	0-2	3.3	Clermont Foot	18748	Stade Matmut-Atlantique	Florent Batta	Match Report	NA
6	1	Sun	2021-08-08	15:00	Strasbourg	0.4	0-2	0.9	Angers	23250	Stade de la Meinau	Jeremy Stinat	Match Report	NA
7	1	Sun	2021-08-08	15:00	Nice	0.8	0-0	0.2	Reims	18030	Stade de Nice	Johan Hamel	Match Report	NA
8	1	Sun	2021-08-08	15:00	Saint-Étienne	2.1	1-1	1.3	Lorient	20461	Stade Geoffroy-Guichard	Thomas L'écuyer	Match Report	NA
9	1	Sun	2021-08-08	17:00	Metz	0.7	3-3	1.4	Lille	15551	Stade Saint-Symphorien	Eric Wattellier	Match Report	NA
10	1	Sun	2021-08-08	20:45	Montpellier	0.5	2-3	2.0	Marseille	13500	Stade de la Mosson	Jérémy Pignard	Match Report	NA
11	2	Fri	2021-08-13	21:00	Lorient	0.8	1-0	0.9	Monaco	12149	Stade Yves Allainmat - Le Moustoir	Hakim Ben El Hadj Salem	Match Report	NA
12	2	Sat	2021-08-14	17:00	Lille	1.0	0-4	2.2	Nice	30144	Stade Pierre-Mauroy	Jérémy Brisard	Match Report	NA
13	2	Sat	2021-08-14	21:00	Paris S-G	2.5	4-2	0.6	Strasbourg	46962	Parc des Princes	Willy Delajod	Match Report	NA
14	2	Sun	2021-08-15	13:00	Angers	1.4	3-0	0.9	Lyon	6154	Stade Raymond Kopa	Clément Turpin	Match Report	NA
15	2	Sun	2021-08-15	15:00	Clermont Foot	2.3	2-0	0.5	Troyes	11005	Stade Gabriel Montpied	Romain Lissorgue	Match Report	NA
16	2	Sun	2021-08-15	15:00	Brest	1.5	1-1	0.9	Rennes	14271	Stade Francis-Le Blé	Benoît Bastien	Match Report	NA
17	2	Sun	2021-08-15	15:00	Nantes	1.4	2-0	1.1	Metz	12054	Stade de la Beaujoire - Louis Fonteneau	Johan Hamel	Match Report	NA

... or kind of worked?



1. Getting started

1.4. Use functions

- That kind of **weird characters** kicks in when there is an **encoding issue**
 - Thankfully, **read.csv()** can take many **other inputs**, including encoding!
 - Usually the UTF-8 encoding is the solution to French characters

```
fb <- read.csv("C:/User/Documents/ligue1.csv", encoding = "UTF-8")
```

- When you will be facing **similar issues**, check out the arguments of read.csv() using **?read.csv**

Arguments	
file	<p>the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an <i>absolute</i> path, the file name is <i>relative</i> to the current working directory, <code>getwd()</code>. Tilde-expansion is performed where supported. This can be a compressed file (see file).</p> <p>Alternatively, <i>file</i> can be a readable text-mode connection (which will be opened for reading if necessary, and if so <code>closed</code> (and hence destroyed) at the end of the function call). (If <code>stdin()</code> is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, Ctrl-D on Unix and Ctrl-Z on Windows. Any pushback on <code>stdin()</code> will be cleared before return.)</p> <p><i>file</i> can also be a complete URL. (For the supported URL schemes, see the 'URLs' section of the help for url.)</p>
header	<p>a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: <code>header</code> is set to <code>TRUE</code> if and only if the first row contains one fewer field than the number of columns.</p>
sep	<p>the field separator character. Values on each line of the file are separated by this character. If <code>sep = ""</code> (the default for <code>read.table</code>) the separator is 'white space', that is one or more spaces, tabs, newlines or carriage returns.</p>
quote	<p>the set of quoting characters. To disable quoting altogether, use <code>quote = ""</code>. See scan for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless <code>colClasses</code> is specified.</p>
dec	<p>the character used in the file for decimal points.</p>



1. Getting started

1.4. Use functions

- From the **documentation** you can see that functions have **many arguments**
 - Some **without default** values: You need to specify the argument for the function to work
 - Some **with default** values: If you don't specify these arguments, defaults will be used

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",  
         dec = ".", fill = TRUE, comment.char = "", ...)
```

- You don't need to write the argument names only for those written in the **correct order**

```
read.csv(file = "dt.csv")
```



```
read.csv("dt.csv")
```

```
read.csv("dt.csv")
```



```
read.csv("dt.csv", sep = ",")
```



1. Getting started

1.4. Use functions

- From the **documentation** you can see that functions have **many arguments**
 - Some **without default** values: You need to specify the argument for the function to work
 - Some **with default** values: If you don't specify these arguments, defaults will be used

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",  
         dec = ".", fill = TRUE, comment.char = "", ...)
```

- You don't need to write the argument names only for those written in the **correct order**

```
read.csv(file = "dt.csv")
```



```
read.csv("dt.csv")
```

```
read.csv("dt.csv")
```



```
read.csv("dt.csv", sep = ",")
```

```
read.csv("dt.csv", sep = ",")
```



```
read.csv("dt.csv", ",")
```



1. Getting started

1.4. Use functions

- From the **documentation** you can see that functions have **many arguments**
 - Some **without default** values: You need to specify the argument for the function to work
 - Some **with default** values: If you don't specify these arguments, defaults will be used

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",  
        dec = ".", fill = TRUE, comment.char = "", ...)
```

- You don't need to write the argument names only for those written in the **correct order**

```
read.csv(file = "dt.csv")
```



```
read.csv("dt.csv")
```

```
read.csv("dt.csv")
```



```
read.csv("dt.csv", sep = ",")
```

```
read.csv("dt.csv", sep = ",")
```



```
read.csv("dt.csv", ",")
```

```
read.csv("dt.csv", sep = ",")
```



```
read.csv("dt.csv", TRUE, ",")
```



1. Getting started

1.4. Use functions

- From the **documentation** you can see that functions have **many arguments**
 - Some **without default** values: You need to specify the argument for the function to work
 - Some **with default** values: If you don't specify these arguments, defaults will be used

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",  
        dec = ".", fill = TRUE, comment.char = "", ...)
```

- You don't need to write the argument names only for those written in the **correct order**

```
read.csv(file = "dt.csv")
```



```
read.csv("dt.csv")
```

```
read.csv("dt.csv")
```



```
read.csv("dt.csv", sep = ",")
```

```
read.csv("dt.csv", sep = ",")
```



```
read.csv("dt.csv", ",")
```

```
read.csv("dt.csv", sep = ",")
```



```
read.csv("dt.csv", TRUE, ",")
```



Overview

1. Getting started ✓

- 1.1. About R
- 1.2. The R Studio IDE
- 1.3. Import and eyeball data
- 1.4. Use functions

2. Anatomy of a data.frame

- 2.1. Data structure
- 2.2. Classes
- 2.3. Vectors
- 2.4. Subsetting

3. Wrap up!



Overview

1. Getting started ✓

- 1.1. About R
- 1.2. The R Studio IDE
- 1.3. Import and eyeball data
- 1.4. Use functions

2. Anatomy of a data.frame

- 2.1. Data structure
- 2.2. Classes
- 2.3. Vectors
- 2.4. Subsetting



2. Anatomy of a `data.frame`

2.1. Data structure

- Now that we imported the data properly, we can check out its `str()` **ucture** in more details

```
str(fb)
```




2. Anatomy of a data.frame

2.1. Data structure

- *Don't be scared of the output!*

```
str(fb)
```

```
## 'data.frame':      380 obs. of  14 variables:
## $ Wk           : int   1 1 1 1 1 1 1 1 1 1 ...
## $ Day          : chr    "Fri" "Sat" "Sat" "Sun" ...
## $ Date         : chr    "2021-08-06" "2021-08-07" "2021-08-07" "2021-08-08" ...
## $ Time         : chr    "21:00" "17:00" "21:00" "13:00" ...
## $ Home         : chr    "Monaco" "Lyon" "Troyes" "Rennes" ...
## $ xG           : num    2 1.4 0.8 0.6 0.7 0.4 0.8 2.1 0.7 0.5 ...
## $ Score        : chr    "1-1" "1-1" "1-2" "1-1" ...
## $ xG.1         : num    0.3 0.8 1.2 2 3.3 0.9 0.2 1.3 1.4 2 ...
## $ Away         : chr    "Nantes" "Brest" "Paris S-G" "Lens" ...
## $ Attendance   : int    7500 29018 15248 22567 18748 23250 18030 20461 15551 13500 ...
## $ Venue        : chr    "Stade Louis II." "Groupama Stadium" "Stade de l'Aube" "Roazhon Park" ...
## $ Referee      : chr    "Antony Gautier" "Mikael Lesage" "Amaury Delerue" "Bastien Dechepy" ...
## $ Match.Report : chr    "Match Report" "Match Report" "Match Report" "Match Report" ...
## $ Notes        : logi   NA NA NA NA NA NA ...
```



2. Anatomy of a `data.frame`

2.1. Data structure

- `str()` says that `fb` is a `data.frame`, and gives its numbers of **observations** (rows) and **variables** (columns)

```
str(fb)
```

```
## 'data.frame':    380 obs. of  14 variables:
```



2. Anatomy of a `data.frame`

2.1. Data structure

- It also gives the **variables names**

```
str(fb)
```

```
## 'data.frame':   380 obs. of  14 variables:
##  $ Wk
##  $ Day
##  $ Date
##  $ Time
##  $ Home
##  $ xG
##  $ Score
##  $ xG.1
##  $ Away
##  $ Attendance
##  $ Venue
##  $ Referee
##  $ Match.Report
##  $ Notes
```



2. Anatomy of a data.frame

2.1. Data structure

- The **first values** of each variable

```
str(fb)
```

```
## 'data.frame':      380 obs. of  14 variables:
## $ Wk           :      1 1 1 1 1 1 1 1 1 1 ...
## $ Day          :      "Fri" "Sat" "Sat" "Sun" ...
## $ Date         :      "2021-08-06" "2021-08-07" "2021-08-07" "2021-08-08" ...
## $ Time         :      "21:00" "17:00" "21:00" "13:00" ...
## $ Home         :      "Monaco" "Lyon" "Troyes" "Rennes" ...
## $ xG           :      2 1.4 0.8 0.6 0.7 0.4 0.8 2.1 0.7 0.5 ...
## $ Score       :      "1-1" "1-1" "1-2" "1-1" ...
## $ xG.1        :      0.3 0.8 1.2 2 3.3 0.9 0.2 1.3 1.4 2 ...
## $ Away        :      "Nantes" "Brest" "Paris S-G" "Lens" ...
## $ Attendance  :      7500 29018 15248 22567 18748 23250 18030 20461 15551 13500 ...
## $ Venue       :      "Stade Louis II." "Groupama Stadium" "Stade de l'Aube" "Roazhon Park" ...
## $ Referee     :      "Antony Gautier" "Mikael Lesage" "Amaury Delerue" "Bastien Dechepy" ...
## $ Match.Report:      "Match Report" "Match Report" "Match Report" "Match Report" ...
## $ Notes       :      NA NA NA NA NA NA ...
```



2. Anatomy of a data.frame

2.1. Data structure

- As well as the **class** of each variable

```
str(fb)
```

```
## 'data.frame':      380 obs. of  14 variables:
## $ Wk           : int  1 1 1 1 1 1 1 1 1 1 ...
## $ Day          : chr   "Fri" "Sat" "Sat" "Sun" ...
## $ Date         : chr   "2021-08-06" "2021-08-07" "2021-08-07" "2021-08-08" ...
## $ Time         : chr   "21:00" "17:00" "21:00" "13:00" ...
## $ Home         : chr   "Monaco" "Lyon" "Troyes" "Rennes" ...
## $ xG           : num   2 1.4 0.8 0.6 0.7 0.4 0.8 2.1 0.7 0.5 ...
## $ Score        : chr   "1-1" "1-1" "1-2" "1-1" ...
## $ xG.1         : num   0.3 0.8 1.2 2 3.3 0.9 0.2 1.3 1.4 2 ...
## $ Away         : chr   "Nantes" "Brest" "Paris S-G" "Lens" ...
## $ Attendance   : int   7500 29018 15248 22567 18748 23250 18030 20461 15551 13500 ...
## $ Venue        : chr   "Stade Louis II." "Groupama Stadium" "Stade de l'Aube" "Roazhon Park" ...
## $ Referee      : chr   "Antony Gautier" "Mikael Lesage" "Amaury Delerue" "Bastien Dechepy" ...
## $ Match.Report : chr   "Match Report" "Match Report" "Match Report" "Match Report" ...
## $ Notes        : logi  NA NA NA NA NA NA ...
```



2. Anatomy of a `data.frame`

2.1. Data structure

- But what does the **class** correspond to?

```
str(fb)
```

```
## 'data.frame':    380 obs. of  14 variables:
##  $ Wk          : int  ?
##  $ Day         : chr  ?
##  $ Date        : chr  ?
##  $ Time        : chr  ?
##  $ Home        : chr  ?
##  $ xG          : num  ?
##  $ Score       : chr  ?
##  $ xG.1        : num  ?
##  $ Away        : chr  ?
##  $ Attendance  : int  ?
##  $ Venue       : chr  ?
##  $ Referee     : chr  ?
##  $ Match.Report: chr  ?
##  $ Notes       : logi  ?
```



2. Anatomy of a `data.frame`

2.2. Classes

Numeric

Character

Logical



2. Anatomy of a `data.frame`

2.2. Classes

Numeric

Character

Logical

These are simply numbers:

```
class(3)
```

```
## [1] "numeric"
```

```
class(-1.6180339)
```

```
## [1] "numeric"
```

Numeric variable classes include:

- **int** for round numbers
- **dbl** for 2-decimal numbers



2. Anatomy of a `data.frame`

2.2. Classes

Numeric

These are simply numbers:

```
class(3)
```

```
## [1] "numeric"
```

```
class(-1.6180339)
```

```
## [1] "numeric"
```

Numeric variable classes include:

- **int** for round numbers
- **dbl** for 2-decimal numbers

Character

They must be surrounded by `"`:

```
class("Roazhon Park")
```

```
## [1] "character"
```

```
class("35")
```

```
## [1] "character"
```

We also call these values:

- Character strings
- Or just strings

Logical



2. Anatomy of a data.frame

2.2. Classes

Numeric

These are simply numbers:

```
class(3)
```

```
## [1] "numeric"
```

```
class(-1.6180339)
```

```
## [1] "numeric"
```

Numeric variable classes include:

- **int** for round numbers
- **dbl** for 2-decimal numbers

Character

They must be surrounded by ":

```
class("Roazhon Park")
```

```
## [1] "character"
```

```
class("35")
```

```
## [1] "character"
```

We also call these values:

- Character strings
- Or just strings

Logical

Something either TRUE or FALSE:

```
3 >= 4
```

```
## [1] FALSE
```

```
class(T)
```

```
## [1] "logical"
```

Most common logical operators:

- == > < <= >=
- & (and) | (or) ! (opposite)



2. Anatomy of a `data.frame`

2.2. Classes

Guess the output!

```
as.numeric("2022")
```

```
## [1] 2022
```

What about this one?

```
as.character(2022-2023)
```

```
## [1] "-1"
```

A final one:

```
as.character(2022>2023)
```

```
## [1] "FALSE"
```



2. Anatomy of a data.frame

2.2. Classes

- To know everything:

Class conversion table:

	numeric	character	logical
as.numeric()	No effect	Converts strings of numbers into numeric values Returns NA if characters in the string	Returns 1 if TRUE Returns 0 if FALSE
as.character()	Converts numeric values into strings of numbers	No effect	Returns "TRUE" if TRUE Returns "FALSE" if FALSE
as.logical()	Returns TRUE if != 0 Returns FALSE if 0	Returns TRUE if "T" or "TRUE" Returns FALSE if "F" or "FALSE" Returns NA otherwise	No effect

NA stands for 'Not Available', it corresponds to a **missing value**



2. Anatomy of a data.frame

2.2. Classes

- Great! But there is one last mystery...

```
str(fb)
```

```
## 'data.frame':      380 obs. of  14 variables:
## $ Wk           : int  1 1 1 1 1 1 1 1 1 1 ...
## $ Day          : chr   "Fri" "Sat" "Sat" "Sun" ...
## $ Date         : chr   "2021-08-06" "2021-08-07" "2021-08-07" "2021-08-08" ...
## $ Time         : chr   "21:00" "17:00" "21:00" "13:00" ...
## $ Home         : chr   "Monaco" "Lyon" "Troyes" "Rennes" ...
## $ xG           : num   2 1.4 0.8 0.6 0.7 0.4 0.8 2.1 0.7 0.5 ...
## $ Score        : chr   "1-1" "1-1" "1-2" "1-1" ...
## $ xG.1         : num   0.3 0.8 1.2 2 3.3 0.9 0.2 1.3 1.4 2 ...
## $ Away         : chr   "Nantes" "Brest" "Paris S-G" "Lens" ...
## $ Attendance   : int   7500 29018 15248 22567 18748 23250 18030 20461 15551 13500 ...
## $ Venue        : chr   "Stade Louis II." "Groupama Stadium" "Stade de l'Aube" "Roazhon Park" ...
## $ Referee      : chr   "Antony Gautier" "Mikael Lesage" "Amaury Delerue" "Bastien Dechepy" ...
## $ Match.Report : chr   "Match Report" "Match Report" "Match Report" "Match Report" ...
## $ Notes        : logi  NA NA NA NA NA NA ...
```



2. Anatomy of a `data.frame`

2.2. Classes

- Are these dollar signs here for a reason?

```
str(fb)
```

```
## 'data.frame':    380 obs. of  14 variables:
##  $ Wk
##  $ Day
##  $ Date
##  $ Time
##  $ Home
##  $ xG
##  $ Score
##  $ xG.1
##  $ Away
##  $ Attendance
##  $ Venue
##  $ Referee
##  $ Match.Report
##  $ Notes
```



2. Anatomy of a data.frame

2.3. Vectors

- It's actually just a reference to the fact that `$` allows to **extract a variable** from a dataset

```
fb$Wk
```

```
##      [1]  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3  3  3
##     [30]  4  4  4  4  4  4  4  4  4  4  4  5  5  5  5  5  5  5  5  5  5  6  6  6  6  6  6  6  6  6
##     [59]  6  7  7  7  7  7  7  7  7  7  7  7  8  8  8  8  8  8  8  8  8  8  9  9  9  9  9  9  9  9  9
##     [88]  9  9 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 11  3 12 12 12 12 12 12 12
##    [117] 12 12 12 12 13 13 13 13 13 13 13 13 13 13 13 14 14 14 14 14 14 14 14 14 14 15 15 15 15 15
##    [146] 15 15 15 15 16 16 16 16 16 16 16 16 16 16 16 17 17 17 17 17 17 17 17 17 17 18 18 18 18 18
##    [175] 18 18 18 18 18 19 19 19 19 19 19 19 19 19 19 20 20 20 20 20 20 20 20 21 21 21 21 21 21 21
##    [204] 21 21 19 20 20 22 22 22 22 22 22 22 22 22 22 22 20 14 23 23 23 23 23 23 23 23 23 23 24 24
##    [233] 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 27
##    [262] 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29
##    [291] 30 30 30 30 30 30 30 30 30 30 30 31 31 31 31 31 31 31 31 31 31 32 32 32 32 32 32 32 32 32
##    [320] 32 33 33 33 33 33 33 33 33 33 33 33 33 33 33 34 34 34 34 34 34 34 34 34 34 35 35 35 35 35
##    [349] 35 35 36 36 36 36 36 36 36 36 36 36 36 36 37 37 37 37 37 37 37 37 37 37 37 38 38 38 38 38
##    [378] 38 38 38
```



2. Anatomy of a `data.frame`

2.3. Vectors

- We call these objects **vectors**
 - Vectors are basically **sequences of values that have the same class**
 - R won't let you create a vector containing elements of different classes
- We make our own vectors using the `c()` **concatenate** function

```
c("Hello world", 35, FALSE)
```

```
## [1] "Hello world" "35"          "FALSE"
```

- The fact that vectors are homogeneous in class allows that **operations apply to all their elements**

```
c(1, 2, 3) / 3
```

```
## [1] 0.3333333 0.6666667 1.0000000
```

```
3 / c(1, 2, 3)
```

```
## [1] 3.0 1.5 1.0
```




2. Anatomy of a `data.frame`

2.4. Subsetting

- But `$` is not the only way to **extract** a variable from a dataset
 - You can also make use of the `[]` subsetting operator

`data[row, columns]`

- Inside the **brackets**, indicate what you want to **keep using**:
 - **Indices**: e.g., the third column has index 3
 - **Logical**: A vector of TRUE and FALSE
 - **Names**: They must be in quotation marks

- Example:

```
fb[1, c("Venue", "Attendance")]
```

```
##           Venue Attendance
## 1 Stade Louis II.      7500
```

- Brackets also work for vectors:

```
vec <- c(3, 2, 1)
vec[c(T, F, T)]
```

```
## [1] 3 1
```

Practice

06:00

- 1) Download and import the dataset if you haven't already
- 2) Combine the use of `[]` and `nrow()` to obtain the last value of the `wk` variable
- 3) Subset the home team, the score, and the away team for matches that occurred during the last week

You've got 6 minutes!

Solution

1) Download and import the dataset if you haven't already

```
fb <- read.csv("C:/User/Documents/ligue1.csv", encoding = "UTF-8")
```

2) Combine the use of `[]` and `nrow()` to obtain the last value of the `wk` variable

```
last_week <- fb[nrow(fb), "Wk"]  
last_week
```

```
## [1] 38
```

Solution

3) Subset the home team, the score, and the away team for matches that occurred during the last week

```
fb[Wk == last_week, c("Home", "Score", "Away")]
```

```
## Error in `[.data.frame`(fb, Wk == last_week, c("Home", "Score", "Away")): object 'Wk' not found
```

- Oops! Seems like **R couldn't find** the Wk variable
 - R was looking for Wk **in our environment**
 - But there is no Wk there
- We must **refer to fb** which is in our environment
 - Then we can **access Wk using the \$** symbol

```
fb[fb$Wk == 38, c("Home", "Score", "Away")]
```

	Home	Score	Away
## 371	Lille	2-2	Rennes
## 372	Brest	2-4	Bordeaux
## 373	Nantes	1-1	Saint-Étienne
## 374	Clermont Foot	1-2	Lyon
## 375	Angers	2-0	Montpellier
## 376	Lorient	1-1	Troyes
## 377	Paris S-G	5-0	Metz
## 378	Reims	2-3	Nice
## 379	Marseille	4-0	Strasbourg
## 380	Lens	2-2	Monaco



Overview

1. Getting started ✓

- 1.1. About R
- 1.2. The R Studio IDE
- 1.3. Import and eyeball data
- 1.4. Use functions

2. Anatomy of a `data.frame` ✓

- 2.1. Data structure
- 2.2. Classes
- 2.3. Vectors
- 2.4. Subsetting

3. Wrap up!



3. Wrap up!

1. Import data

```
fb <- read.csv("C:/User/Documents/ligue1.csv", encoding = "UTF-8")
```

2. Class

```
is.numeric("1.6180339") # What would be the output?
```

```
## [1] FALSE
```

3. Subsetting

```
fb$Home[3]
```

```
## [1] "Troyes"
```