Introduction to R, RStudio and Quarto

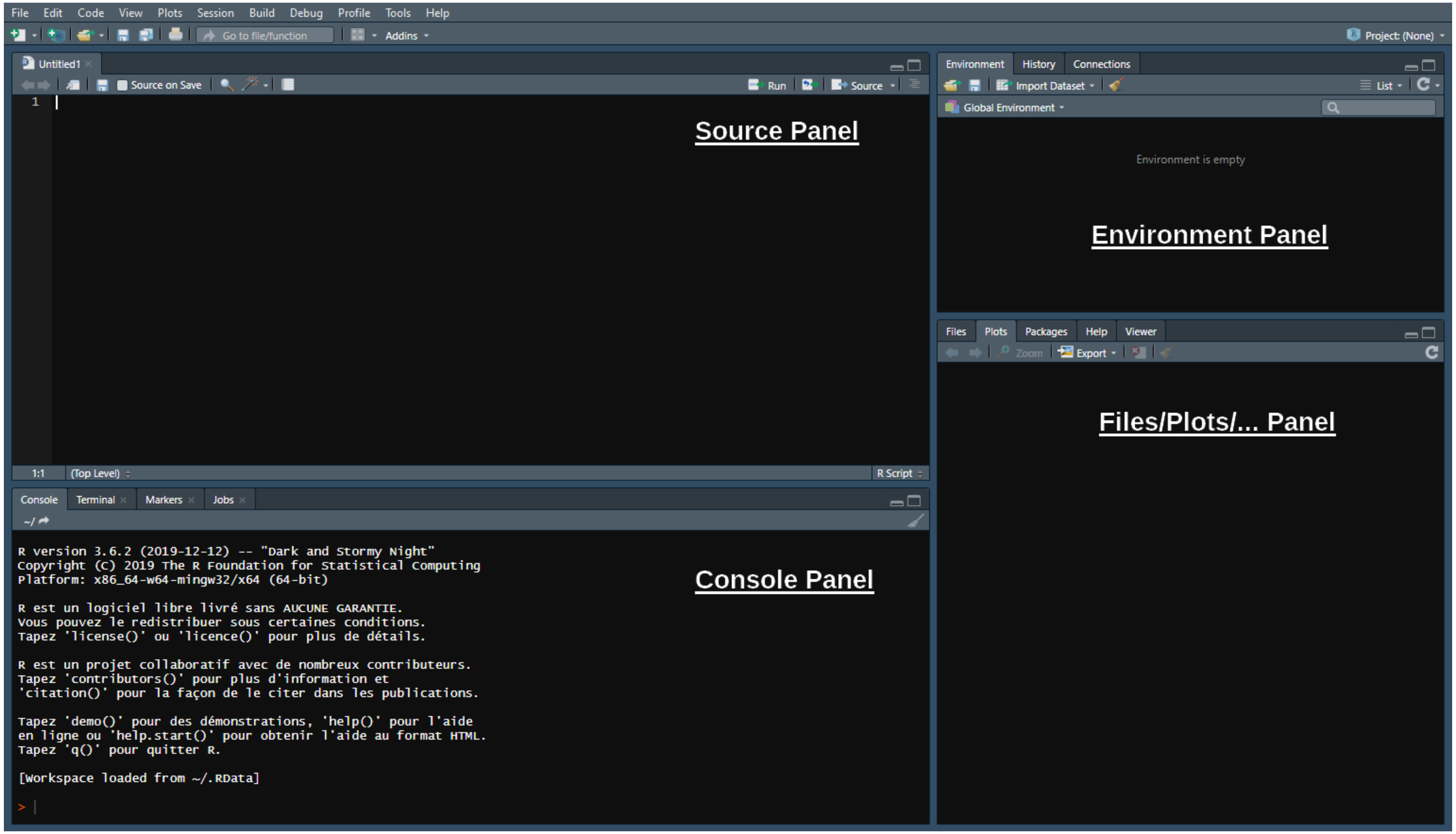
Overview

Getting Started

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
    - Biostatistics

The R Studio IDE



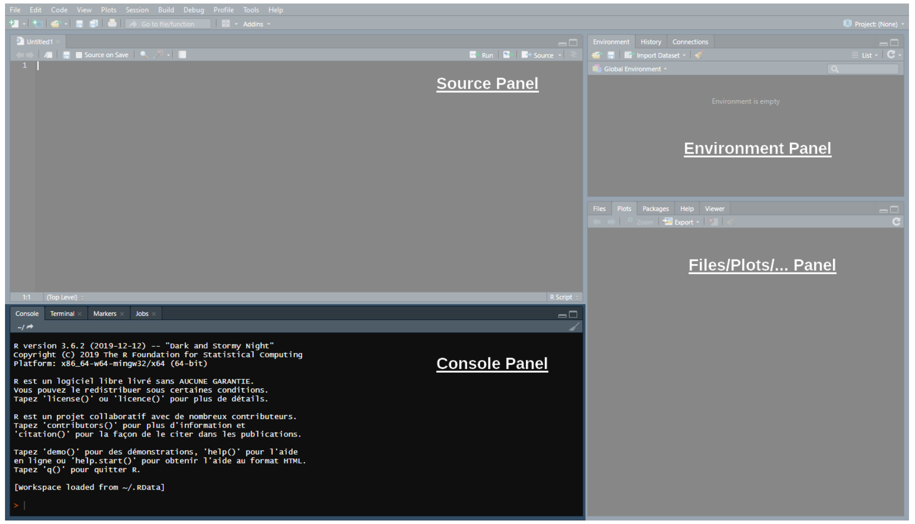
```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Workspace loaded from ~/.RData]

>
```

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**:

```r
1+1
```

```
[1] 2
```

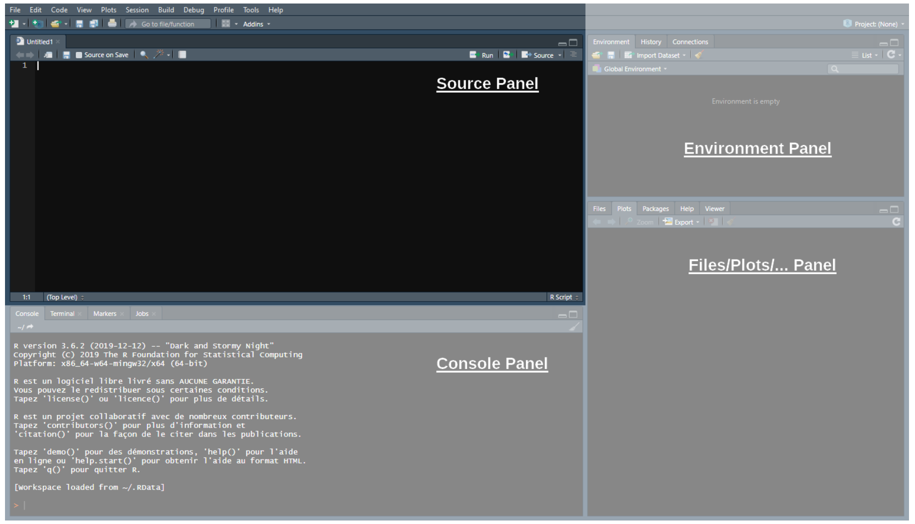


```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Workspace loaded from ~/.RData]

> 
```

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 you to **comment** your code
    - Everything after the **#** will be **ignored** by R until the next line break

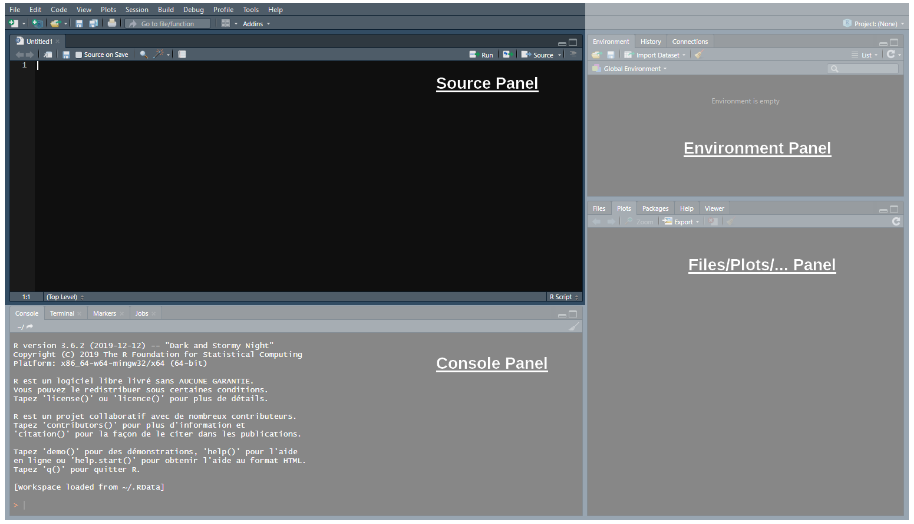
```r
1+1 # Do not put 2+2 on the same line, press enter
2+2
```



```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Workspace loaded from ~/.RData]

>
```

The Source panel

- To send the command from the source panel to the console panel:
  1. **Click on/Highlight** the line(s) 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



```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Workspace loaded from ~/.RData]

>
```

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
```

- Now that the object x is stored in your environment, you can use it

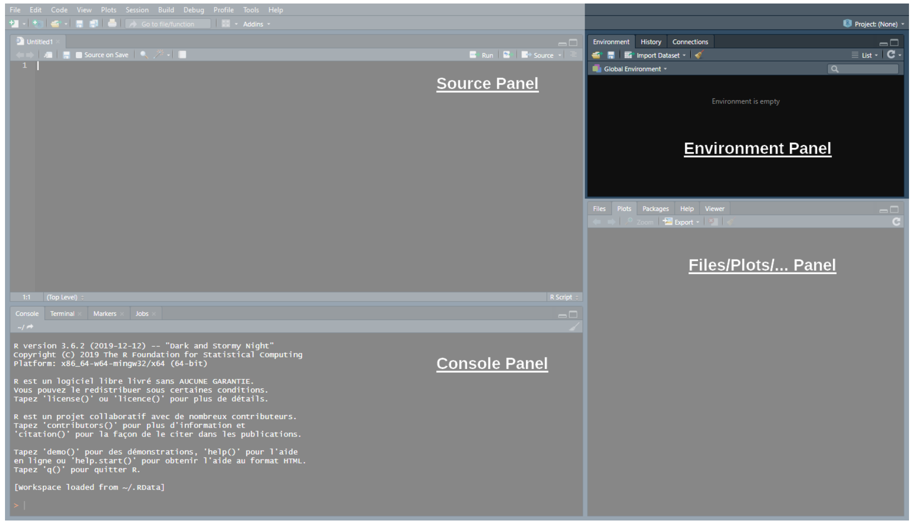
```
x + 1
```

```
[1] 2
```

- You can also modify that object at any point:

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

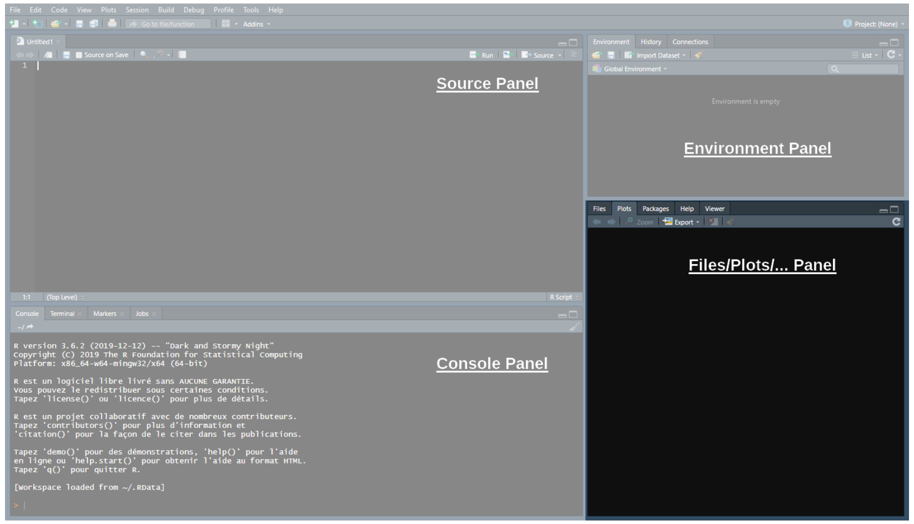
```
[1] 2
```



```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Workspace loaded from ~/.RData]

>
```

The Files/Plots/... panel

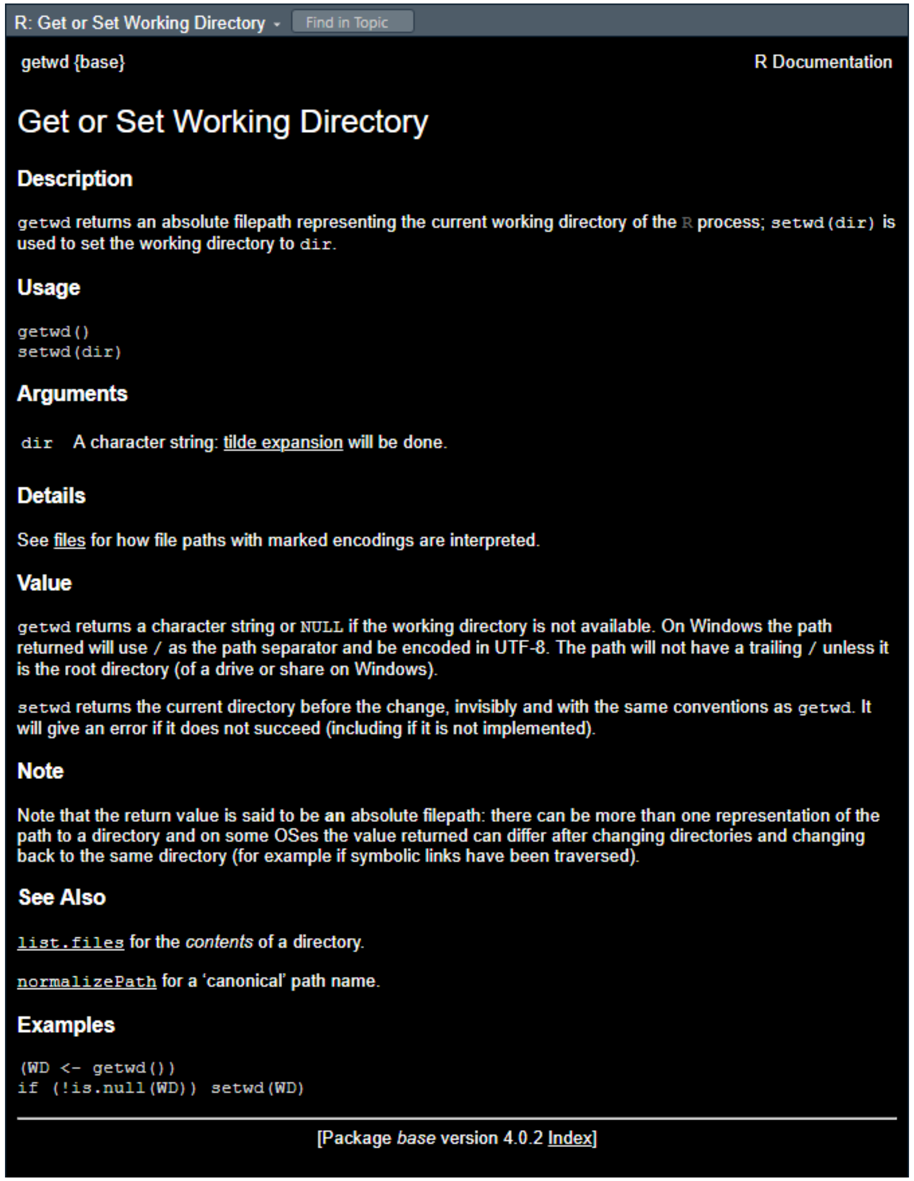
- In this panel, we'll mainly be interested in the following 4 tabs:
  - **Files:** Shows your working directory
  - **Plots:** Where R returns plots
  - **Packages:** A library of tools that we can load if needed
  - **Help:** Where to look for documentation on R functions



```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Workspace loaded from ~/.RData]

>
```

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



getwd {base} R Documentation

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

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 on your computer (Ctrl+S)

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

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

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

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

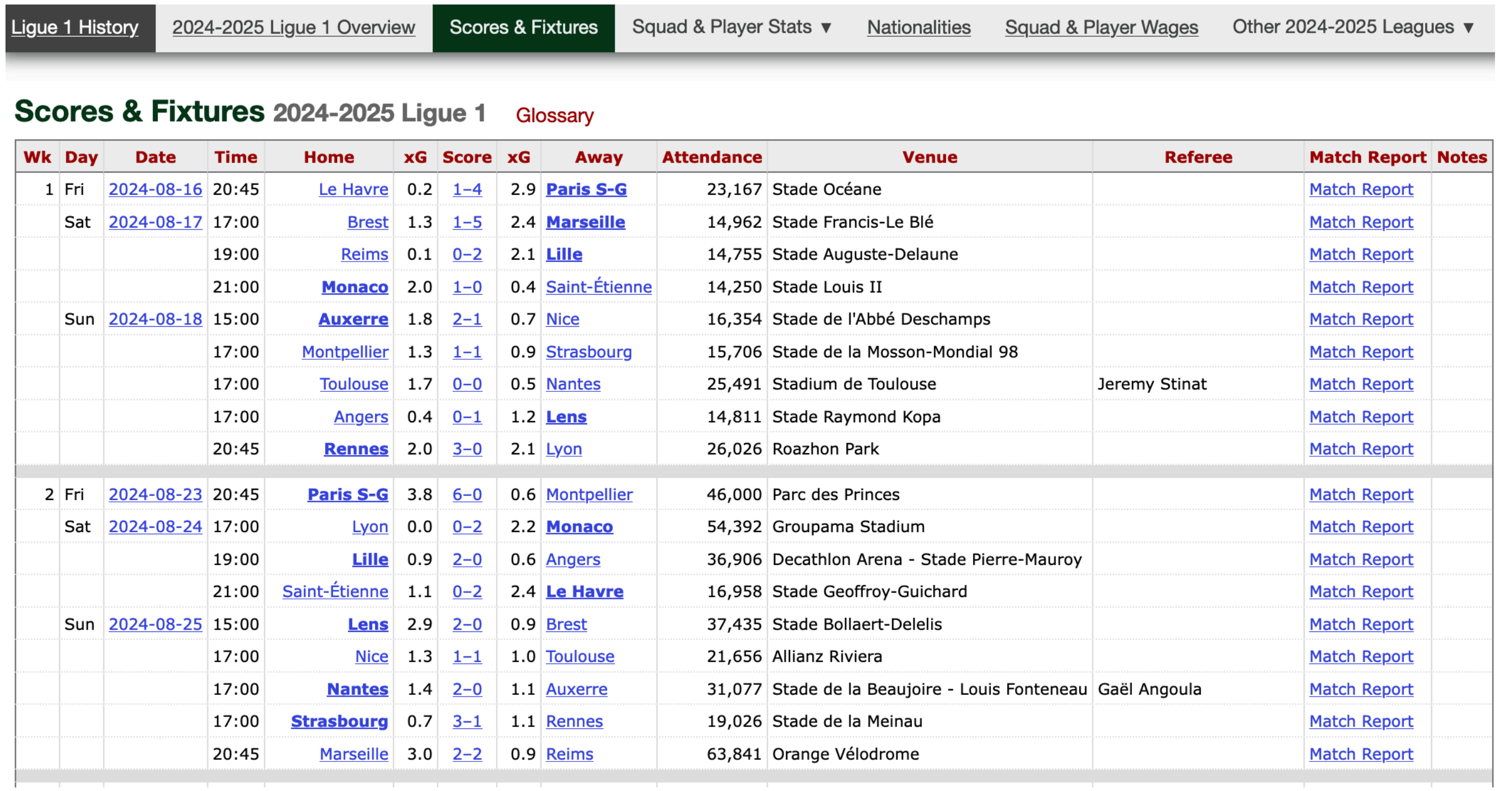
3. Print `result` in your console and save your script somewhere on your computer (Ctrl+S)

```r
result
```

```
[1] 42
```

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 2024-2025 Ligue 1 Glossary

Wk	Day	Date	Time	Home	xG	Score	xG	Away	Attendance	Venue	Referee	Match Report	Notes
1	Fri	2024-08-16	20:45	Le Havre	0.2	1–4	2.9	**Paris S-G**	23,167	Stade Océane		Match Report	
	Sat	2024-08-17	17:00	Brest	1.3	1–5	2.4	**Marseille**	14,962	Stade Francis-Le Blé		Match Report	
			19:00	Reims	0.1	0–2	2.1	**Lille**	14,755	Stade Auguste-Delaune		Match Report	
			21:00	**Monaco**	2.0	1–0	0.4	Saint-Étienne	14,250	Stade Louis II		Match Report	
	Sun	2024-08-18	15:00	**Auxerre**	1.8	2–1	0.7	Nice	16,354	Stade de l'Abbé Deschamps		Match Report	
			17:00	Montpellier	1.3	1–1	0.9	Strasbourg	15,706	Stade de la Mosson-Mondial 98		Match Report	
			17:00	Toulouse	1.7	0–0	0.5	Nantes	25,491	Stadium de Toulouse	Jeremy Stinat	Match Report	
			17:00	Angers	0.4	0–1	1.2	**Lens**	14,811	Stade Raymond Kopa		Match Report	
			20:45	**Rennes**	2.0	3–0	2.1	Lyon	26,026	Roazhon Park		Match Report	
2	Fri	2024-08-23	20:45	**Paris S-G**	3.8	6–0	0.6	Montpellier	46,000	Parc des Princes		Match Report	
	Sat	2024-08-24	17:00	Lyon	0.0	0–2	2.2	**Monaco**	54,392	Groupama Stadium		Match Report	
			19:00	**Lille**	0.9	2–0	0.6	Angers	36,906	Decathlon Arena - Stade Pierre-Mauroy		Match Report	
			21:00	Saint-Étienne	1.1	0–2	2.4	**Le Havre**	16,958	Stade Geoffroy-Guichard		Match Report	
	Sun	2024-08-25	15:00	**Lens**	2.9	2–0	0.9	Brest	37,435	Stade Bollaert-Delelis		Match Report	
			17:00	Nice	1.3	1–1	1.0	Toulouse	21,656	Allianz Riviera		Match Report	
			17:00	**Nantes**	1.4	2–0	1.1	Auxerre	31,077	Stade de la Beaujoire - Louis Fonteneau	Gaël Angoula	Match Report	
			17:00	**Strasbourg**	0.7	3–1	1.1	Rennes	19,026	Stade de la Meinau		Match Report	
			20:45	Marseille	3.0	2–2	0.9	Reims	63,841	Orange Vélodrome		Match Report	

Import and Eyeball Data

- You can **download** the dataset for the 2021/22 season by clicking here or from the course webpage.
  - Note that the file extension is **.csv** (for **Comma Separated Values**).
  - Let's take a look at the **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 follows a specific structure:
  - Each line corresponds to a row (the first row typically contains column names).
  - For each row, values of each column are separated by commas.
- *But how do we get it into our RStudio environment?*

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` (later we will mostly use `read_csv`)
- Remember we use the arrow (`<-`) to create objects in R

```
data <- read.csv("/Users/jan/Downloads/ligue1.csv")
```

Important

Make sure you have the right path to your data file. Also, make sure you use correct backlashes "/". Do NOT use "\".

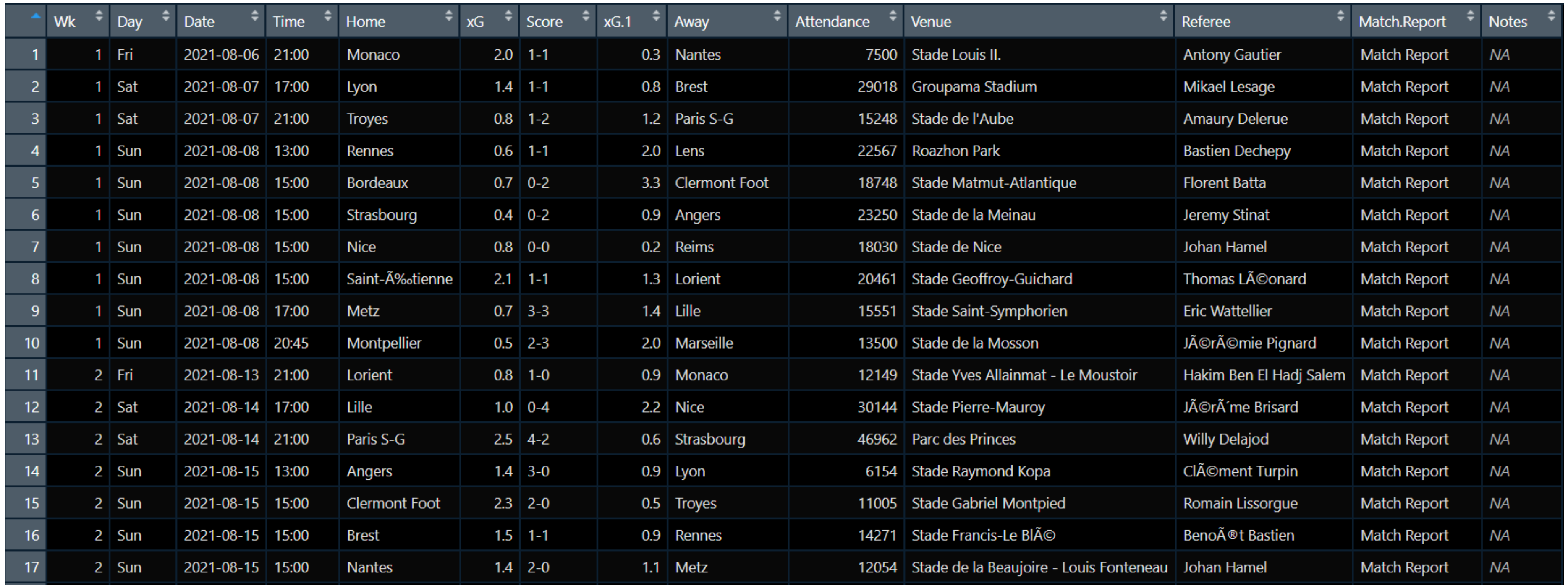
- Let's inspect this new object

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

```
head(data, 3)
```

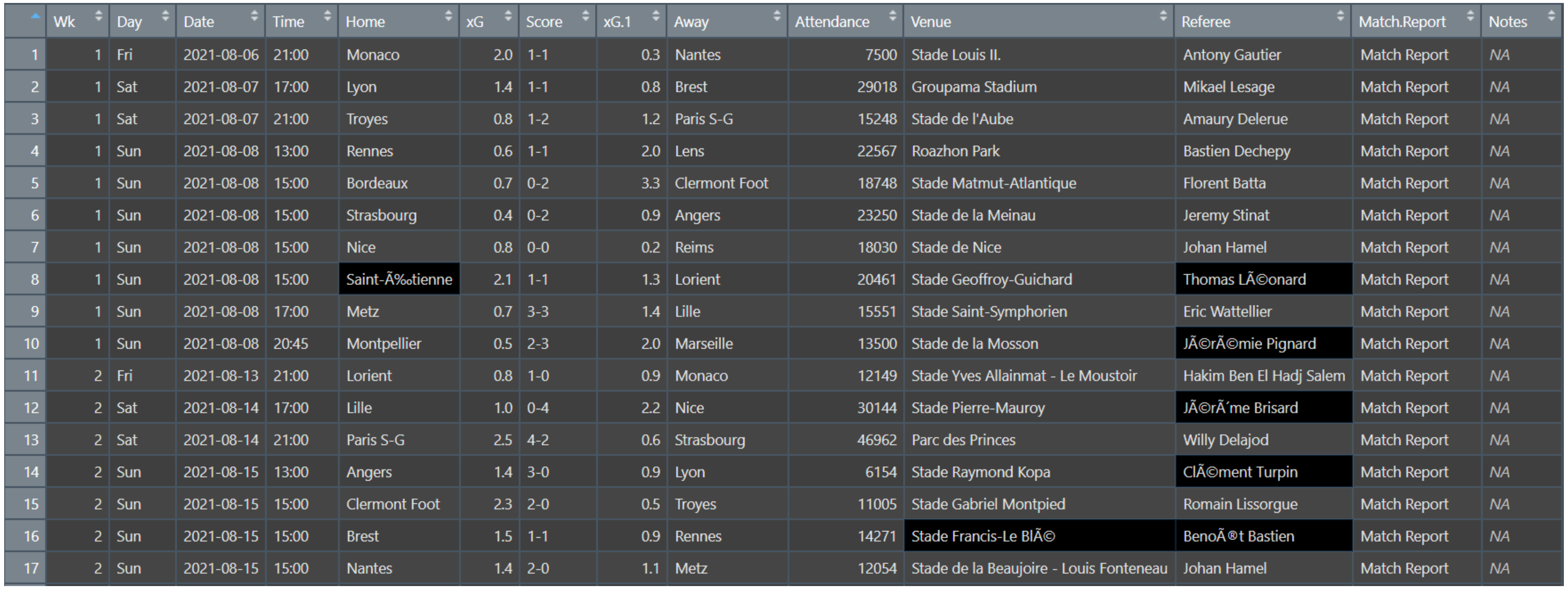
```
  Wk Day       Date  Time   Home  xG Score xG.1      Away Attendance
1  1 Fri 2021-08-06 21:00 Monaco 2.0   1-1  0.3    Nantes       7500
2  1 Sat 2021-08-07 17:00   Lyon 1.4   1-1  0.8     Brest      29018
3  1 Sat 2021-08-07 21:00 Troyes 0.8   1-2  1.2 Paris S-G      15248
             Venue         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
```

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



	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-Ã‰tienne	2.1	1-1	1.3	Lorient	20461	Stade Geoffroy-Guichard	Thomas LÃ©onard	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	JÃ©rÃ©mie 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	JÃ©rÃ´me 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	ClÃ©ment 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 BlÃ©	BenoÃ®t 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!



	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-Ã‰tienne	2.1	1-1	1.3	Lorient	20461	Stade Geoffroy-Guichard	Thomas LÃ©onard	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	JÃ©rÃ©mie 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	JÃ©rÃ´me 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	ClÃ©ment 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 BlÃ©	BenoÃ®t 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

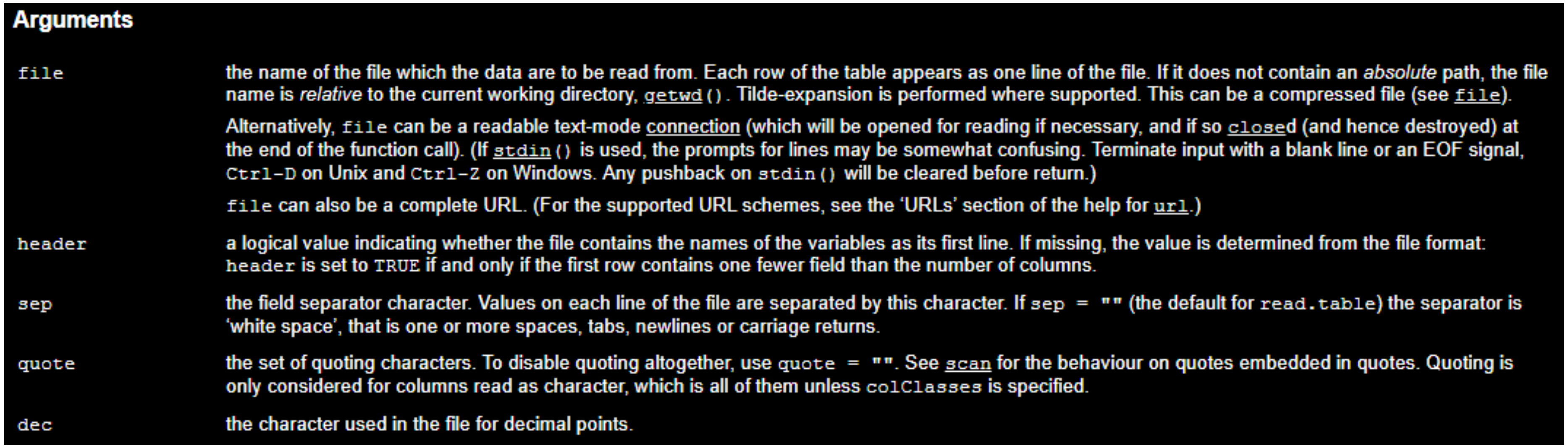
Or kind of worked…

These kind of weird characters pop up when there is an encoding issue

- Thankfully, `read.csv()` has many options that can be set as inputs, including encoding!
- Usually the UTF-8 encoding is the solution to French characters

```r
data <- read.csv("/Users/jan/Downloads/ligue1.csv", encoding = "UTF-8")
```

When you will be facing similar issues, check out the arguments of `read.csv()` by typing `?read.csv`



Arguments

`file`	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 *absolute* path, the file name is *relative* to the current working directory, `getwd()`. Tilde-expansion is performed where supported. This can be a compressed file (see `file`). Alternatively, `file` can be a readable text-mode connection (which will be opened for reading if necessary, and if so `close`d (and hence destroyed) at the end of the function call). (If `stdin()` 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 `stdin()` will be cleared before return.) `file` can also be a complete URL. (For the supported URL schemes, see the 'URLs' section of the help for `url`.)
`header`	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: `header` is set to `TRUE` if and only if the first row contains one fewer field than the number of columns.
`sep`	the field separator character. Values on each line of the file are separated by this character. If `sep = ""` (the default for `read.table`) the separator is 'white space', that is one or more spaces, tabs, newlines or carriage returns.
`quote`	the set of quoting characters. To disable quoting altogether, use `quote = ""`. 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 `colClasses` is specified.
`dec`	the character used in the file for decimal points.

Overview

1. **Getting Started**
    - About R
    - The R Studio IDE
    - Import and eyeball data

2. **Anatomy of a `data.frame`**
    - Data structure
    - Classes
    - Vectors
    - Subsetting

Anatomy of a `data.frame`

Data Structure

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

```r
str(data)
```

Data Structure

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

```r
str(data)
```

```
'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 ...
```

Data Structure

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

```
str(data)
```

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

Data Structure

- It also gives the **variables names**

```r
str(data)
```

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

Data Structure

- The **first values** of each variable

```r
str(data)
```

```
## '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 ...
```

Data Structure

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

```r
str(data)
```

```
## '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 ...
```

Data Structure

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

```r
str(data)
```

```
## '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  ?
```

Classes

Numeric

These are simply numbers:

```r
class(3)
```

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

```r
class(-1.89278)
```

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

- Numeric variable classes include:
  - int for round numbers
  - dbl for 2-decimal numbers

Character

They must be surrounded by " or ':

```r
class("Paris Saint-Germain")
```

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

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

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

- We also call these values:
  - Character strings
  - Or just strings

Logical

Something either TRUE of FALSE:

```r
3 >= 4
```

```
[1] FALSE
```

```r
class(3 >= 4)
```

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

```r
class(TRUE)
```

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

Operator	Meaning
==	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
&	And
|	Or
!	Opposite

Classes

Guess the output!

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

```
[1] 2022
```

What about this one?

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

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

And a final one.

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

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

Classes

	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**

Classes

One last mystery...

```r
str(data)
```

```
'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 ...
```

Classes

Are these dollar signs here for a reason?

```r
str(data)
```

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

Vectors

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

```r
data$Home
```

```
 [1] "Monaco"        "Lyon"          "Troyes"        "Rennes"
 [5] "Bordeaux"      "Strasbourg"    "Nice"          "Saint-Étienne"
 [9] "Metz"          "Montpellier"   "Lorient"       "Lille"
[13] "Paris S-G"     "Angers"        "Clermont Foot" "Brest"
[17] "Nantes"        "Reims"         "Lens"          "Marseille"
[21] "Brest"         "Monaco"        "Saint-Étienne" "Lyon"
[25] "Strasbourg"    "Metz"          "Montpellier"   "Bordeaux"
[29] "Rennes"        "Nantes"        "Nice"          "Marseille"
[33] "Troyes"        "Strasbourg"    "Angers"        "Lens"
[37] "Clermont Foot" "Lille"         "Reims"         "Lorient"
[41] "Paris S-G"     "Monaco"        "Montpellier"   "Rennes"
[45] "Bordeaux"      "Brest"         "Metz"          "Nantes"
[49] "Lyon"          "Strasbourg"    "Lens"          "Saint-Étienne"
[53] "Nice"          "Troyes"        "Clermont Foot" "Reims"
[57] "Angers"        "Marseille"     "Paris S-G"     "Rennes"
[61] "Nantes"        "Lille"         "Montpellier"   "Monaco"
[65] "Lyon"          "Lens"          "Lorient"       "Angers"
[69] "Metz"          "Saint-Étienne" "Strasbourg"    "Paris S-G"
[73] "Lyon"          "Bordeaux"      "Troyes"        "Brest"
[77] "Reims"         "Clermont Foot" "Marseille"     "Lens"
[81] "Montpellier"   "Nice"          "Rennes"        "Lorient"
[85] "Monaco"        "Angers"        "Nantes"        "Lille"
[89] "Saint-Étienne" "Paris S-G"     "Clermont Foot" "Lyon"
```

Vectors

- Variables are basically objects that we call vectors
  - Vectors are 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()`oncatenate function

```r
some_vector <- c("Hello world", 35, FALSE)
some_vector
```

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

Note that R will coerce the different elements into the same class when we create a vector (in this case character)

```r
class(some_vector)
```

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

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

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

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

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

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

Subsetting

- With `$`, you can extract a single variable from a dataset
- You can extract several variables and specific observations from a data frame using `[]`

data[row(s),   column(s)]

- 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:

```r
data[1, c("Venue", "Attendance")]
```

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

We can also subset single vectors:

```r
vector <- c(3, 2, 1)
vector[c(TRUE, TRUE, FALSE)]
```

```
[1] 3 2
```

Practice

04:00

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

Tip

Instead of `str()`, you can use the `names()` function to display all the variable names of a data frame.

Solution

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

```
data <- read.csv("/Users/jan/Downloads/ligue1.csv")
```

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

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

```
[1] 38
```

3. Subset the home team, the score, and the away team for matches that occured during the last week

```r
names(data)
```

```
 [1] "Wk"           "Day"          "Date"         "Time"         "Home"        
 [6] "xG"           "Score"        "xG.1"         "Away"         "Attendance"  
[11] "Venue"        "Referee"      "Match.Report" "Notes"       
```

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

```
Error: 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 the data frame `data` which is in our environment
  - Then we can access Wk using the `$` symbol

```r
data[data$Wk == last_week, 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
```

Overview

1. **Getting Started**
    - About R
    - The R Studio IDE
    - Import and eyeball data
2. **Anatomy of a `data.frame`**
    - Data structure
    - Classes
    - Vectors
    - Subsetting
3. **Wrap Up**
    - Summary and key take-aways

Wrap Up

Import data

```r
data <- read.csv("/Users/jan/Downloads/ligue1.csv")
```

Class

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

```
[1] FALSE
```

Subsetting

```r
data$Home[3] # What would be the output?
```

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

Overview

1. **Getting Started**
    - About R
    - The R Studio IDE
    - Import and eyeball data
2. **Anatomy of a `data.frame`**
    - Data structure
    - Classes
    - Vectors
    - Subsetting
3. **Wrap Up**
    - Summary and key take-aways
4. **Markdown and universal writing**
    - Office Model vs. Engineering Model
    - Excel failures
    - Markdown

Markdown and universal writing

Office Model vs. Engineering Model

Writing up research is a complicated, messy process!

Office Model vs. Engineering Model

- Loads of puzzle pieces:
  - Data
  - Statistical results
  - Fieldwork
  - Analysis
  - Figures
  - Tables
  - Citations
  - Text
- Each of these comes from a different place

Office Model vs. Engineering Model

Two general approaches for this mess:

The **Office** model

- Manually put everything in one document (and repeat often)

The **Engineering** model

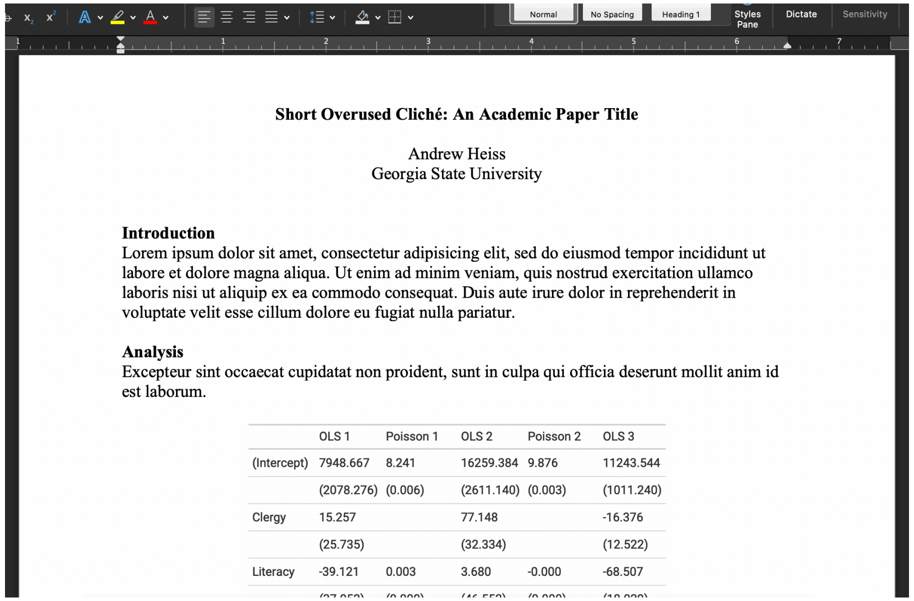
- Work with the raw pieces and compile it all in the end

The Office Model

Everything lives in one `.docx` file

- Drag images in
- Copy/paste stats from R
- Connect Word to Zotero or Endnote
- Track versions with filenames:
  - `ms.docx`, `ms2_final.docx`, `ms2_final_final.docx`

Final output = `.docx` file



Short Overused Cliché: An Academic Paper Title

Andrew Heiss
Georgia State University

Introduction
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Analysis
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

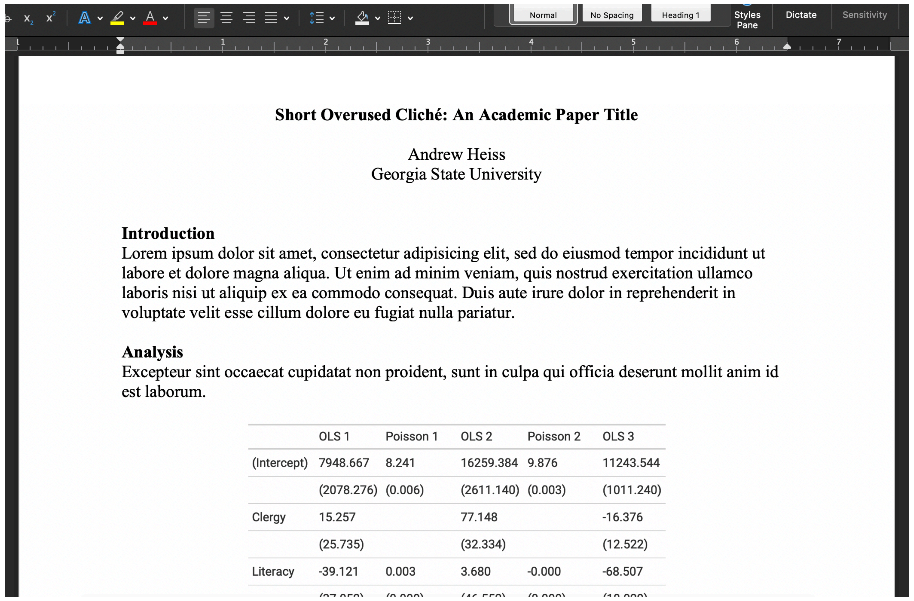
	OLS 1	Poisson 1	OLS 2	Poisson 2	OLS 3
(Intercept)	7948.667	8.241	16259.384	9.876	11243.544
	(2078.276)	(0.006)	(2611.140)	(0.003)	(1011.240)
Clergy	15.257		77.148		-16.376
	(25.735)		(32.334)		(12.522)
Literacy	-39.121	0.003	3.680	-0.000	-68.507

The Engineering Model

Everything lives separately and is combined in the end

- Type text in a plain text document
- Import images automatically
- Import stats automatically from R scripts (`.R` or `.qmd`) or `.do` files
- Store citations in reference manager
- Track versions with git

Final output = whatever you want (Word, PDF, HTML)



Short Overused Cliché: An Academic Paper Title

Andrew Heiss
Georgia State University

Introduction
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Analysis
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

	OLS 1	Poisson 1	OLS 2	Poisson 2	OLS 3
(Intercept)	7948.667	8.241	16259.384	9.876	11243.544
	(2078.276)	(0.006)	(2611.140)	(0.003)	(1011.240)
Clergy	15.257		77.148		-16.376
	(25.735)		(32.334)		(12.522)
Literacy	-39.121	0.003	3.680	-0.000	-68.507

Office Model vs. Engineering Model

There is no one right way!

The **Office** model

Cons:

- With changing analyses or data, manually updating your doc is laborous
- Chaos-prone:
  - You got to remember which script generated what)
- Error-prone:
  - It is easy to forget to update all figures, tables, results in text, etc.

The **Engineering** model

Cons:

- A bit of an entry cost
  - Need to learn a new coding language
- You'll always work with people who only use Word

Office Model vs. Engineering Model

The **Office** model

Pros:

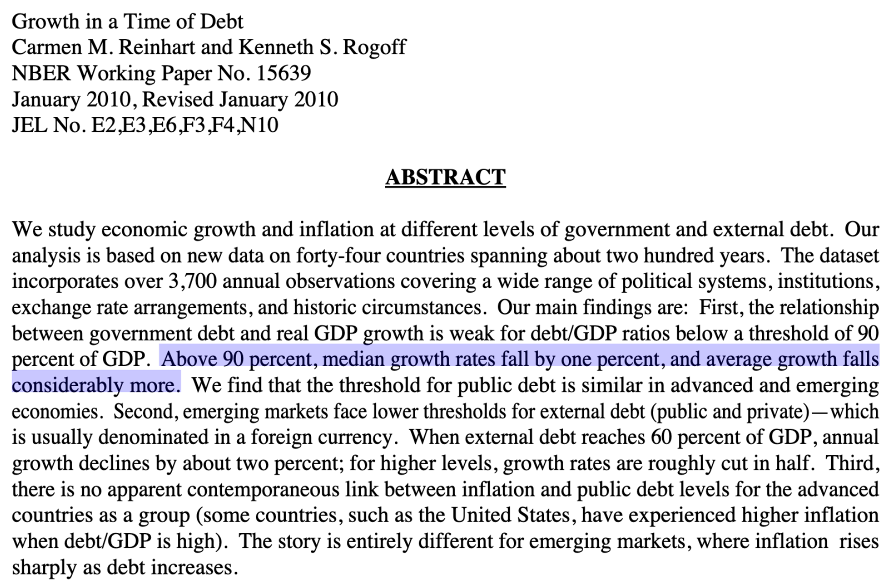
- No coding, easy environments
- The whole world runs on Word

The **Engineering** model

Pros:

- Less **cognitive** load
  - While everything seems complex in the beginning, no chaos because all is documented and transparent
- Less **work** load (in the long run)
  - No need to copy/paste new results, add updated figures, reformat citation, etc.
- Transparency
  - There's a record of everything you do
  - Your findings are reproducible by anyone (and yourself!)

Excel failures

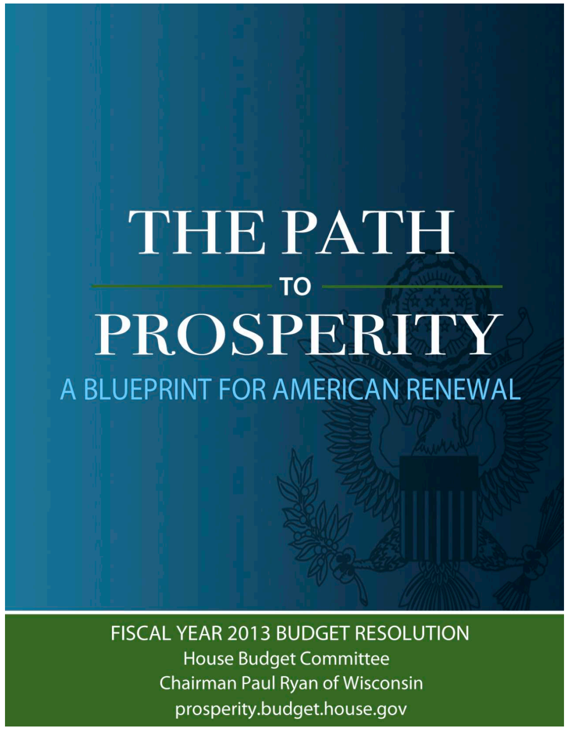


Growth in a Time of Debt
Carmen M. Reinhart and Kenneth S. Rogoff
NBER Working Paper No. 15639
January 2010, Revised January 2010
JEL No. E2,E3,E6,F3,F4,N10

ABSTRACT

We study economic growth and inflation at different levels of government and external debt. Our analysis is based on new data on forty-four countries spanning about two hundred years. The dataset incorporates over 3,700 annual observations covering a wide range of political systems, institutions, exchange rate arrangements, and historic circumstances. Our main findings are: First, the relationship between government debt and real GDP growth is weak for debt/GDP ratios below a threshold of 90 percent of GDP. Above 90 percent, median growth rates fall by one percent, and average growth falls considerably more. We find that the threshold for public debt is similar in advanced and emerging economies. Second, emerging markets face lower thresholds for external debt (public and private)—which is usually denominated in a foreign currency. When external debt reaches 60 percent of GDP, annual growth declines by about two percent; for higher levels, growth rates are roughly cut in half. Third, there is no apparent contemporaneous link between inflation and public debt levels for the advanced countries as a group (some countries, such as the United States, have experienced higher inflation when debt/GDP is high). The story is entirely different for emerging markets, where inflation rises sharply as debt increases.

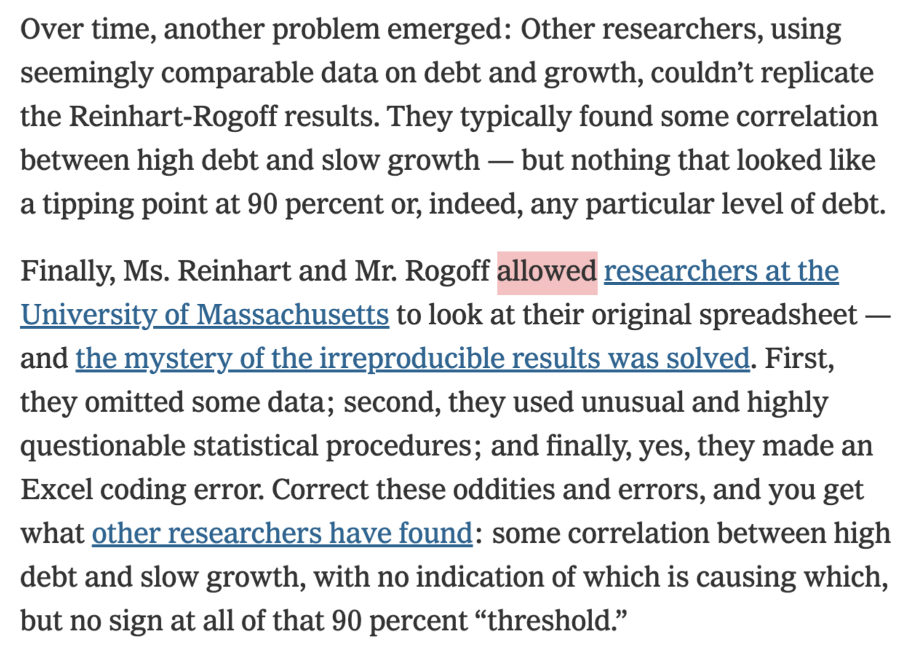
Dept:GDP ratio 90%+ → -0.1% growth


THE PATH
TO
PROSPERITY
A BLUEPRINT FOR AMERICAN RENEWAL
FISCAL YEAR 2013 BUDGET RESOLUTION
House Budget Committee
Chairman Paul Ryan of Wisconsin
prosperity.budget.house.gov

Paul Ryan's 2013 House budget resolution

Excel failures

Thomas Herndon



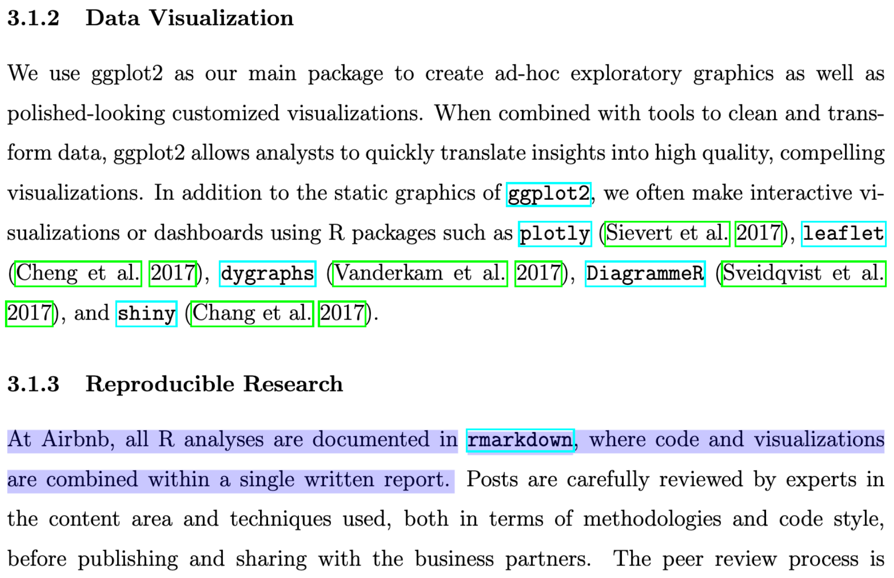
Over time, another problem emerged: Other researchers, using seemingly comparable data on debt and growth, couldn't replicate the Reinhart-Rogoff results. They typically found some correlation between high debt and slow growth — but nothing that looked like a tipping point at 90 percent or, indeed, any particular level of debt.

Finally, Ms. Reinhart and Mr. Rogoff allowed researchers at the University of Massachusetts to look at their original spreadsheet — and the mystery of the irreproducible results was solved. First, they omitted some data; second, they used unusual and highly questionable statistical procedures; and finally, yes, they made an Excel coding error. Correct these oddities and errors, and you get what other researchers have found: some correlation between high debt and slow growth, with no indication of which is causing which, but no sign at all of that 90 percent "threshold."

From Paul Krugman, "The Excel Depression"

Engineering model in real life

Private companies and governments use the engineering model to write reports on data



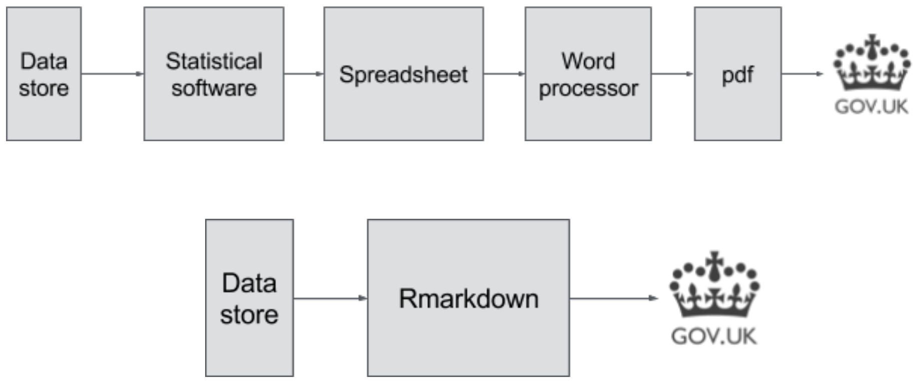
3.1.2 Data Visualization

We use ggplot2 as our main package to create ad-hoc exploratory graphics as well as polished-looking customized visualizations. When combined with tools to clean and transform data, ggplot2 allows analysts to quickly translate insights into high quality, compelling visualizations. In addition to the static graphics of `ggplot2`, we often make interactive visualizations or dashboards using R packages such as `plotly` (Sievert et al. 2017), `leaflet` (Cheng et al. 2017), `dygraphs` (Vanderkam et al. 2017), `DiagrammeR` (Sveidqvist et al. 2017), and `shiny` (Chang et al. 2017).

3.1.3 Reproducible Research

At Airbnb, all R analyses are documented in `rmarkdown`, where code and visualizations are combined within a single written report. Posts are carefully reviewed by experts in the content area and techniques used, both in terms of methodologies and code style, before publishing and sharing with the business partners. The peer review process is

Airbnb

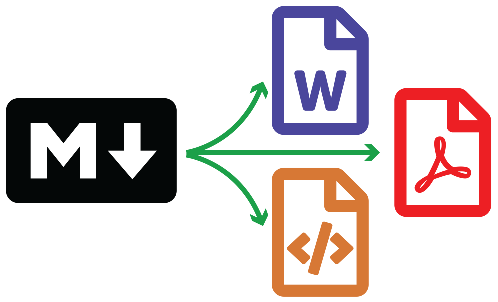


The UK's reproducible analysis pipeline

So, what is Markdown?

- There are many different typesetting languages in which you can write.
- The most widely used ones are perhaps LaTeX (mostly scientific work), HTML (web-based stuff) and Word.

- How to get around learning all of them and having to switch for different outputs?
  - Write in one simplified syntax (Markdown)
  - Convert to whatever output you want



And what is RMarkdown/Quarto?

- Quarto and RMarkdown are publishing systems which use markdown language.
- They allow you to render to different outcome formats

```
# To HTML
quarto render manuscript.qmd --to html

# To Word
quarto render manuscript.qmd --to docx

# To PDF (through LaTeX)
quarto render manuscript.qmd --to pdf
```

- They allow to combine code, figures, tables, images, text etc. (more on that now)

Overview

1. **Getting Started**
    - About R
    - The R Studio IDE
    - Import and eyeball data
    - Use functions
2. **Anatomy of a `data.frame`**
    - Data structure
    - Classes
    - Vectors
    - Subsetting
3. **Wrap Up**
    - Summary and key take-aways
4. **Markdown and universal writing**
    - Office Model vs. Engineering Model
    - Excel failures
    - Markdown
5. **Writing reports in Quarto**
    - What is Quarto?
    - YAML header
    - Code chunks
    - Text formatting
    - Run and render your code
    - Inline code
    - Tables
    - Preset themes
    - Report parameters

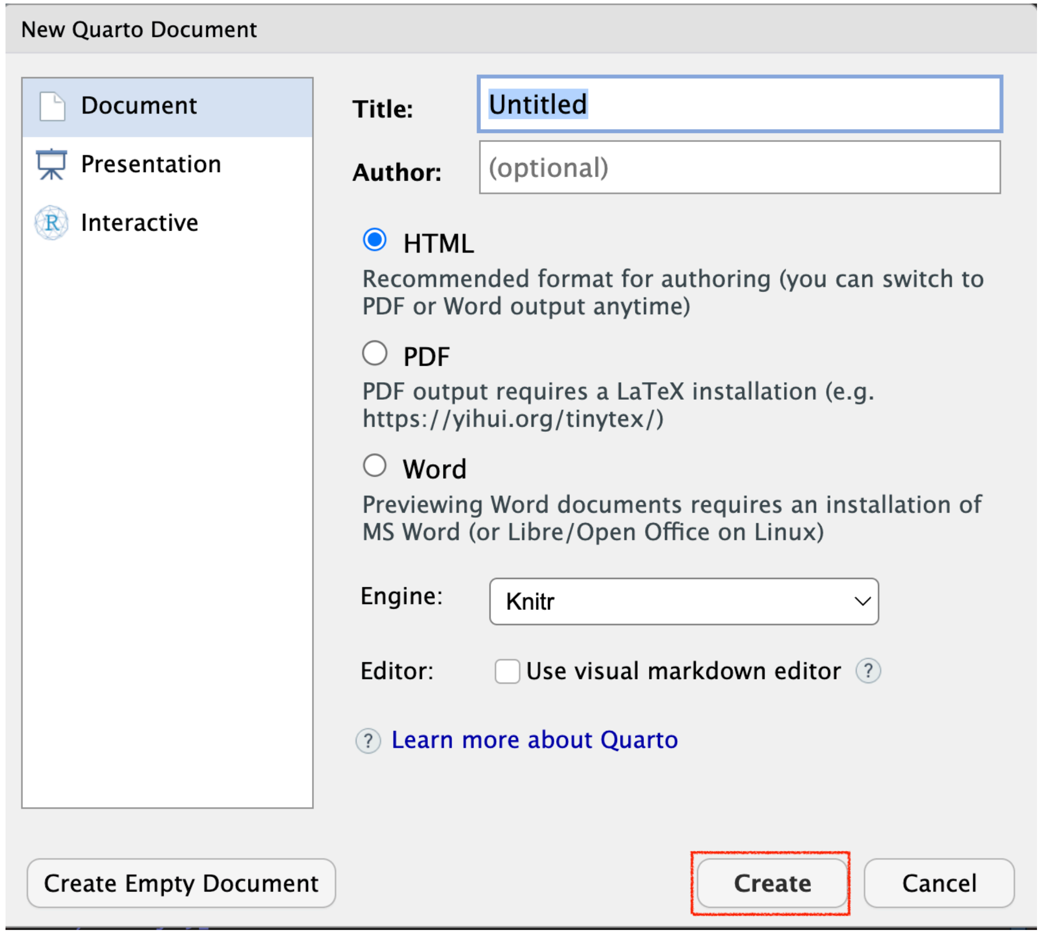
Writing reports in Quarto

What is Quarto?

- Quarto is an open-source publishing system in which you can both write/run code (R/Python/Julia/Observable) and edit text
- Quarto is the newer, fancier version of RMarkdown (which only worked with R code)
- It is structured around 3 types of content:
  - Code chunks to run and render the output
  - Editable text to display
  - YAML metadata for the Quarto build process

What is Quarto?

- Let’s create our first Quarto document!
- Click on File > New File > Quarto document


New Quarto Document

Title: Untitled

Author: (optional)

● HTML
Recommended format for authoring (you can switch to PDF or Word output anytime)

○ PDF
PDF output requires a LaTeX installation (e.g. https://yihui.org/tinytex/)

○ Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux)

Engine: Knitr

Editor: ☐ Use visual markdown editor

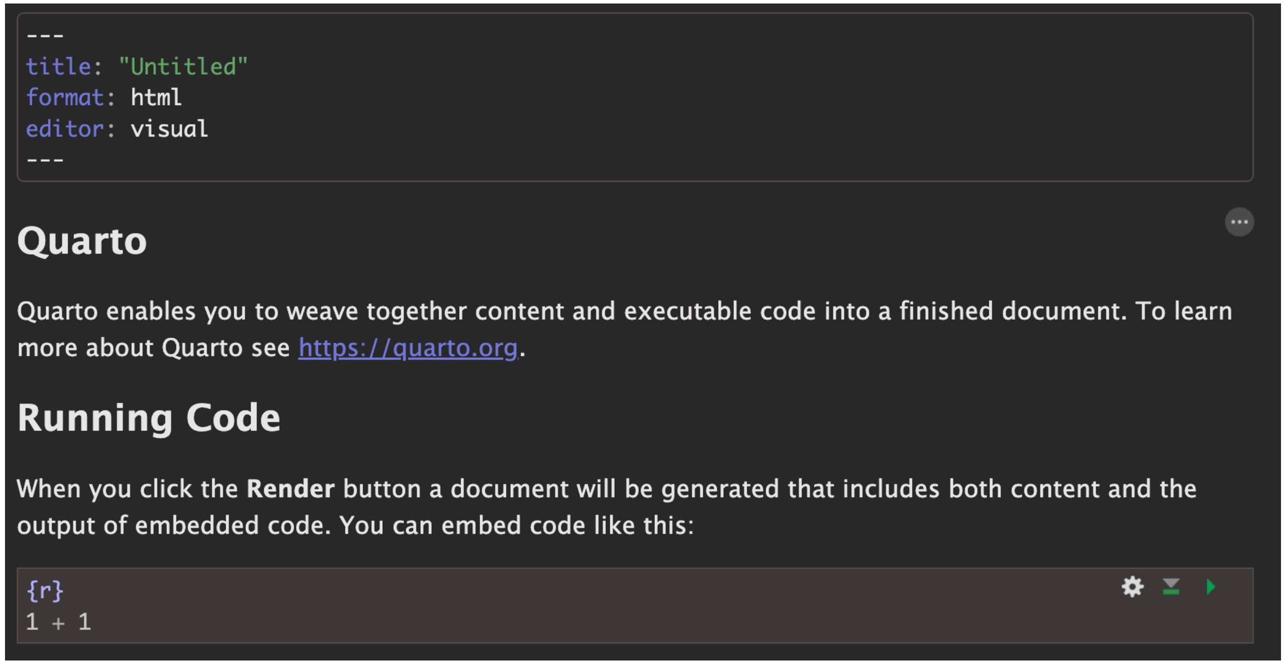
What is Quarto?

It creates a template containing the 3 types of content:

YAML header

Text

Code Chunk



```
---
title: "Untitled"
format: html
editor: visual
---
```

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
{r}
1 + 1
```

Basic principles

YAML Header

- The YAML header contains general information related to the file configuration:
  - Title/subtitle (in quotes)
  - Author/date (in quotes)
  - Output type (html/pdf)
  - Editor configuration (use source, not visual)
  - …
- It should be specified at the very beginning of the document and surrounded by three dashes like this:

```yaml
---
title: "My first Quarto document"
subtitle: "What a blast"
author: "My Name"
date: "05/01/2024"
format: html
editor: source
---
```

Basic principles

Code Chunks

- Code chunks are blocks of R code that can be run when working on and rendering the .qmd file

- You can insert a code chunk using `Ctrl + Alt + i` or by typing the backticks chunk delimiters as follows

```r
1 + 1
```

- When rendering the document, R will execute the code
  - Both the code and the output will appear in the document like so

```r
1 + 1
```

```
[1] 2
```

Basic principles

Code Chunks

- The **content** to be **displayed** from the code chunk can be specified in **chunk options**
  - For instance, to display only the output and not the code chunk, you can set `echo` to `FALSE`

~~~
```{r, echo = F}
1+1
```
~~~

~~~
```{r}
#| echo: false
1+1
```
~~~

- And the output will only be

```
[1] 2
```

- Instead of

```r
1 + 1
```

```
[1] 2
```

Basic principles

Code Chunks

Chunk Options to Know

Option	Default	Effect
eval	TRUE	Whether to evaluate the code and include its results
echo	TRUE	Whether to display code along with its results
warning	TRUE	Whether to display warnings
error	TRUE	Whether to display errors
message	TRUE	Whether to display messages
results	'markup'	'hide' to hide the output
fig.width	7	Width in inches for plots created in chunk
fig.height	7	Height in inches for plots created in chunk

Basic principles

Code Chunks

- For an option to be the default for the whole document, set it up in the YAML header:

```yaml
---
title: "My first Quarto document"
format: html
execute:
  echo: false
  warning: false
---
```

Basic principles

Text Formatting

- Quarto is not only about rendering code but also about **writing** actual **text**
- You can write paragraphs as you would normally do on a typical report
- And Quarto provides convenient ways to format your text
- Unlike most text editing software, in source Quarto text formatting isn't about clicking on dedicated buttonds
- It relies on symbols that should be written along with the text

Basic principles

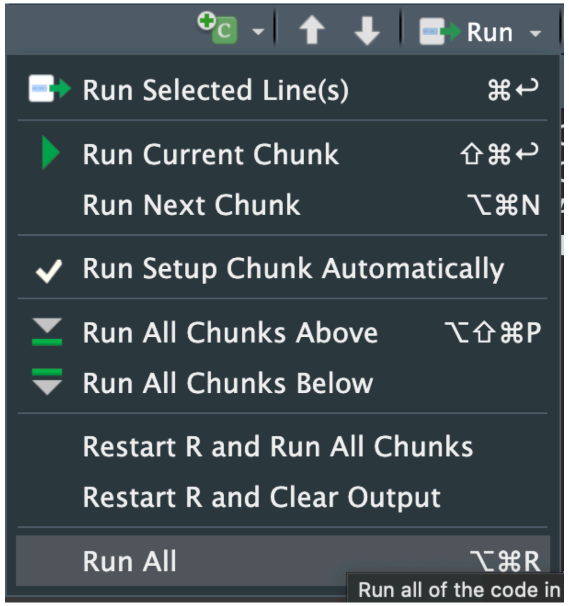
Text Formatting

Type...	...to get
Some text in a paragraph. More text in the next paragraph. Always use empty lines between paragraphs.	Some text in a paragraph. More text in the next paragraph. Always use empty lines between paragraphs.
Italic or _Italic_	*Italic*
Bold or __Bold__	**Bold**
# Heading 1	Heading 1
## Heading 2	Heading 2
### Heading 3	Heading 3
(Go up to heading level 6 with ######)	
[Link text](https://www.example.com)	Link text

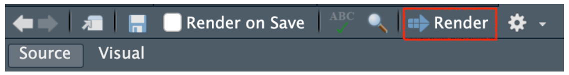
Basic principles

Run and render your code

- You have different options to execute the content of a code chunk in Quarto
  - Check out the buttons at the top right of the chunk



- To render a Quarto file, click on the render button



Useful features

Inline code

- Quarto allows to include R output directly in text
- To do this, use `` `r r_code_here` ``

~~~
```{r}
#| label: find-avg-mpg
#| echo: false
number_of_days <- 5
```

We are `r number_of_days` days into the week.
~~~

… would render to this:

> We are 5 days into the week.

That’s it for today :)