Quarto & LaTeX

Lecture 5

Louis SIRUGUE

CPES 2 - Fall 2023

Quick reminder

The 3 core components of the ggplot() function

Component	Contribution	Implementation
Data	Underlying values	ggplot(data, | data %>% ggplot(.,
Mapping	Axis assignment	aes(x = V1, y = V2, ...))
Geometry	Type of plot	+ geom_point() + geom_line() + ...

- Any **other element** should be added with a `+` **sign**

```
ggplot(data, aes(x = V1, y = V2)) +
  geom_point() + geom_line() +
  anything_else()
```

Quick reminder

Main customization tools

Item to customize	Main functions
Axes	scale_[x/y]_[continuous/discrete]
Baseline theme	theme_[void/minimal/.../dark]()
Annotations	geom_[[h/v]line/text](), annotate()
Theme	theme(axis.[line/ticks].[x/y] = ...,

Main types of geometry

Geometry	Function
Bar plot	geom_bar()
Histogram	geom_histogram()
Area	geom_area()
Line	geom_line()
Density	geom_density()
Boxplot	geom_boxplot()
Violin	geom_violin()
Scatter plot	geom_point()

Quick reminder

Main types of aesthetics

Argument	Meaning
alpha	opacity from 0 to 1
color	color of the geometry
fill	fill color of the geometry
size	size of the geometry
shape	shape for geometries like points
linetype	solid, dashed, dotted, etc.

- If specified **in the geometry**
  - It will apply uniformly to **all the geometry**
- If assigned to a variable **in aes**
  - It will **vary with the variable** according to a scale documented in legend

```
ggplot(data, aes(x = V1, y = V2, size = V3)) +
  geom_point(color = "steelblue", alpha = .6)
```

Warm up practice

- Today we're going to use the ***"Fichier des prénoms"***
  - This is where the INSEE reports the **birth count** associated with **each first name in France**
  - It is **virtually exhaustive from 1946**, when the INSEE was founded

```
names <- read.csv("C:/User/Documents/fichier_prenoms.csv", sep = ";", encoding = "UTF-8")
str(names)
```

```
## 'data.frame':    686538 obs. of  4 variables:
##  $ sexe    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ preusuel: chr  "_PRENOMS_RARES" "_PRENOMS_RARES" "_PRENOMS_RARES" "_PRENOMS_RARES" ...
##  $ annais  : chr  "1900" "1901" "1902" "1903" ...
##  $ nombre  : int  1249 1342 1330 1286 1430 1472 1451 1514 1509 1526 ...
```

- `sexe` `1` for Male and `2` for Female
- `preusuel` first name (`_PRENOMS_RARES` *gathers rare first names for anonymity considerations)*
- `annais` birth year (`XXXX` *groups unknown birth years)*
- `nombre` number of newborns for the corresponding sex/name/year

Warm up practice

10:00

1) Recode the `sexe` variable with `Male` and `Female` instead of `1` and `2`

2) Filter out observations for which `annais` is `XXXX` and convert `annais` to numeric

3) Summarise your data into the total number of births per year

4) Plot the evolution of the number of births over time using a line geometry

You've got 10 minutes!

Solution

Load the necessary packages

```
library(dplyr)
library(ggplot2)
```

1) Recode the `sexe` variable with `Male` and `Female` instead of `1` and `2`

```
names %>%
  mutate(sexe = ifelse(sexe == 1, "Male", "Female"))
```

2) Filter out observations for which `annais` is `XXXX` and convert `annais` to numeric

```
names %>%
  mutate(sexe = ifelse(sexe == 1, "Male", "Female")) %>%
  filter(annais != "XXXX") %>%
  mutate(annais = as.numeric(annais))
```

Solution

3) Summarise your data into the total number of births per year

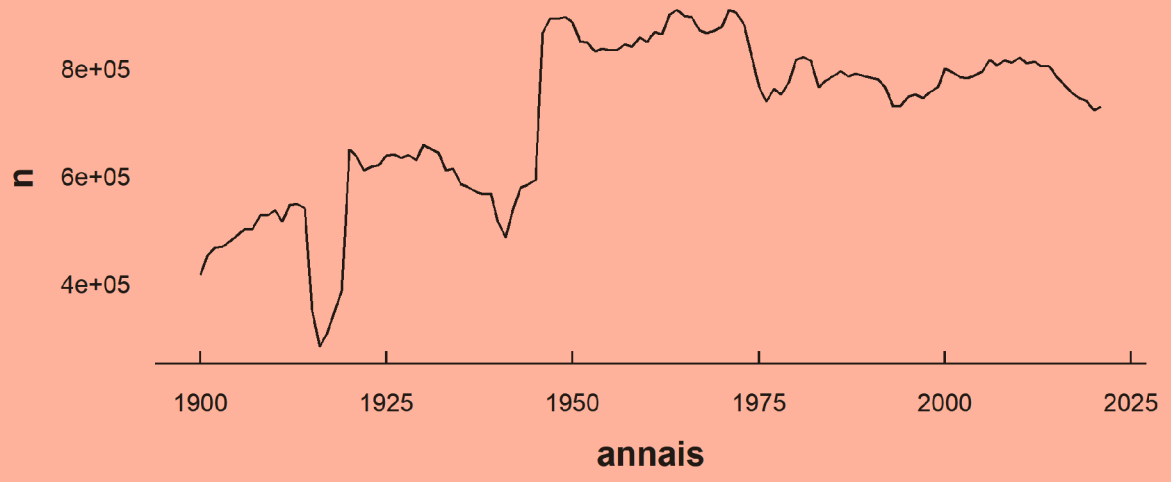
```
names %>%
  mutate(sexe = ifelse(sexe == 1, "Male", "Female")) %>%
  filter(annais != "XXXX") %>%
  mutate(annais = as.numeric(annais)) %>%
  group_by(annais) %>%
  summarise(n = sum(nombre))
```

```
## # A tibble: 8 × 2
##   annais      n
##    <dbl>  <int>
## 1   1900 415040
## 2   1901 453456
## 3   1902 465791
## 4   1903 468810
## 5   1904 478962
## 6   1905 489697
## 7   1906 501745
## 8   1907 501025
```

Solution

4) Plot the evolution of the number of births over time using a line geometry

```
names %>%
  mutate(sexe = ifelse(sexe == 1, "Male", "Female")) %>%
  filter(annais != "XXXX") %>%
  mutate(annais = as.numeric(annais)) %>%
  group_by(annais) %>%
  summarise(n = sum(nombre)) %>%
  ggplot(aes(x = annais, y = n)) + geom_line()
```



Today we learn how to make reports with Quarto!

Today we learn how to make reports with Quarto!

1. Basic principles

1.1. 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**

- Here are some examples of Quarto documents
  - Homework
  - Website
  - Slides

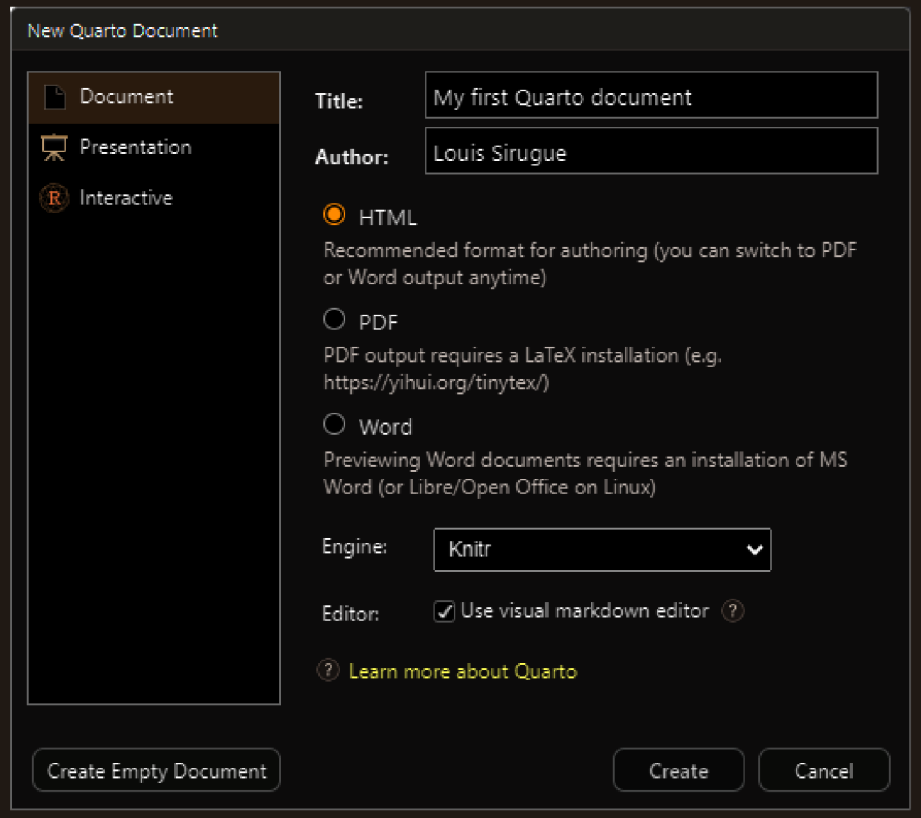
- It is structured around **3 types of content**:
  - **Code chunks** to run and render the output
  - **Editable text** to display
  - **YAML metadata** for the Quarto build process

➜ Let's go through them by creating our first Quarto document!

1. Basic principles

1.1. What is Quarto?

→ Click on **File > New File > Quarto document**



1. Fill out the information and select **HTML**

2. Click on **OK**

1. Basic principles

1.1. What is Quarto?

- It creates a **template** containing the **3 types of content**:

YAML header →

Text →

Code chunks →

```yaml
---
title: "My first Quarto document"
author: "Louis Sirugue"
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
```

1. Basic principles

1.2. 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:

```
---
title: "My first Quarto document"
subtitle: "A step-by-step introduction"
author: "Louis Sirugue"
date: "10/02/2023"
format: html
editor: source
---
```

1. Basic principles

1.3. 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:

```
1+1
```

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

1. Basic principles

1.3. 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
```
```

- And the output will only be

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

- Instead of

```
1+1
```

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

1. Basic principles

1.3. 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

1. Basic principles

1.3. Code chunks

- For an option to apply to the **whole document**, set it up in the YAML header:

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

- For an option to apply to a **specific chunk**, two possibilities:

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

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

1. Basic principles

1.4. 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

- Basic formatting includes:
  - Italics
  - Bold
  - Hyperlinks
  - Headers
  - Block quotes
  - Un/ordered lists
  - ...

- Unlike most text editing software, in *source* Quarto **text formatting** isn't about clicking on dedicated buttons
  - It **relies on symbols** that should be written along with the text

1. Basic principles

1.4. Text formatting

Syntax	Output
`Plain text` `End a line with two spaces for line break`	Plain text End a line with two spaces for line break
`*italics*`	*italics*
`**bold**`	**bold**
`# Header 1`	# Header 1
`## Header 2`	## Header 2
...	...
`###### Header 6`	###### Header 6
`[link](https://www.rstudio.com)`	link

1. Basic principles

1.4. Text formatting

Syntax

```
> block quote
```

Horizontal rule:

```
***
```

```
* unordered list
* item 2
  + sub-item 1
  + sub-item 2

1. ordered list
2. item 2
  + sub-item 1
  + sub-item 2
```

Output

> block quote

Horizontal rule:

- unordered list
- item 2
  - sub-item 1
  - sub-item 2

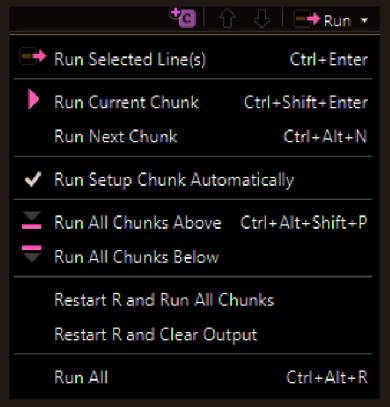
1. ordered list
2. item 2
   - sub-item 1
   - sub-item 2

1. Basic principles

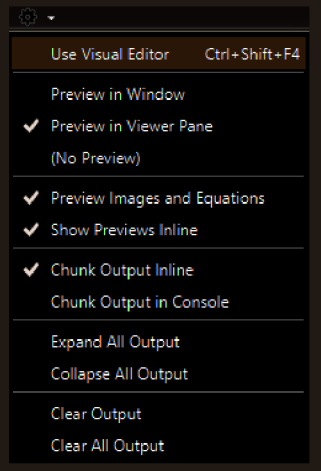
1.5. Run and render your code

- To **execute** the content of a **code** chunk in Quarto
  - Click on the **green play button** at the top right of the chunk

- You can also:
  - **Run all chunks above** the current chunk
  - **Run all chunks** from the Run drop down menu at the top right (or Ctrl+Alt+R)



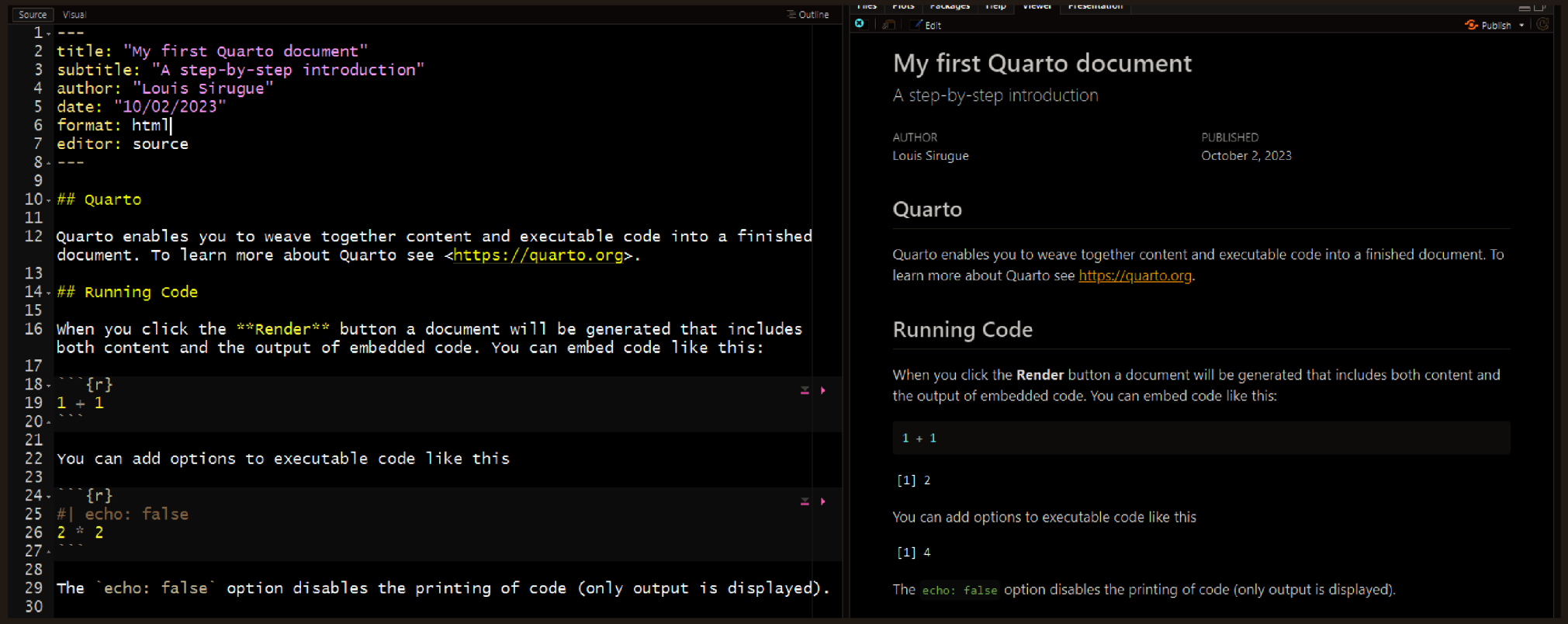
- To choose where the **output** must be **displayed**, click on the *"Options"* button
  - **Chunk output inline** (below the chunk)
  - **Chunk output in console**



1. Basic principles

1.5. Run and render your code

- To **render** a Quarto file, click on the **render button** Render (`ctrl + shift + k`)



Overview

Overview

2. Useful features

2.1. Inline code

- A big advantage of Quarto is that you can **automate** your **reports**

Why is it useful?

- You might figure out quite late in the process that you need to **make a change** at the beginning of the analysis
  - A change that potentially **impacts everything** that comes after in the report

- Imagine that you forgot to filter out an irrelevant group of observations at the beginning
  - If you simply filter your data at the beginning in a code chunk
  - All your tables and figures will **update automatically**

- But what if you wrote some of your results **within paragraphs?**
  - In a usual text formatting software you would have to update everything manually
  - But here you can also make it **update automatically!**

2. Useful features

2.1. Inline code

- Consider the following report :

Source Visual Outline

```
---
title: "Report example"
subtitle: "Motivation for using inline code in Quarto"
author: "Louis Sirugue"
date: "10-02-2023"
format: html
editor: source
---

## Data description

```{r}
dt <- data.frame(Name = c("Bob", "Emmy", "Mark", "Ann", "Tom", "Sam"),
                 Height = c(1.72, 1.80, 1.94, 1.60, .190, 1.84))

mean(dt$Height)
```

The average height in the sample is 1.5 meters.
```

Edit Publish

Report example

Motivation for using inline code in Quarto

AUTHOR
Louis Sirugue

PUBLISHED
October 2, 2023

Data description

```
dt <- data.frame(Name = c("Bob", "Emmy", "Mark", "Ann", "Tom", "Sam"),
                 Height = c(1.72, 1.80, 1.94, 1.60, .190, 1.84))

mean(dt$Height)
```

```
[1] 1.515
```

The average height in the sample is 1.5 meters.

2. Useful features

2.1. Inline code

- Imagine that there is a problem with the data and that you must use an updated version

Source Visual Outline

```
---
title: "Report example"
subtitle: "Motivation for using inline code in Quarto"
author: "Louis Sirugue"
date: "10-02-2023"
format: html
editor: source
---

## Data description

```{r}
dt <- data.frame(Name = c("Bob", "Emmy", "Mark", "Ann", "Tom", "Sam"),
                 Height = c(1.72, 1.80, 1.94, 1.60, 1.90, 1.84))

mean(dt$Height)
```

The average height in the sample is 1.5 meters.
```

Edit Publish

Report example

Motivation for using inline code in Quarto

AUTHOR
Louis Sirugue

PUBLISHED
October 2, 2023

Data description

```r
dt <- data.frame(Name = c("Bob", "Emmy", "Mark", "Ann", "Tom", "Sam"),
                 Height = c(1.72, 1.80, 1.94, 1.60, 1.90, 1.84))

mean(dt$Height)
```

```
[1] 1.8
```

The average height in the sample is 1.5 meters.

2. Useful features

2.1. Inline code

- All the results were updated automatically but not the text
  - That's where **inline code** comes in!

→ **Inline code** allows to include the output of some **R code within text areas** of your report

- R code outside code chunks should be included between backticks:
  - Surrounding code with **backticks** in a text area will **change** the **format** to that of the code chunk
  - **Adding** the **`r`** letter right after the first backtick will **show** the **output** of the code instead of the code

Syntax	Output
`` `paste("a", "b", sep = "-")` ``	`paste("a", "b", sep = "-")`
`` `r paste("a", "b", sep = "-")` ``	a-b

2. Useful features

2.1. Inline code

- With inline code, **paragraphs** also do **update automatically**:

Source Visual Outline

```
---
title: "Report example"
subtitle: "Motivation for using inline code in Quarto"
author: "Louis Sirugue"
date: "10-02-2023"
format: html
editor: source
---

## Data description

```{r}
dt <- data.frame(Name = c("Bob", "Emmy", "Mark", "Ann", "Tom", "Sam"),
                 Height = c(1.72, 1.80, 1.94, 1.60, 1.90, 1.84))

mean(dt$Height)
```

The average height in the sample is `r mean(dt$Height)` meters.
```

Edit Publish

Report example

Motivation for using inline code in Quarto

AUTHOR
Louis Sirugue

PUBLISHED
October 2, 2023

Data description

```
dt <- data.frame(Name = c("Bob", "Emmy", "Mark", "Ann", "Tom", "Sam"),
                 Height = c(1.72, 1.80, 1.94, 1.60, 1.90, 1.84))

mean(dt$Height)
```

```
[1] 1.8
```

The average height in the sample is 1.8 meters.

2. Useful features

2.2. Tables

- Displaying a table as a raw output can be unpleasant to read

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

- The `kable()` function from the `knitr` package allows to display tables in a nice way

```
library("knitr")
```

2. Useful features

2.2. Tables

- You just need to put the table you want to display inside the `kable()` function

```
kable(head(mtcars), caption = "First rows of the dataset")
```

First rows of the dataset

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.21	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1

2. Useful features

2.2. Tables

- For **big tables**, one solution is the `datatable()` function from the `DT` package

- As with `kable()`, you just need to put the table you want to display inside the `datatable()` function

```
library("DT")
datatable(mtcars)
```

- The output will be an **interactive table** which allows to:
  - Navigate in the table by displaying a limited number of rows at a time
  - Choose the number of rows to display
  - Search for a given element in the table

- You can select the default number of rows to display as follows

```
datatable(mtcars, options = list(pageLength = 5))
```

2. Useful features

2.2. Tables

Show 5 entries Search:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2

Showing 1 to 5 of 32 entries Previous 1 2 3 4 5 6 7 Next

→ *Try to search for **"Toyota"** for instance*

2. Useful features

2.3. Preset themes

- The **default theme** of Quarto might seem **a bit dull**
  - The look of your reports can easily be **enhanced** using a variety of **preset** themes
  - The preset theme to use should be specified in the **YAML header**
  - Add a theme argument to the html_document format specified as output

```
---
title: "My first Quarto document"
subtitle: "A step-by-step introduction"
author: "Louis Sirugue"
date: "10/02/2023"
format: html
editor: source
theme: "cosmo"
---
```

- When using themes from downloaded packages, the way you set the theme can be slightly different
  - Check the online documentation

2. Useful features

2.3. Preset themes

yeti

Typography

Heading 1
Heading 2
Heading 3
Heading 4
Heading 5
Heading 6
Heading with faded secondary text

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor.

Example body text

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

This line of text is meant to be treated as fine print.

The following is **rendered as bold text**.

The following is *rendered as italicized text*.

An abbreviation of the word attribute is attr.

sandstone

Typography

Heading 1
Heading 2
Heading 3
Heading 4
Heading 5
Heading 6
Heading with faded secondary text

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor.

Example body text

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

This line of text is meant to be treated as fine print.

The following is **rendered as bold text**.

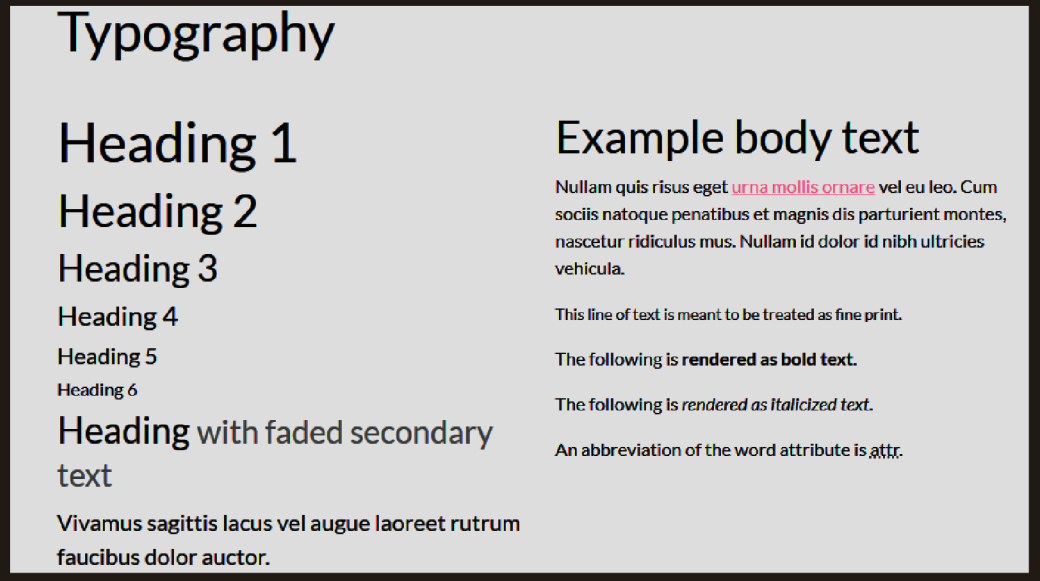
The following is *rendered as italicized text*.

An abbreviation of the word attribute is attr.

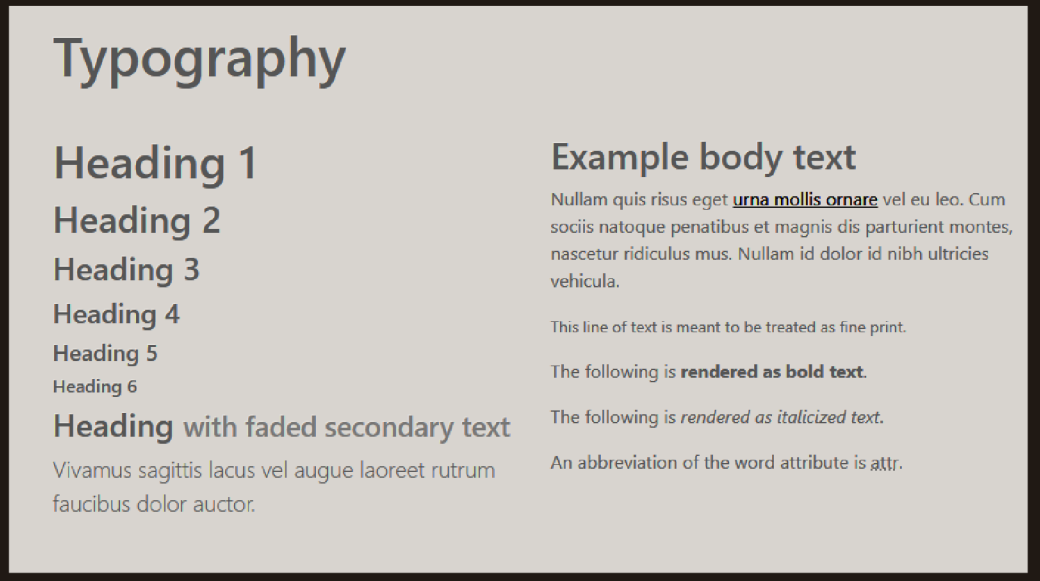
2. Useful features

2.3. Preset themes

darkly



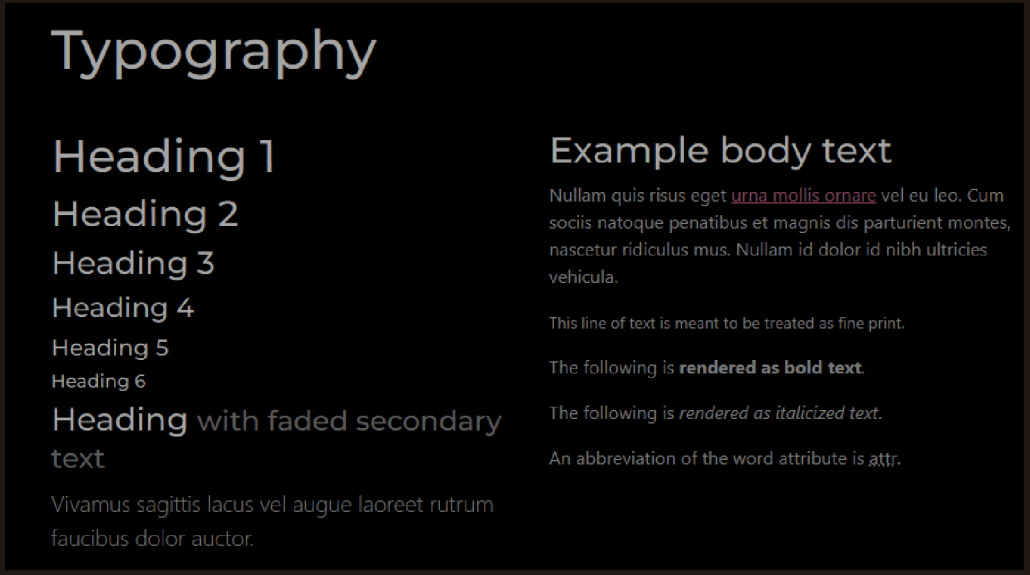
slate



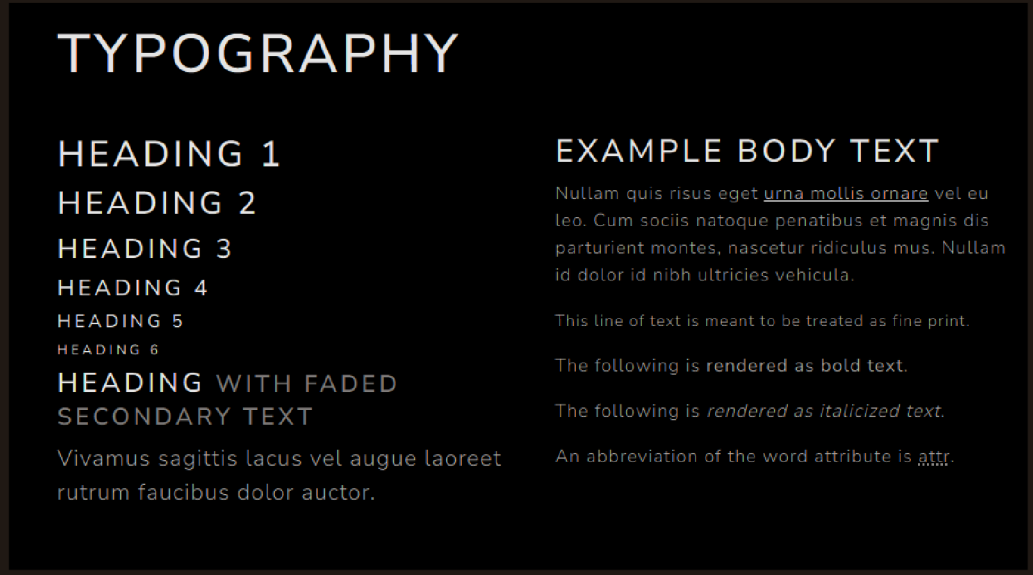
2. Useful features

2.3. Preset themes

minty



lux



Practice

Reproduce the following html using Quarto

Copy raw output

You've got 15 minutes!

15:00

Report on the first name LOUIS

AUTHOR
Your name

PUBLISHED
October 2, 2023

1. Setup

The packages needed in a qmd must *always* be loaded in a code chunk at the beginning of the file.

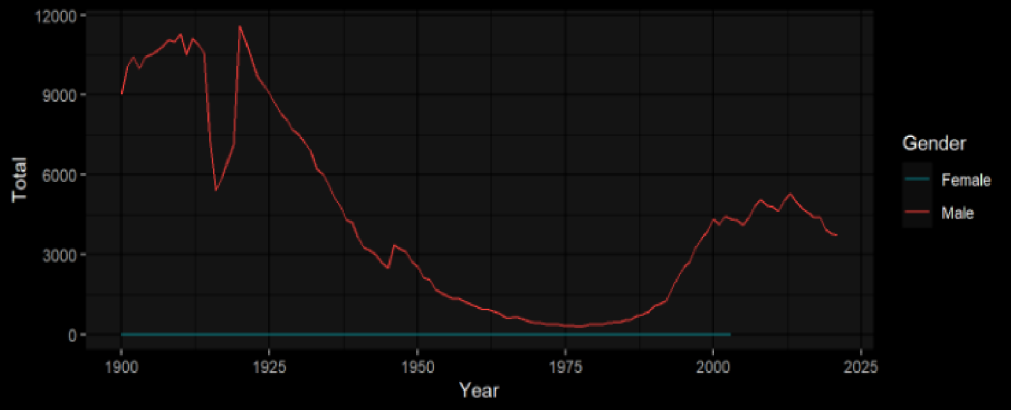
```
library(dplyr)
library(ggplot2)
```

However, the command `install.packages()` must **not** be written in a Quarto document.
It should be run only once in the console.

2. Data cleaning

```
names <- read.csv('fichier_prenoms.csv', encoding = 'UTF-8', sep = ';') %>%
  mutate(Gender = ifelse(sexe == 1, 'Male', 'Female')) %>%
  filter(annais != 'XXXX') %>%
  mutate(Year = as.numeric(annais))
```

3. Evolution of the first name LOUIS over time



3715 children were born under the name LOUIS in 2021. This statistic is written in inline code such that it updates automatically.

Solution

```
---
title: "Report on the first name LOUIS"
author: "Your name"
date: "10-02-2023"
format: html
editor: source
theme: "cosmo"
---

### 1. Setup

The packages needed in a qmd must *always* be loaded in a code chunk at the beginning of the file.

```{r, message = F, warning = F}
library(dplyr)
library(ggplot2)
```

However, the command `install.packages()` must **not** be written in a Quarto document.
It should be run only once in the console.
```

Solution

```
### 2) Data cleaning

```{r}
names <- read.csv('fichier_prenoms.csv', encoding = 'UTF-8', sep = ';') %>%
  mutate(Gender = ifelse(sexe == 1, 'Male', 'Female')) %>%
  filter(annais != 'XXXX') %>%
  mutate(Year = as.numeric(annais))
```

3) Evolution of the first name LOUIS over time

```{r, echo = F, message = F, fig.height = 3, fig.width = 8}
names %>% filter(preusuel == "LOUIS") %>%
  group_by(Year, Gender) %>%
  summarise(Total = sum(nombre)) %>% ungroup() %>%
  ggplot(aes(x = Year, y = Total, color = Gender)) + geom_line()
```

```{r, echo = F}
n_louis <- names %>% filter(Year == 2021 & preusuel == "LOUIS") %>% summarise(n = sum(nombre))
```

`r n_louis$n` children were born under the name LOUIS in 2021. This statistic is written in
inline code such that it updates automatically.
```

2. Useful features

2.4. Report parameters

- It may sometimes be useful to produce **separate html reports for differents groups** in your data
  - Country/state-specific reports
  - Here, a different report for each first name

- **YAML parameters** are very useful for that
  - They are accessible **like any object** in your environment
  - They must be specified as follows

```
title: "Report on the first name `r params$name`"
author: "Your name"
date: "10-02-2023"
format: html
editor: source
theme: "cosmo"
params:
   name: "LOUIS"
```

2. Useful features

2.4. Report parameters

- You simply have to call that object in your code chunks or inline code when needed

```
### 3) Evolution of the first name `r params$name` over time

```{r, echo = F, message = F, fig.height = 3}
  names %>% filter(preusuel == params$name) %>%
  group_by(Year, Gender) %>%
  summarise(Total = sum(nombre)) %>% ungroup() %>%
  ggplot(aes(x = Year, y = Total, color = Gender)) + geom_line()
```


```{r, echo = F}
  n_louis <- names %>% filter(Year == 2021 & preusuel == params$name) %>% summarise(n = sum(nombre))
```

`r n_louis$n` children were born under the name `r params$name` in 2021. This statistic is
written in inline code such that it updates automatically.
```

➜ Let's render our .qmd with different values of that parameter!

Report on the first name LOUIS

AUTHOR
Your name

PUBLISHED
October 2, 2023

1. Setup

The packages needed in a qmd must *always* be loaded in a code chunk at the beginning of the file.

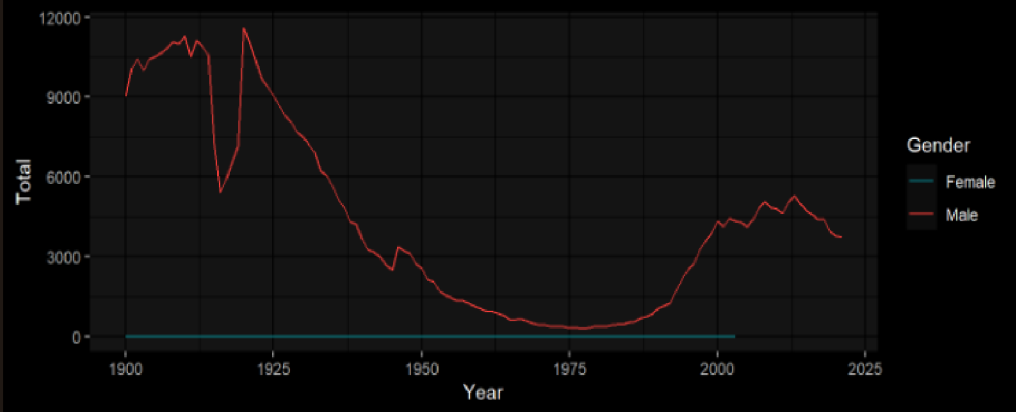
```
library(dplyr)
library(ggplot2)
```

However, the command `install.packages()` must **not** be written in a Quarto document.
It should be run only once in the console.

2. Data cleaning

```
names <- read.csv('fichier_prenoms.csv', encoding = 'UTF-8', sep = ';') %>%
  mutate(Gender = ifelse(sexe == 1, 'Male', 'Female')) %>%
  filter(annais != 'XXXX') %>%
  mutate(Year = as.numeric(annais))
```

3. Evolution of the first name LOUIS over time



3715 children were born under the name LOUIS in 2021. This statistic is written in inline code such that it updates automatically.

Report on the first name DIDIER

AUTHOR
Your name

PUBLISHED
October 2, 2023

1. Setup

The packages needed in a qmd must *always* be loaded in a code chunk at the beginning of the file.

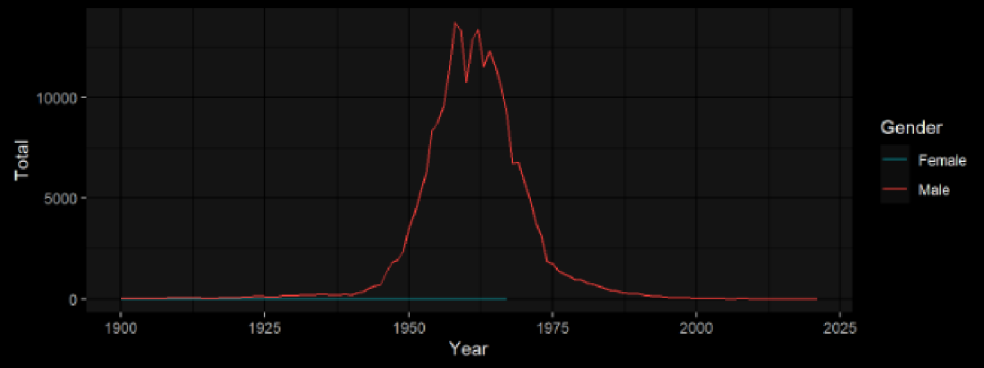
```
library(dplyr)
library(ggplot2)
```

However, the command `install.packages()` must **not** be written in a Quarto document.
It should be run only once in the console.

2. Data cleaning

```
names <- read.csv('fichier_prenoms.csv', encoding = 'UTF-8', sep = ';') %>%
  mutate(Gender = ifelse(sexe == 1, 'Male', 'Female')) %>%
  filter(annais != 'XXXX') %>%
  mutate(Year = as.numeric(annais))
```

3. Evolution of the first name DIDIER over time



3 children were born under the name DIDIER in 2021. This statistic is written in inline code such that it updates automatically.

Report on the first name PAULINE

AUTHOR
Your name

PUBLISHED
October 2, 2023

1. Setup

The packages needed in a qmd must *always* be loaded in a code chunk at the beginning of the file.

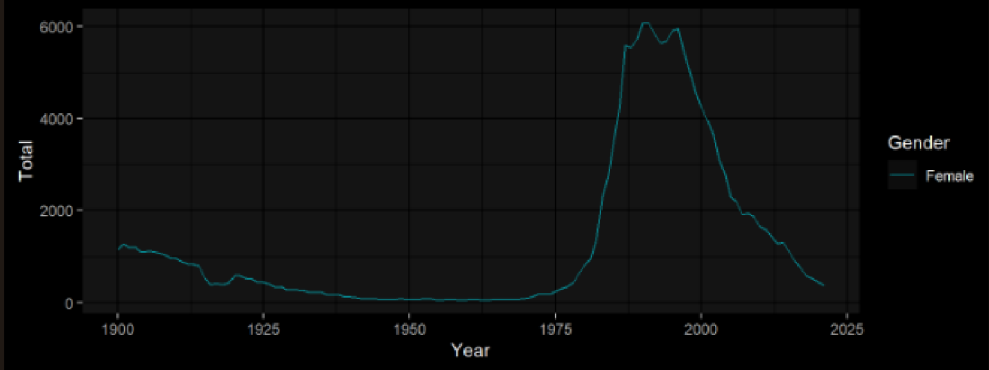
```
library(dplyr)
library(ggplot2)
```

However, the command `install.packages()` must **not** be written in a Quarto document. It should be run only once in the console.

2. Data cleaning

```
names <- read.csv('fichier_prenoms.csv', encoding = 'UTF-8', sep = ';') %>%
  mutate(Gender = ifelse(sexe == 1, 'Male', 'Female')) %>%
  filter(annais != 'XXXX') %>%
  mutate(Year = as.numeric(annais))
```

3. Evolution of the first name PAULINE over time



366 children were born under the name PAULINE in 2021. This statistic is written in inline code such that it updates automatically.

Report on the first name CAMILLE

AUTHOR
Your name

PUBLISHED
October 2, 2023

1. Setup

The packages needed in a qmd must *always* be loaded in a code chunk at the beginning of the file.

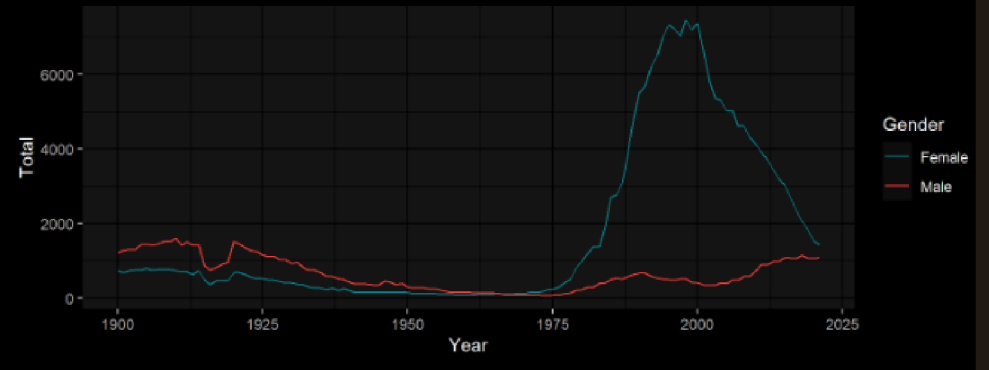
```
library(dplyr)
library(ggplot2)
```

However, the command `install.packages()` must **not** be written in a Quarto document. It should be run only once in the console.

2. Data cleaning

```
names <- read.csv('fichier_prenoms.csv', encoding = 'UTF-8', sep = ';') %>%
  mutate(Gender = ifelse(sexe == 1, 'Male', 'Female')) %>%
  filter(annais != 'XXXX') %>%
  mutate(Year = as.numeric(annais))
```

3. Evolution of the first name CAMILLE over time



2524 children were born under the name CAMILLE in 2021. This statistic is written in inline code such that it updates automatically.

2. Useful features

2.4. Report parameters

- But by **default** the **name of the .html** output will be the name of your .qmd
  - So if you **render report.qmd** for the first name Louis it will save the report under **report.html**
  - And if you **render it a second time** for the first name Didier it will **override** the first .html

- The **solution** is to **render** your .qmd **externally**
  - You can do that with the **render()** function of the rmarkdown package
  - Save your .qmd and **open a new .R script** to try it out

```r
library(rmarkdown)

render(
  input = "C:/User/Documents/prenom.qmd",                    # Specify the input .qmd
  output_file = "C:/User/Documents/LOUIS.html",              # Specify the output file
  params = list(name = "LOUIS")                              # Specify the YAML parameter(s)
)
```

2. Useful features

2.4. Report parameters

- To **avoid copy-pasting** this command for each name we want a report on, we must **use a loop**
  1.
  2.
  3.
  4.

```
for ( in ) {



}
```

2. Useful features

2.4. Report parameters

- To **avoid copy-pasting** this command for each name we want a report on, we must **use a loop**
  1. First we should name the object that will successively take the value of each first name
  2.
  3.
  4.

```
for (i in ) {



}
```

2. Useful features

2.4. Report parameters

- To **avoid copy-pasting** this command for each name we want a report on, we must **use a loop**
  1. First we should name the object that will successively take the value of each first name
  2. Then indicate which values this object must successively take
  3.
  4.

```
for (i in c("LOUIS", "DIDER", "PAULINE", "CAMILLE")) {





}
```

2. Useful features

2.4. Report parameters

- To **avoid copy-pasting** this command for each name we want a report on, we must **use a loop**
  1. First we should name the object that will successively take the value of each first name
  2. Then indicate which values this object must successively take
  3. Then indicate what to do at each iteration
  4.

```
for (i in c("LOUIS", "DIDER", "PAULINE", "CAMILLE")) {

  render(
    input = "C:/User/Documents/prenom.qmd",
    output_file = "C:/User/Documents/LOUIS.html",
    params = list(name = "LOUIS")
  )

}
```

2. Useful features

2.4. Report parameters

- To **avoid copy-pasting** this command for each name we want a report on, we must **use a loop**
  1. First we should name the object that will successively take the value of each first name
  2. Then indicate which values this object must successively take
  3. Then indicate what to do at each iteration
  4. And this should depend on the object that successively take each value

```
for (i in c("LOUIS", "DIDER", "PAULINE", "CAMILLE")) {

  render(
    input = "C:/User/Documents/prenom.qmd",
    output_file = paste0("C:/User/Documents/", i, ".html"),
    params = list(name = i)
  )

}
```

Overview

Overview

3. LaTeX for equations

3.1. What is LaTeX?

- $\LaTeX$ is a document preparation system

- But LaTeX is not a *"what you see is what you get"* system
  - In Microsoft Word or Google doc, you work directly on the "output document"
  - **LaTeX** works more like Quarto: **Edit** your text **in a script using commands and symbols**
    Compile the script to **get the output**

- LaTeX is the **preferred** typesetting system for most **academic** fields mainly because:
  - Many things can be **automated** in LaTeX
  - It has a good way to typeset **mathematical formulas**

- We're not gonna learn how to make $\LaTeX$ documents (do it in 30mn), but just how to make equations

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

3. LaTeX for equations

3.2. LaTeX syntax

- To include a **LaTeX equation** in Quarto, you simply have to surround it with the **`$` sign**:

Syntax	Output
`1 + 1`	1 + 1
`$1 + 1$`	$1 + 1$

- LaTeX is a convenient way to display **mathematical symbols** and to **structure equations**
  - The **syntax** is mainly based on **backslashes \ and braces {}**

Example:

→ What you type in the text area: `$x \neq \frac{\alpha \times \beta}{2}$`

→ What is rendered as an output: $x \neq \frac{\alpha \times \beta}{2}$

3. LaTeX for equations

3.2. LaTeX syntax

→ Common greek letters

Syntax	Output
`$\alpha$`	α
`$\beta$`	β
`$\gamma$ $\Gamma$`	γ Γ
`$\delta$ $\Delta$`	δ Δ
`$\epsilon$ $\varepsilon$`	ϵ ε
`$\lambda \Lambda$`	λ Λ
`$\phi$ $\Phi$`	ϕ Φ
`$\pi$ $\Pi$`	π Π
`$\psi$ $\Psi$`	ψ Ψ
`$\theta$ $\Theta$`	θ Θ
`$\sigma$ $\Sigma$`	σ Σ
...	...

3. LaTeX for equations

3.2. LaTeX syntax

→ Common symbols

Syntax	Output
`$+ - \pm$`	$+ - \pm$
`$\times \div$`	$\times \div$
`$= \neq \equiv \approx$`	$= \neq \equiv \approx$
`$> < \geq \leq \lessgtr$`	$> < \geq \leq \lessgtr$
`$\rightarrow \leftarrow \Leftrightarrow$`	$\rightarrow \leftarrow \Leftrightarrow$
`$\in \notin$`	$\in \notin$
`$\forall \exists \nexists$`	$\forall \exists \nexists$
`$\infty$`	∞
`$\sum \prod \int$`	$\sum \prod \int$
...	...

3. LaTeX for equations

3.2. LaTeX syntax

→ Exponents and accentuation

Syntax	Output
`$x^a$`	x^a
`$x_b$`	x_b
`$x^a_b$`	x^a_b
`$x^{a, i}_{b, j}$`	$x^{a,i}_{b,j}$
`$\hat{\beta} \widehat{\beta_{i,j}}$`	$\hat{\beta}\ \widehat{\beta_{i,j}}$
`$\tilde{\beta} \widetilde{\beta_{i,j}}$`	$\tilde{\beta}\ \widetilde{\beta_{i,j}}$
`$\overline{x} \underline{x}$`	$\overline{x}\ \underline{x}$
`$\overrightarrow{x} \underleftarrow{x}$`	$\overrightarrow{x}\ \underleftarrow{x}$
...	...

3. LaTeX for equations

3.2. LaTeX syntax

→ Math constructs and variable sized symbols

Syntax	Output
`$\frac{a \times b}{c}$`	$\frac{a \times b}{c}$
`$\sqrt{x} \sqrt[n]{x}$`	$\sqrt{x} \sqrt[n]{x}$
`$\sum_{i = 1}^N$`	$\sum_{i = 1}^N$
`$\prod_{i = 1}^N$`	$\prod_{i = 1}^N$
`$\int_a^b$`	$\int_a^b$
`$\overline{x}=\frac{1}{N}\sum_{i=1}^N x_i$`	$\overline{x}=\frac{1}{N}\sum_{i=1}^N x_i$
...	...

3. LaTeX for equations

3.3. Large equations

- Surrounding a LaTeX input with **one $** on each side is suitable for **inline equation**

- You can also surround a LaTeX input with **two $** on each side
  - It puts the equation at the **center of a new line**
  - And gives **more vertical space** to the equation

- Surrounding a LaTeX input with two $ is usually good for:
  - Large equations
  - Equations that should be emphasized

The mean formula with one `$` on each side

→ For inline equations

$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$

The mean formula with two `$` on each side

→ For large/emphasized equations

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

3. LaTeX for equations

3.3. Large equations

- Sometimes you do not want two **consecutive lines** of equations to be centered
  - You may want to **align** them based on a **common part** within the equations

- This should be done in an **`aligned` environment** (`$\begin{aligned}...\end{aligned}$`)
  - Place the **"&"** symbol where the equations should be aligned
  - And break a line using **"\\"**

```
$$
\begin{aligned}
x & = (a + b) \times c \\
  & = (a \times c) + (b \times c)
\end{aligned}
$$
```

$$
\begin{aligned}
x & = (a + b) \times c \\
  & = (a \times c) + (b \times c)
\end{aligned}
$$

3. LaTeX for equations

3.3. Large equations

- The same principle applies within **`cases` environment**

```
$$\text{Med}(x) = \begin{cases}
  x[\frac{N+1}{2}]                          & \text{if } N \text{ is odd}\\
  \frac{x[\frac{N}{2}]+x[\frac{N}{2}+1]}{2} & \text{if } N \text{ is even}
\end{cases}$$
```

$$\text{Med}(x) = \begin{cases} x[\frac{N+1}{2}] & \text{if } N \text{ is odd} \\ \frac{x[\frac{N}{2}]+x[\frac{N}{2}+1]}{2} & \text{if } N \text{ is even} \end{cases}$$

- Note that the **`text` function** allows to write text without it being interpreted as mathematical letters:

```
$$Mean(x)=\frac{1}{N}\sum_{i=1}^N x_i$$
```

$$Mean(x)=\frac{1}{N}\sum_{i=1}^N x_i$$

```
$$\text{Mean}(x)=\frac{1}{N}\sum_{i=1}^N x_i$$
```

$$\text{Mean}(x)=\frac{1}{N}\sum_{i=1}^N x_i$$

Practice

03:00

1) Inside your .qmd, reproduce the following mathematical expression

$$Y_i = \alpha + \beta X_i + \varepsilon_i$$

2) Then reproduce the following sentence

$\hat{Y}_i$ denote the fitted values of the model.

You've got 3 minutes!

Solution

1) Inside your .qmd, reproduce the following mathematical expression

$$Y_i = \alpha + \beta X_i + \varepsilon_i$$

```
$$Y_i = \alpha + \beta X_i + \varepsilon_i$$
```

2) Then reproduce the following sentence

$\hat{Y_i}$ denote the fitted values of the model.

```
$\hat{Y_i}$ denote the fitted values of the model.
```

Overview

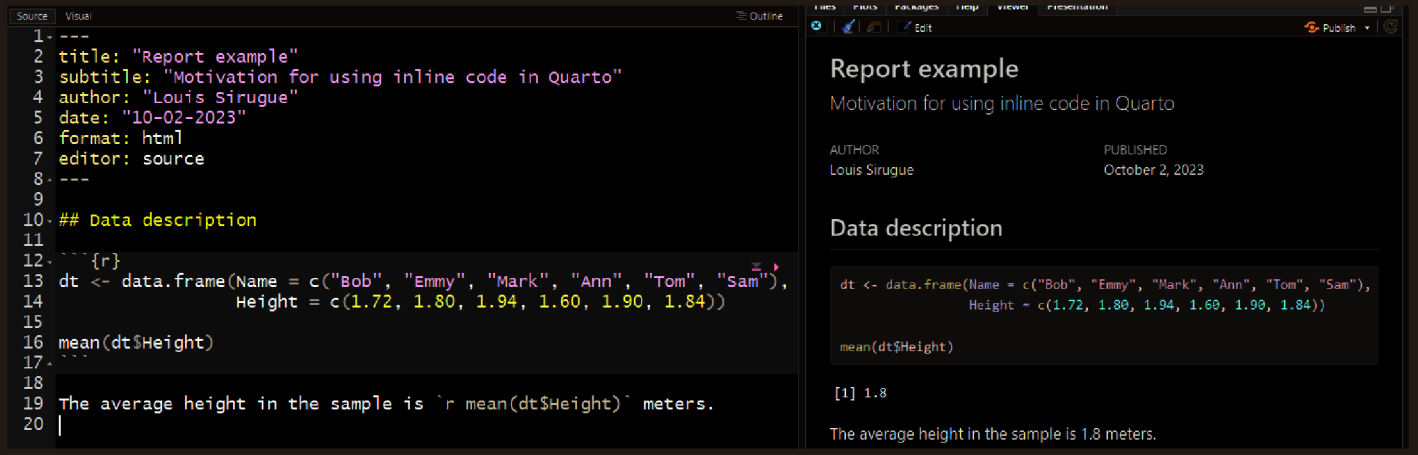
4. Wrap up!

1. Three types of contents

YAML header →

Code chunks →

Text →



4. Wrap up!

2. Useful features

→ **Inline code** allows to include the output of some **R code within text areas** of your report

Syntax	Output
`` `paste("a", "b", sep = "-")` ``	`paste("a", "b", sep = "-")`
`` `r paste("a", "b", sep = "-")` ``	a-b

→ **`kable()`** for clean **html tables** and **`datatable()`** to navigate in **large tables**

```
kable(results_table)
datatable(results_table)
```

4. Wrap up!

3. LaTeX for equations

- $\LaTeX$ is a convenient way to display **mathematical** symbols and to structure **equations**
  - The **syntax** is mainly based on **backslashes \ and braces {}**

→ What you **type** in the text area: `$x \neq \frac{\alpha \times \beta}{2}$`
→ What is **rendered** as an output: $x \neq \frac{\alpha \times \beta}{2}$

To **include** a **LaTeX equation** in Quarto, you simply have to surround it with the **$ sign**

The mean formula with one $ on each side

→ For inline equations

$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$

The mean formula with two $ on each side

→ For large/emphasized equations

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$