

# Applied Data Analysis for Public Policy Studies

## Introduction

Michele Fioretti  
Sciences Po Paris  
2020-09-15

# Welcome to Applied Data Analysis for Public Policy Studies



# Welcome to Applied Data Analysis for Public Policy Studies

- This course will teach you the core tools of *econometrics*. The goal of the course is to provide you with *essential data analysis tools*.
- We will meet on *Wednesdays at 8.00 AM (GMT+2)* on *Zoom* (Pass: AD2020).
- You will meet with the TA, Eléonore Richard, every couple of weeks to discuss the material and review problem sets. Her email address is *eleonore.richard@sciencespo.fr*



# Welcome to Applied Data Analysis for Public Policy Studies

- This course will teach you the core tools of *econometrics*. The goal of the course is to provide you with *essential data analysis tools*.
- We will meet on *Wednesdays at 8.00 AM (GMT+2)* on *Zoom* (Pass: AD2020).
- You will meet with the TA, Eléonore Richard, every couple of weeks to discuss the material and review problem sets. Her email address is *eleonore.richard@sciencespo.fr*

## Welcome to ScPoEconometrics!

- We will learn these tools through example based on *the programming language R*.
- This course builds on the (amazing) *openSource lectures* by the ScPo Econometrics team (Florian Oswald, Gustave Kenedi and Pierre Villedieu).
- A *book overview* of the material is also available.



# What is *econometrics*?

- A set of techniques and methods to answer questions with data.
- Econometrics shares many things with *applied statistics* and *machine learning*.
- Some examples!



# Answering Important Questions with Econometrics

Does immigration *cause* lower wages and higher unemployment for locals?



# Answering Important Questions with Econometrics

Does immigration *cause* lower wages and higher unemployment for locals?

Does increasing the minimum wage *cause* greater unemployment?



# Answering Important Questions with Econometrics

Does immigration *cause* lower wages and higher unemployment for locals?

Does increasing the minimum wage *cause* greater unemployment?

Does more education *cause* higher wages?





# Answering Important Questions with Econometrics

Does immigration *cause* lower wages and higher unemployment for locals?

Does increasing the minimum wage *cause* greater unemployment?

Does more education *cause* higher wages?

Does higher public debt levels *cause* lower economic growth?



# Answering Important Questions with Econometrics

Does immigration *cause* lower wages and higher unemployment for locals?

Does increasing the minimum wage *cause* greater unemployment?

Does more education *cause* higher wages?

Does higher public debt levels *cause* lower economic growth?

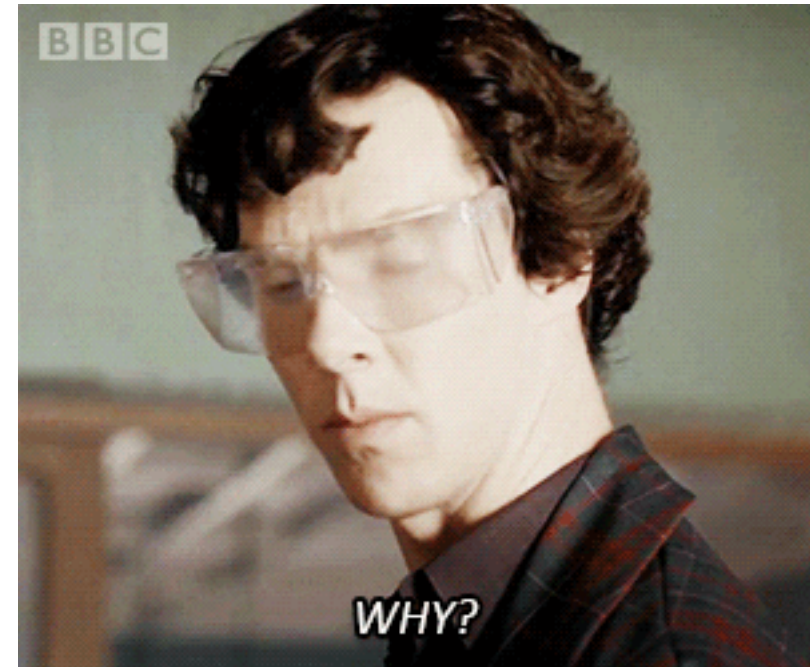
Does birth order *cause* differing education trajectories?



# Causality

- Notice the keyword **cause** in all of the above.
- Notice also that *many other factors could have caused* each of those outcomes.
- Econometrics is often about spelling out conditions under which we can claim to measure causal relationships.
- We will encounter the most basic of those conditions, and talk about some potential pitfalls.

As in the acclaimed **Book of Why** we often ask *why* did something happen?



# This Course

- Teach you the basics of *linear regression*, *statistical inference* and *impact evaluation*.
- Equip you with a framework to think more deeply about causality.
- Introduce you to the R software environment.
- ⚠ This is *not* a course about R.



# This Course

- Teach you the basics of *linear regression*, *statistical inference* and *impact evaluation*.
- Equip you with a framework to think more deeply about causality.
- Introduce you to the R software environment.
- ⚠ This is *not* a course about R.

## Grading

1. There will be *periodic quizzes* on Moodle roughly every two weeks => 20%



# This Course

- Teach you the basics of *linear regression*, *statistical inference* and *impact evaluation*.
- Equip you with a framework to think more deeply about causality.
- Introduce you to the R software environment.
- ⚠ This is *not* a course about R.

## Grading

1. There will be *periodic quizzes* on Moodle roughly every two weeks => 20%
2. There will be *a problem set / case study* => 30%



# This Course

- Teach you the basics of *linear regression*, *statistical inference* and *impact evaluation*.
- Equip you with a framework to think more deeply about causality.
- Introduce you to the R software environment.
- ⚠ This is *not* a course about R.

## Grading

1. There will be *periodic quizzes* on Moodle roughly every two weeks => 20%
2. There will be *a problem set / case study* => 30%
3. There will be *a take home exam* => 50%



# Course Materials

1. The **Book**
2. The **Slides**
3. The code repository for the **R package**
4. Quizzes on **Moodle**





# Syllabus 🙌

Lecture 1: **Introduction**

*Quiz 1 (after lecture 2)*

Lecture 2/3: **Summarising Data**

*Quiz 2*

Lecture 4: **Simple Linear Regression**

Lecture 5: **Introduction to Causality**

*Midterm Project*

Lecture 6: **Multiple Linear Regression**

Lecture 7: **Sampling**

*Quiz 3*

Lecture 8/9: **Statistical Inference**

*Quiz 4*

Lecture 10: **Differences-in-Differences**

Lecture 11: **Regression Discontinuity**

*Quiz 5*

Lecture 12: **Recap**

*Final Project*



# Useful Resources (Other Than our *Book*)

## Econometrics

- *Mastering Metrics* by Angrist and Pischke
- *Modern Introduction to Econometrics* by Wooldridge
- *Introduction to Econometrics* by Stock and Watson
- *Causal Inference: The Mixtape* by Cunningham
- Ben Lambert's youtube channel

## Metrics and R

- ModernDive
- Introduction to Econometrics with R
- R for Data Science



R

# What is R?

R is a **programming language** with powerful statistical and graphic capabilities.

## Why are we using R?<sup>1</sup>

1. R is **free** and **open source**—saving both you and the university 💰💰💰.
1. R is very **flexible and powerful**—adaptable to nearly any task, *e.g.*, data cleaning, data visualization, econometrics, spatial data analysis, machine learning, web scraping, ...
1. R has a vibrant, thriving online community that will (almost) always have a solution to your problem. (**stack overflow**)
1. If you put in the work<sup>2</sup>, you will come away with a **very valuable and marketable** tool.

[1]: This list has been inspired by **Ed Rubin's**.

[2]: Learning R definitely requires time and effort but it's worth it, trust me! 💪.



# Why can't we just use Excel?

Many reasons but here are just a few:



# Why can't we just use Excel?

Many reasons but here are just a few:

- Not reproducible.



# Why can't we just use Excel?

Many reasons but here are just a few:

- Not reproducible.
- Not straightforward to merge datasets together.



# Why can't we just use Excel?

Many reasons but here are just a few:

- Not reproducible.
- Not straightforward to merge datasets together.
- Very fastidious to clean data.





# Why can't we just use Excel?

Many reasons but here are just a few:

- Not reproducible.
- Not straightforward to merge datasets together.
- Very fastidious to clean data.
- Limited to small datasets



# Why can't we just use Excel?

Many reasons but here are just a few:

- Not reproducible.
- Not straightforward to merge datasets together.
- Very fastidious to clean data.
- Limited to small datasets
- Not designed for proper econometric analyses, maps, complex visualisations, etc.



# R SHOWCASE

# Showcase #1: Spatial Data

- **R** is very strong with spatial data. In particular via the **sf** package.
- We can represent *any* shape or geometry.
- Maps are the most obvious example:

```
library(sf)
library(tmap)
iris_shfl <- read_sf("chapter1_files/figure-html/cont")
  mutate(dep = substr(INSEE_COM,1,2)) %>%
  select(CODE_IRIS, dep, geometry) %>%
  filter(dep == "75")
iris_income <- readRDS("../rds/iris_inc.rds") %>%
  mutate(CODE_IRIS = IRIS) %>%
  select(CODE_IRIS, DISP_MED15)
iris_map <- left_join(iris_shfl, iris_income, by = "CODE_IRIS")
tmap_mode("plot")
tm_shape(iris_map) +
  tm_borders() +
  tm_fill(col = "DISP_MED15", title = "Median househo")
```

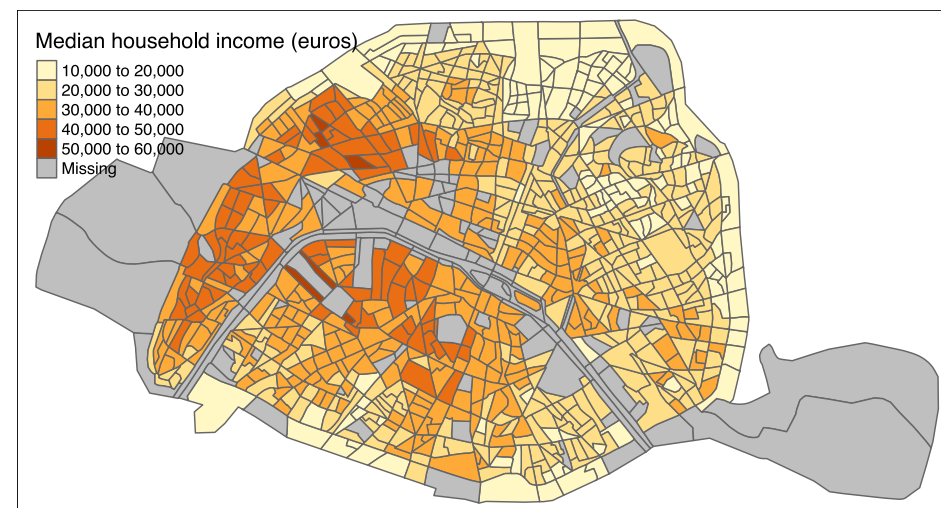


# Showcase #1: Spatial Data

- **R** is very strong with spatial data. In particular via the **sf** package.
- We can represent *any* shape or geometry.
- Maps are the most obvious example:

```
library(sf)
library(tmap)
iris_shfl <- read_sf("chapter1_files/figure-html/cont")
  mutate(dep = substr(INSEE_COM,1,2)) %>%
  select(CODE_IRIS, dep, geometry) %>%
  filter(dep == "75")
iris_income <- readRDS("../rds/iris_inc.rds") %>%
  mutate(CODE_IRIS = IRIS) %>%
  select(CODE_IRIS, DISP_MED15)
iris_map <- left_join(iris_shfl, iris_income, by = "C")
tmap_mode("plot")
tm_shape(iris_map) +
  tm_borders() +
  tm_fill(col = "DISP_MED15", title = "Median househo
```

- Can be improved but you get this with only **14 lines of code!**

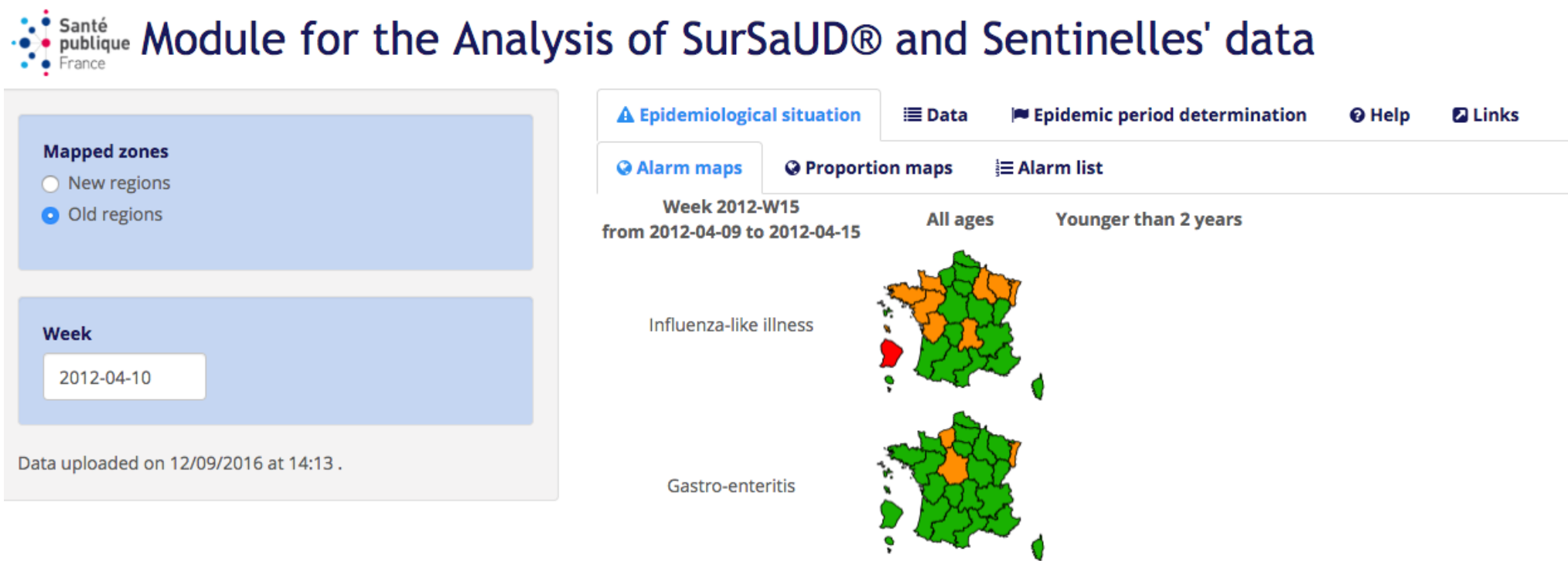


# Showcase #2: Interactive web applications



# Showcase #2: Interactive web applications

- Santé publique France has created a **simple web application** to track the epidemiological situation in French regions per week



# In Practice: Data Wrangling

- You will spend a lot of time preparing data for further analysis.





# In Practice: Data Wrangling

- You will spend a lot of time preparing data for further analysis.
- The `gapminder` dataset contains data on life expectancy, GDP per capita and population by country between 1952 and 2007.
- Suppose we want to know the average life expectancy and average GDP per capita for each continent in each year.
- We need to group the data by continent *and* year, then compute the average life expectancy and average GDP per capita

```
# load gapminder package
library(gapminder)
# load the dataset in object
gapminder = gapminder::gapminder
# display variables in the dataset
names(gapminder)
# show first 4 lines of the dataset
head(gapminder, n = 4)
```

```
## [1] "country" "continent" "year" "lifeExp" "pop" "gdpPercap"
## # A tibble: 4 x 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>    <int> <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952  28.8  8425333  779.
## 2 Afghanistan Asia      1957  30.3  9240934  821.
## 3 Afghanistan Asia      1962  32.0 10267083  853.
## 4 Afghanistan Asia      1967  34.0 11537966  836.
```



# In Practice: Data Wrangling

- There are always several ways to achieve a goal. (As in life 😊)
- Here we will only focus on the `dplyr` way:

```
# You may want to install the dplyr package first: install.packages("dplyr")
library(dplyr)
# compute the required statistics
gapminder_dplyr <- gapminder %>%
  group_by(continent, year) %>%
  summarise(
    count = n(),
    mean_lifeexp = mean(lifeExp, na.rm = TRUE),
    mean_gdppercap = mean(gdpPercap, na.rm = TRUE)
  )
```

```
# show first 4 lines of this dataframe
head(gapminder_dplyr, n = 4)
```

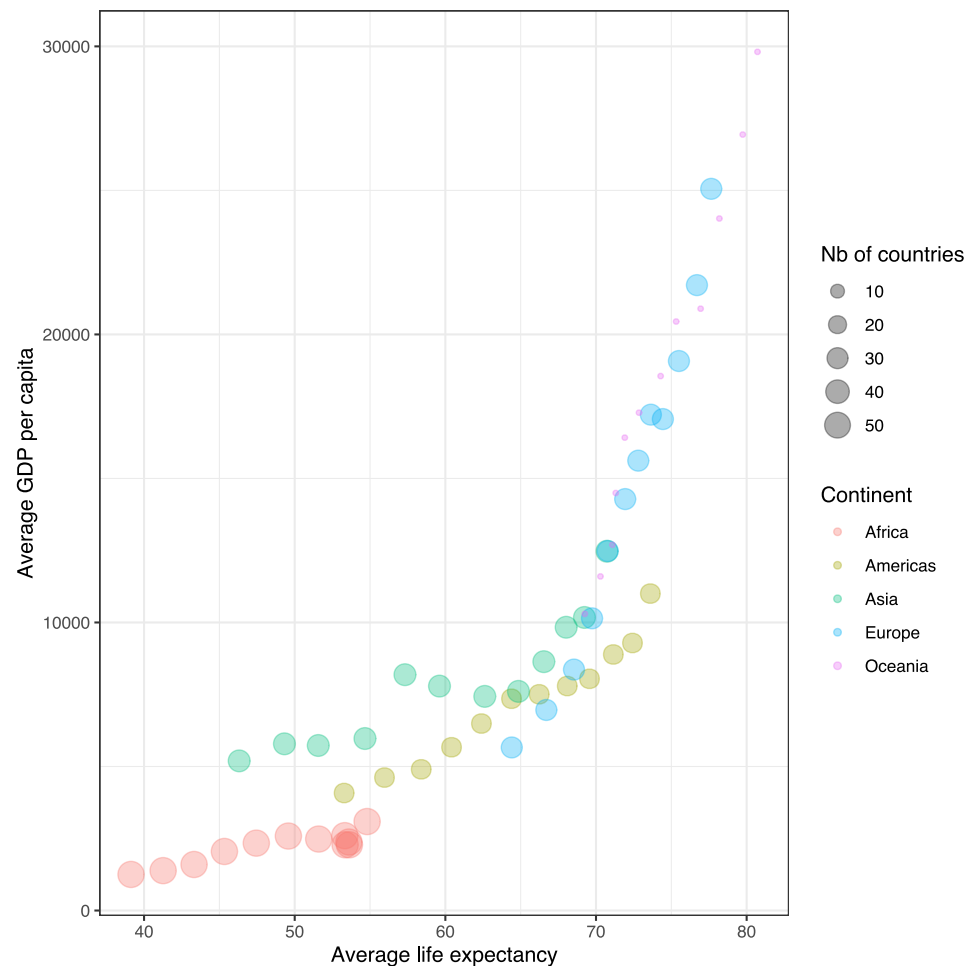
```
## # A tibble: 4 x 5
## # Groups:   continent [1]
##   continent  year count mean_lifeexp mean_gdppercap
##   <fct>      <int> <int>      <dbl>      <dbl>
## 1 Africa    1952     52        39.1       1253.
## 2 Africa    1957     52        41.3       1385.
## 3 Africa    1962     52        43.3       1598.
## 4 Africa    1967     52        45.3       2050.
```



# Visualisation

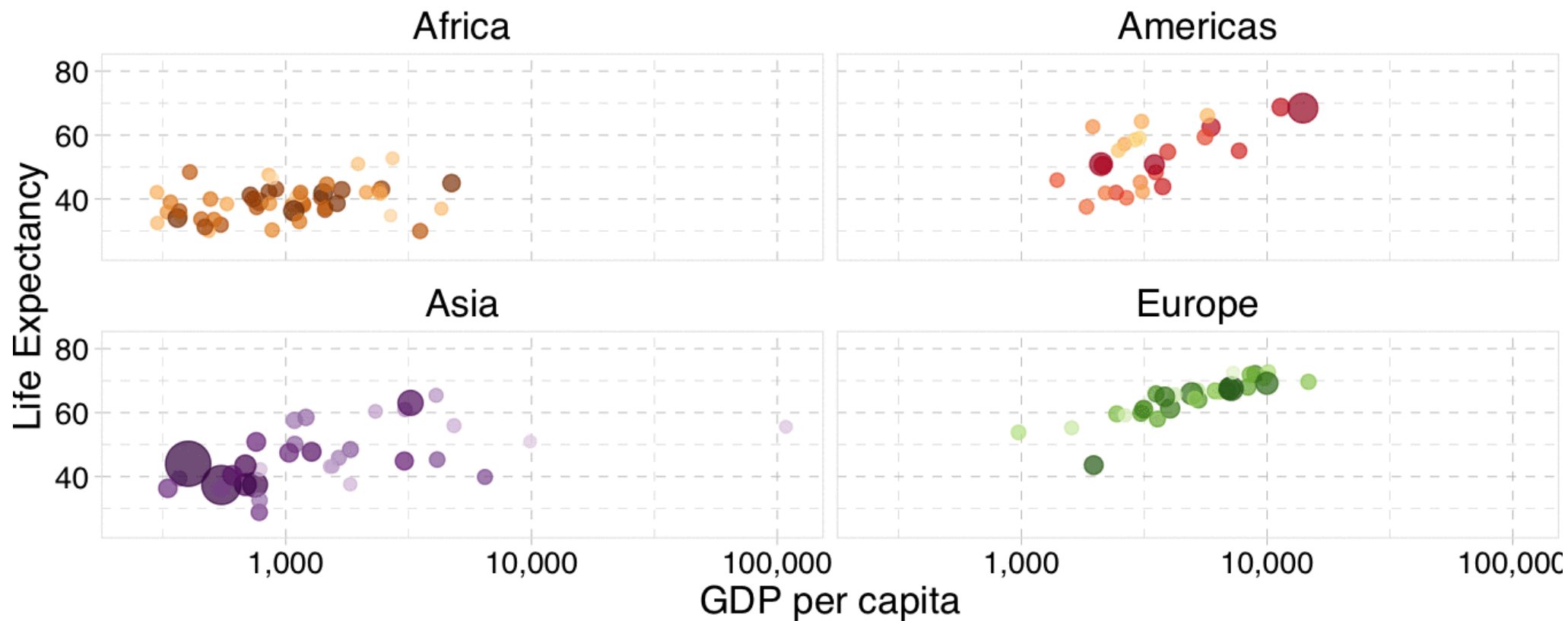
- Now we could *look* at the result in `gapminder_dplyr`, or compute some statistics from it.
- Nothing beats a picture, though:

```
# to install the dplyr package: install.packages("dplyr")
library(ggplot2)
# create a plot
ggplot(data = gapminder_dplyr,
       mapping = aes(x = mean_lifeexp,
                     y = mean_gdppercap,
                     color = continent,
                     size = count)) +
  geom_point(alpha = 1/3) +
  labs(x = "Average life expectancy",
       y = "Average GDP per capita",
       color = "Continent",
       size = "Nb of countries") +
  theme_bw() # white background
```



# Animated Plotting 🙌 1

Year: 1952



[1]: This animation is taken from [Ed Rubin](#).

# R 101: Here Is Where You Start

# Tool Time!

## Getting R and Rstudio

- Download **R** from **CRAN** for your OS.
- Download **RStudio** from **here** for your OS.



# Start your RStudio!

## First Glossary of Terms

- R: a programming language.
- RStudio: an integrated development environment (IDE) to work with R.



# Start your RStudio!

## First Glossary of Terms

- **R**: a programming language.
- **RStudio**: an integrated development environment (IDE) to work with **R**.
- *command*: user input (text or numbers) that **R** *understands*.
- *script*: a list of commands collected in a text file, each separated by a new line, to be run one after the other.





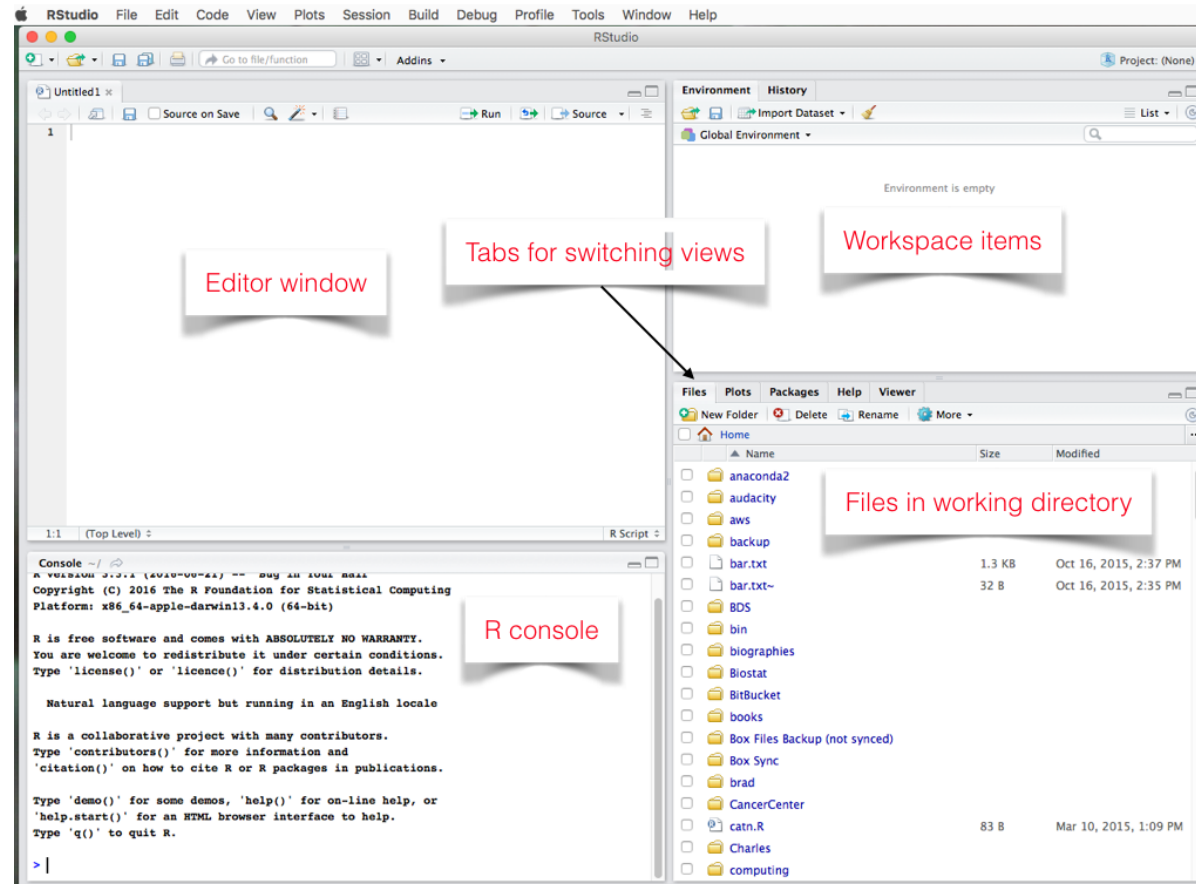
# Start your RStudio!

## First Glossary of Terms

- **R**: a programming language.
- **RStudio**: an integrated development environment (IDE) to work with **R**.
- *command*: user input (text or numbers) that **R** *understands*.
- *script*: a list of commands collected in a text file, each separated by a new line, to be run one after the other.
- To run a script, you need to highlight the relevant code lines and hit **Ctrl+Enter** (Windows) or **Cmd+Enter** (Mac).



# RStudio Layout



# R as a Calculator

- You can use the `R` console like a calculator
- Just type an arithmetic operation after `>` and hit `Enter`!



# R as a Calculator

- You can use the **R** console like a calculator
- Just type an arithmetic operation after **>** and hit **Enter**!

- Some basic arithmetic first:

```
4 + 1
```

```
## [1] 5
```

```
8 / 2
```

```
## [1] 4
```

- Great! What about this?

```
log(exp(1))
```

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

```
# by the way: this is a comment! (R disregards it)
```



# Task 1 (5 minutes)

- Create a new R script (File → New File → R Script). Save it somewhere as `lecture_intro.R`.
- Write in your script and run the following code: (Ctrl or Cmd + Enter)

```
4 * 8
```

- Write in your script and run the following code. What happens if you only run the first line of the code?

```
x = 5 # equivalently x <- 5  
x
```

- Congratulations, you have created your first R "object"! Everything is an object in R! Objects are assigned using `=` or `<-`.
- Lastly, find the cube of `x` and assign that value to `x_3`.



# Where to get Help?

- R built-in help:

```
?log
?sin
?paste
?lm
help(lm)    # help() is equivalent
??plot    # get all help on keyword "plot"
help(ggplot,package="ggplot2") # show help from a certain package
```

- Help from Humans!
  - Google is your best friend!
  - [stackoverflow.com](https://stackoverflow.com)
  - Your classroom channel on Slack. If you haven't joined click [here](#) to join
  - [rstudio forum](#)



# HOW to get Help? Follow this for Slack questions!

1. Describe what you want to do.
2. Describe what you *expect* your code to do.
3. Describe what your code *does instead*.
  - Provide the entire error message.
4. Provide enough code to *reproduce* your error.
  - You can post code snippets on Slack and Stack Overflow



# R Packages

- R users contribute add-on data and functions as *packages*
- Installing packages is easy!

```
install.packages("ggplot2")
```

- To *use* the contents of a package, we must load it from our library:

```
library(ggplot2)
```





# ScPoApps package

- The ScPoEconometrics team wrote an **R** package for you, which is hosted on **GitHub**
- You can install (and frequently update!) from here:

```
if (!require(devtools)) {install.packages("devtools"); library(devtools)}  
install_github(repo = "ScPoEcon/ScPoApps") # Let me know if you encounter an error
```



# ScPoApps package

- The ScPoEconometrics team wrote an **R** package for you, which is hosted on **GitHub**
- You can install (and frequently update!) from here:

```
if (!require(devtools)) {install.packages("devtools"); library(devtools)}  
install_github(repo = "ScPoEcon/ScPoApps") # Let me know if you encounter an error
```

- Did it work?

```
library(ScPoApps)  
packageVersion("ScPoApps") # latest release is 0.1  
  
## [1] '0.1'
```

- Check also if the package finds the tutorials

```
ScPoApps::runTutorial()
```

```
## Error: Please run `runTutorial()` with a valid tutorial as an argument.  
## Valid tutorial are: 'airline-safety', 'chapter2', 'chapter2-script', 'correlation', 'lm-example', 'non-normal', 'res
```



# Vectors

- What is a **vector**?
- The **c** function creates vectors.

```
c(1, 3, 5, 7, 8, 9)
```

```
## [1] 1 3 5 7 8 9
```

- Coercion to unique types:

```
c(42, "Statistics", TRUE)
```

```
## [1] "42" "Statistics" "TRUE"
```

- Creating a *range*

```
c(y = 1:6)
```

```
## y1 y2 y3 y4 y5 y6
```

```
## 1 2 3 4 5 6
```



# data.frame's

data.frames are like spreadsheets.

```
example_data = data.frame(x = c(1, 3, 5, 7),  
                           y = c(rep("Hello", 3), "Goodbye"),  
                           z = sample(c(TRUE, FALSE), size=4, replace=TRUE))
```

example\_data

```
##   x     y     z  
## 1 1  Hello FALSE  
## 2 3  Hello  TRUE  
## 3 5  Hello FALSE  
## 4 7 Goodbye TRUE
```

In practice, you will be importing files that contain the data into R rather than creating data.frames by hand.



## Task 2 (10 minutes)

- Find out (using google) how to import a .csv file.
- Import `gun_murders.csv`<sup>1</sup> in a new object `murders`. This file contains data on gun murders by US state in 2010. (Hint: objects are created using `=`)
- Ensure that `murders` is a data.frame by running:

```
# Check class  
class(murders)
```

- Find out what variables are contained in `murders` by running:

```
# Obtain variable names  
names(murders)
```

- View the contents of `murders` by clicking on `murders` in your workspace
  - What does the `total` variable correspond to?

[1]: This dataset is taken from the `dslabs` package.



# data.frames

- Useful methods for a dataframe:

```
str(murders) # describes the data.frame
```

```
## 'data.frame':    51 obs. of  5 variables:
## $ state      : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
## $ abb       : chr  "AL" "AK" "AZ" "AR" ...
## $ region     : Factor w/ 4 levels "Northeast","South",...: 2 4 4 2 4 4 1 2 2 2 ...
## $ population: num  4779736 710231 6392017 2915918 37253956 ...
## $ total      : num  135 19 232 93 1257 ...
```

```
names(murders) # column names
```

```
## [1] "state"      "abb"        "region"     "population" "total"
```

```
nrow(murders) # number of rows
```

```
## [1] 51
```

```
ncol(murders) # number of columns
```

```
## [1] 5
```



# Data on Gun Murders in the US

- Let's **View** the data in a table format

```
View(murders) # Open a data.frame/vector in an Excel style table (up to 50 columns)
```

- Let's look at the first rows of `murders`.

```
head(murders, n = 3) # show first 3 rows
```

```
##      state abb region population total
## 1 Alabama  AL  South    4779736    135
## 2  Alaska   AK   West     710231     19
## 3 Arizona  AZ   West    6392017    232
```

- To access one of the variables **as a vector**, we use the `$` operator as in `murders$state`. We can check the type of `murders$state` with

```
class(murders$state) # type of the state variable in the murders data.frame
## [1] "character"
```

- Or we use the column name or index: `murders[, "state"]` or `murders[, 1]`



# Subsetting data.frames

- Subsetting a data.frame: `murders[row condition, column number]` or `murders[row condition, "column name"]`

```
# Only keep states with over 500 gun murders and keep only the "state" and "total" variables  
murders[murders$total > 500, c("state", "total")]
```

```
##           state total  
## 5  California 1257  
## 10   Florida  669  
## 33   New York  517  
## 44    Texas   805
```

- There is also a special function for subsetting data:

```
subset(murders, subset = total > 500, select = c("state", "total"))
```

```
##           state total  
## 5  California 1257  
## 10   Florida  669  
## 33   New York  517  
## 44    Texas   805
```





## Task 3 (10 minutes)

1. How many observations are there in `murders`?
2. How many variables? What are the data types of each variable?
3. Notice that the colon operator `a:b` is just short for *construct a sequence from `a` to `b`*. Create a new object `murders_2` containing the rows 10 to 25 of `murders`.
4. Create a new object `murders_3` which only contains the columns `state` and `total`. (Recall that `c` creates vectors.)
5. What is the average value of `total`?
6. What is the average value of `total` for state's in the "South", i.e. with `region == "South"`?
7. Create a `total_percap` variable:

```
murders$total_percap = (murders$total / murders$population) * 10000
```



Congratulations, you've created your first variable!

**SEE YOU NEXT WEEK!**

---

 [michele.fioretti@sciencespo.fr](mailto:michele.fioretti@sciencespo.fr)

 Slides

 Book

 @ScPoEcon

 @ScPoEcon

---

