SciencesPo
DEPARTMENT OF ECONOMICS

ScPoEconometrics

Introduction

Constance Frohly
Sciences Po Paris
2025-01-31

Welcome to ScPoEconometrics!

Welcome to ScPoEconometrics!

- In this course you will learn the core tools of ***econometrics***.

Welcome to ScPoEconometrics!

- In this course you will learn the core tools of ***econometrics***.
- You will also learn to use the `R` programming language!

Welcome to ScPoEconometrics!

- In this course you will learn the core tools of ***econometrics***.
- You will also learn to use the `R` programming language!

What is *econometrics*?

- A set of ***techniques and methods*** to answer (economic) questions with ***data***.
- Some examples!

Answering Important Questions with Econometrics

> Does immigration *lead to* lower wages and/or higher unemployment for locals?

Answering Important Questions with Econometrics

Does immigration *lead to* lower wages and/or higher unemployment for locals?

Does raising the minimum wage *reduce* employment for the low-skilled?

Answering Important Questions with Econometrics

Does immigration *lead to* lower wages and/or higher unemployment for locals?

Does raising the minimum wage *reduce* employment for the low-skilled?

Does getting a college degree *afford* higher wages?

Answering Important Questions with Econometrics

- Does immigration *lead to* lower wages and/or higher unemployment for locals?
- Does raising the minimum wage *reduce* employment for the low-skilled?
- Does getting a college degree *afford* higher wages?
- Do higher public debt levels *lead to* lower economic growth?

Answering Important Questions with Econometrics

- Does immigration *lead to* lower wages and/or higher unemployment for locals?
- Does raising the minimum wage *reduce* employment for the low-skilled?
- Does getting a college degree *afford* higher wages?
- Do higher public debt levels *lead to* lower economic growth?
- Does the neighborhood you grew up in have an *impact* on your life outcomes?

Causality

- Notice that ***many other factors could have caused*** each of the outcomes mentioned.

Causality

- Notice that ***many other factors could have caused*** each of the outcomes mentioned.
- Often, we'll want to focus on the ***causal impact*** of just one of these factors (immigration, minimum wage, education ,etc.)

Causality

- Notice that ***many other factors could have caused*** each of the outcomes mentioned.
- Often, we'll want to focus on the ***causal impact*** of just one of these factors (immigration, minimum wage, education ,etc.)
- Econometrics is 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.
- "Credibility Revolution" in econometrics over the past 30 years (2022 Economics Nobel awarded to some of the main protagonists of this "revolution")

R

What is R?

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

What is R?

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

Why are we using R?[1]

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

What is R?

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

Why are we using R?[1]

1. R is **free** and **open source**—saving both you and the university 💰💵💰.

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

What is R?

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

Why are we using R?[1]

1. R is **free** and **open source**—saving both you and the university 💰💵💰.
2. R is very **flexible and powerful**—adaptable to nearly any task, (data cleaning, data visualization, econometrics, spatial data analysis, machine learning, web scraping, etc.)

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

What is R?

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

Why are we using R?[1]

1. R is **free** and **open source**—saving both you and the university 💰💵💰.
2. R is very **flexible and powerful**—adaptable to nearly any task, (data cleaning, data visualization, econometrics, spatial data analysis, machine learning, web scraping, etc.)
3. R has a vibrant, thriving online community that will (almost) always have a solution to your problem.

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

What is R?

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

Why are we using R?[1]

1. R is **free** and **open source**—saving both you and the university 💰💵💰.
2. R is very **flexible and powerful**—adaptable to nearly any task, (data cleaning, data visualization, econometrics, spatial data analysis, machine learning, web scraping, etc.)
3. R has a vibrant, thriving online community that will (almost) always have a solution to your problem.
4. If you put in the work[2], you will come away with a **very valuable and useful** 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!

First Taste of R

In Practice: Data Wrangling

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.

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

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 from the gapminder package
data(gapminder, package = "gapminder")
# show first 4 lines of this dataframe
head(gapminder,n = 4)
```

```
## # A tibble: 4 × 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:

```
# compute the required statistics
gapminder_dplyr <- gapminder %>%
  group_by(continent, year) %>%
  summarise(count = n(),
            mean_lifeexp = mean(lifeExp),
            mean_gdppercap = mean(gdpPercap))
```

```
# show first 5 lines of the new data
head(gapminder_dplyr, n = 5)
```

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:

```
# compute the required statistics
gapminder_dplyr <- gapminder %>%
  group_by(continent, year) %>%
  summarise(count = n(),
            mean_lifeexp = mean(lifeExp),
            mean_gdppercap = mean(gdpPercap))
```

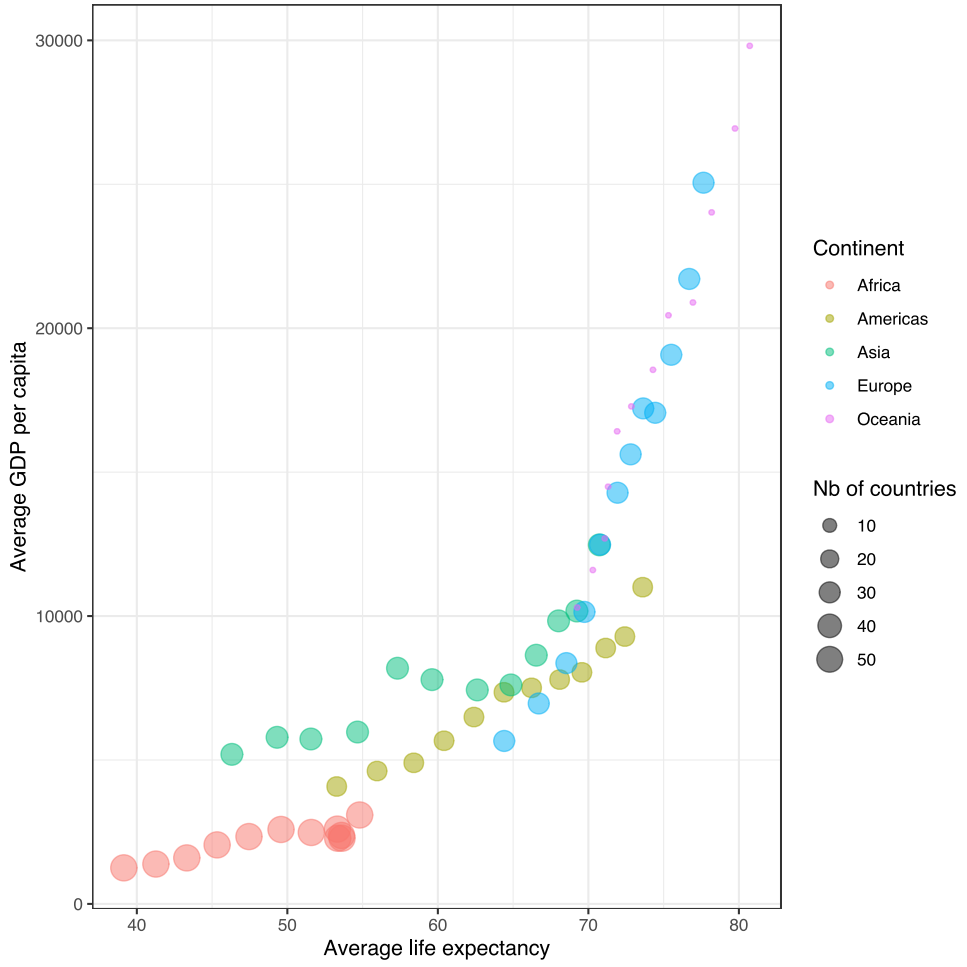
```
# show first 5 lines of the new data
head(gapminder_dplyr, n = 5)
```

```
## # A tibble: 5 × 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.
## 5 Africa     1972    52         47.5          2340.
```

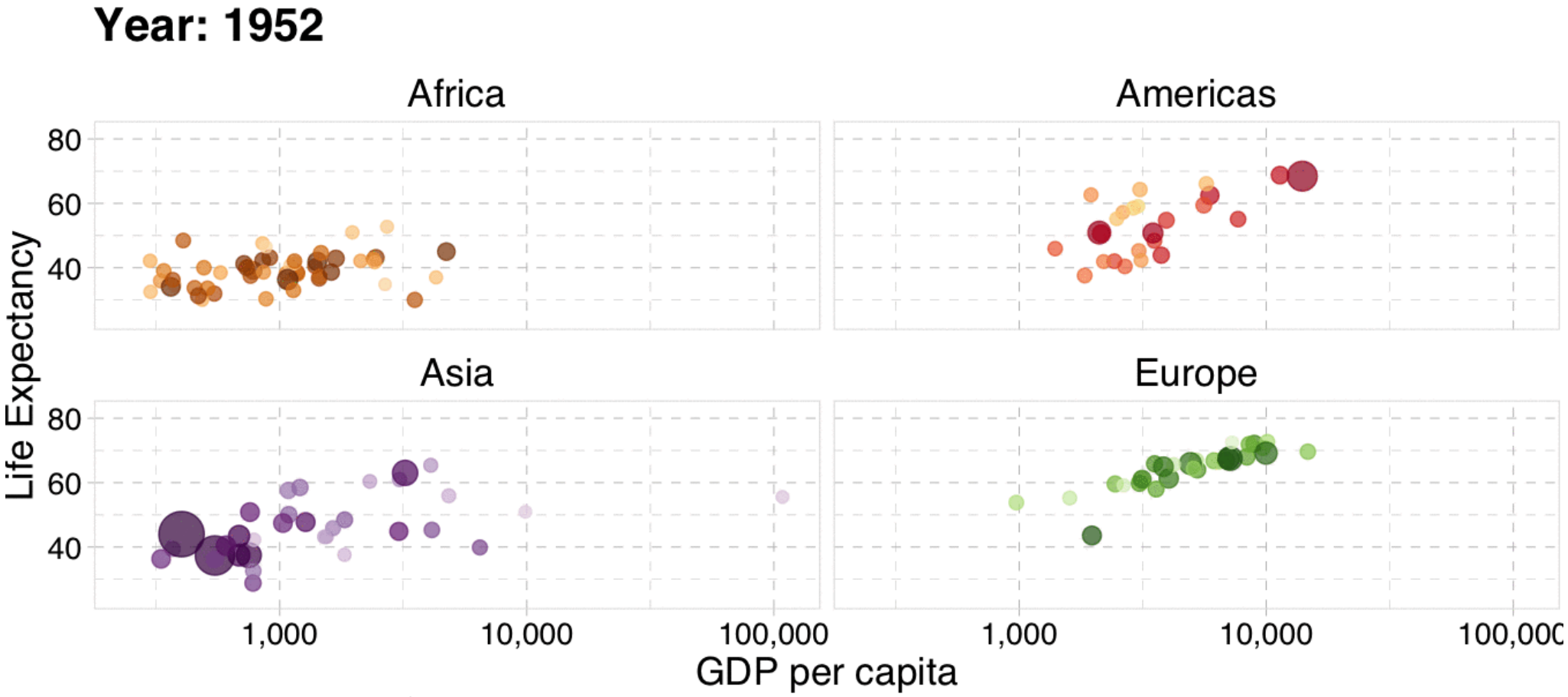
Visualisation

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

```
ggplot(data = gapminder_dplyr,
       mapping = aes(x = mean_lifeexp,
                     y = mean_gdppercap,
                     color = continent,
                     size = count)) +
  geom_point(alpha = 1/2) +
  labs(x = "Average life expectancy",
       y = "Average GDP per capita",
       color = "Continent",
       size = "Nb of countries") +
  theme_bw()
```



Animated Plotting 👌 [1]



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

R 101: Here Is Where You Start

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

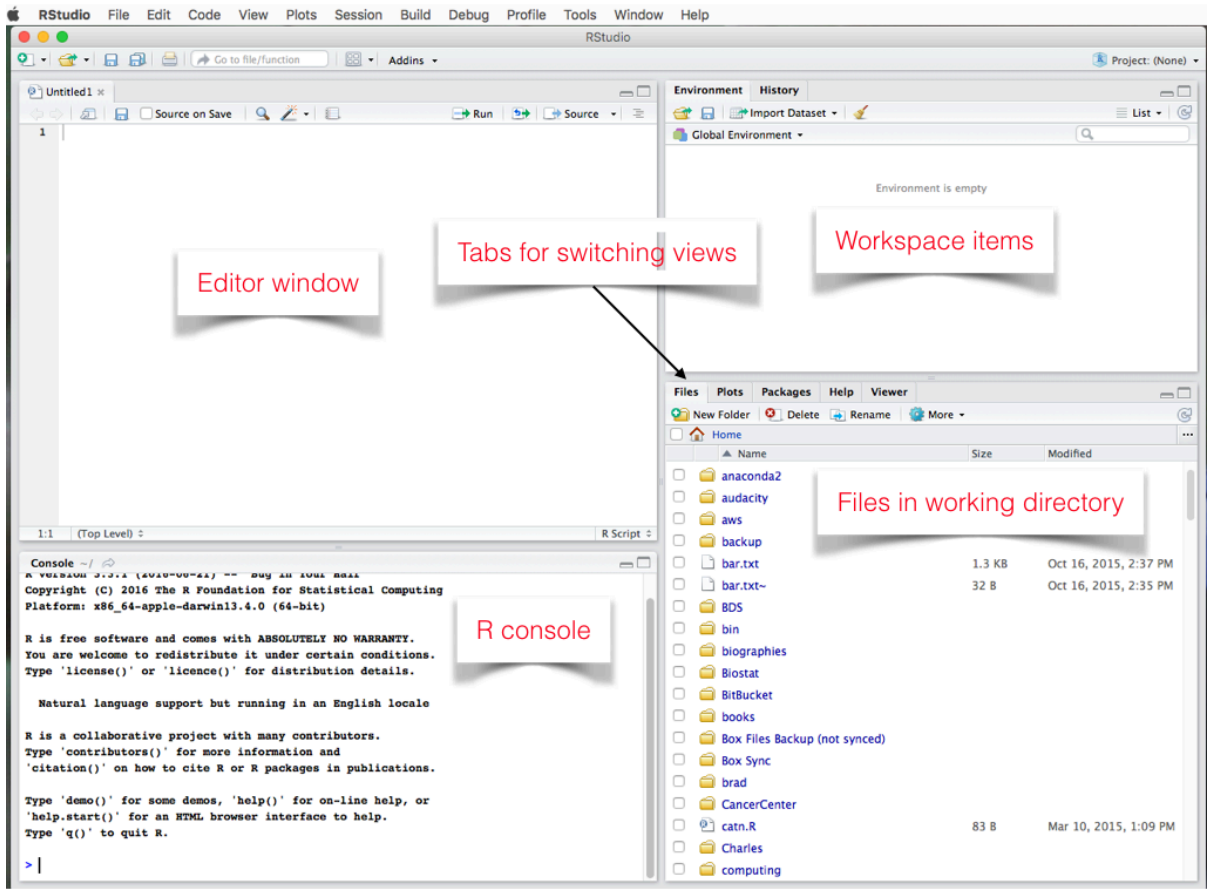
- `R`: a programming language.
- `RStudio`: an integrated development environment (IDE) to work with `R`.
- *command*: user input (text or numbers) that `R` *understands*.
- *script*: a list of commands collected in a text file, each separated by a new line, to be run one after the other.

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?

```
2^3
```

```
## [1] 8
```

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

Task 1

05:00

1. Create a new R script (File → New File → R Script). Save it somewhere as `lecture_intro.R`.

2. Type the following code in your script and run it. To run the code press `Ctrl` or `Cmd` + `Enter` (you can either highlight the code or just put your cursor at the end of the line)

```
4 * 8
```

3. Type the following code in your script and run it. 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 `<-`.

4. Create a new object named `x_3` to which you assign the cube of `x`. Note that to assign you need to use `=` or `<-`. Use code to compute the cube, not a calculator.

Where to get Help?

`R` built-in `help`:

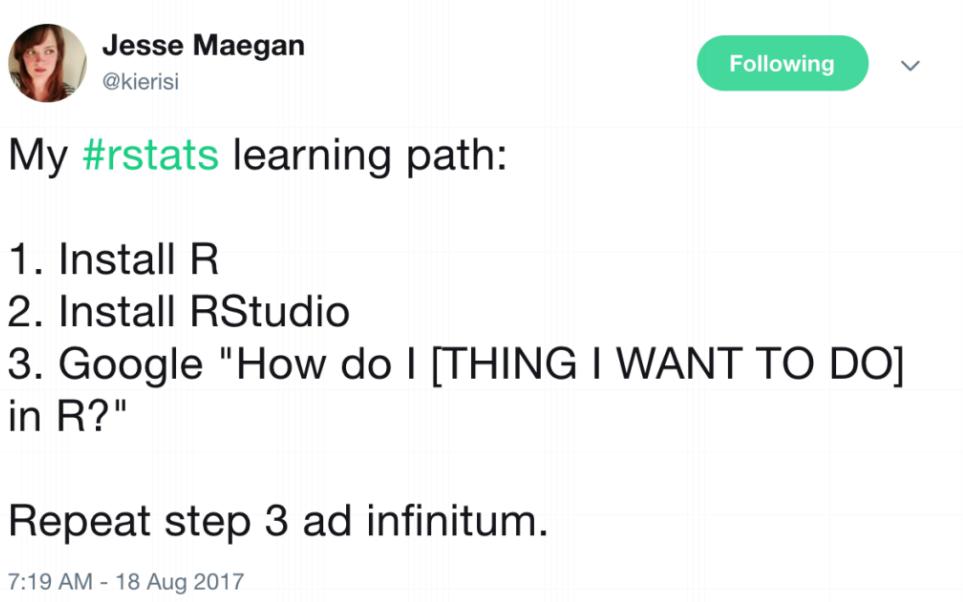
```
?log #? in front of function
help(lm)   # help() is equivalent
??plot  # get all help on keyword "plot"
```

Where to get Help?

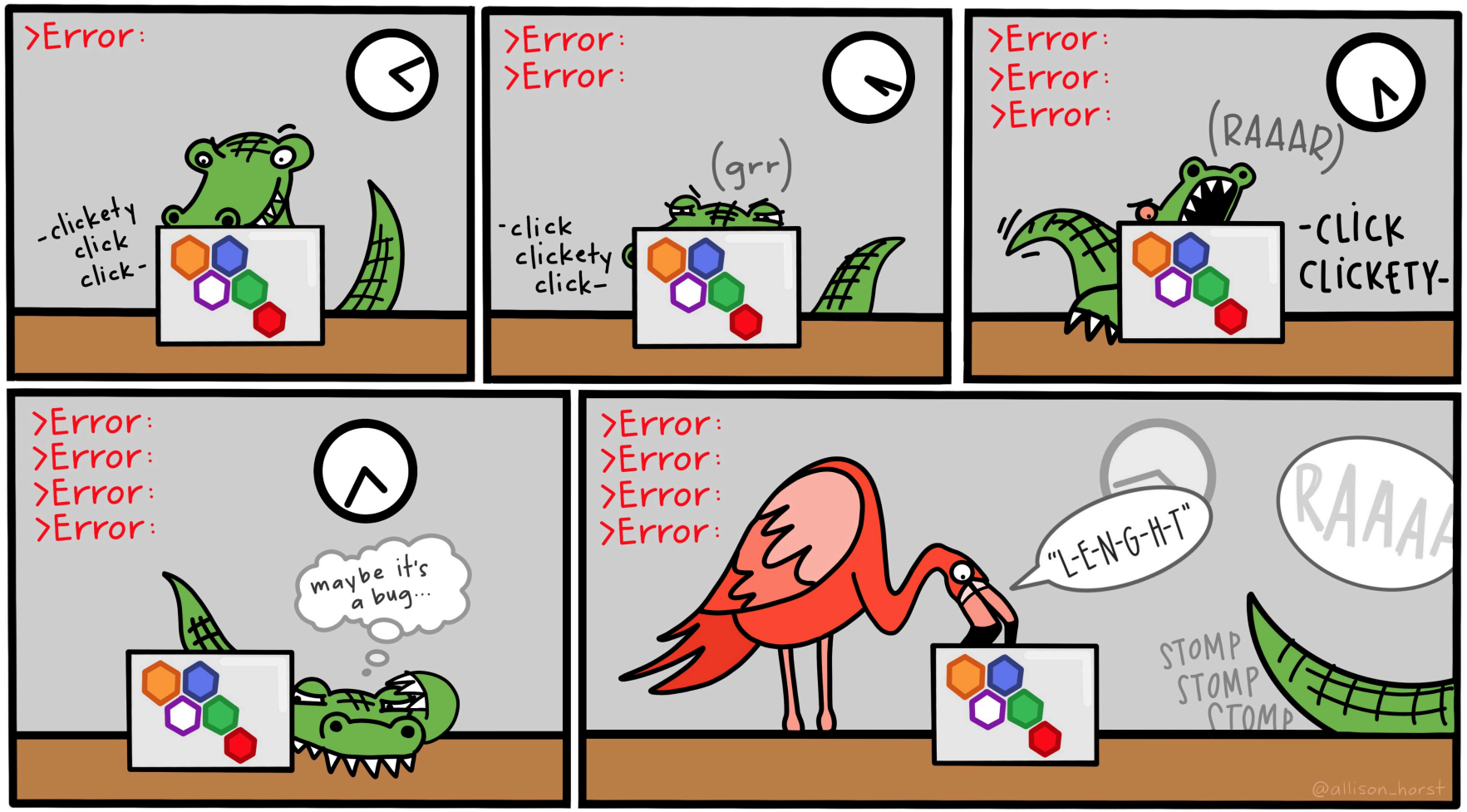
`R` built-in `help`:

```
?log #? in front of function
help(lm)   # help() is equivalent
??plot  # get all help on keyword "plot"
```

In practice:



Collaborate!



R Packages

- `R` users contribute add-on data and functions as *packages*
- Installing packages is easy! Just use the `install.packages` function:

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

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

```
library(ggplot2)
```

Vectors

- The `c` function creates vectors, i.e. *one-dimensional arrays*.

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

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

- Coercion to unique types:

```
(v <- c(42, "Statistics", TRUE))
```

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

Vectors

- The `c` function creates vectors, i.e. *one-dimensional arrays*.

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

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

- Coercion to unique types:

```
(v <- c(42, "Statistics", TRUE))
```

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

- Creating a *range*

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

- get vector elements with square bracket operator `[index]`:

```
v[c(1,3)]
```

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

`data.frame`'s

`data.frame`s represent **tabular data**. Like spreadsheets.

```
example_data = data.frame(x = c(1, 3, 5, 7),
                          y = c(rep("Hello", 3), "Goodbye"),
                          z = c("one", 2, "three", 4))
example_data
```

```
##   x       y     z
## 1 1   Hello   one
## 2 3   Hello     2
## 3 5   Hello three
## 4 7 Goodbye     4
```

- A `data.frame` has 2 dimensions: *rows* and *columns*. Like a *matrix*. Can get elements with `[row_index,col_index]`.
- In practice, you will be importing files that contain the data into `R` rather than creating `data.frame`s by hand.

Task 2

07:00

1. Find out (using `help()` or google first, then ChatGPT) how to import a .csv file. Do NOT use the "Import Dataset" button, nor install a package.

2. Import gun_murders.csv[1] in a new object `murders`. This file contains data on gun murders by US state in 2010. (Hint: objects are created using `=` or `<-`).

3. Ensure that `murders` is a data.frame by running:

```
class(murders) # check class
```

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

```
names(murders) # obtain variable names
```

5. 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 functions to describe a dataframe:

```
str(murders) # `str` describes structure of any R object
```

```
## 'data.frame':    51 obs. of  5 variables:
##  $ state     : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ abb       : chr  "AL" "AK" "AZ" "AR" ...
##  $ region    : chr  "South" "West" "West" "South" ...
##  $ population: int  4779736 710231 6392017 2915918 37253956 5029196 3574097 897934 601723 19687653 ...
##  $ total     : int  135 19 232 93 1257 65 97 38 99 669 ...
```

data.frames

Useful functions to describe a dataframe:

```
str(murders) # `str` describes structure of any R object
```

```
## 'data.frame':    51 obs. of  5 variables:
##  $ state     : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ abb       : chr  "AL" "AK" "AZ" "AR" ...
##  $ region    : chr  "South" "West" "West" "South" ...
##  $ population: int  4779736 710231 6392017 2915918 37253956 5029196 3574097 897934 601723 19687653 ...
##  $ total     : int  135 19 232 93 1257 65 97 38 99 669 ...
```

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

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

data.frames

Useful functions to describe a dataframe:

```
str(murders) # `str` describes structure of any R object
```

```
## 'data.frame':    51 obs. of  5 variables:
##  $ state     : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ abb       : chr  "AL" "AK" "AZ" "AR" ...
##  $ region    : chr  "South" "West" "West" "South" ...
##  $ population: int  4779736 710231 6392017 2915918 37253956 5029196 3574097 897934 601723 19687653 ...
##  $ total     : int  135 19 232 93 1257 65 97 38 99 669 ...
```

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

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

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

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

data.frames

Useful functions to describe a dataframe:

```
str(murders) # `str` describes structure of any R object
```

```
## 'data.frame':    51 obs. of  5 variables:
##  $ state     : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ abb       : chr  "AL" "AK" "AZ" "AR" ...
##  $ region    : chr  "South" "West" "West" "South" ...
##  $ population: int  4779736 710231 6392017 2915918 37253956 5029196 3574097 897934 601723 19687653 ...
##  $ total     : int  135 19 232 93 1257 65 97 38 99 669 ...
```

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

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

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

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

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

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

Accessing `data.frame` Columns

- To extract one column **as a vector** we can use the `$` operator (as in `murders$state`), or the square bracket operator `[which_index]` where each element of a matrix is defined by matrix[row, col].

```
first5 <- murders[1:5, ]  # take first 5 states only
first5$state  # extract with $ operator
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

```
first5[ ,"state"]  # extract with column name
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

```
first5[ ,1] # get first column
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

Accessing `data.frame` Columns

- To extract one column **as a vector** we can use the `$` operator (as in `murders$state`), or the square bracket operator `[which_index]` where each element of a matrix is defined by matrix[row, col].

```
first5 <- murders[1:5, ]  # take first 5 states only
first5$state  # extract with $ operator
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

```
first5[ ,"state"]  # extract with column name
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

```
first5[ ,1] # get first column
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

- Check `class` of an object:

```
class(murders)
```

```
## [1] "data.frame"
```

Accessing `data.frame` Columns

- To extract one column **as a vector** we can use the `$` operator (as in `murders$state`), or the square bracket operator `[which_index]` where each element of a matrix is defined by matrix[row, col].

```
first5 <- murders[1:5, ]  # take first 5 states only
first5$state  # extract with $ operator
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

```
first5[ ,"state"]  # extract with column name
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

```
first5[ ,1] # get first column
```

```
## [1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"
```

- Check `class` of an object:

```
class(murders)
```

```
## [1] "data.frame"
```

- `typeof` gives the R-internal data type:

```
typeof(murders)
```

```
## [1] "list"
```

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
```

```
# Only keep California and Texas and keep only the "state" and "total" variables
murders[murders$state %in% c("California", "Texas"), c("state", "total")]
```

```
##         state total
## 5  California  1257
## 44      Texas   805
```

Task 3

10:00

1. How many observations are there in `murders`?
2. How many variables? What are the data types of each variable?
3. Remember that the colon operator `1:10` is just short for *construct a sequence from `1` to `10`* (i.e. 1, 2, 3, etc). 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. Create a `total_percap` variable equal to the number of murders per 10,000 inhabitants by running the following code.

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

Congratulations, you've created your first variable! Click on the `murders` object to see the new variable.

Course Details

This Course

- Teach you the basics of *linear regression*, *statistical inference* and *impact evaluation*.

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***.

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 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 ***five quizzes*** on Moodle roughly every two weeks $\longrightarrow$ 40%

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 ***five quizzes*** on Moodle roughly every two weeks $\rightarrow$ 40%
2. There will be ***two take home exams / case studies*** $\rightarrow$ 60%

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 ***five quizzes*** on Moodle roughly every two weeks $\rightarrow$ 40%
2. There will be ***two take home exams / case studies*** $\rightarrow$ 60%
3. There will be *no* final exam 😅

Syllabus 🤞

Lecture 1 (today): ***Introduction***

Quiz 1 (after lecture 2)

Lectures 2 and 3: ***Tidying, Visualising and Summarising Data***

Quiz 2

Lecture 4: ***Simple Linear Regression***

Lecture 5: ***Introduction to Causality***

Midterm Project

Lecture 6: ***Multiple Linear Regression***

Lecture 7: ***Linear Regression Extensions***

Quiz 3

Lecture 8: ***Sampling***

Lecture 9: ***Confidence Intervals and Hypothesis Testing***

Quiz 4

Lecture 10: ***Statistical Inference***

Lecture 11: ***Regression Discontinuity***

Quiz 5

Lecture 12: ***Recap***

Final Project

Course Materials and Useful Resources

Econometrics

- *Mastering Metrics* by Angrist and Pischke
- *Introductory Econometrics: A Modern Approach* 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
- Awesome R Learning Resources

Course Policies

Be nice. Be honest. Don't cheat.

Class conduct and expectations

Submissions: Submit the assignments on Moodle and via email with the object "Econometrics - Midterm/Final Project - Your Name".

Late work: Won't be accepted unless you have a very good reason.

Cheating: You will get 0. Just don't cheat, it's honestly not worth it.

Work in groups: You can/should work in groups of 2-3 on the quizzes.

Mental Health

> ***Pay attention to your mental health in the same way you would your physical health.***

Seeking help is a sign of perceptive self-awareness, not weakness.

Don't hesitate to ***reach out*** to your friends, family, any professor (including myself!) if you feel the need.

Sciences Po offers free and confidential psychological support services with professionals.

Last But Not Least: Class Rep