Lecture 1: Introduction

L2 - Statistics

Gustave Kenedi

This lecture is heavily inspired by those of Marion Monnet, Youssef Souidi and Kyle Butts.

2026-01-13

Welcome to this course!

- I joined CYU in September 2025
- PhD Sciences Po (2023), postdoc in London (2023-2025)
- **Applied microeconomist**, working on **intergenerational mobility** and **educational inequalities**
- I use statistics and R programming daily
- If interested, you can find my research on my website
- **First time** teaching this course ⟶ doing my best, **mistakes** are possible
- **Icebreaker:** tennis (Federer) fan, podcast listener, economics nerd
- **Introduce yourselves**, and where you're from 🙂

About the course

Objectives:

- Give you a background on **statistical theory** and their **application**
- Teach you how to perform **statistical analyses** in the R programming language
- Prepare you to succeed in **econometrics courses**

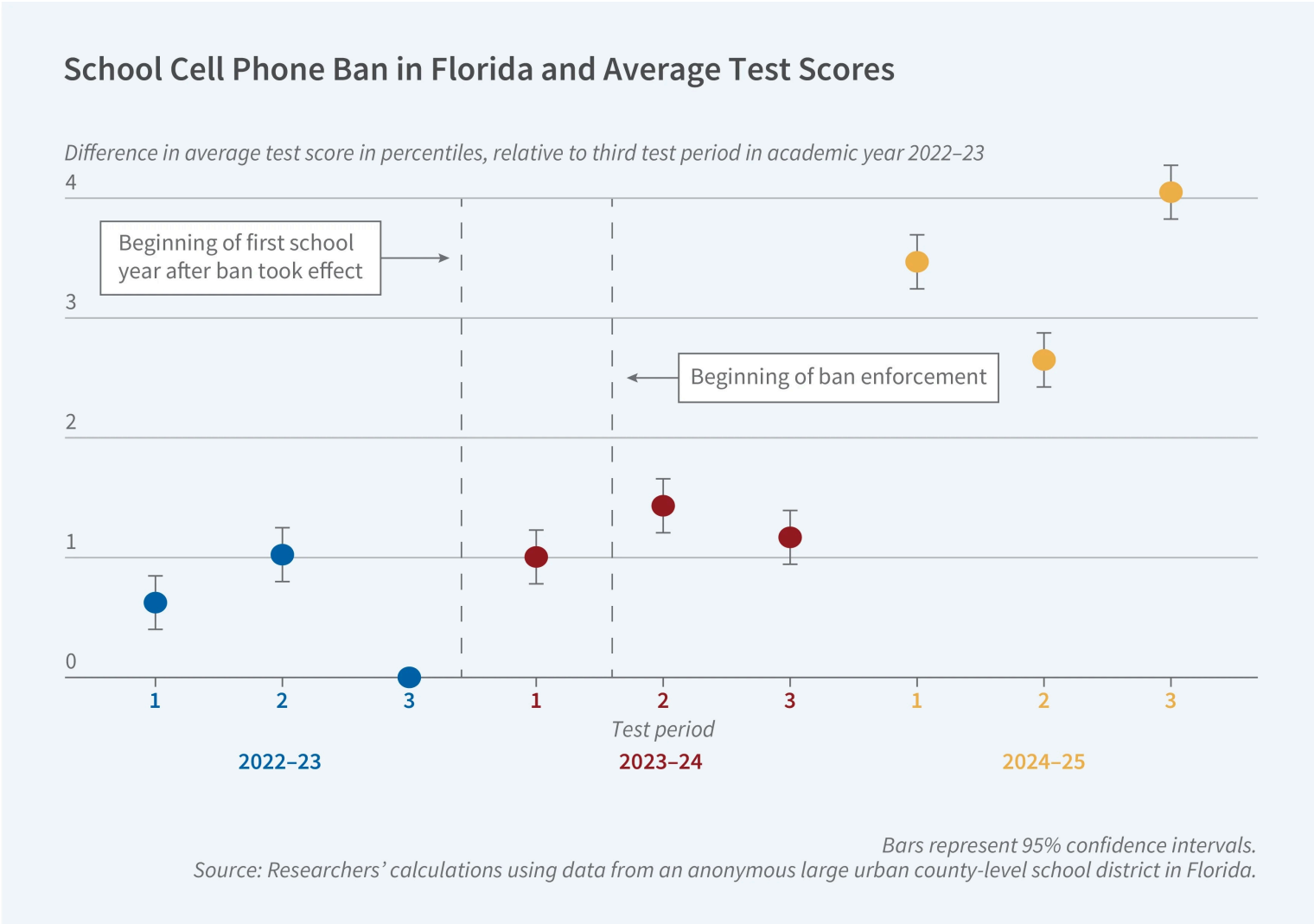
Prerequisites:

- Probability theory
- **Appetite for learning**, enthusiasm, hard work, and perseverance

What you'll learn:

- Manipulate, summarize and visualize data with R
- The main tools of **statistical inference**

A quick word about in-class technology and learning



Source: NBER Digest (2025), *School Cell Phone Bans and Student Achievement*

A quick word about in-class technology and learning

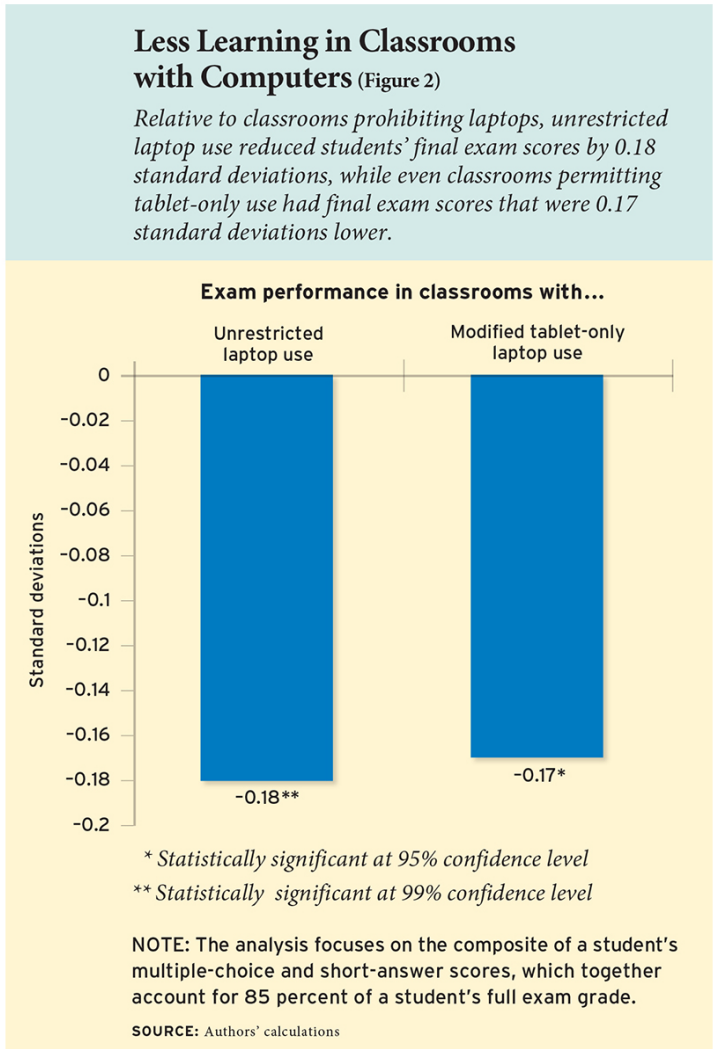
RESEARCH

For better learning in college lectures, lay down the laptop and pick up a pen

Susan M. Dynarski
August 10, 2017

Source: Dynarski (2017)

A quick word about in-class technology and learning



Source: Carter, Gerrnberg and Walker (2017), Should Professors Ban Laptops?, *Education Next*

A quick word about in-class technology and learning

- I'm not saying you should *not* use your laptop/tablet in class
- You should use it **wisely**
- Listen to me, ask questions, and look at the class slides
- Don't randomly check your **phone** to see if you have notifications (I'm the same, trust me!)
- When doing R try to focus on the task at hand

Today's lecture

1. Why is statistics useful?

2. The data science process

3. R101

4. Your first `R` plot

5. Anatomy of a `data.frame`

6. Course details

Why is statistics useful?

What is *statistics*?

Statistics gives us a way of linking *economic theory* with the real world through data analysis

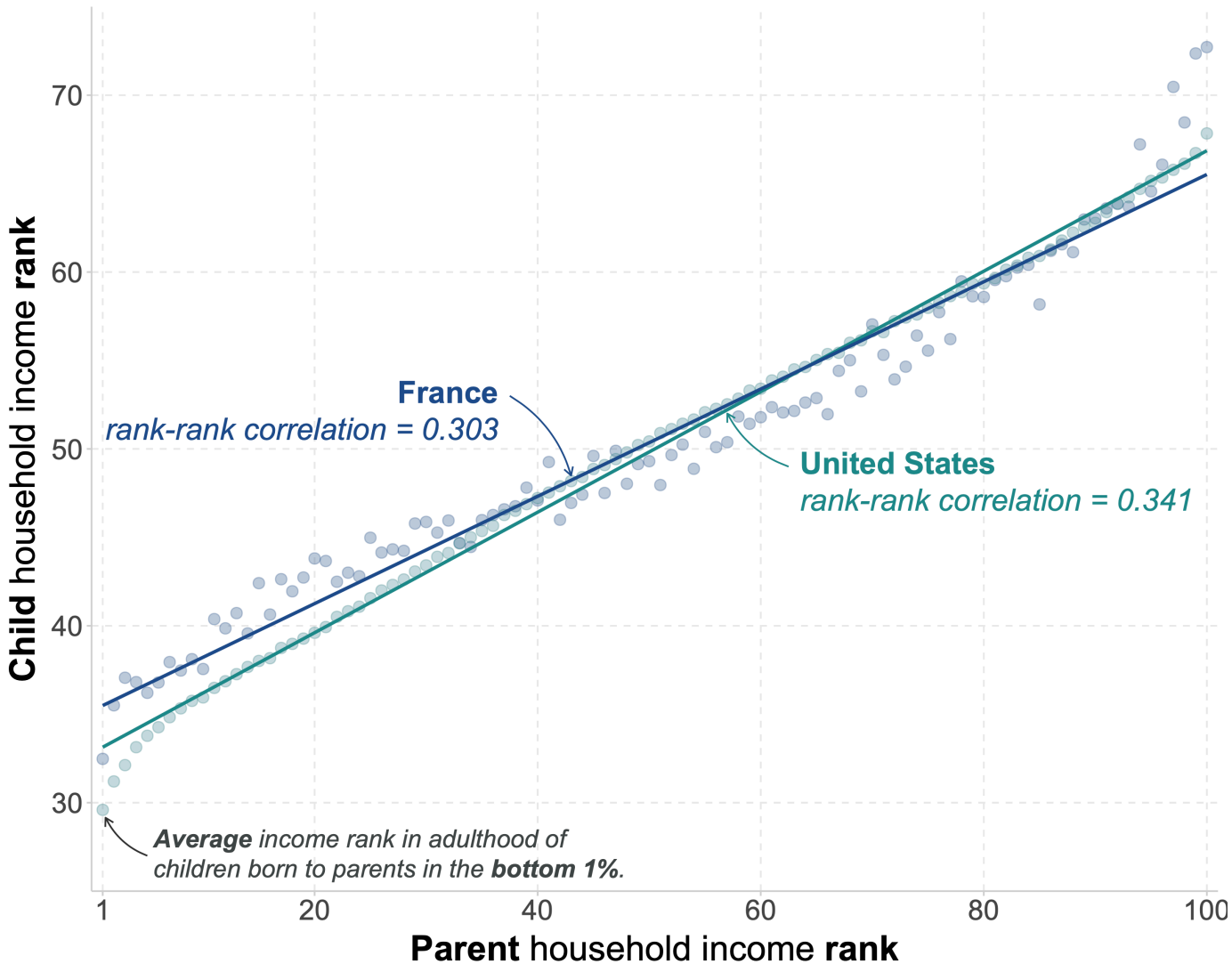
- How did prices change after the tax increase?
- How did students respond to tuition fees?
- Does more education lead to higher wages?

Statistics allows us to translate datasets into usable information

- Summary statistics help describe large groups of data
- Use statistics to make predictions
- Statistics helps us inform our decision making

Real-world examples

Measuring intergenerational mobility



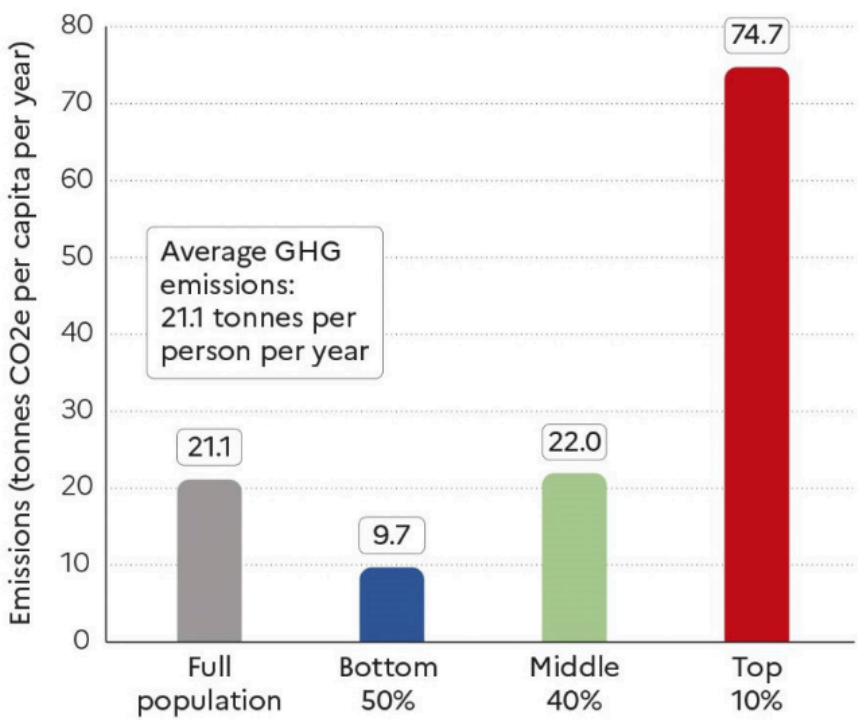
→ to summarize this relationship, we use **percentile ranks**

Source: Kenedi and Sirugue (2023)

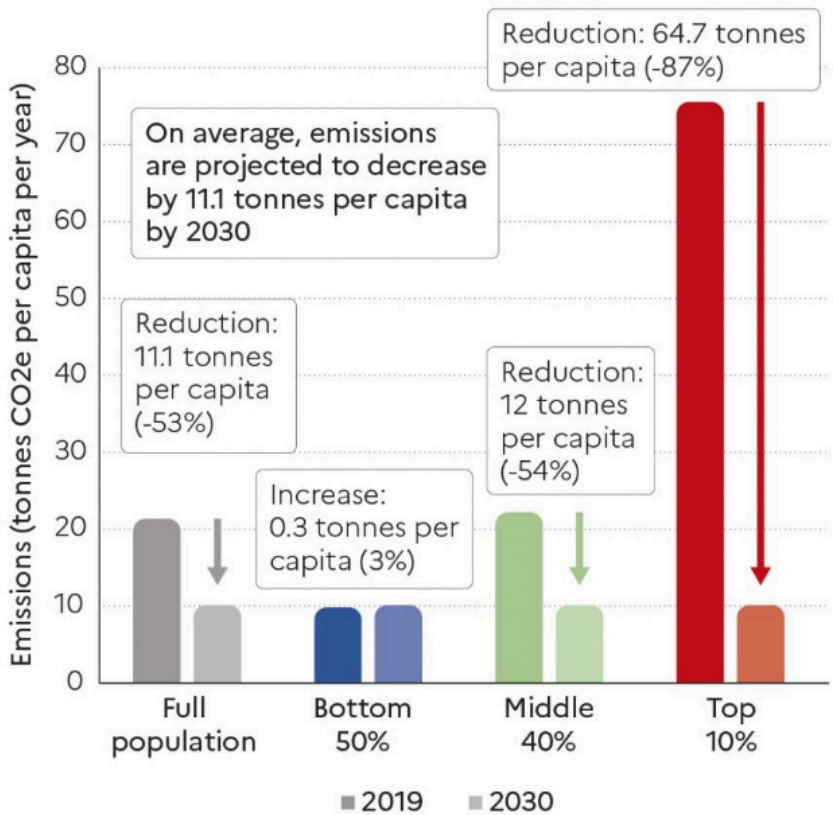
How to meet CO2 emissions target?

The US case

Per capita emissions by income group in the US, 2019 estimates



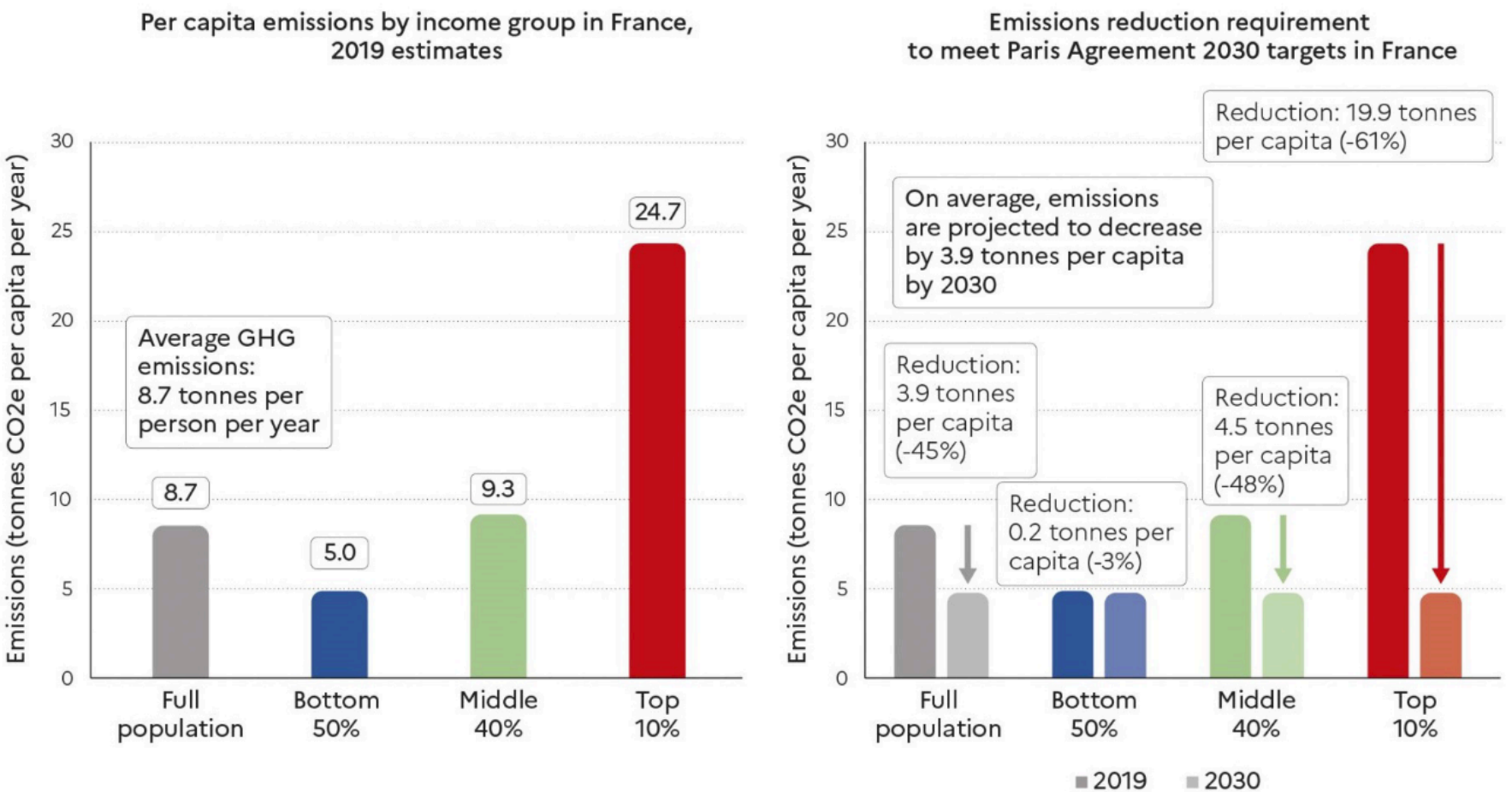
Emissions reduction requirement to meet Paris Agreement 2030 targets in the US



Source: World Inequality Report 2022

How to meet CO2 emissions target?

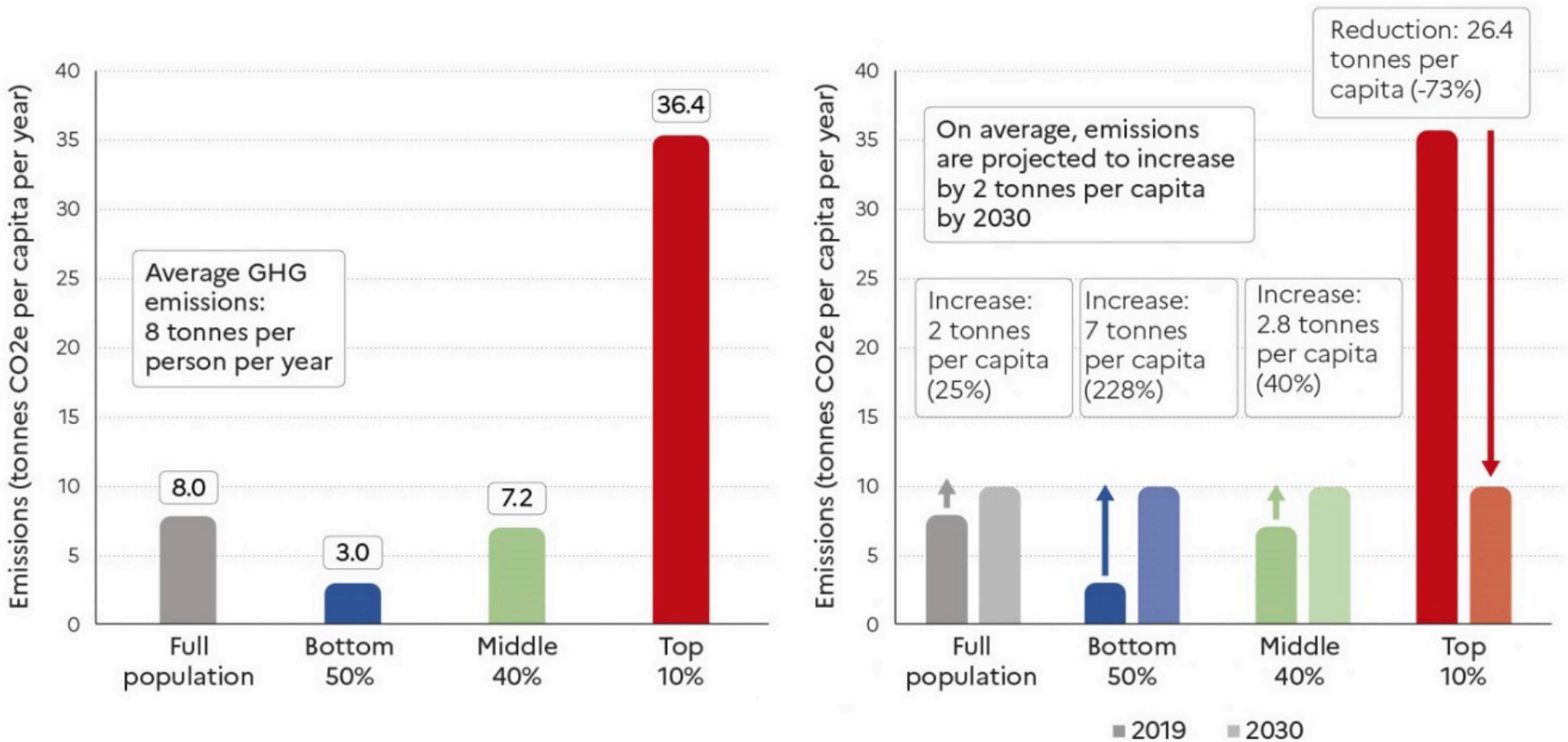
The French case



Source: World Inequality Report 2022

How to meet CO2 emissions target?

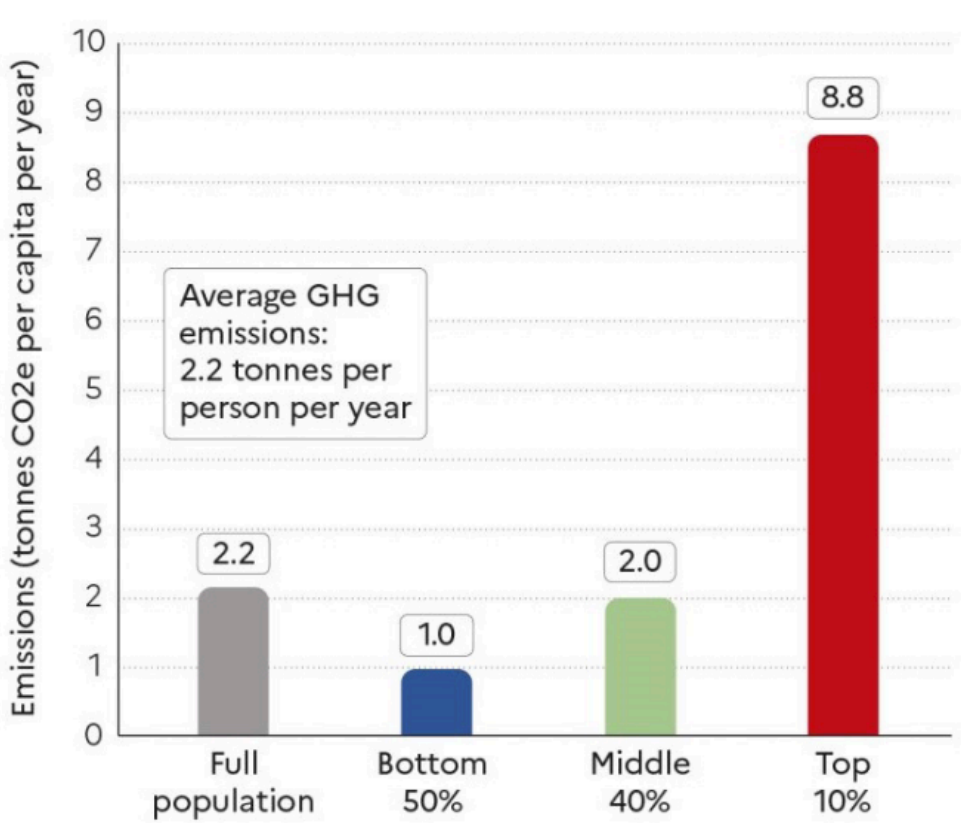
The Chinese case

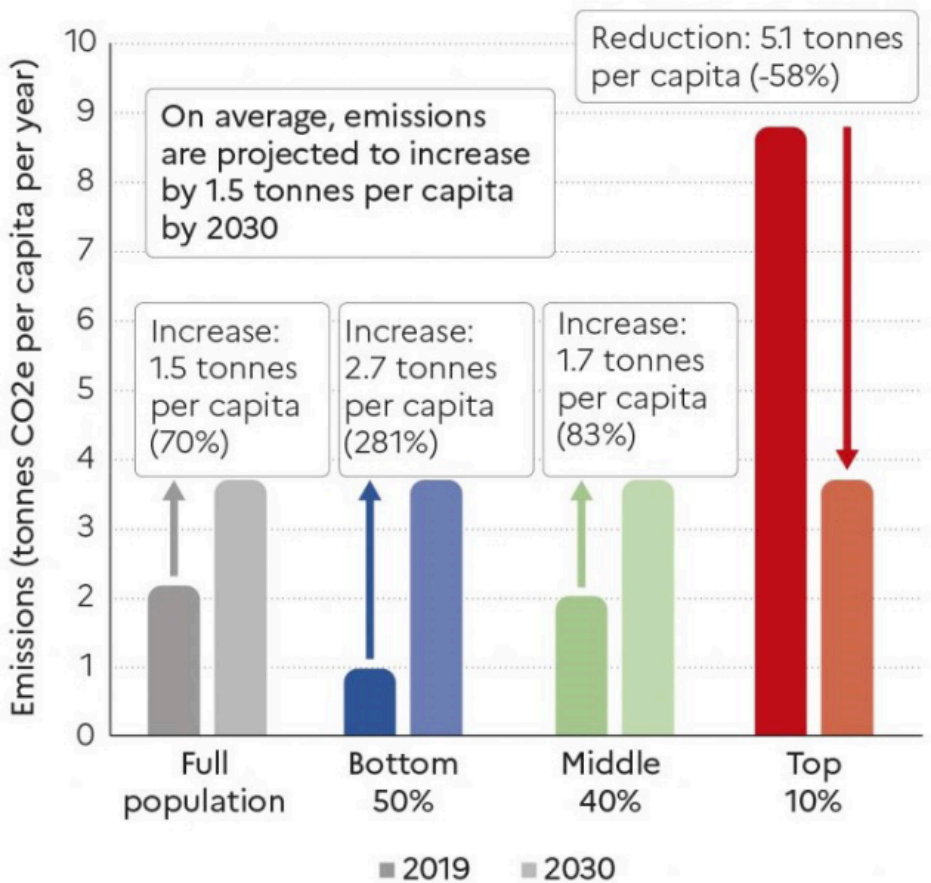


Source: World Inequality Report 2022

How to meet CO2 emissions target?

The Indian case



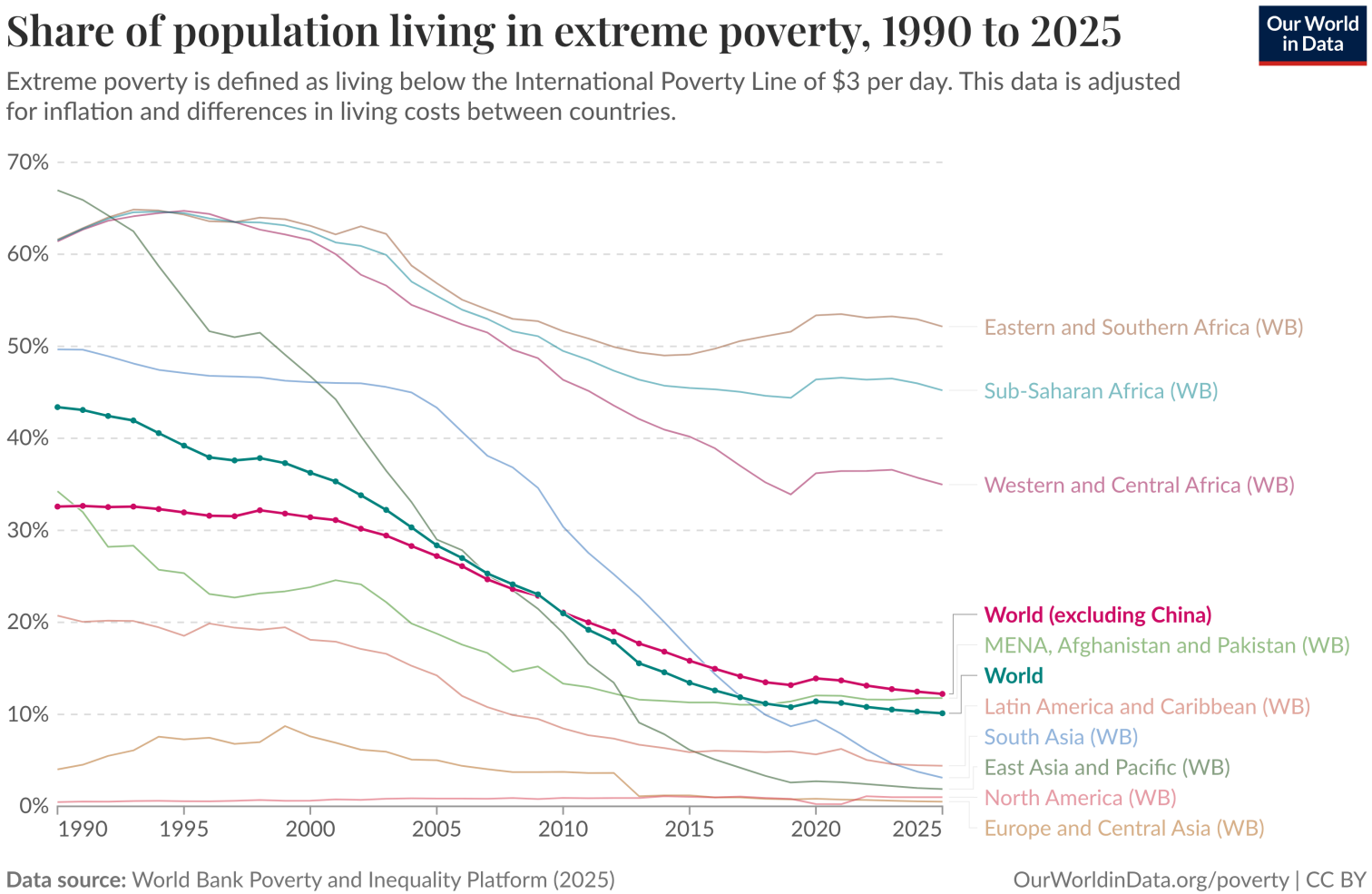


Source: World Inequality Report 2022

How to meet CO2 emissions target?

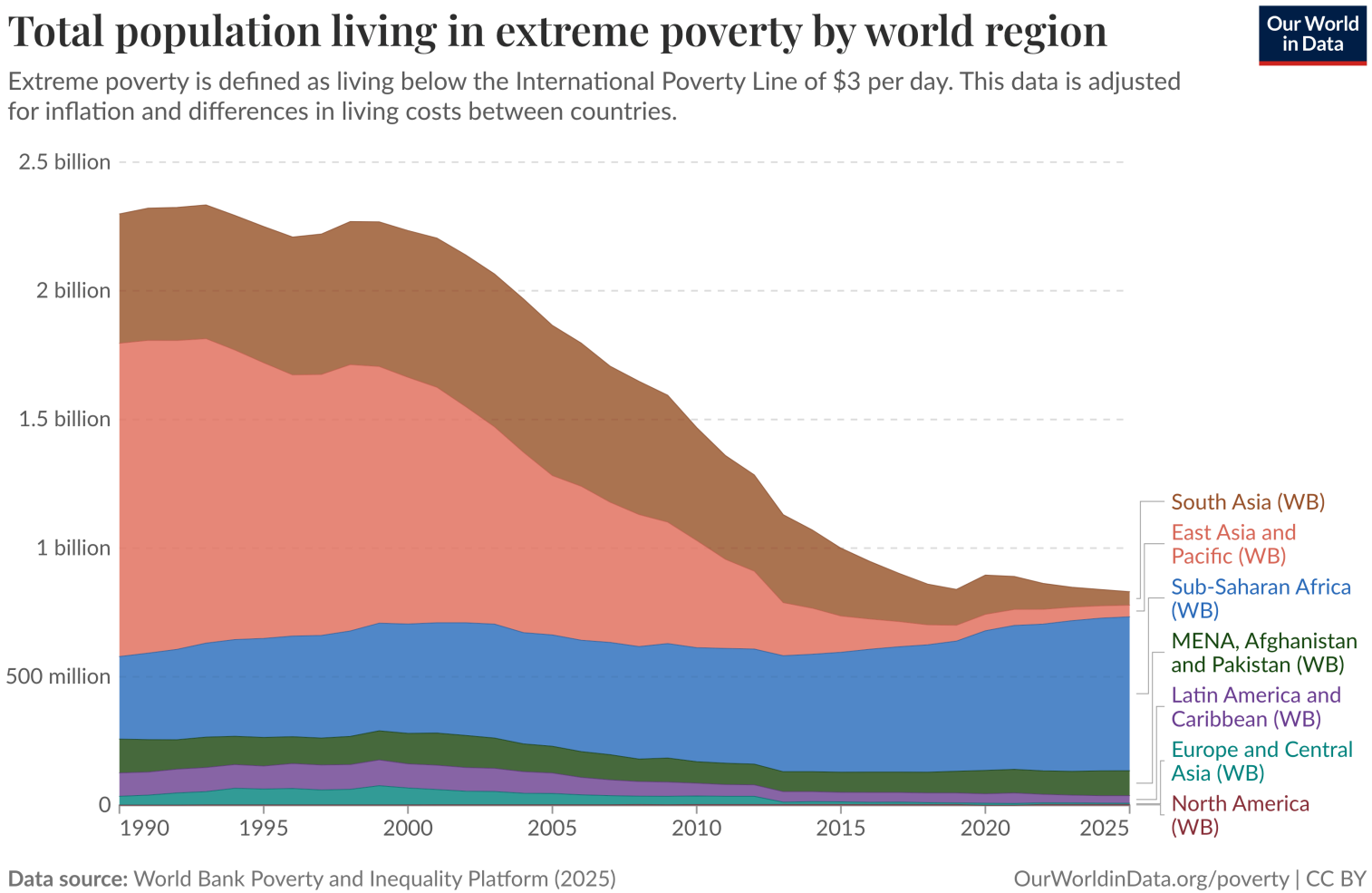
- CO2 emissions are highly unequally distributed among the population
- Per capita emissions are higher in developed countries
- Within countries, richer people emit more CO2
- Huge implications for public policies aiming to adress this issue
- To summarize these informations, the World Inequality Report uses **quantiles**

Massive decline in extreme poverty



Source: Our World in Data

Massive decline in extreme poverty

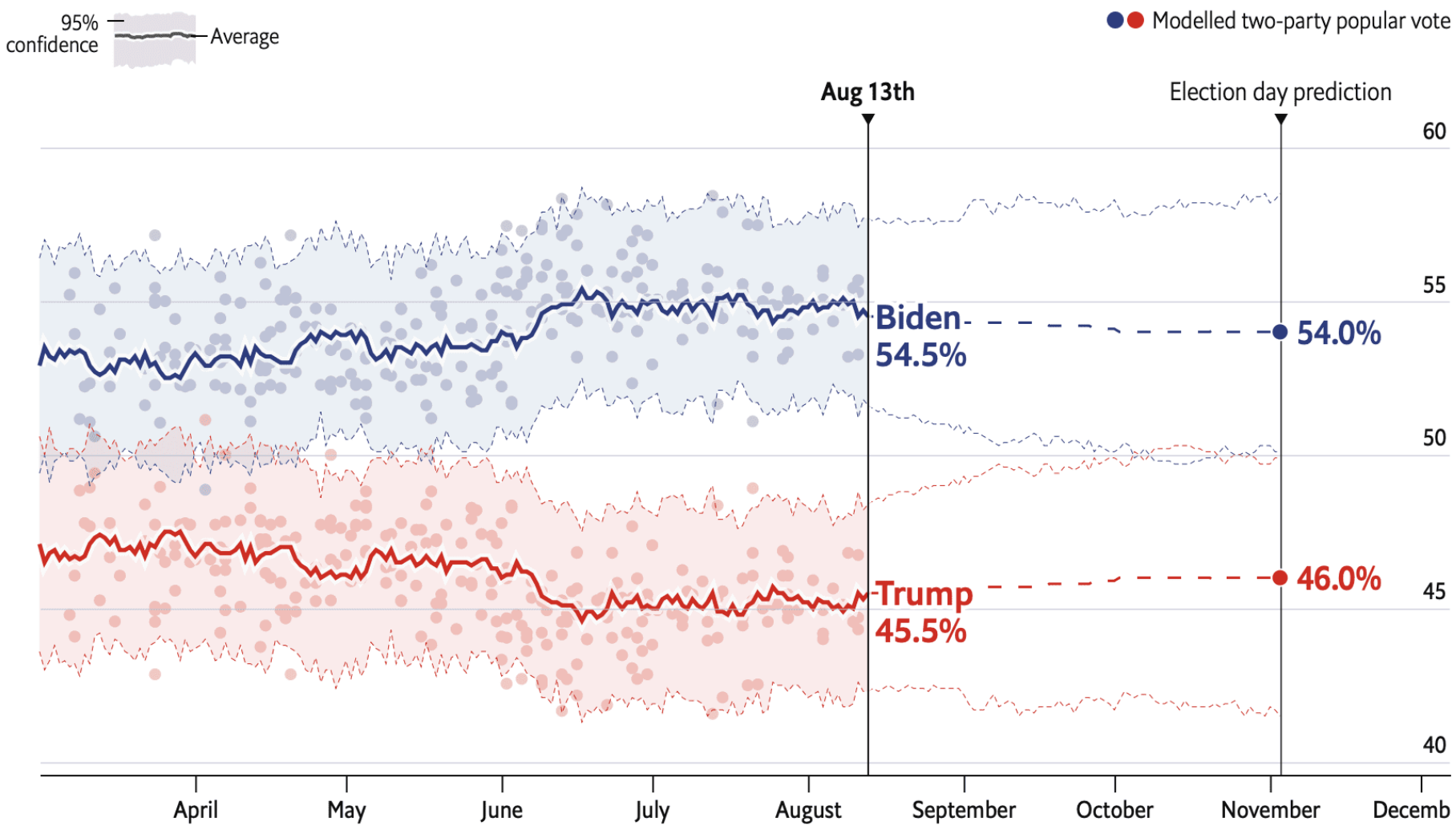


Source: Our World in Data

Massive decline in extreme poverty

- The share and number of people living in extreme poverty went through a dramatic reduction during the last 35 years
- However, this evolution does not apply equally to all regions
- The vast majority of remaining extreme poverty is in Sub-Saharan Africa
- To summarize this information, Our World in Data uses **statistical measures of evolution**

Forecasting election results



Source: Andrew Gelman

Forecasting election results

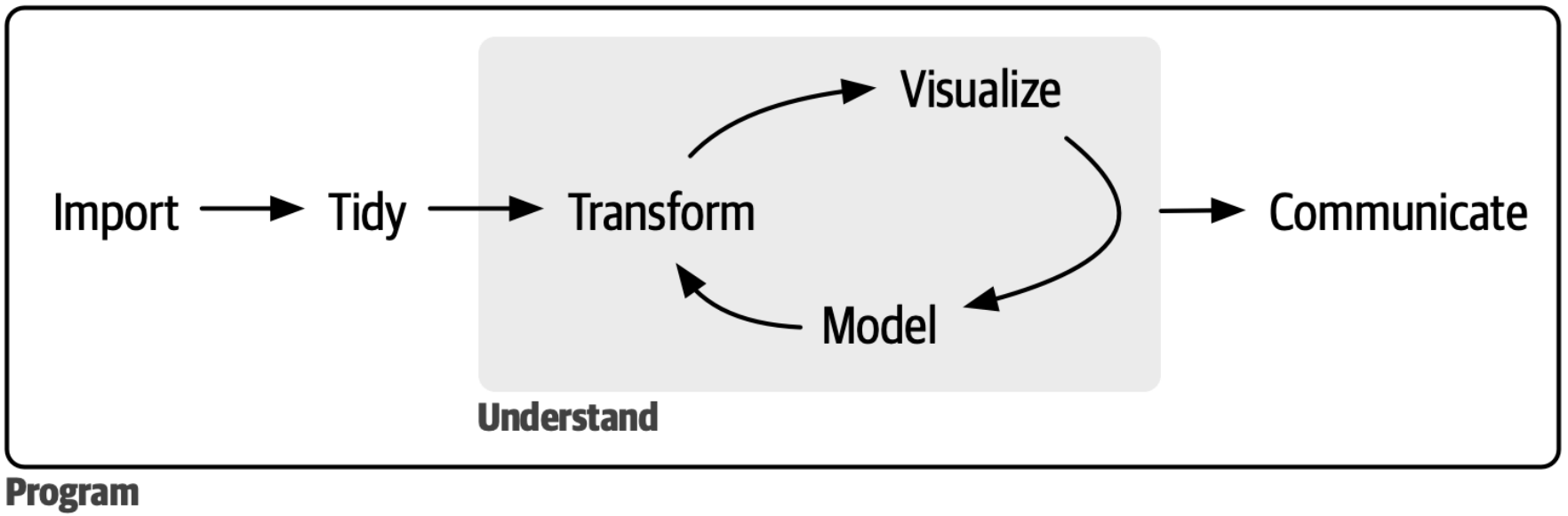
- Biden was predicted to obtain 55% of the vote vs 45% for Trump in August 2020
- To say so, the following statistical tools were used: **mean**, standard deviation, normal distribution, **confidence intervals**

Today's lecture

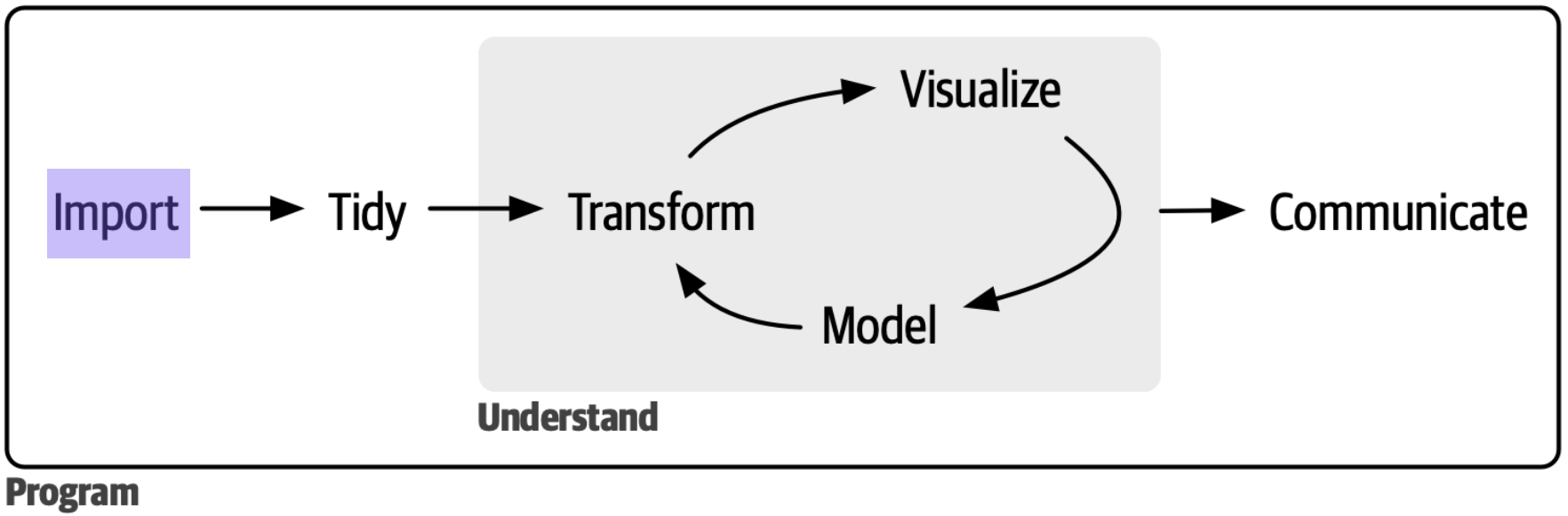
1. Why is statistics useful? ✓
2. The data science process
3. R101
4. Your first `R` plot
5. Anatomy of a `data.frame`
6. Course details

The data science process

The data science process

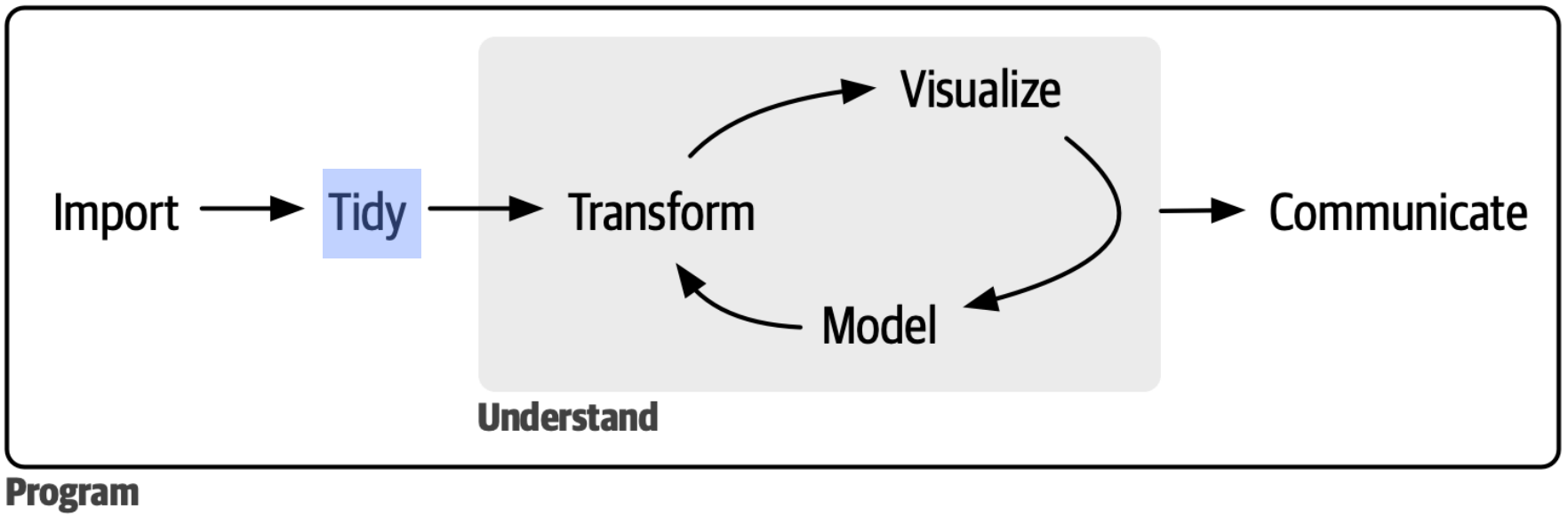


Step 1: import data



Step 1: Import: load a data file

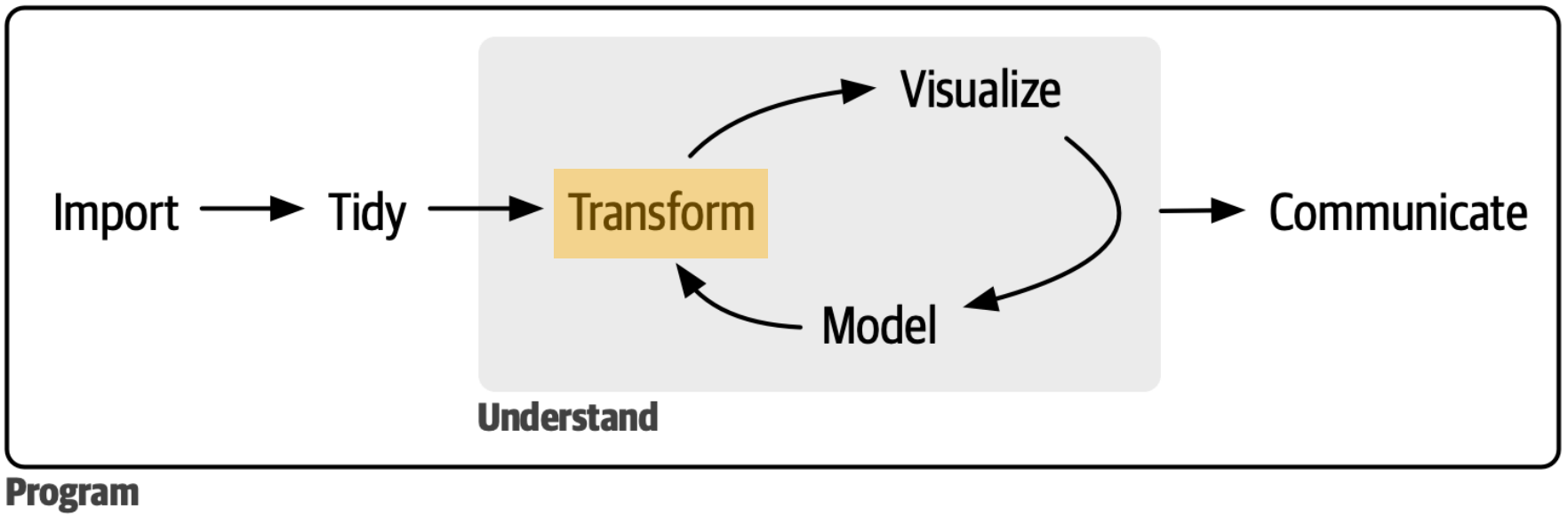
Step 2: tidy data



Step 1: Import: load a data file

Step 2: Tidy: each column ⟶ a variable; each row ⟶ an observation

Step 3: transform data

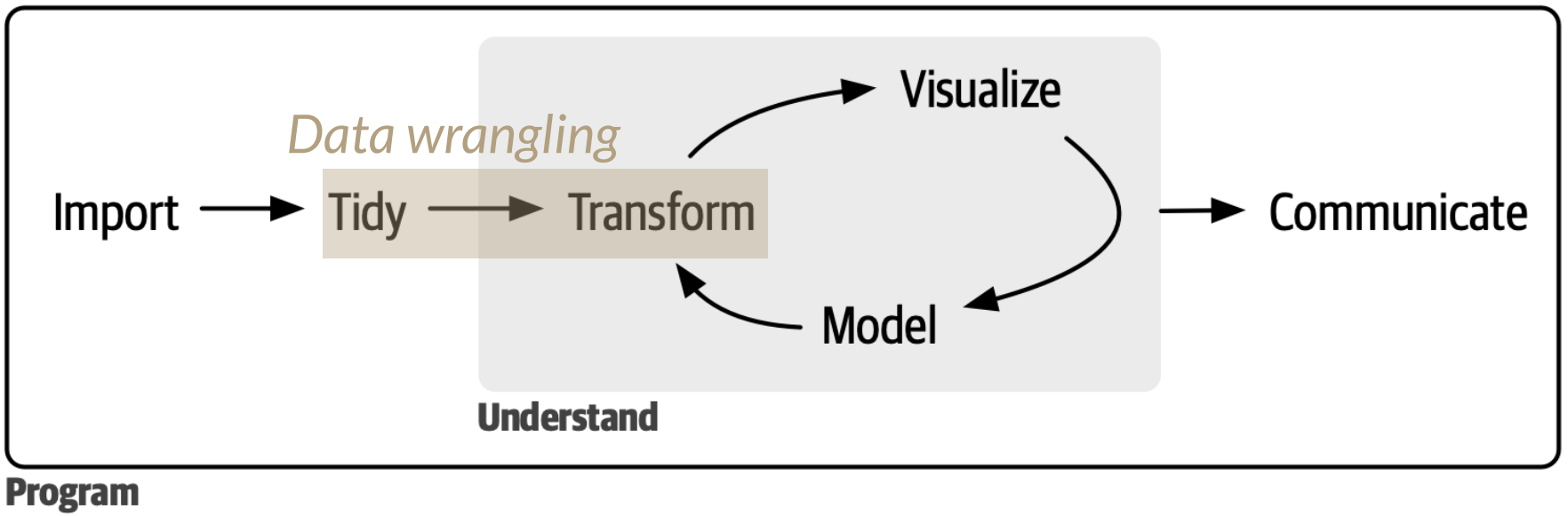


Step 1: Import: load a data file

Step 2: Tidy: each column ⟶ a variable; each row ⟶ an observation

Step 3: Transform: filter data, create new variables, compute summary stats

Step 3: transform data

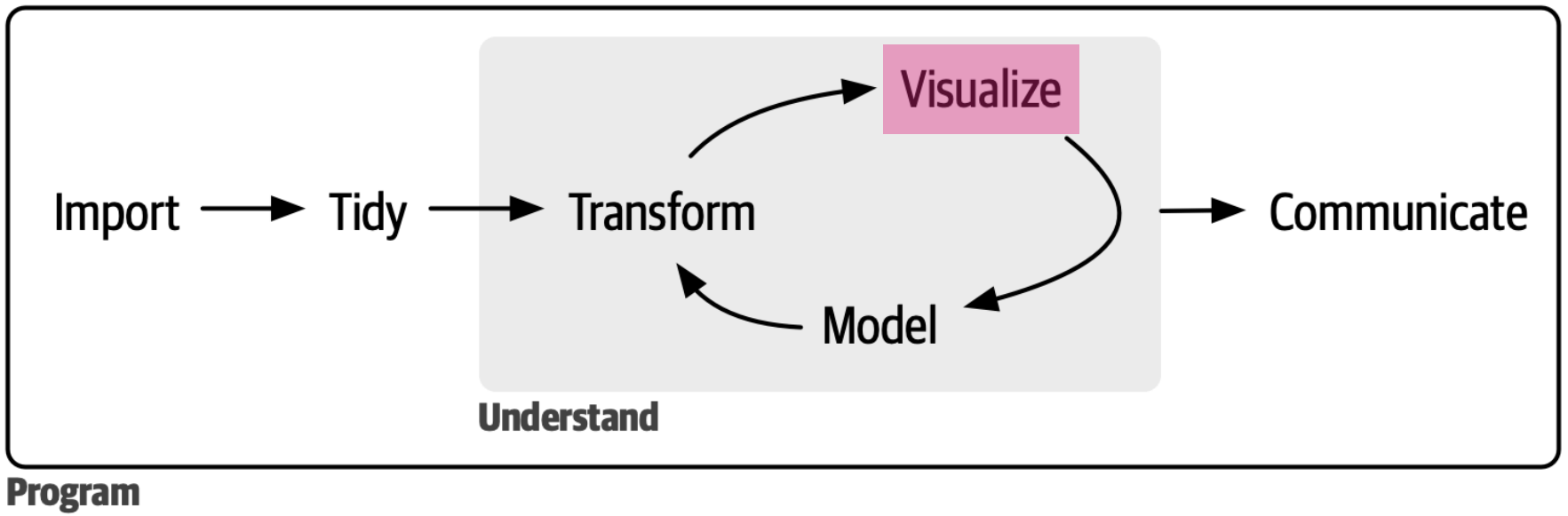


Step 1: Import: load a data file

Step 2: Tidy: each column ⟶ a variable; each row ⟶ an observation

Step 3: Transform: filter data, create new variables, compute summary stats

Step 4: visualize data



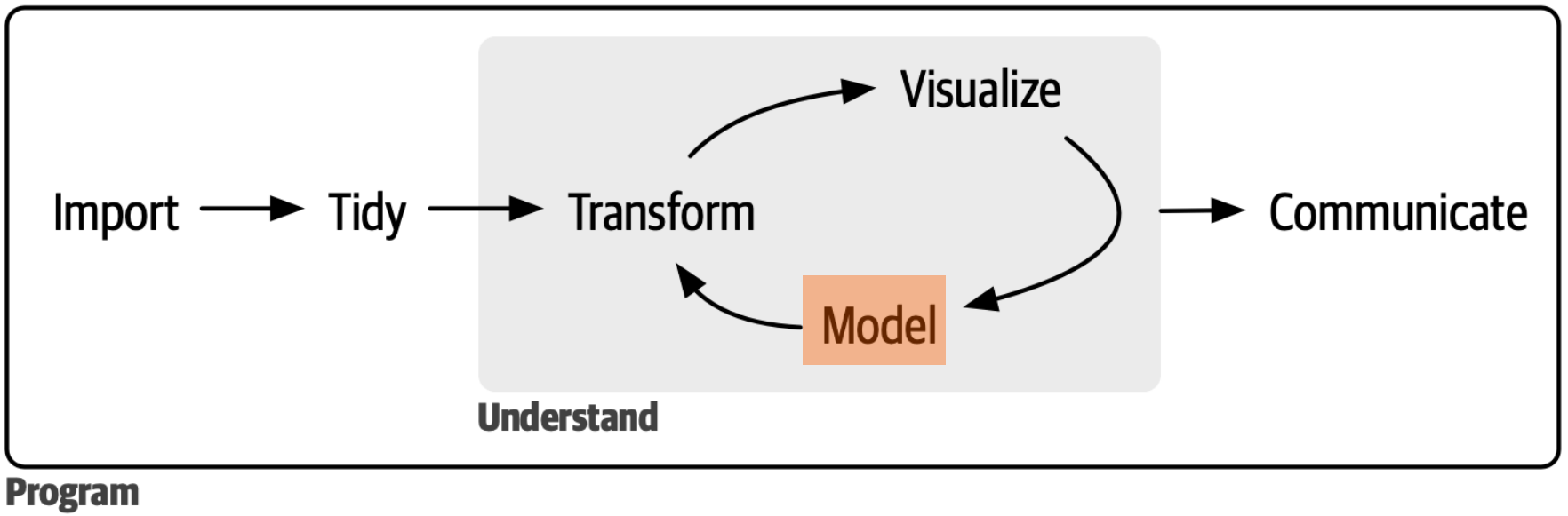
Step 1: Import: load a data file

Step 2: Tidy: each column ⟶ a variable; each row ⟶ an observation

Step 3: Transform: filter data, create new variables, compute summary stats

Step 4: Visualize: produce plots to understand the data

Step 5: model data



Step 1: Import: load a data file

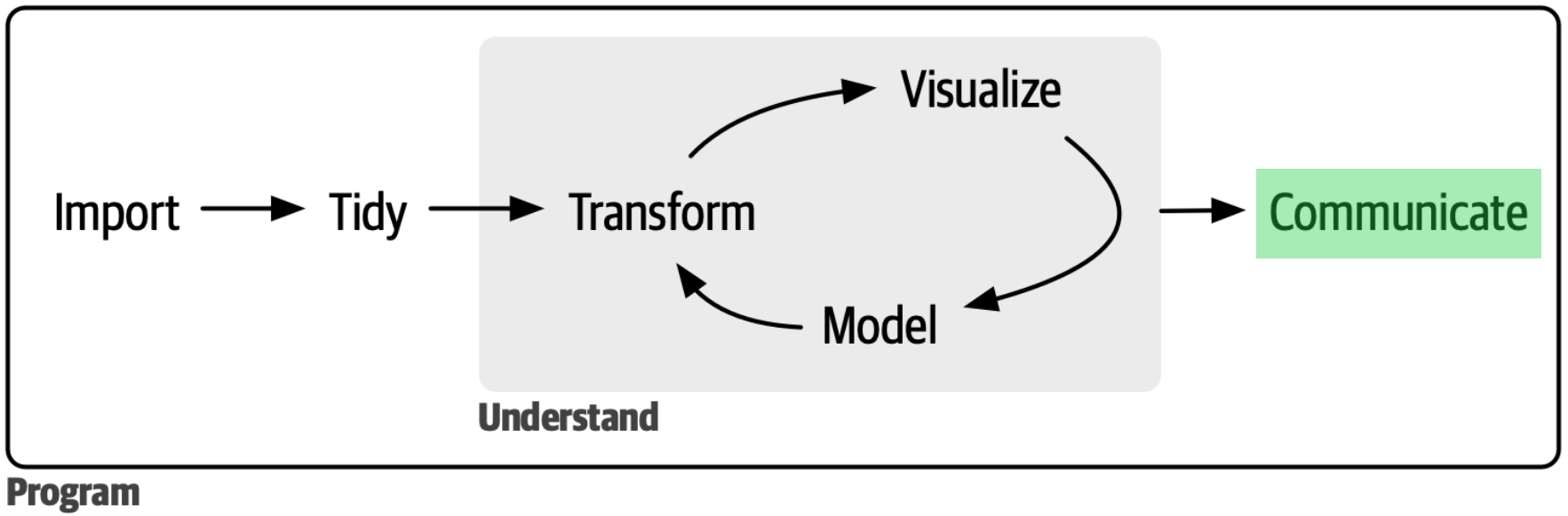
Step 2: Tidy: each column ⟶ a variable; each row ⟶ an observation

Step 3: Transform: filter data, create new variables, compute summary stats

Step 4: Visualize: produce plots to understand the data

Step 5: Model: make assumptions about data to answer question

Step 6: communicate



Step 1: Import: load a data file

Step 2: Tidy: each column ⟶ a variable; each row ⟶ an observation

Step 3: Transform: filter data, create new variables, compute summary stats

Step 4: Visualize: produce plots to understand the data

Step 5: Model: make assumptions about data to answer question

Step 6: Communicate: communicate results to others

The data science process

Step 1: Import: load a data file

Step 2: Tidy: each column ⟶ a variable; each row ⟶ an observation

Step 3: Transform: filter data, create new variables, compute summary stats

Step 4: Visualize: produce plots to understand the data

Step 5: Model: make assumptions about data to answer question

Step 6: Communicate: communicate results to others

To undertake all these steps we need a **tool**.

⟶ we will use the programming language R

Today's lecture

1. Why is statistics useful? ✓
2. The data science process ✓
3. R101
4. Your first `R` plot
5. Anatomy of a `data.frame`
6. Course details

R 101

What is R?

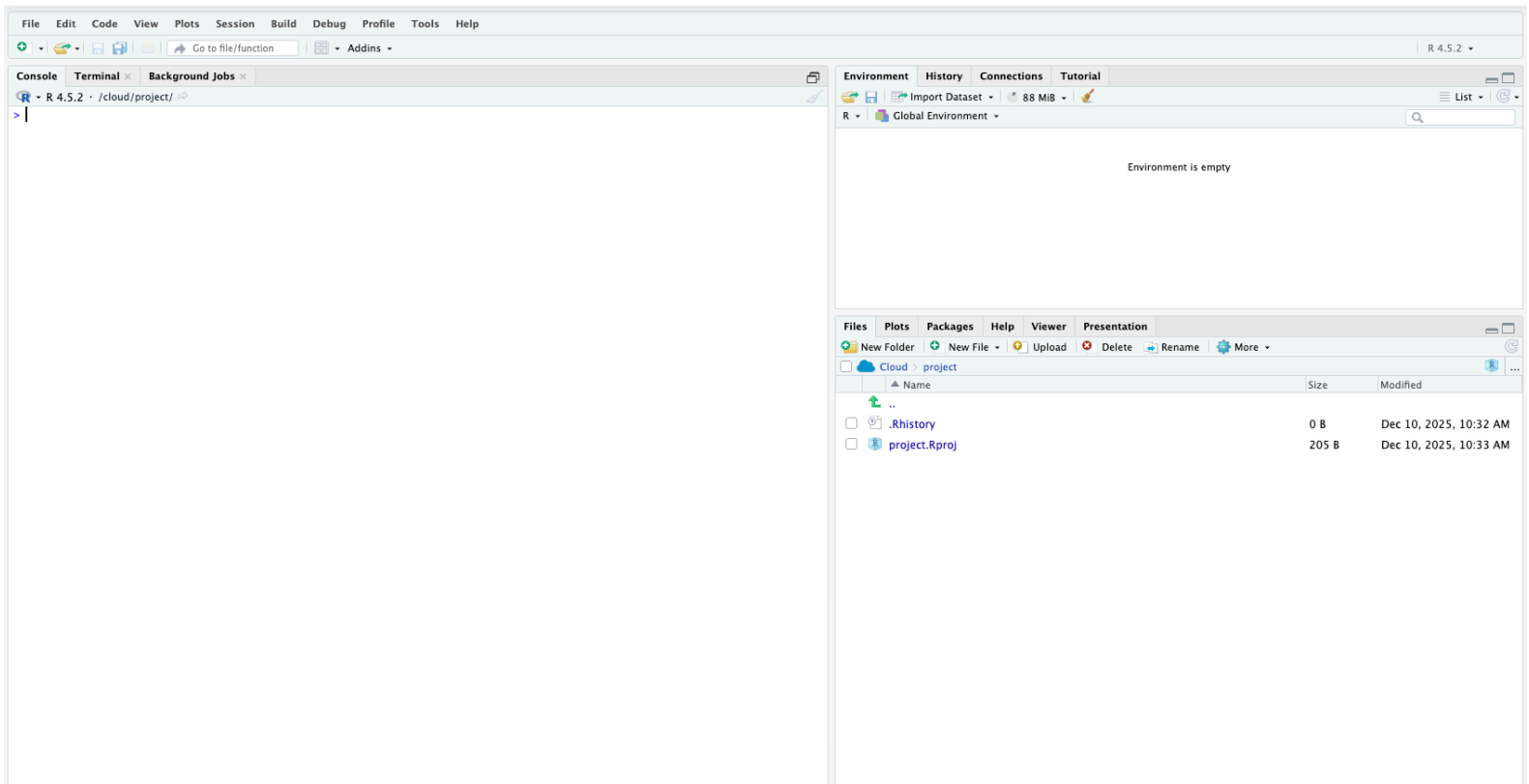
- R is a **programming language** and free software environment for **statistical computing and graphics**
- The R language is widely (and increasingly) used in **academic and non-academic research** in fields like:
  - economics, statistics, biostatistics
- Things you can do with R:
  - Reports
  - Nice plots
  - Interactive dashboards
  - Art
  - These slides

Why are we using R? [1]

1. Free and open source: saves both you and the university  .
2. Very flexible and powerful: adaptable to nearly any task (data cleaning, data visualization, statistics, econometrics, spatial data analysis, machine learning, web scraping, etc.).
3. Thriving online community: (almost) always have a solution to your problem(s), AIs very good as well.
4. Very valuable and useful tool: worth putting in the work [2].

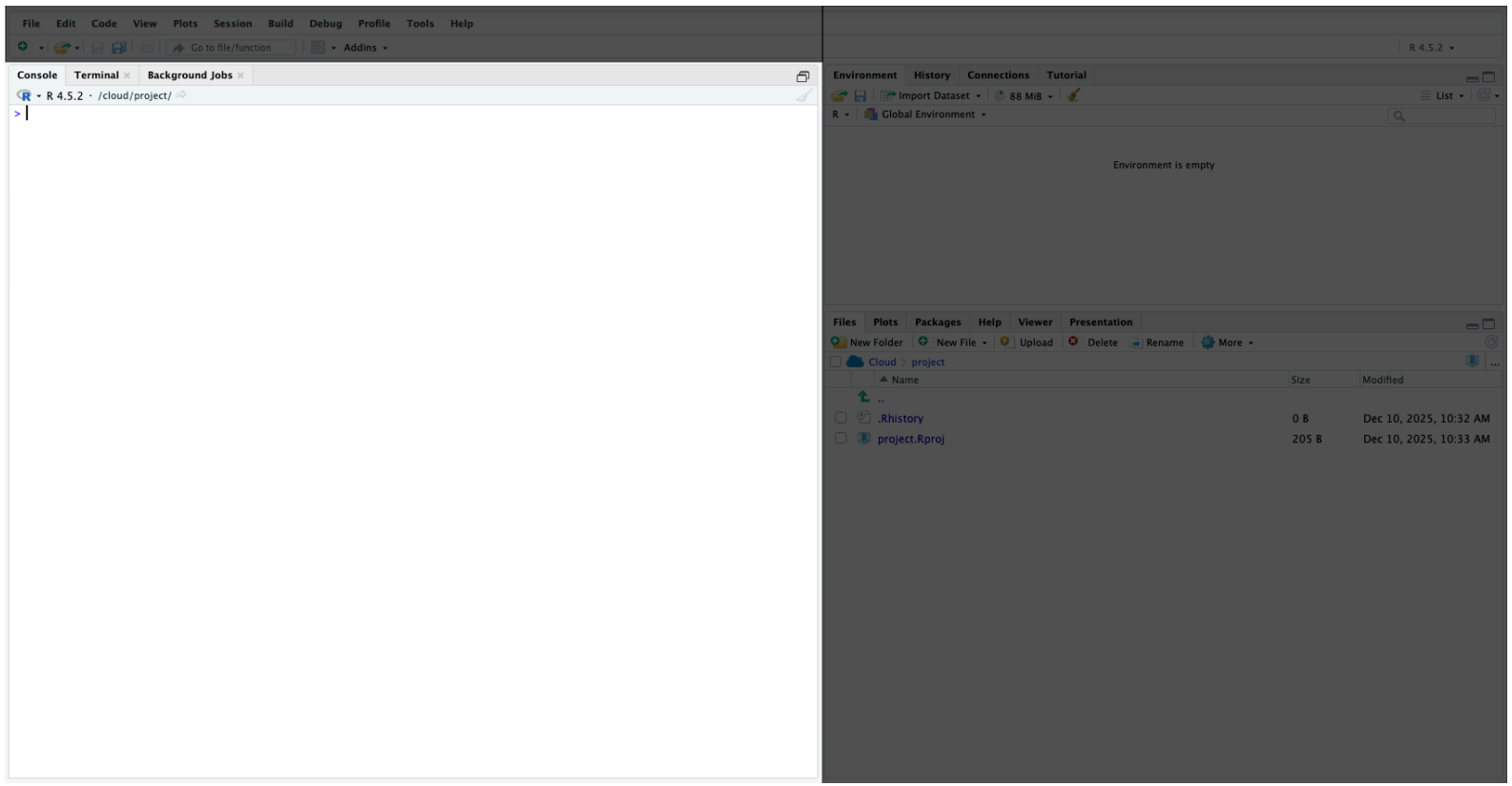
The `RStudio` interface

- `R`: a **programming language**
- `RStudio` or `posit.cloud`: an **integrated development environment** (IDE) to work with `R`.



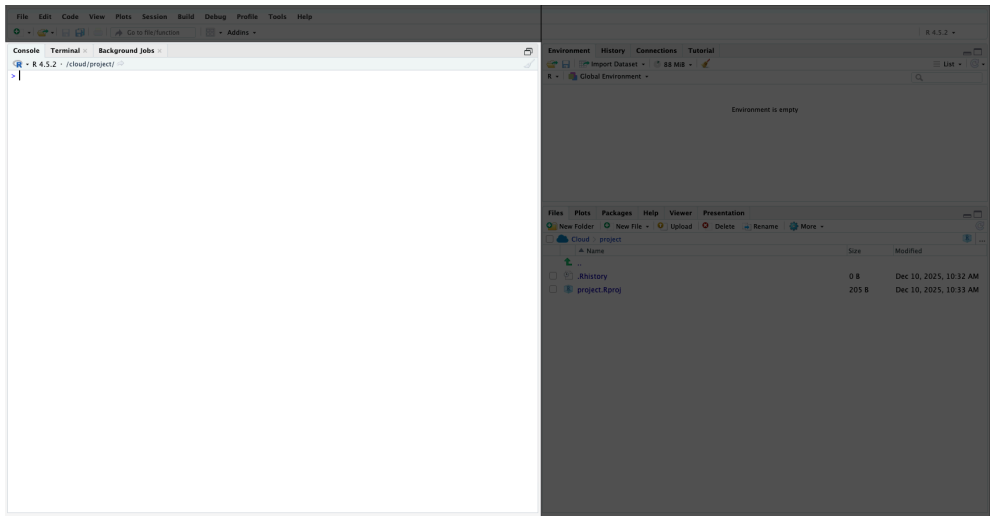
The `RStudio` interface

- R: a **programming language**
- `RStudio` or `posit.cloud`: an **integrated development environment** (IDE) to work with R.



The `console` panel

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



```
1  1 + 1
```

```
[1] 2
```

- You can also write comments with #

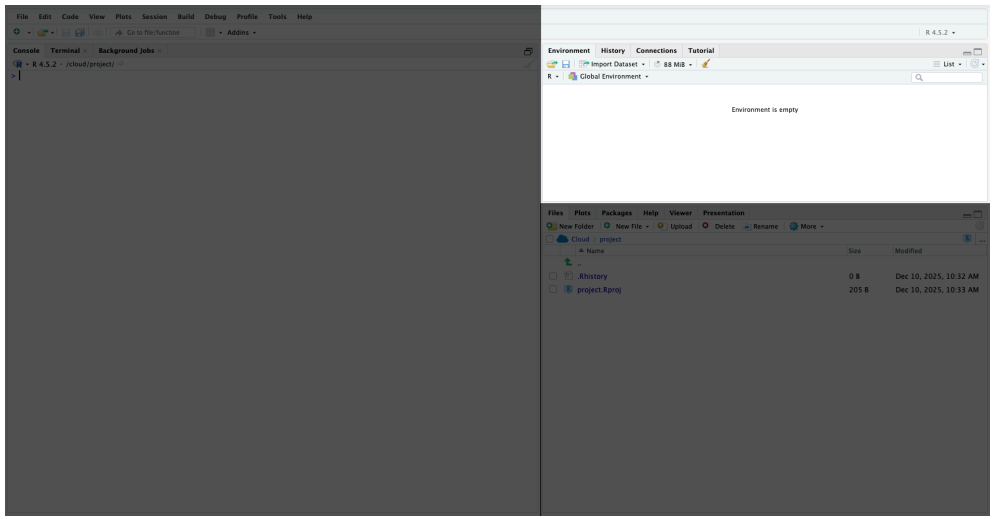
```
1  # 1 + 1 this code will not run
```

The `environment` panel

- This is your `data` objects are stored
  - For instance we can assign a value to an `object` using `<-`

```
1  x <- 1
2  x
```

```
[1] 1
```



Your environment should now have an object called `x`, which takes value 1

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

```
1  x + 1
```

```
[1] 2
```

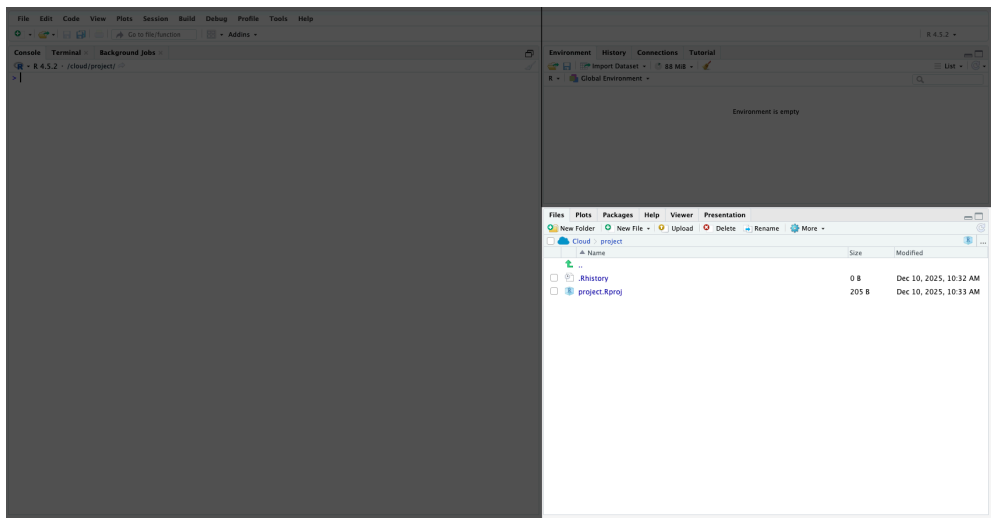
You can also modify that object at any point:

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

```
[1] 2
```

The `files/plots/...` panel

- **Files:** shows your working directory
- **Plots:** where R returns plots
- **Packages:** library of tools that we can load if needed
- **Help:** documentation on R functions

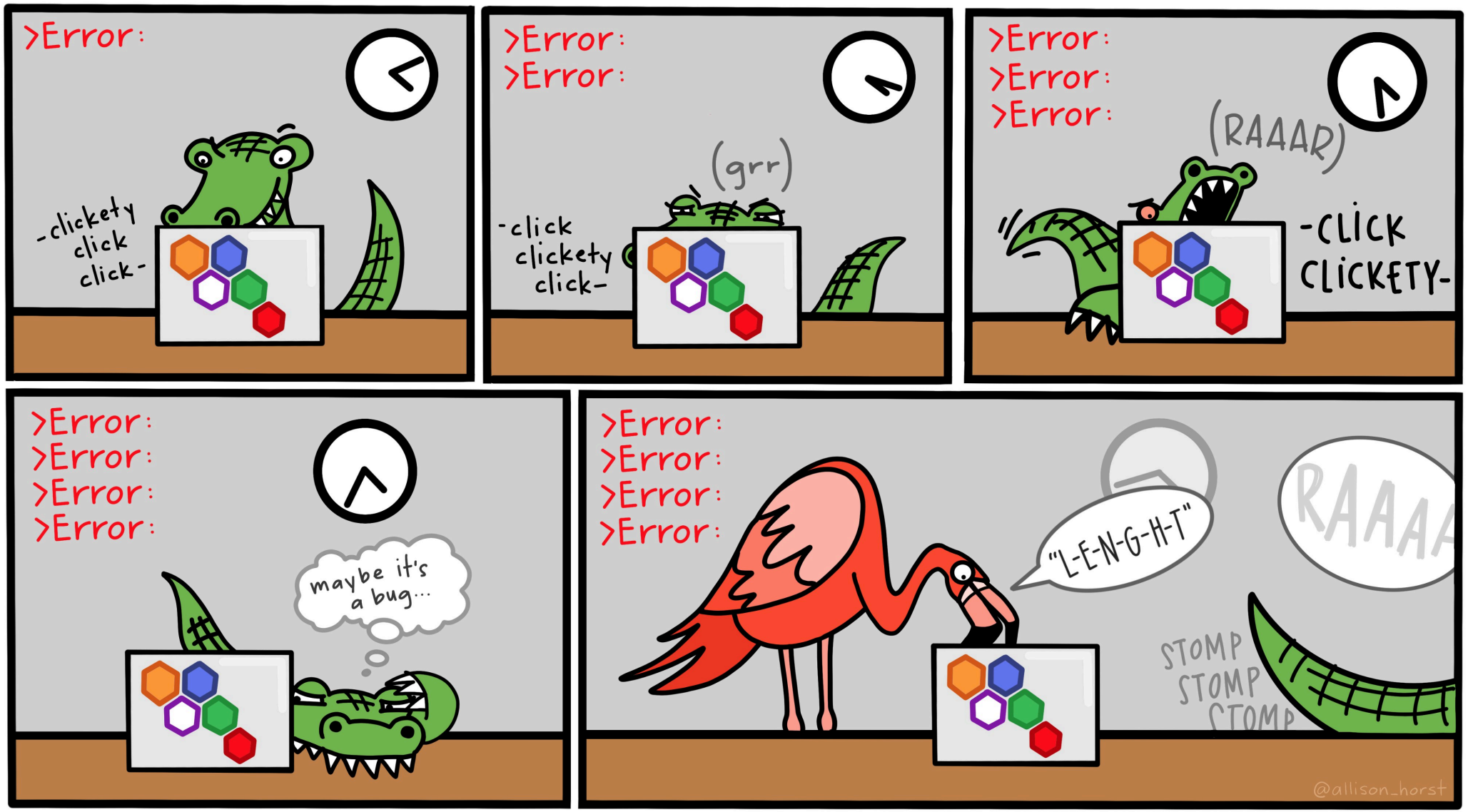


- Enter `?getwd()` in the console to see what a help file looks like

```
1  ?getwd()
```

- The help page:
  - *describes* what the command does
  - *explains* the different parameters of the command
  - gives *examples* of how to use the command

Collaboration is encouraged!



Your turn! #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 `<-`. What is `x_3` equal to?

R Packages

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

```
1  install.packages("tidyverse") # we will use the tidyverse a lot!
```

- The `tidyverse` package is **now installed** on your computer: you won't have to do it again
- To *use* the contents of a package, we must load it from our library using `library`:

```
1  library(tidyverse)
```

- Each time you start a **new R session**, you'll have to load the packages you need with `library()`
- To make it clear which package an object comes from you can use the package name followed by two colons, e.g. `dplyr::mutate()`

Today's lecture

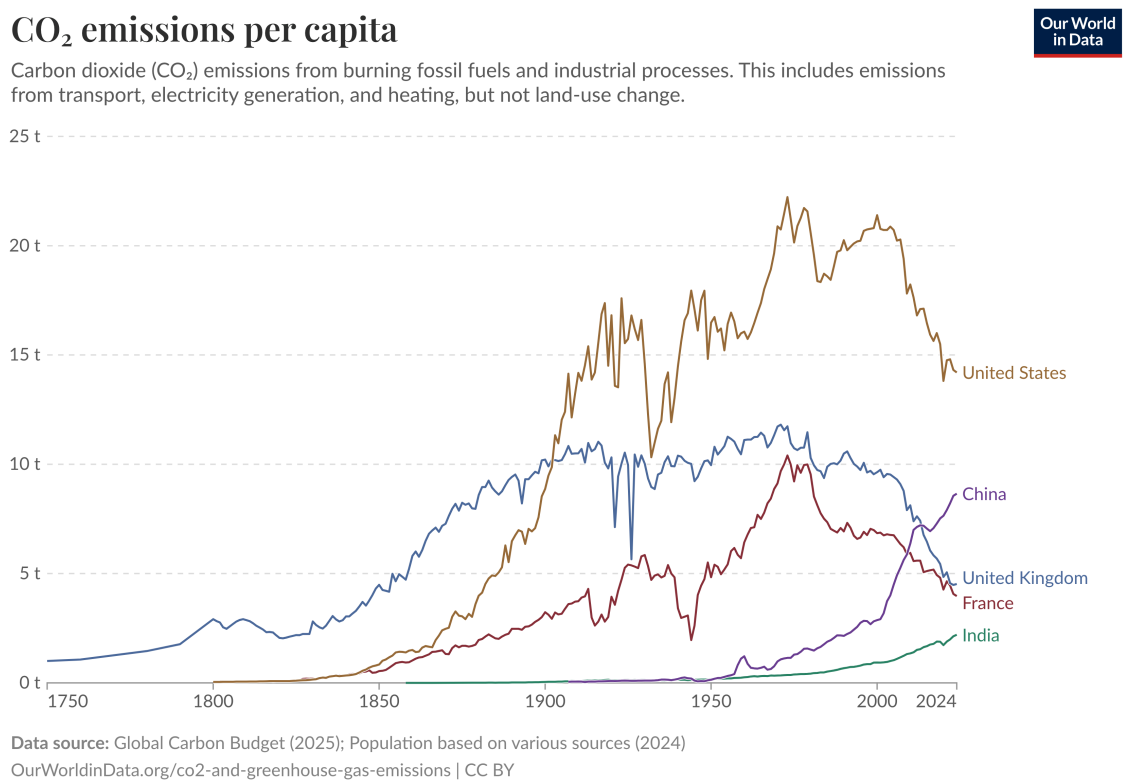
1. Why is statistics useful? ✓
2. The data science process ✓
3. R101 ✓
4. Your first `R` plot
5. Anatomy of a `data.frame`
6. Course details

Your first R plot

CO2 emissions per capita over time

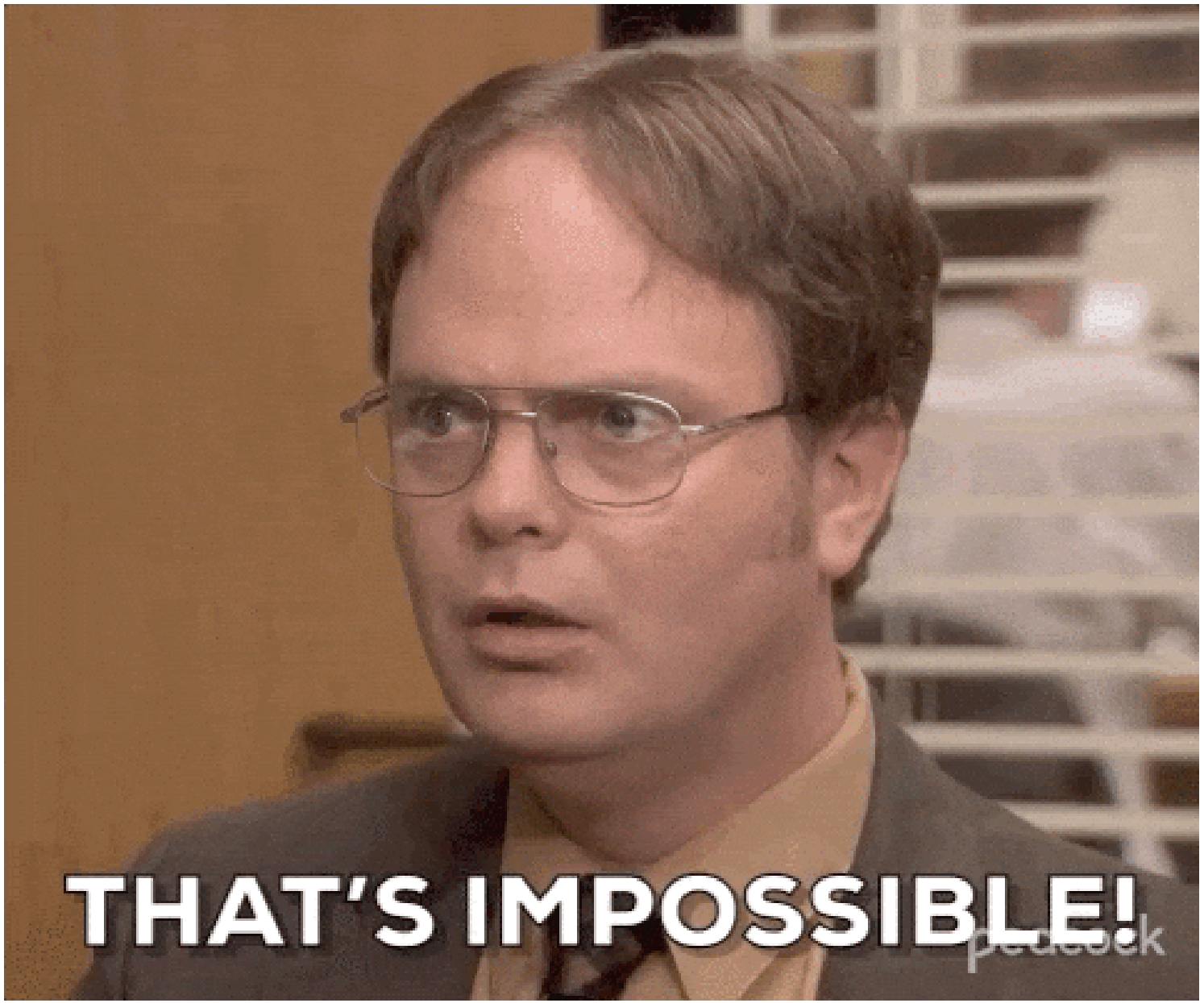
Our goal is to **analyze data** and **do statistics**, not calculate 1 + 1!

Let's analyse CO2 emissions data from Our World in Data



→ **Today's challenge:** reproduce the figure above!

What you're probably thinking



Step 1: import the data

You can download the dataset from here.

- Note: this is a `.csv` file (for *Comma Separated Values*) ⟶ think of it as a dumbed down version of Excel file (`.xls`/`.xlsx`)
- Let's have a look at the first 5 rows of the raw csv file:

```
1  entity,code,year,annual_co2_emissions_per_capita
2  Afghanistan,AFG,1949,0.0019921463
3  Afghanistan,AFG,1950,0.010837197
4  Afghanistan,AFG,1951,0.011625335
5  Afghanistan,AFG,1952,0.011467511
```

- Each row of data corresponds to a country
- The first row contains the column names
- In each row, values are separated by a **comma**

How the heck do I get this into R?

Importing data

To **import** stuff in R we use **read** functions.

The read function dedicated to .csv files is `read.csv()`:

```
1  co2_emissions <- read.csv("data\co2_emissions_historical.csv")
```

```
Error: '\c' is an unrecognized escape in character string (<input>:1:33)
```

Error! Who can guess what the problem is?

"\" must be the other way ⟶ "/"

```
1  co2_emissions <- read.csv("data/co2_emissions_historical.csv")
2  # co2_emissions <- read.csv(file = "data/co2_emissions_historical.csv") does the same thing
```

You can also load the data from the link directly:

```
1  co2_emissions <- read.csv("https://www.dropbox.com/scl/fi/3jefa7xz8fyvqh76e0k9l/co2_emissions_histori
```

Your turn! #2

05:00

1. Load the CO2 emissions dataset in a new object `co2_emissions`. (Hint: objects are created using `=` or `<-`.)
2. Ensure that `co2_emissions` is a `data.frame` by running:

```
class(co2_emissions) # check class of object
```

3. Find out what the variables contained in `co2_emissions` by running:

```
names(co2_emissions) # obtain variable names
```

4. View the contents of `co2_emissions` by clicking on `co2_emissions` in your environment or by running the code below. What do the different variables correspond to?

```
View(co2_emissions)
```

Congratulations, you have imported your first dataset in `R`!

Step 2: inspecting the data

- Before creating our plot, let's get quickly inspect the data.
- The first thing we can do is to print the top rows using the `head()` function:

```
1 head(co2_emissions)
```

```
      entity code year annual_co2_emissions_per_capita
1 Afghanistan  AFG 1949                      0.001992146
2 Afghanistan  AFG 1950                      0.010837197
3 Afghanistan  AFG 1951                      0.011625335
4 Afghanistan  AFG 1952                      0.011467511
5 Afghanistan  AFG 1953                      0.013123366
6 Afghanistan  AFG 1954                      0.012945492
```

- This is a very simple dataset: only 4 variables

Step 2: inspecting the data

- We can also look at the data's structure:

```
1 str(co2_emissions)
```

```
'data.frame':   22976 obs. of  4 variables:
 $ entity                        : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
 $ code                          : chr  "AFG" "AFG" "AFG" "AFG" ...
 $ year                          : int  1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 ...
 $ annual_co2_emissions_per_capita: num  0.00199 0.01084 0.01163 0.01147 0.01312 ...
```

⟶ 22,976 observations and 4 variables

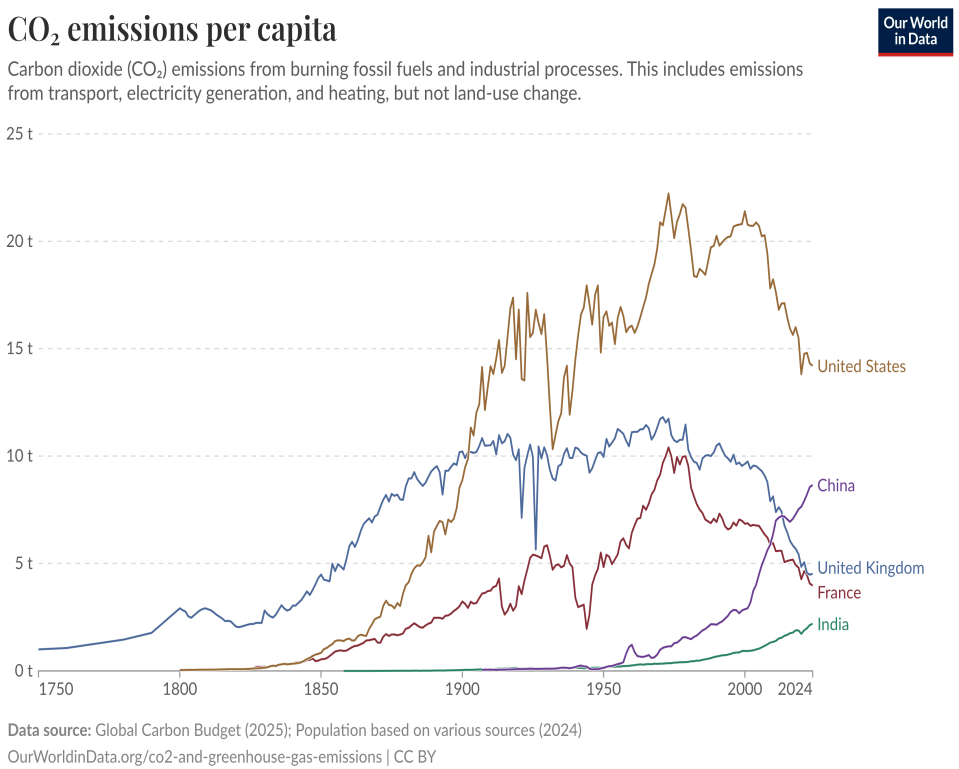
> **Definitions:**
>
> **Variable:** *quantity, quality, or property that you can measure.*
>
> **Observation:** *set of measurements made under similar conditions. An observation will contain several values, each associated with a different variable. We'll sometimes refer to an observation as a data point.*

⟶ Variables are: `entity`, `code`, `year`, `annual_co2_emissions_per_capita`

⟶ Class of variables: number (`int`, `num`), text (`char`)

⟶ First few values seem coherent

Step 3: graph ingredients

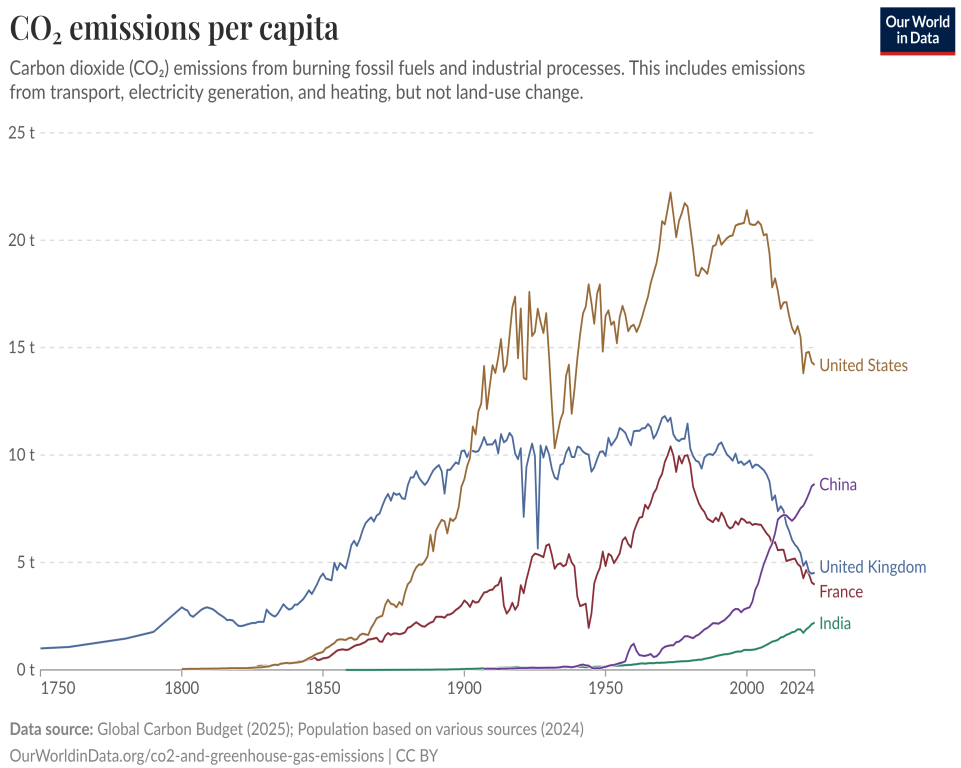


Ingredients:

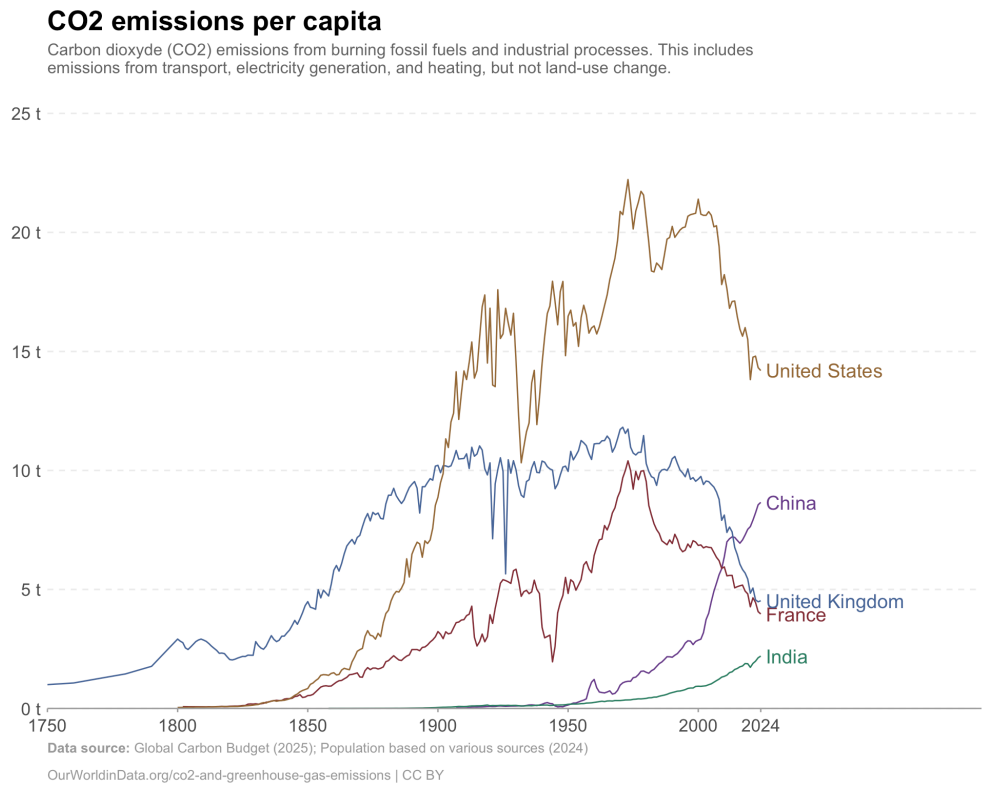
- only 5 countries
- line plot
- plot title and subtitle
- other styling elements:
  - legend in graph,
  - y-axis unit,
  - specific years in x-axis

Achievable in 10 minutes (ok ok maybe 20)

Original version:



My version:



Not perfect, but close enough

Let’s build the graph together

Start with your `data.frame`

```
1  co2_emissions # data.frame will we use for the graph
```

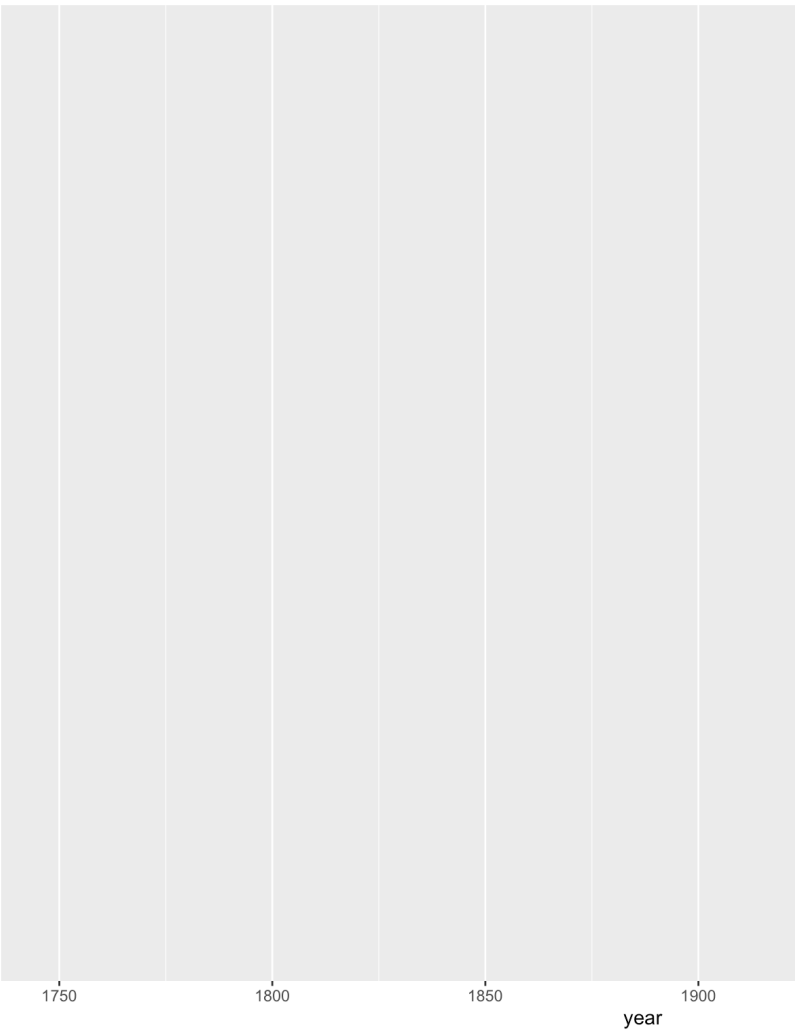
	entity
1	Afghanistan
2	Afghanistan
3	Afghanistan
4	Afghanistan
5	Afghanistan
6	Afghanistan
7	Afghanistan
8	Afghanistan
9	Afghanistan
10	Afghanistan
11	Afghanistan
12	Afghanistan
13	Afghanistan
14	Afghanistan
15	Afghanistan
16	Afghanistan
17	Afghanistan
18	Afghanistan

Initiate `ggplot`

```
1  co2_emissions |>
2    ggplot() # start a graph
```

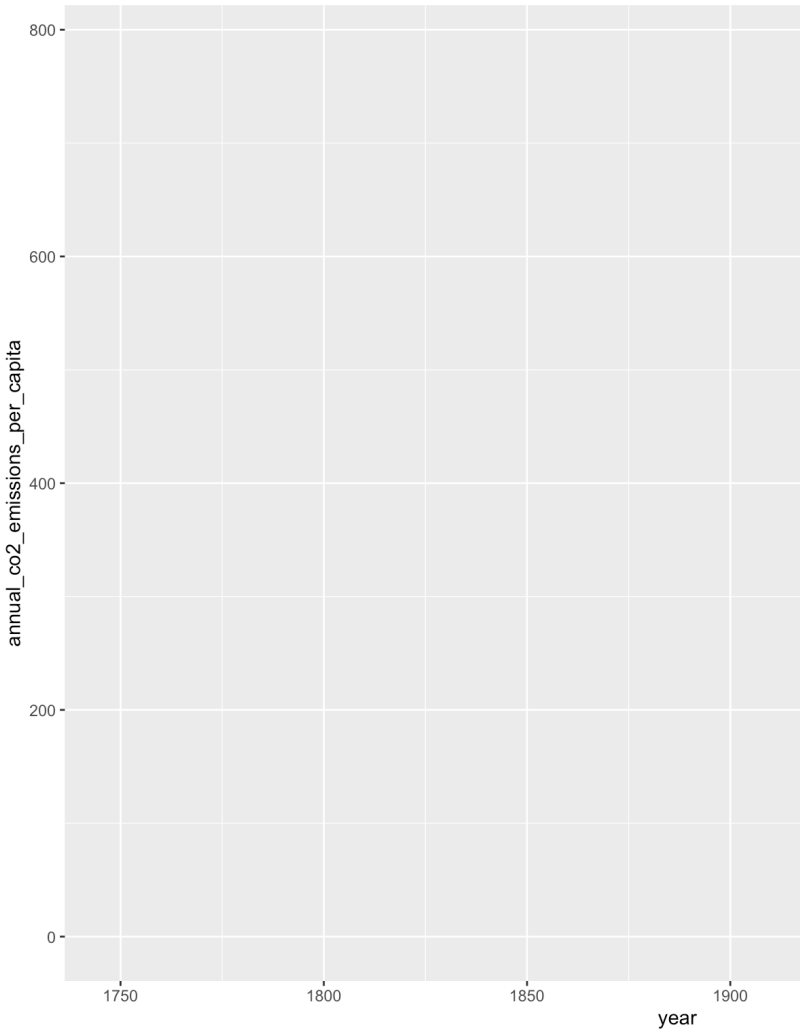
Add `x-axis` variable

```
1  co2_emissions |>
2    ggplot(aes(x = year)) # x-axis = year variable
```



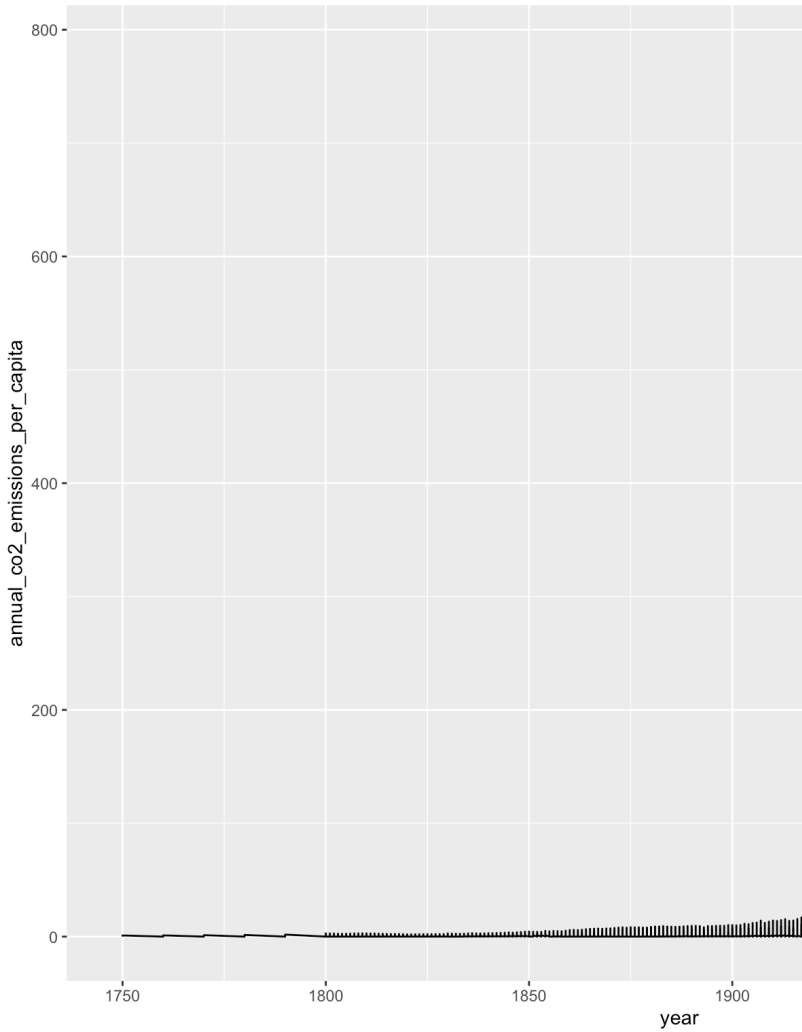
Add `y-axis` variable

```
1  co2_emissions |>
2    ggplot(aes(x = year,
3               y = annual_co2_emissions_per_capita)) # y-axis
```



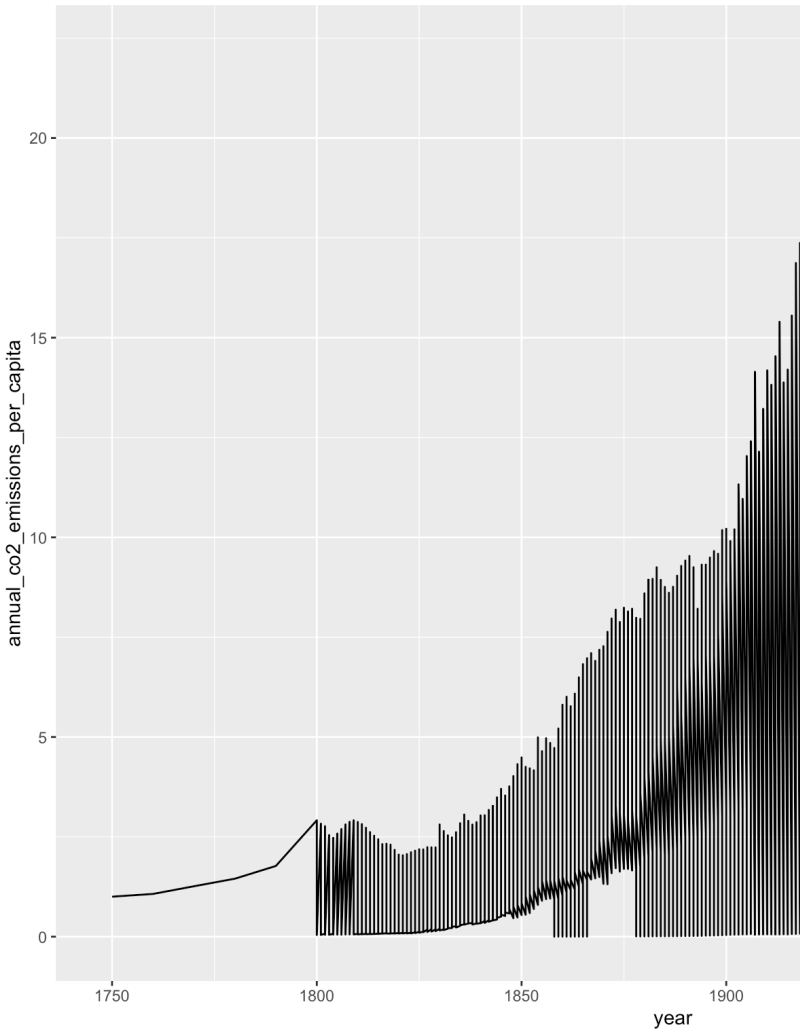
Plot data as `line`

```
1  co2_emissions |>
2    ggplot(aes(x = year,
3               y = annual_co2_emissions_per_capita)) +
4    geom_line() # line plot
```



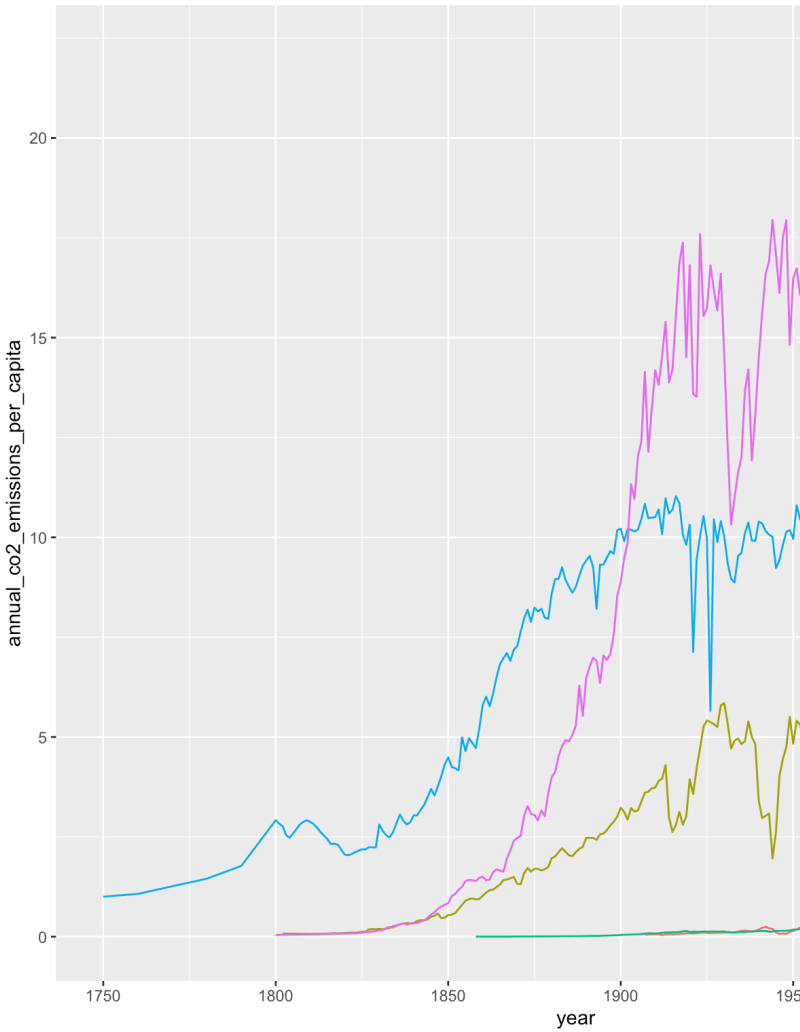
`Filter` to only keep relevant countries

```
1  co2_emissions |>
2    filter(entity %in% c("United States", "China",
3                         "United Kingdom", "France",
4                         "India")) |> # keep only 5 relevant count
5    ggplot(aes(x = year,
6               y = annual_co2_emissions_per_capita)) +
7    geom_line()
```



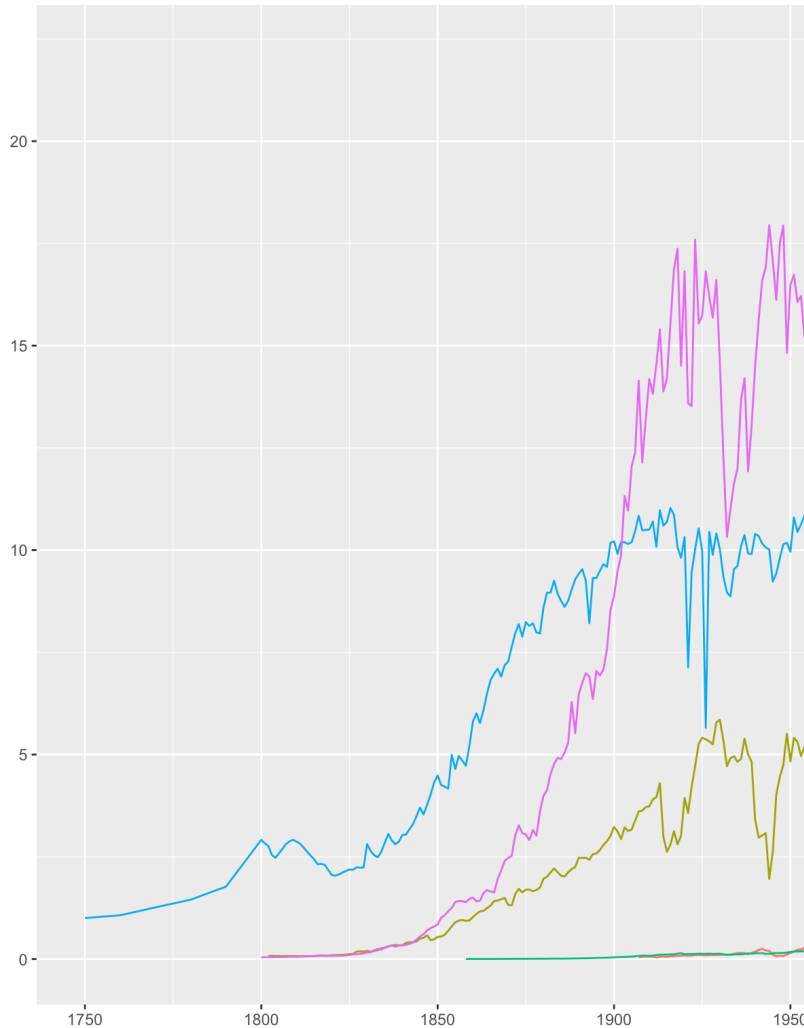
One `color` per country

```
1  co2_emissions |>
2    filter(entity %in% c("United States", "China",
3                         "United Kingdom", "France",
4                         "India")) |>
5    ggplot(aes(x = year,
6               y = annual_co2_emissions_per_capita,
7               color = entity)) + # one line per country
8    geom_line()
```



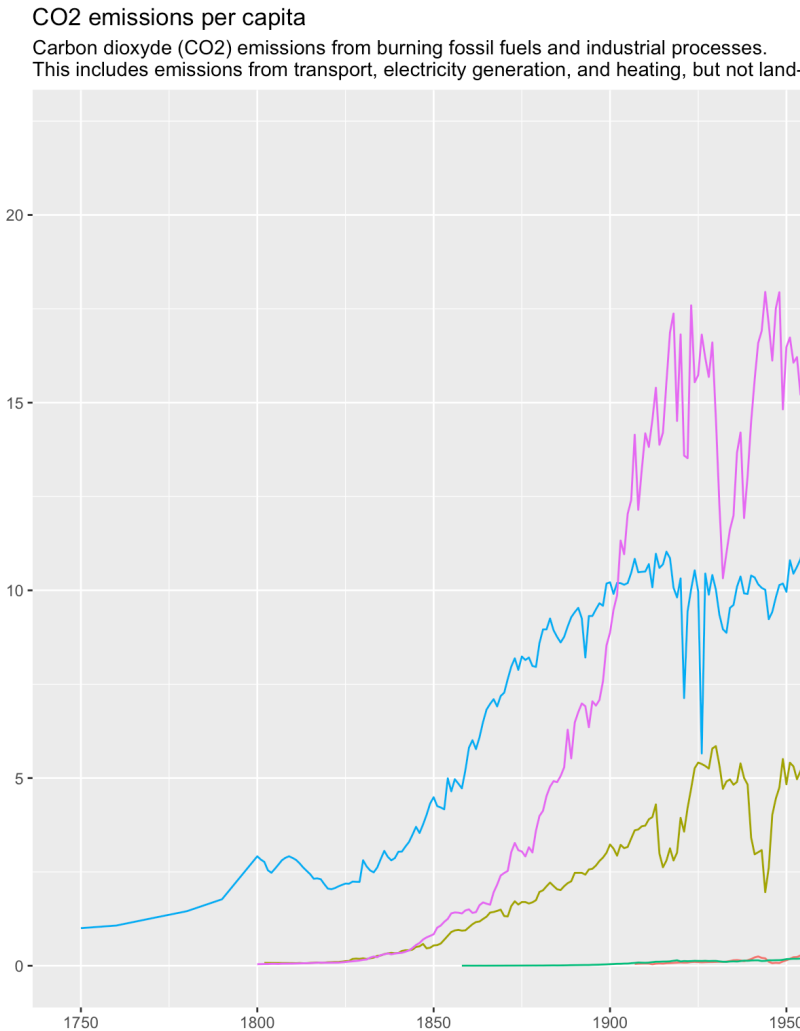
Remove axes `labels`

```
co2_emissions |>
  filter(entity %in% c("United States", "China",
                       "United Kingdom", "France",
                       "India")) |>
  ggplot(aes(x = year,
             y = annual_co2_emissions_per_capita,
             color = entity)) +
  geom_line() +
  labs(x = NULL, # no x-axis
       y = NULL) # no y-axis
```



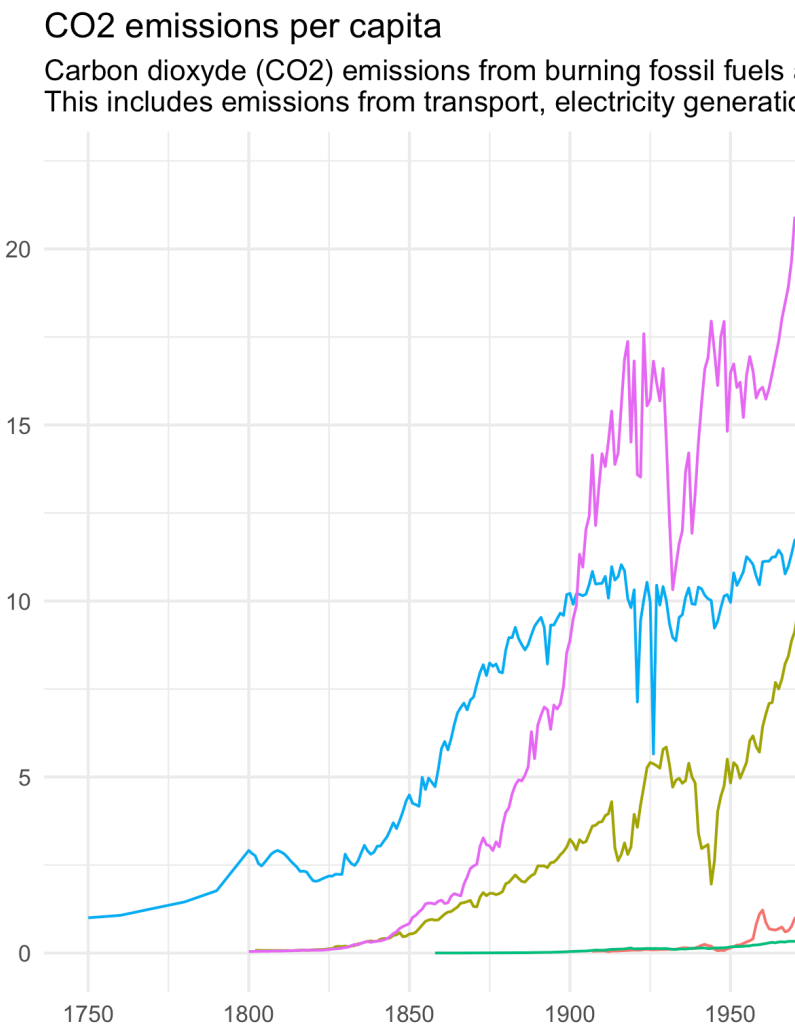
Add plot `title`, `subtitle` and `legend title`

```
co2_emissions |>
  filter(entity %in% c("United States", "China",
                       "United Kingdom", "France",
                       "India")) |>
  ggplot(aes(x = year,
             y = annual_co2_emissions_per_capita,
             color = entity)) +
  geom_line() +
  labs(x = NULL,
       y = NULL,
       color = "Country:", # legend for color
       title = "CO2 emissions per capita", #title
       subtitle = "Carbon dioxyde (CO2) emissions from burning f
```



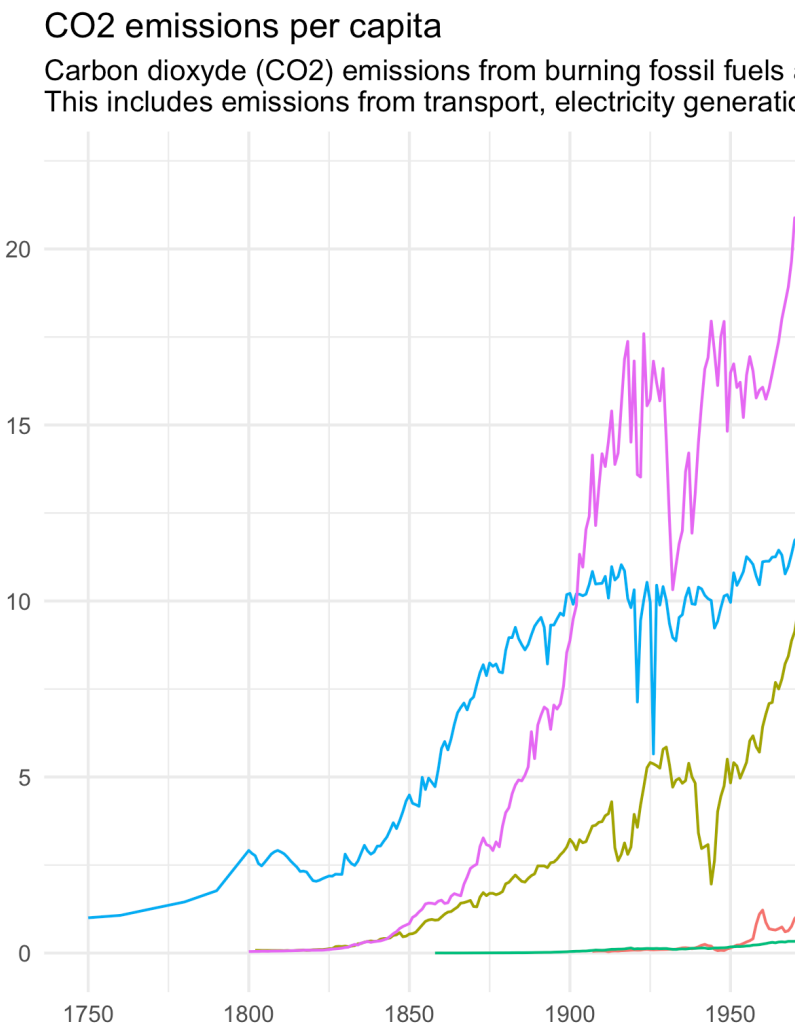
Change ggplot `theme`

```
co2_emissions |>
  filter(entity %in% c("United States", "China",
                       "United Kingdom", "France",
                       "India")) |>
  ggplot(aes(x = year,
             y = annual_co2_emissions_per_capita,
             color = entity)) +
  geom_line() +
  labs(x = NULL,
       y = NULL,
       color = "Country:",
       title = "CO2 emissions per capita",
       subtitle = "Carbon dioxyde (CO2) emissions from burning f
  theme_minimal(base_size = 16) # minimal theme with text size 1
```



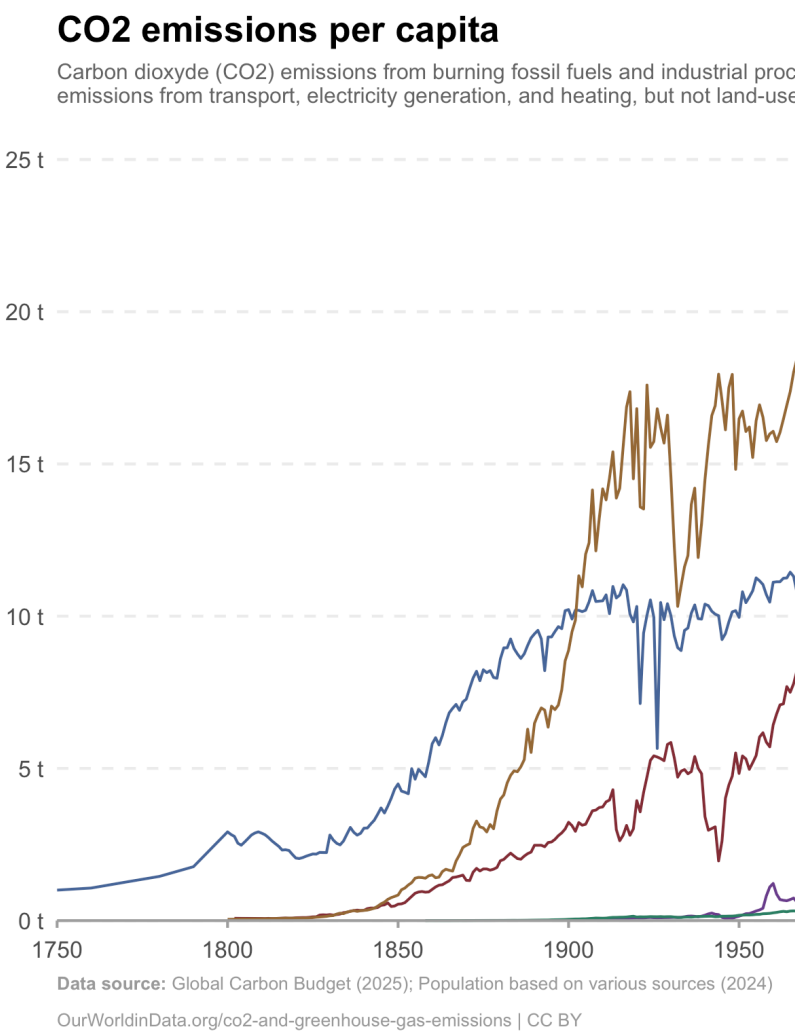
Add 2024 to `x-axis` breaks

```
co2_emissions |>
  filter(entity %in% c("United States", "China",
                       "United Kingdom", "France",
                       "India")) |>
  ggplot(aes(x = year,
             y = annual_co2_emissions_per_capita,
             color = entity)) +
  geom_line() +
  scale_x_continuous(breaks = c(1750, 1800, 1850, 1900, 1950, 20
  labs(x = NULL,
       y = NULL,
       color = "Country:",
       title = "CO2 emissions per capita",
       subtitle = "Carbon dioxyde (CO2) emissions from burning f
  theme_minimal(base_size = 16)
```



Full code

```
 1 library(ggtext)
 2
 3 keep_countries <- c("United States", "China", "United Kingdom",
 4
 5 country_labs <- co2_emissions |>
 6   filter(entity %in% keep_countries) |>
 7   filter(year == 2024)
 8
 9 co2_emissions |>
10   filter(entity %in% keep_countries) |>
11   ggplot(aes(x = year,
12              y = annual_co2_emissions_per_capita,
13              color = entity)) +
14   geom_line() +
15   geom_text(data = country_labs,
16             aes(label = entity, x = 2026), hjust = 0, size = 5)
17   scale_x_continuous(breaks = c(seq(1750, 2000, 50), 2024), expa
18   scale_y_continuous(lim = c(0, 25), labels = scales::label_numb
19   scale_color_manual(values = c("#6B3D8E", "#852F38", "#2B8163",
20   coord_cartesian(clip = "off") +
21   labs(x = NULL,
22        y = NULL,
23        color = "Country:",
24        title = "CO2 emissions per capita"
```



Your turn! #3

10:00

1. Copy the code from slide 51 and run the code
2. Add your country to the graph. If your country is already plotted, add one of your choice. *Hint*:

```
c("United States", "China",
  "United Kingdom", "France",
  "India",
  "MyCountry")
```

3. Add the following code to your graph. What does this do?

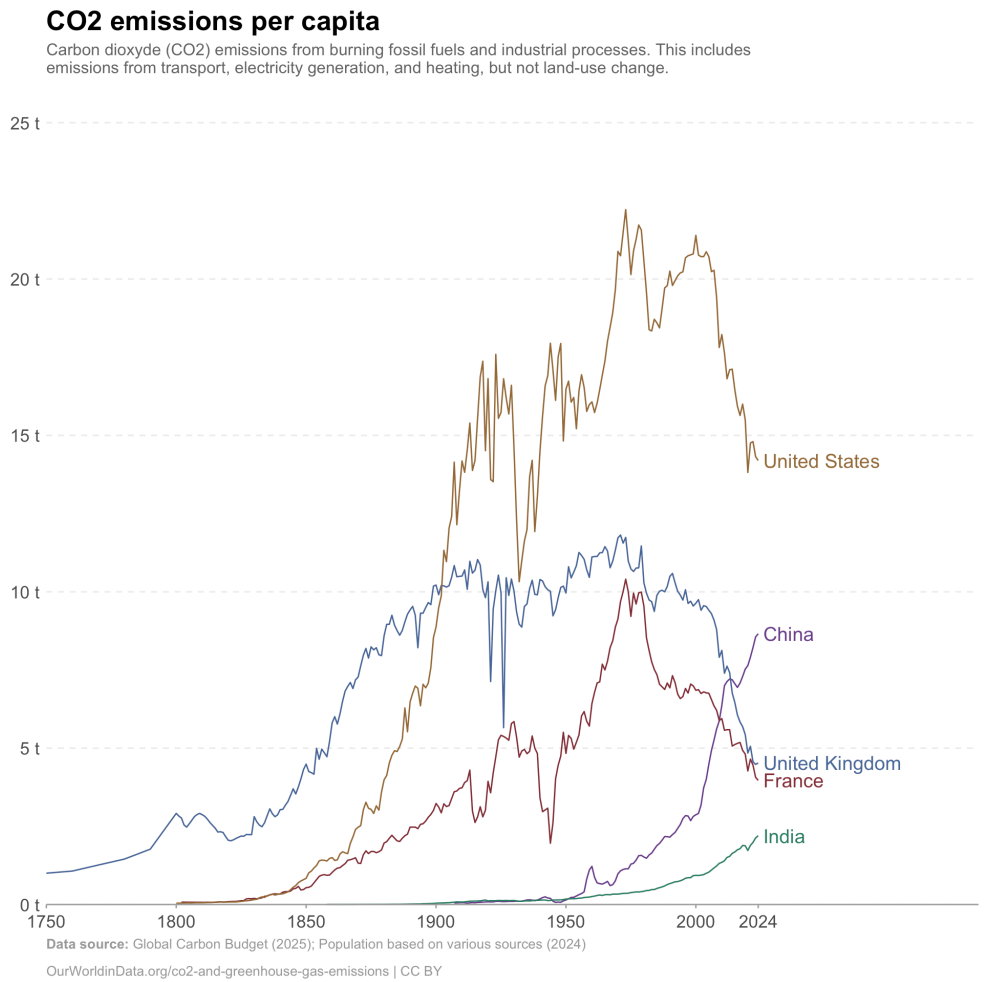
```
  + geom_point()
```

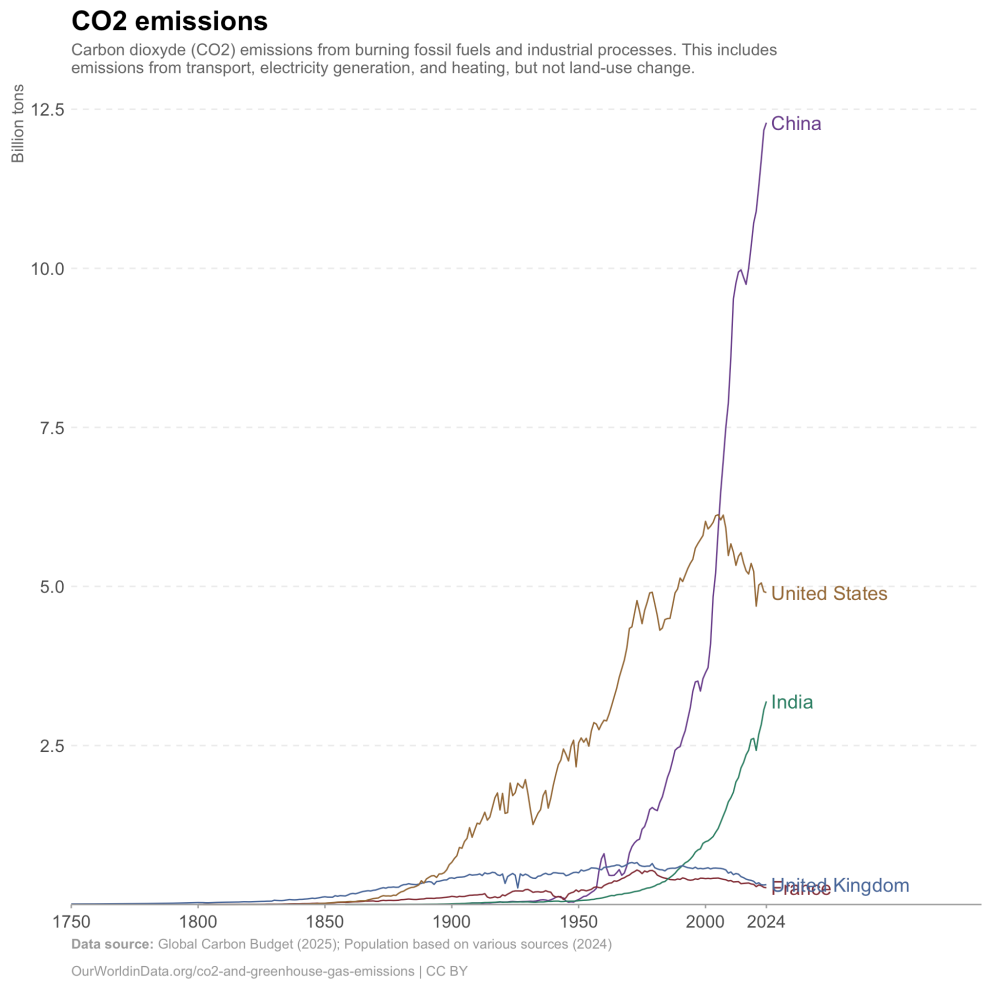
4. Load this dataset. It contains `co2_emissions` with an additional variable. What is it?

```
co2_emissions_new <- read.csv("https://www.dropbox.com/scl/fi/hkjwayxaq46oqataxpe5p/co2_emissions_his
```

5. Adapt the plot code so that the `y-axis` corresponds to annual CO2 emissions (instead of per capita).

Lesson 1 of statistics: units matter!





Today's lecture

1. Why is statistics useful? ✓
2. The data science process ✓
3. R101 ✓
4. Your first `R` plot ✓
5. Anatomy of a `data.frame`
6. Course details

Anatomy of a `data.frame`

Data.frame structure

Recall our data's `str`ucture:

```
1 str(co2_emissions)
```

```
'data.frame':   22976 obs. of  4 variables:
 $ entity                         : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
 $ code                           : chr  "AFG" "AFG" "AFG" "AFG" ...
 $ year                           : int  1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 ...
 $ annual_co2_emissions_per_capita: num  0.00199 0.01084 0.01163 0.01147 0.01312 ...
```

⟶ what do `chr`, `int`, `num` actually mean?

Data types

Numeric (`num`, `int`)

→ numbers:

```
1 class(3)
```

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

```
1 class(-3.14159)
```

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

Character (`chr`)

→ strings of text (`""`):

```
1 class("CY Cergy")
```

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

```
1 class("2025")
```

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

Important: numbers are coerced to character if surrounded by `"`.

Logical (`logi`)

→ `TRUE` or `FALSE`:

```
1 3 > 4
```

```
[1] FALSE
```

```
1 class(T) # T = TRUE
```

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

Logical operators:

- equal to: `==`
- in: `%in%`
- comparisons: `>`, `<`, `>=`, `<=`
- `&` (and), `|` (or), `!` (not)

Very important: `NA` (*not available*) corresponds to a `missing value`

Guess the output

What is the output of this code?

```
1 as.numeric("2026")
```

```
[1] 2026
```

What about?

```
1 as.character(2026-2025)
```

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

Last one:

```
1 as.character(2026>2025)
```

```
[1] "TRUE"
```

Vectors

There's one element we haven't discussed: what are these `$`?

```
1 str(co2_emissions)
```

```
'data.frame':   22976 obs. of  4 variables:
 $ entity                        : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
 $ code                          : chr  "AFG" "AFG" "AFG" "AFG" ...
 $ year                          : int  1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 ...
 $ annual_co2_emissions_per_capita: num  0.00199 0.01084 0.01163 0.01147 0.01312 ...
```

- refers to the fact that `$` allows to **extract a variable** from a dataset

```
1 head(co2_emissions$year, n = 100) # first 100 elements of the year variable
```

```
  [1] 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963
 [16] 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978
 [31] 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993
 [46] 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
 [61] 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
 [76] 2024 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946
 [91] 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956
```

⟶ we call these objects **vectors**

Vectors

- Vectors are basically sequences of values that have the same class
- R won't let you create a vector containing elements of different classes
- We can create vectors using concatenate function:

```
1 c(1,2,3,4,5)
```

```
[1] 1 2 3 4 5
```

What will this code yield?

```
1 c("Hello World", 2026, FALSE)
```

```
[1] "Hello World" "2026"        "FALSE"
```

⟶ coercion to unique types!

- Operations apply to all vector elements:

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

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

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

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

Subsetting

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

$$\text{data}[\text{row}, \ \text{columns}]$$

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

Example:

```
1  co2_emissions[1, c("entity", "year")]
```

```
      entity year
1 Afghanistan 1949
```

Brackets also work for vectors:

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

```
[1] 3 1
```

Your turn! #4

10:00

1. Keep only observations for the `United States` in `1995` from `co2_emissions`. *Hint*: remember that to access a variable you need to use `$`, i.e.:

```
co2_emissions[co2_emissions$entity == "COUNTRY"]
```

2. Create a new object containing `co2_emissions`' rows 10 to 20. You can create sequences using `a:b`.

```
1:10
```

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

3. Load this dataset in an object called `co2_emissions_gdp`. It contains `co2_emissions` as well as a column for GDP (in 2015 dollars).

4. Create a `co2_emissions_per_gdp` variable equal to annual CO2 emissions per dollar of GDP by running the following code:

```
co2_emissions_gdp$co2_emissions_per_gdp <- co2_emissions_gdp$annual_co2_emissions / co2_emissions_gdp
```

Congratulations, you've created your first variable!

Today's lecture

1. Why is statistics useful? ✓
2. The data science process ✓
3. R101 ✓
4. Your first `R` plot ✓
5. Anatomy of a `data.frame` ✓
6. Course details

Course details

This course

- Teach you the basics of **statistics**.
- Equip you with a framework to think more deeply about **inference**: drawing conclusions about a *`population`* from a *`sample`*.
- Introduce you to the `R` software environment.
- ⚠️ This is not a course about R.

Grading

1. Short **Moodle quizzes** ⟶ 4 x 5% = 20%

2. Statistics **data project** ⟶ 40%

3. **Final exam** ⟶ 40%

Details about data project:

- Produce short (~10 pages max, include code) statistical analysis in groups of 2
- Analyze data of your choice to understand a specific question
- Follow data science process: import, tidy, transform, visualize, model
- Demonstrate that you understand concepts learned in class
- Have fun!

Syllabus

Lecture 1. **Introduction to R** (today)

Moodle quiz 1 (after lecture 2)

Lectures 2 and 3. **Summarizing and visualizing data**

Moodle quiz 2

Lecture 4. **Sampling and inferential statistics**

Lecture 5. **Confidence intervals**

Moodle quiz 3

Lecture 6. **Hypothesis testing**

Lecture 7. **Statistical power**

Moodle quiz 4

Lecture 8 **Correlation and causality**

Statistics data project

Lecture 9 **Linear regression**

Final exam

Course materials and useful resources

Course materials:

- on github and on Moodle

Useful resources:

- R for Data Sciences
- ModernDive
- Awesome R Learning Resources
- Kyle Butts Econ 3818

Course policies

Be nice. Be honest. Don't cheat.

(BTW you should apply these principles outside of the classroom as well `#lifeadvice` 😉)

Office hours: Tuesdays and Thursdays 4-5pm ⟶ book slot here

Slack

We will **exclusively** use Slack for our interactions.

Please ***DO NOT*** contact me by email (unless there's a legitimate reason to).

I will be checking Slack sparingly so please help one another!

Class conduct and expectations

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.

Some words about AI

Generative AI is useful but ethically questionnable

- I use it in my research and for my teaching preparation
- Powerful tool that you should learn to use **efficiently and carefully**
- AI does NOT replace coding from scratch, thinking, and hard work!
- **Ethical questions:** energy use, intellectual property

Generative AI is often wrong and has a terrible mindset

- Always keep in mind that AI hallucinates very easily ⟶ **double-check**!
- It (still) lacks the `ability to doubt`, navigate nuances, and be rigorous

What I ask from you

- For the statistics data project, tell me broadly how you used AI
- Be honest, you will not be penalized

Last but not least: 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.

CYU offers free and confidential psychological support services with professionals.

See you next week!