- **M1 MIDS & MFA**
- **Université Paris Cité**
- Année 2023-2024
- Course Homepage
- Moodle

We will use the following packages. If needed, we install them.

```
to_be_loaded <- c("tidyverse",
                  "patchwork",
                  "glue",
                  "ggforce",
                  "plotly",
                  "ggthemes",
                  "gapminder",
                  "ggrepel")

for (pck in to_be_loaded) {
  if (!require(pck, character.only = T)) {
    install.packages(pck, repos="http://cran.rstudio.com/")
    stopifnot(require(pck, character.only = T))
  }
}
```

Grammar of Graphics

We will use the *Grammar of Graphics* approach to visualization

The expression *Grammar of Graphics* was coined by Leiland Wilkinson to describe a principled approach to visualization in Data Analysis (EDA)

A plot is organized around data (a table with rows (observations) and columns (variables))

A *plot* is a *graphical object* that can be built *layer* by *layer*

Building a graphical object consists in *chaining* elementary operations

The acclaimed TED presentation by Hans Rosling illustrates the Grammar of Graphics approach

Visit https://www.youtube.com/embed/jbkSRLYSojo

Do It Yourself with `R`

We will reproduce the animated demonstration using

- `ggplot2`: an implementation of *grammar of graphics* in 'R
- `plotly`: a bridge between `R` and the javascript library `D3.js`
- Using `plotly`, opting for `html` ouput, brings the possibility of interactivity and animation

Install and load packages

```
require("gapminder")
```

Insist on the difference between *installing* and *loading* a package

- How do we get the list of installed packages?
- How do we get the list of loaded packages?
- Which objects are made available by a package?

Have a look at `gapminder` dataset

- A table has a *schema*: a list of named *columns*, each with a given type
- A table has a *content*: *rows*. Each row is a collection of items, corresponding to the columns
- `glimpse()` allows to see the schema and the first rows
- `head()` allows to see the first rows

Get a feeling of the dataset

Pick two random rows for each continent using `slice_sample()`

What makes a table *tidy*?

Have a look at Data tidying in R for Data Science (2nd ed.)

Is the `gapminder` table redundant?

Gapminder tibble (extract)

Extract/filter a subset of rows using `dplyr::filter(...)`

Filtering (selection σ from database theory) : Picking one year of data

There is simple way to filter rows satisfying some condition. It consists in mimicking indexation in a matrix, leaving the colum index empty, replacing the row index by a condition statement (a logical expression) also called a mask.

```
gapminder_2002 <- gapminder[gapminder$year==2002, ]
```

Have a look at `gapminder$year==2002`. What is the type/class of this expression?

This is possible in base `R` and very often convenient.

Nevertheless, this way of performing row filtering does not emphasize the connection between the dataframe and the condition. Any logical vector with the right length could be used as a mask. Moreover, this way of performing filtering is not very functional.

> In the parlance of Relational Algebra, `filter` performs a *selection* of rows. Relational expression
>
> $$\sigma_{\text{condition}}(\text{Table})$$
>
> translates to
>
> ```
> filter(Table, condition)
> ```
>
> where condition is a boolean expression that can be evaluated on each row of Table. In `SQL`, the relational expression would translate into
>
> ```
> SELECT *
> FROM Table
> WHERE condition
> ```
>
> Check Package `dplyr` docs
> The `posit` cheatsheet on `dplyr` is an unvaluable resource for table manipulation.

Use `dplyr::filter()` to perform row filtering

Static plotting

First attempt

- Define a plot with respect to `gapminder_2002`

> You should define a `ggplot` object with data layer `gapminder_2022` and call this object `p` for further reuse.

- Map variables `gdpPercap` and `lifeExp` to axes `x` and `y`

> Use `ggplot` object `p` and add a global aesthetic mapping `gdpPercap` and `lifeExp` to axes `x` and `y` (using `+` from `ggplot2`) .

- For each row, draw a point at coordinates defined by the mapping

> You need to add a `geom_` layer to your `ggplot` object, in this case `geom_point()` will do.

We are building a graphical object (a `ggplot` object) around a data frame (`gapminder`)

We supply *aesthetic mappings* (`aes()`) that can be either global or bound to some *geometries* (`geom_point()`)or *statistics*

The global aesthetic mapping defines which columns are

- mapped to which axes,
- possibly mapped to colours, linetypes, shapes, …

Geometries and Statistics describe the building blocks of graphics

What's missing here?

when comparing to the Gapminder demonstration, we can spot that

- colors are missing

- bubble sizes are all the same. They should reflect the population size of the country
- titles and legends are missing. This means the graphic object is useless.

We will add layers to the graphical object to complete the plot

Second attempt: display more information

- Map `continent` to color (use `aes()`)
- Map `pop` to bubble size (use `aes()`)
- Make point transparent by tuning `alpha` (inside `geom_point()` avoid *overplotting*)

Scaling

In order to pay tribute to Hans Rosling, we need to take care of two *scaling* issues:

- the gdp per capita axis should be *logarithmic* `scale_x_log10()`
- the *area* of the point should be proportional to the population `scale_size_area()`

Motivate the proposed scalings.
- Why is it important to use logarithmic scaling for gdp per capita?
- When is it important to use logarithmic scaling on some axis (in other contexts)?
- Why is it important to specify `scale_size_area()` ?

In perspective

- Add a plot title
- Make axes titles
  - explicit
  - readable
- Use `labs(...)`

What should be the respective purposes of Title, Subtitle, Caption, … ?

Theming using `ggthemes`

- Theming

```
require("ggthemes")
```

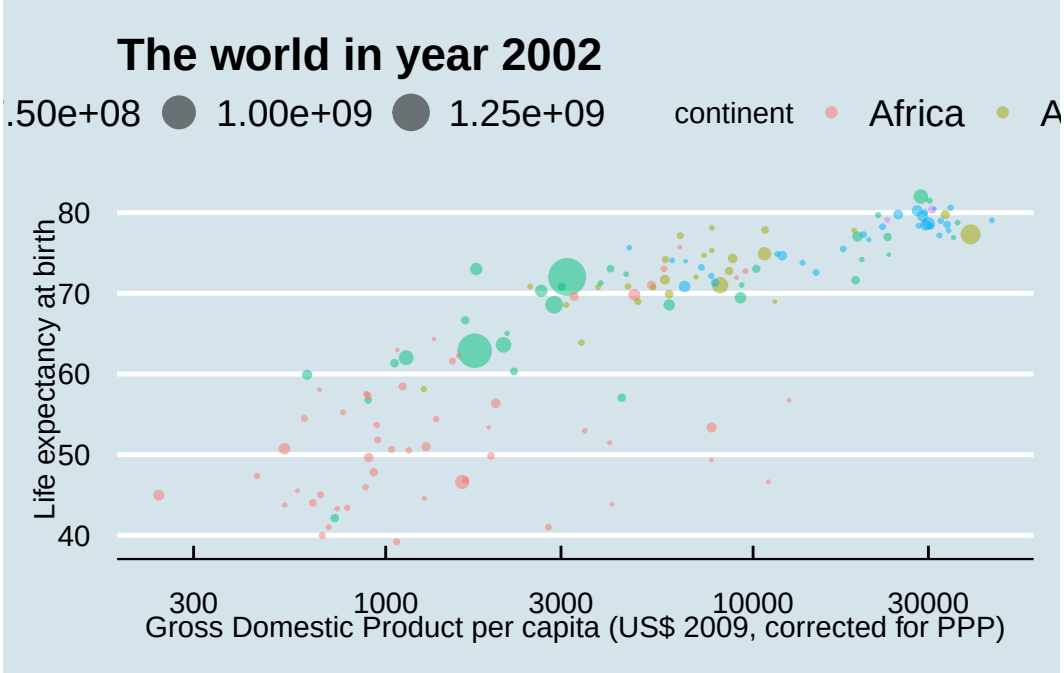
Look at the online help on `pacman::p_load()`, how does `pacman::p_load()` relate to `require()` and `library()`?

A theme defines the *look and feel* of plots

Within a single document, we should use only one theme

See Getting the theme for a gallery of available themes

```
p +
  theme_economist()
```



Tuning scales

Use `scale_color_manual(...)` to hand-tune the color aesthetic mapping.

```
neat_color_scale <-
      c("Africa" = "#01d4e5",
        "Americas" = "#7dea01" ,
        "Asia" = "#fc5173",
        "Europe" = "#fde803",
        "Oceania" = "#536227")
```

Choosing a color scale is a difficult task

`viridis` is often a good pick.

Zooming on a continent

```
zoom_continent <- 'Europe'  # choose another continent at your convenience
```

Use `facet_zoom()` from package `ggforce`

Adding labels

Facetting

So far we have only presented one year of data (2002)

Rosling used an *animation* to display the flow of time

If we have to deliver a printable report, we cannot rely on animation, but we can rely on *facetting*

Facets are collections of small plots constructed in the same way on subsets of the data

We add a layer to the graphical object using `facet_wrap()`

As all rows in `gapminder_2002` are all related to `year` 2002, we need to rebuild the graphical object along the same lines (using the same *graphical pipeline*) but starting from the whole `gapminder` dataset.

Should we do this using *cut and paste*?

No

Don't Repeat Yoursel (DRY)

> Abide to the DRY principle using operator `%+%`: the `ggplot2` object `p` can be fed with another dataframe and all you need is proper facetting.

Animate for free with `plotly`

Use `plotly::ggplotly()`