General Introduction to R

Session 2 - Graphic representations with GGPLOT

Jean-Baptiste Guiffard and Florence Lecuit

October 18, 2024

Graphic representations with R

The main packages for the realization of graphics in R

It is possible to create a multitude of graphs on R with many options, from simple to complex. For this, specialized packages exist:

1. The **graphics** package: already existing by default in R
2. The **lattice** package : adds functionalities to graphics.
3. The **ggplot2** package : the one we will see the most, because it is very complete and offers a modern approach to create very good quality graphics.

Load data

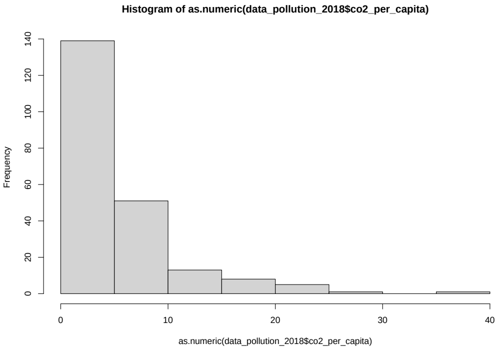
Upload the data

```
data_pollution <- read.csv2('DATA/co2_clean.csv', sep=";")
Metadata_Country <- read.csv2('DATA/Metadata_Country.csv', sep=",") %>%
  rename("Country_code" = "Country.Code")
```

Some basic graphic functions: hist

A **histogram** represents the distribution of numerical data

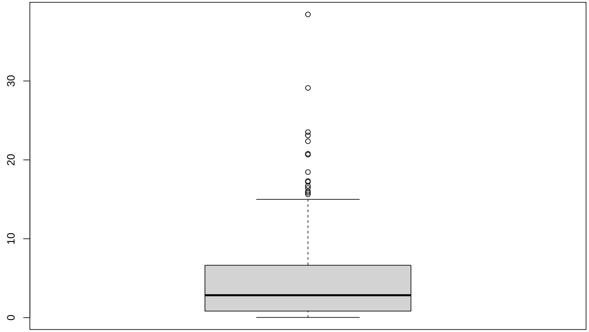
```
data_pollution_2018 <- data_pollution %>%
  filter(year==2018)
hist(as.numeric(data_pollution_2018$co2_per_capita))
```



Some basic graphic functions: boxplot (I)

A **boxplot** represents graphically the locality, spread and skewness groups of numerical data through their quartiles

```
boxplot(data_pollution_2018$co2_per_capita)
```

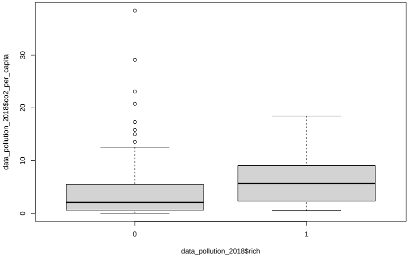


Some basic graphic functions: boxplot (II)

```r
data_pollution_2018$rich <- ifelse(data_pollution_2018$gdp > 500000000000,1,0)
table(data_pollution_2018$rich)
```

```
##
##   0   1
## 128  37
```

```r
boxplot(data_pollution_2018$co2_per_capita ~ data_pollution_2018$rich)
```



The function plot()

The plot function is commonly used to produce graphs, it is a generic function that adapts automatically according to the arguments introduced in the function.

Two possible syntaxes:

- classical syntax:

```
plot(x = varX, y = varY)
```

with x, the variable to put on the x-axis and y, the variable to put on the y-axis

- formula-based syntax:

```
plot(varY ~ varX)
```

Example 1: represent a scatter plot

Create variables to plot:

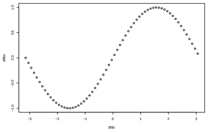
```
x <- seq(-pi, pi, 0.1)
y <- sin(x)
```

```
plot(x, y)    # format classique
```

```
plot(y ~ x)   # formule ---> plot.formula
```

Plot variables from data frame:

```
dt <- data.frame(z = x, w = y)
plot(dt$w ~ dt$z)  # or: plot(w ~ z, data = dt)
```



Example 2: represent a scatter plot (I)

A **scatter plot** is used to present the measurement of two or more related variables → Useful when the values of the variables on the y axis depend on the values of the variable on the x axis.

```
join_pollution_wb_data <- data_pollution %>%
  dplyr::inner_join(Metadata_Country, by = c("iso_code" = "Country_code"))

join_pollution_wb_data <- join_pollution_wb_data %>%
  filter(country != "") %>%
  filter(IncomeGroup !="")
```

Exercise:

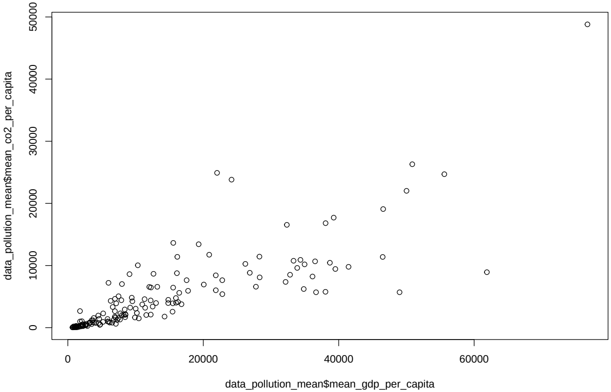
- Create two variables GDP per capita and CO2 per capita in kg ;
- Create a new database that, for the period [1990;2020], gives the average of these two variables by country;
- Delete the columns with missing values.

Example 2: represent a scatter plot (II)

```
join_pollution_wb_data <- join_pollution_wb_data %>%
  mutate(gdp_per_capita = gdp/population,
         co2_per_capita_en_kg = co2/population*1000000000)


data_pollution_mean <- join_pollution_wb_data %>%
  filter(year >= 1990 & year <= 2020) %>%
  group_by(country,IncomeGroup) %>%
  summarise(mean_gdp_per_capita = mean(gdp_per_capita, na.rm=T),
            mean_co2_per_capita = mean(co2_per_capita_en_kg, na.rm=T))


data_pollution_mean <- data_pollution_mean %>%
  na.omit()
```

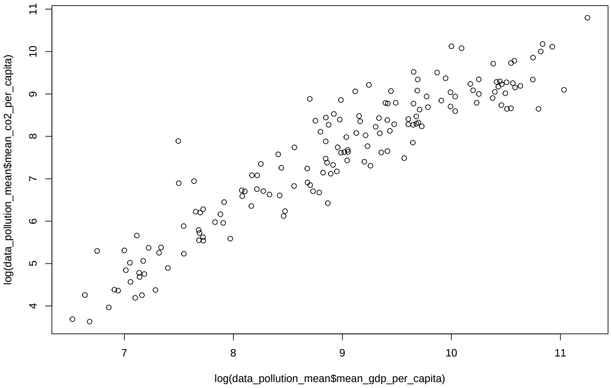
Example 2: represent a scatter plot (III)

```
plot(data_pollution_mean$mean_gdp_per_capita,
     data_pollution_mean$mean_co2_per_capita)
```



Example 2: represent a scatter plot (VI)

```
plot(log(data_pollution_mean$mean_gdp_per_capita), log(data_pollution_mean$mean_co
```

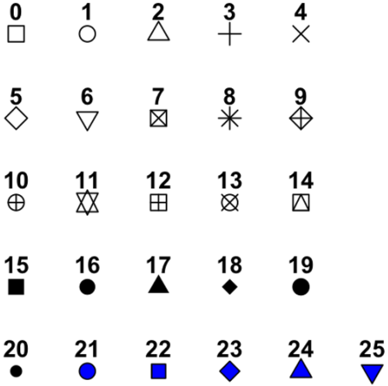


A function with many parameters. . .

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

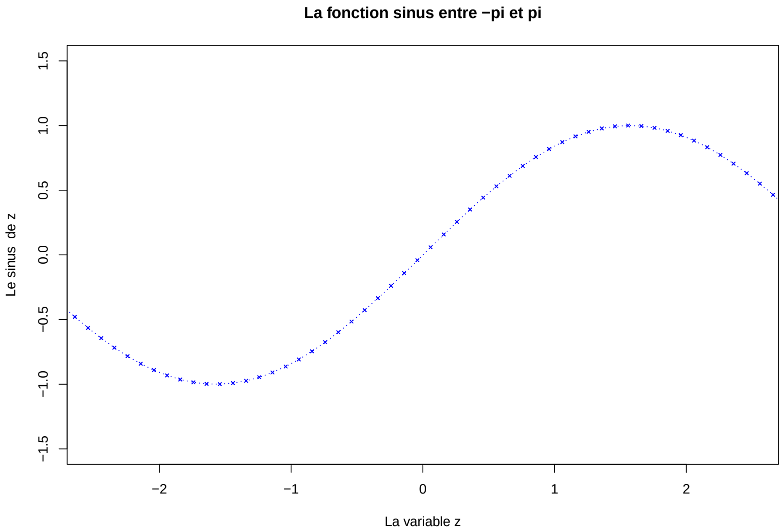
```
plot(w ~ z, data = dt,
  type = "o", # type de tracé: points ("p"), lignes ("l"), les deux ("b" ou "o"),
  col = "blue", # couleur, tapez `colours()` pour la liste complète
  pch = 4, # type de symboles, un chiffre entre 0 et 25, tapez `?points`
  cex = 0.5, # taille des symboles
  lty = 3, # type de lignes, un chiffre entre 1 et 6
  lwd = 1.2, # taille de lignes
  xlim = c(-2.5, 2.5), # limites de l'axe des x
  ylim = c(-1.5, 1.5), # limites de l'axe des y)
  xlab = "La variable z", # titre pour l'axe des x
  ylab = "Le sinus  de z", # titre pour l'axe des y
  main = "La fonction sinus entre -pi et pi" # titre général pour le graphique
)
```

Customizing your graph... points



Point shapes (pch symbols)

Customizing your graph: example



GGPLOT

Introducing GGPLOT

GGPLOT2

- A package used to make graphics;
- The way of coding respects a grammar which is specific to this package... Inspired by the book “The Grammar of Graphics” (Leland Wilkinson), where the name comes from.
- Distinguishes itself from other graphical production tools under R. Allows to produce more elaborated and better finalized graphs than the graphs produced with classical R functions.
- Allows for example to obtain graphical representations by subgroups of individuals with very few lines of code.

In writing the command for creating a graphic, we will consider an assembly of layers $\rightarrow$ split the instructions.

```r
library(ggplot2)
```

```
## Warning: le package 'ggplot2' a été compilé avec la version R 4.3.3
```

The grammar of ggplot

Construction of a ggplot from a set of independent elements

- **Data**: the data set containing the variables used;
- **Aesthetucs**: variables to represent, (here you can add colors or sizes if associated to variables);
- **Geometrics**: type of graphical representation desired;
- **Statistics**: possible transformations of the data for the desired representation;
- **Scales**: control the link between the data and the aesthetics (change of colors, management of the axes...)

Steps in the realization of a graph for a continuous variable

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

The data set

```
etape1 <- ggplot(data_pollution_mean)
```

The variable to be represented

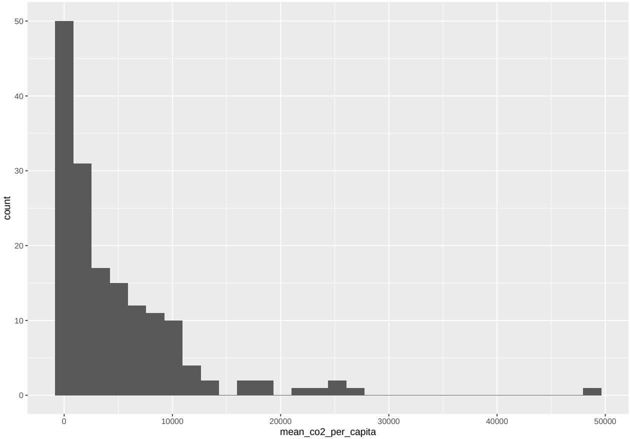
```
etape2 <- etape1 + aes(x=mean_co2_per_capita)
```

The desired representation type

```
etape3 <- etape2 + geom_histogram()
```

Histogram with ggplot

```r
plot(etape3)
```



Steps in the realization of a graph for a discrete variable

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

The data set

```
etape1 <- ggplot(data_pollution_mean)
```

The variable to be represented

```
etape2 <- etape1 + aes(x=IncomeGroup)
```

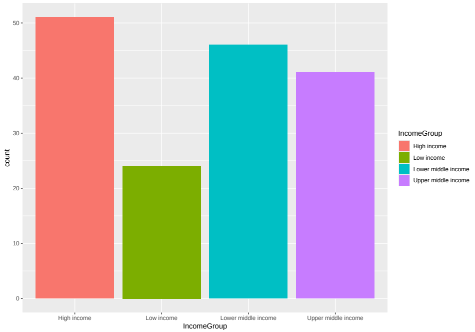
The desired representation type

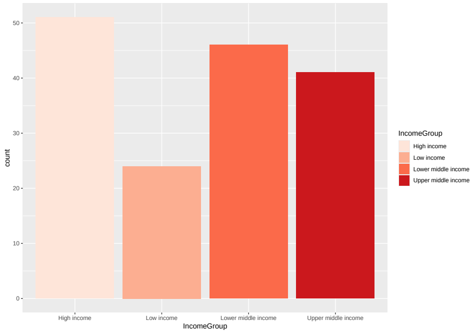
```
etape3 <- etape2 + geom_bar(aes(fill=IncomeGroup))
```

Change of colors

```
etape4 <- etape3 + scale_fill_brewer(palette = "Reds")
```

Bar chart with ggplot

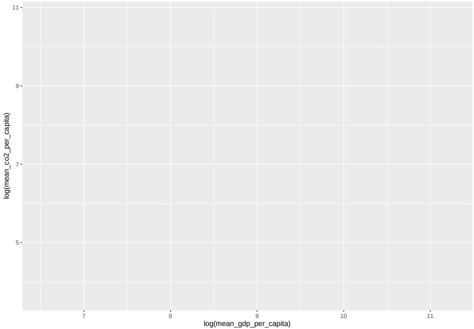




First steps: the data set and the variables to represent

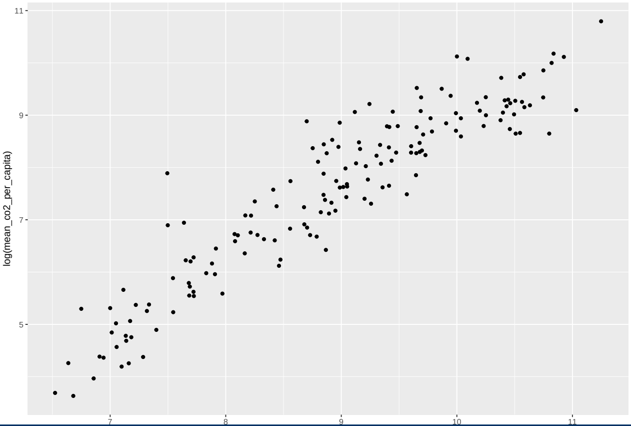
Example with two continuous variables...

```
graph1 <- ggplot(data_pollution_mean,
                 aes(x=log(mean_gdp_per_capita),
                     y=log(mean_co2_per_capita)))
plot(graph1)
```



How will the information be represented?

```
graph1_b <- graph1 + geom_point()
plot(graph1_b)
```



Step 3: Add graphic parameters (I)

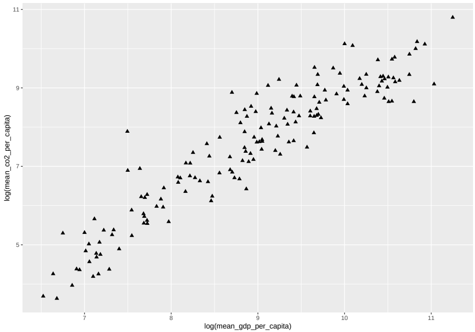
Shape of the points

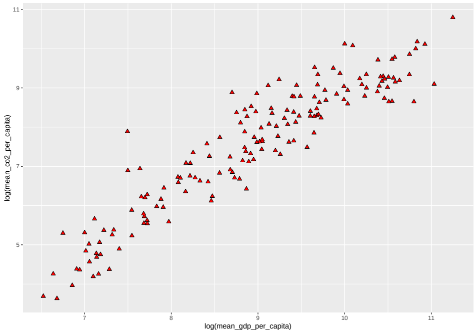
```
graph1_1 <- graph1 +
  geom_point(size=2, shape=17)
```

Colors

```
graph1_2 <- graph1 +
  geom_point(size=2, shape=24, colour='black', fill="red")
```

Step 3: Add graphic parameters (I)





Step 3: Add graphic parameters (II)

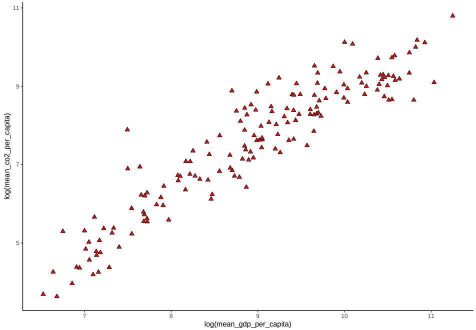
Background

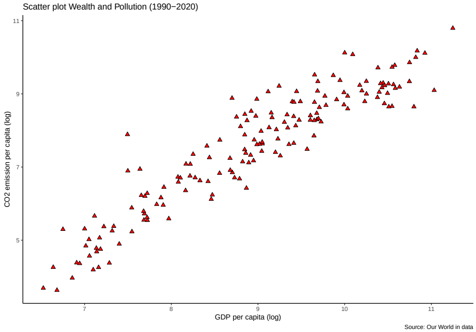
```
graph1_3 <- graph1  +
  geom_point(size=2, shape=24, colour='black', fill="red")+
  theme_classic()
```

Caption, title...

```
graph1_4 <- graph1  +
  geom_point(size=2, shape=24, colour='black', fill="red") +
  ggtitle('Scatter plot Wealth and Pollution (1990-2020)')+
  theme_classic()+
  labs(caption ="Source: Our World in data",
       x='GDP per capita (log)',
       y='CO2 emission per capita (log)')
```

Step 3: Add graphic parameters (II)





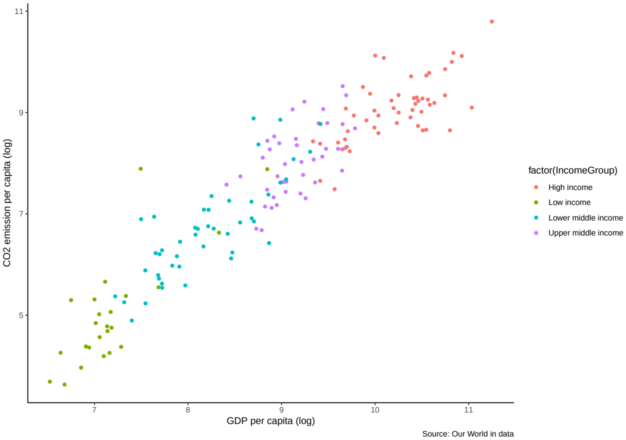
Step 3: Add graphic parameters (III)

Scatter plot - color according to development level groups

```
graph1_5 <- ggplot(data_pollution_mean,
                   aes(x=log(mean_gdp_per_capita),
                       y=log(mean_co2_per_capita),
                       colour = factor(IncomeGroup)))+
  geom_point()+
  theme_classic() +
  labs(caption ="Source: Our World in data",
       x='GDP per capita (log)',
       y='CO2 emission per capita (log)')
```

Step 3: Add graphic parameters (III)

```
plot(graph1_5)
```



Fourth step: Models (I)

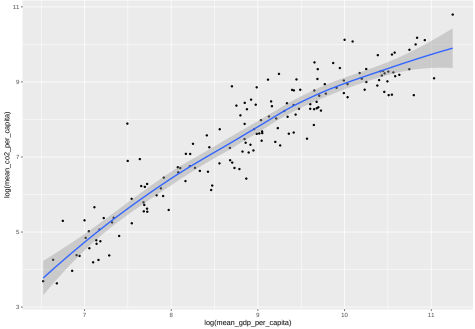
We can work on the link between two quantitative variables (regression models)

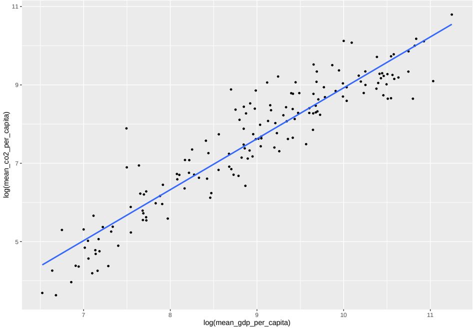
- Regression with the **geom_smooth** function (by default loess regression)

```
graph1_6 <- graph1 +
  geom_point(size=1) +
  geom_smooth()

graph1_7 <- graph1 +
  geom_point(size=1) +
  geom_smooth(method=lm, se=FALSE)
```

Fourth step: Models (I)

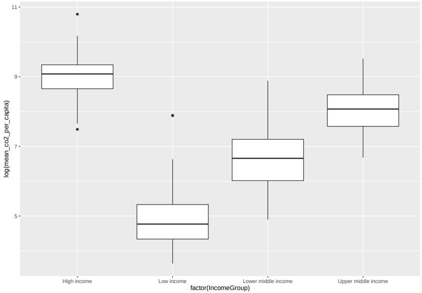
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```





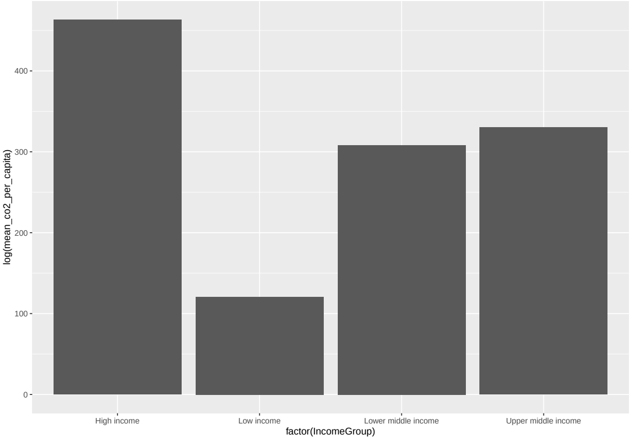
Discrete and continuous variable (I)

```
graph2 <- ggplot(data_pollution_mean,
                 aes(x=factor(IncomeGroup),
                     y=log(mean_co2_per_capita)))
graph2_1 <- graph2 +
  geom_boxplot()
plot(graph2_1)
```



Discrete and continuous variable (I)

```
graph2_2 <- graph2 + geom_bar(stat = "identity")

plot(graph2_2)
```



What graphs from this dataset?

Carbon footprint trajectory by country group

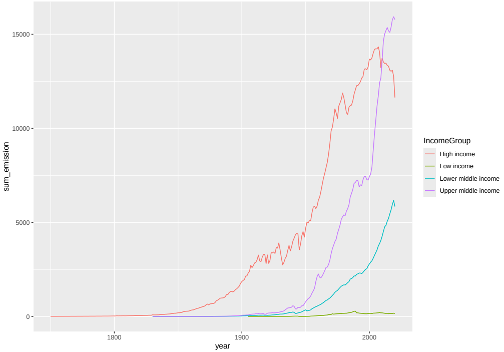
```
data_pollution_by_group <- join_pollution_wb_data %>%
  group_by(IncomeGroup, year) %>%
  summarise(sum_emission = sum(co2, na.rm=T))
```

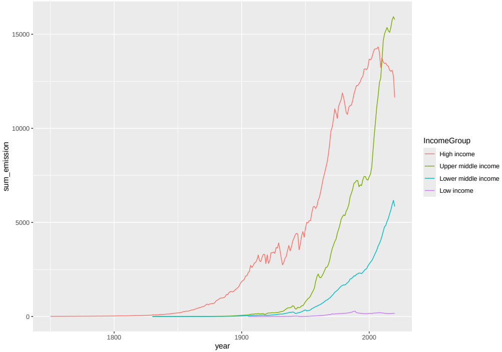
```
graph3_1 <- ggplot(data_pollution_by_group)
graph3_2 <- graph3_1 + aes(x = year,
                           y = sum_emission,
                           group = IncomeGroup,
                           colour = IncomeGroup)
graph3_3 <- graph3_2 + geom_line()
```

Reorganize the legend with ggplot

```r
data_pollution_by_group$IncomeGroup <- factor(data_pollution_by_group$IncomeGroup
graph4_1 <- ggplot(data_pollution_by_group)
graph4_2 <- graph4_1 + aes(x = year,
                           y = sum_emission,
                           group = IncomeGroup,
                           colour = IncomeGroup)
graph4_3 <- graph4_2 + geom_line()
```

Carbon footprint trajectory by country group



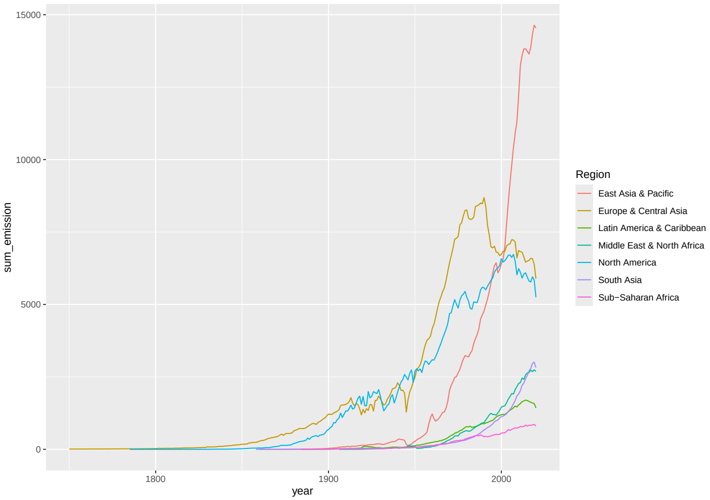


CO2 emissions trajectory by continent


```
data_pollution_by_cont <- join_pollution_wb_data %>%
  group_by(Region, year) %>%
  summarise(sum_emission = sum(co2, na.rm=T))
```

```
data_pollution_by_cont$Region <- factor(data_pollution_by_cont$Region)
graph5_1 <- ggplot(data_pollution_by_cont)
graph5_2 <- graph5_1 + aes(x = year,
                           y = sum_emission,
                           group = Region,
                           colour = Region)
graph5_3 <- graph5_2 + geom_line()
```

CO2 emissions trajectory by continent



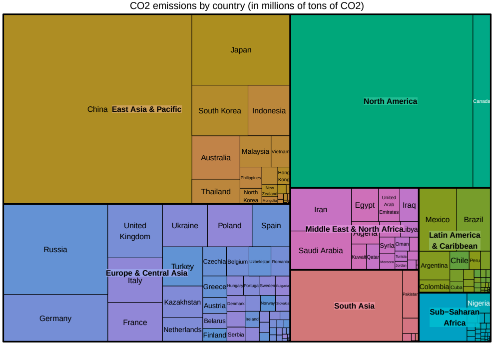
Treemap (I)

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

```
#install.packages("treemap")
library(treemap)

data_pollution_region_mean <- join_pollution_wb_data %>%
  filter(year >= 1990 & year <= 2020) %>%
  group_by(country,Region) %>%
  summarise(mean_gdp_per_capita = mean(gdp_per_capita, na.rm=T),
            mean_co2_per_capita = mean(co2_per_capita_en_kg, na.rm=T),
            mean_co2 = mean(co2, na.rm=T))

selection_1 <- data_pollution_region_mean %>%
  filter(mean_co2_per_capita>5000)

selection_2 <- data_pollution_region_mean %>%
  filter(mean_co2_per_capita>140)
```

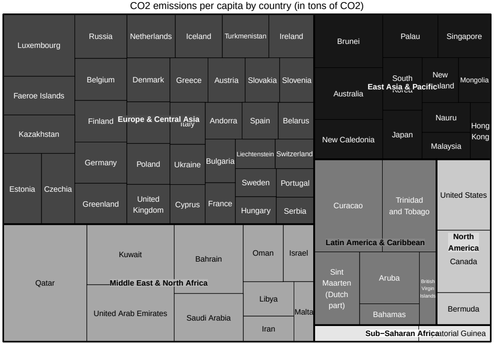
Treemap (II)

```
treemap(selection_2, index=c("Region","country"),
        vSize="mean_co2",
        type="index",
        title="CO2 emissions by country (in millions of tons of CO2)")
```



Treemap (III)

```
treemap(selection_1, index=c("Region","country"),
        vSize="mean_co2_per_capita", type="index",
        palette="-RdGy",
        title="CO2 emissions per capita by country (in tons of CO2)")
```



Sources of pollution (energy)

```
pollution_energy_mean <- join_pollution_wb_data %>%
  filter(year >= 1990 & year <= 2020) %>%
  group_by(country,Region) %>%
  summarise(mean_gas = mean(gas_co2/co2, na.rm=T),
            mean_cement = mean(cement_co2/co2, na.rm=T),
            mean_oil = mean(oil_co2/co2, na.rm=T),
            mean_coal = mean(coal_co2/co2, na.rm=T),
            mean_flaring = mean(flaring_co2/co2, na.rm=T)) %>%
  mutate(mean_gas = ifelse(is.na(mean_gas),0,mean_gas),
         mean_cement= ifelse(is.na(mean_cement),0,mean_cement),
         mean_oil = ifelse(is.na(mean_oil),0,mean_oil),
         mean_coal = ifelse(is.na(mean_coal),0,mean_coal),
         mean_flaring = ifelse(is.na(mean_flaring),0,mean_flaring)) %>%
  mutate(sum_test = mean_gas + mean_cement + mean_oil + mean_coal + mean_flaring)
```