Data and Climate

Session 1

Jean-Baptiste Guiffard

2023-12-04

Course structure

Objectives

1. Intensive training in R software
2. Advanced skills in data collection, processing, and analysis on R (graphical representations, maps, data extractions, web scraping, textual analysis. . .)
3. A common theme: climate data and sustainable development indicators.

→ Course evaluation: a common project probably

Sessions

Sessions	Topics
Session 1	The Basics of R / Manipulating dataset with the DPLYR package
Session 2	Graphic representations with GGPLOT
Session 3	Making maps with R
Session 4	Extracting and analyzing textual data using R
Session 5	Web scraping with R
Session 6	Produce documents with Rmarkdown. . .

Introducing R

R software

R is:

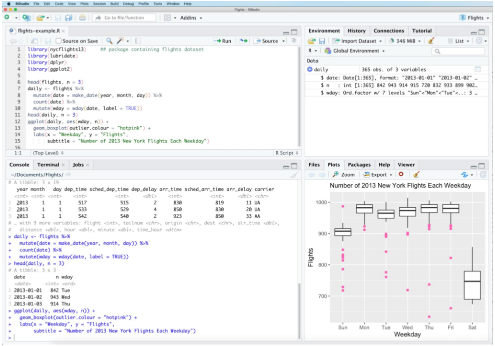
- a platform for the object-oriented statistical programming language S
- Freely distributed by the CRAN (Comprehensive R Archive Network).
- and Open-source

S is a very high level programming language (a programming language with a high level of abstraction that allows to write programs using common natural language words - very often English - and familiar mathematical symbols) and a data analysis environment designed in the 1970s by John Chambers (statistician at Harvard) → the two modern implementation of S are R and S-PLUS.

Getting R and RStudio

- To obtain R for any operating system, just go to CRAN at the following address: https://cran.r-project.org/
- The installation of R is simple by following the instructions.
- The use of R software is facilitated by the use of an integrated development environment (IDE) → RStudio
- After having installed R beforehand, you can download and install RStudio from the following site: https://posit.co/.

RStudio



```r
library(nycflights13)     ## package containing flights dataset
library(lubridate)
library(dplyr)
library(ggplot2)

head(flights, n = 3)
daily <- flights %>%
  mutate(date = make_date(year, month, day)) %>%
  count(date) %>%
  mutate(wday = wday(date, label = TRUE))
head(daily, n = 3)
ggplot(daily, aes(wday, n)) +
  geom_boxplot(outlier.colour = "hotpink") +
  labs(x = "Weekday", y = "Flights",
       subtitle = "Number of 2013 New York Flights Each Weekday")
```

```
daily          365 obs. of 3 variables
   $ date: Date[1:365], format: "2013-01-01" "2013-01-02" …
   $ n   : int [1:365] 842 943 914 915 720 832 933 899 902…
   $ wday: Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 3 …
```

```
# A tibble: 3 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>
1  2013     1     1      517            515         2      830            819        11 UA
2  2013     1     1      533            529         4      850            830        20 UA
3  2013     1     1      542            540         2      923            850        33 AA
# … with 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
> daily <- flights %>%
+   mutate(date = make_date(year, month, day)) %>%
+   count(date) %>%
+   mutate(wday = wday(date, label = TRUE))
> head(daily, n = 3)
# A tibble: 3 x 3
  date           n wday
  <date>     <int> <ord>
1 2013-01-01   842 Tue
2 2013-01-02   943 Wed
3 2013-01-03   914 Thu
> ggplot(daily, aes(wday, n)) +
+   geom_boxplot(outlier.colour = "hotpink") +
+   labs(x = "Weekday", y = "Flights",
+        subtitle = "Number of 2013 New York Flights Each Weekday")
> |
```

Figure 1: RStudio, a free and open source IDE for R

The RStudio environment is presented as a global window divided into 4 distinct sub-windows:

- The script window
- The console
- The environment and history window
- The files, graphs, packages and help window.

A few Resources

Help on a function:

```
?mean
help(mean)
```

Other resources are interesting:

- the help section in RStudio
- The CRAN site has manuals, mailing lists, FAQ's to facilitate exploration. . .
- do a direct web search using a search engine with the keywords R and CRAN.
- the site www.r-bloggers.com.
- Books. . .

The basics of the R language

R, a calculator (I)

```r
2+2.5
```

```
## [1] 4.5
```

```r
7*5
```

```
## [1] 35
```

```r
100/25
```

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

R, a calculator (II)

Operators	Definitions
+	Addition
-	Substraction
*	Multiplication
/	Division
^	Exponent
sqrt	Square Root
log	Logarithm
exp	Exponential
abs	Absolute Value
...	...

R, a calculator (III)

Operators	Definitions
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to
&	And
|	Or

Some basic instructions about R environment

This instruction gives a list of all objects in a working environment:

```r
objects()
ls()
```

If I want to completely clear the environment, I can write the following command:

```r
rm(list=ls(all=TRUE))
```

To delete a particular object:

```r
rm(oneobject)
```

And to quit R...

```r
q()
```

Load data

Working Directory

```r
setwd("C://Folder")
```

Upload the data

```r
data <- read.csv2('DATA/owid-co2-data.csv', sep=",")
```

Packages

A package (or library) is a set of programs that completes and increases the functionalities of R → generally dedicated to a particular method or to a specific application domain.

Downloading a package

```
install.packages('tidyverse')
```

Using a package

```
library(tidyverse)
require(tidyverse)
```

Tidyverse

Figure 2: tidyverse, a collection of extensions designed to work together and based on a common philosophy

- Visualization: ggplot2
- Data manipulation: dplyr, tidyr, tibble
- Data import/export: readr
- Variable manipulation: forcats (qualitative variables), stringr (string)
- Data extraction from the Web
- Programming: purrr

Objects in R

The scalar

An object of type “scalar” can be :

- null
- logical
- numeric
- complex
- character

```
#Scalar
a <- 1
b <- "Initiation à R"
b1 <- FALSE
```

To know the class of an object:

```
class(object)
mode(object)
```

Vector

Numeric vector:

```
#Vecteur
c <- c(3, 4, 5, 6)
d <- c(7, 8, 9, 10)
```

Build a numerical vector:

```
seq(1,10,by=1)
```

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

```
rep(1,8)
```

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

Character vector:

```
e <- c("Initiation", "_", "R")
```

Matrix and list

Matrix → atomic objects i.e. same mode or type for all values.

```
#Matrix
matrix_1 <- matrix(1, nrow = 4, ncol=1)

length(matrix_1)
dim(matrix_1)
```

List → a heterogeneous object i.e. an ordered set of objects that do not always have the same mode or the same length.

```
#Liste
i <- list(b, d, matrix_1, "h")
j <- list(i, "Poupée russe de liste")
```

Data-frame

Data-frame → particular list whose components are of the same length, but whose modes can differ (quantative and qualitative variables measured on the same individuals).

```
#Data frame
taille <- c(152, 156, 160, 160, 163, 167, 169, 173, 174, 174)
masse <- c(51, 51, 54, 60, 61, 64, 70, 71, 72, 73)
sexe <-c("M","F","F","M", "M","F","F","M", "F", "F")
df <- data.frame(taille,masse,sexe)
print(df)
```

```
##    taille masse sexe
## 1     152    51    M
## 2     156    51    F
## 3     160    54    F
## 4     160    60    M
## 5     163    61    M
## 6     167    64    F
## 7     169    70    F
## 8     173    71    M
## 9     174    72    F
## 10    174    73    F
```

Functions (I)

- An R object, many of which are already predefined in R, but which can also be created.
- A function admits arguments as input and returns a result as output.

Functions mean and sd :

```
mean(df$taille)
```

```
## [1] 164.8
```

```
var(df$taille)
```

```
## [1] 61.06667
```

```
sd(df$taille)
```

```
## [1] 7.814516
```

Functions (II)

```r
summary(df$taille)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    152.0   160.0   165.0   164.8   172.0   174.0 
```

Functions (III)

Function rnorm :

- Generates random numbers following a normal distribution
- Takes three arguments: n the number of values, mean the mean (default =0) and sd the standard deviation of the law (default =1).

```r
set.seed(140) # fix the seed of the generator ...
# allows to find the same results from one simulation to another.
rnorm(n=4)
```

```
## [1] 1.9279015 0.7317210 0.9546176 0.7312800
```

Manipulate datasets with Dplyr

Presentation of the database

CO2 and Greenhouse Gas Emissions dataset (Our World in data)

Data on CO2 emissions (annual, per capita, cumulative and consumption-based), other greenhouse gases, energy mix, and other relevant metrics [1]:

- ISO-CODE (alpha-3);
- "population": Population of each country or region;
- "gdp": Gross Domestic product in $ (2011 prices);
- "co2": Annual production-based emissions of carbon dioxide (CO2) in million tonnes;
- "co2_per_capita"
- "cumulative_co2"
- "share_global_co2"
- . . .

[1] Codebook: https://github.com/owid/co2-data/blob/master/owid-co2-codebook.csv

Pipping with “%>%”

- Puts the object on the left side as the first argument of the function on the right side

```
as.numeric(data_pollution$co2) %>% mean(na.rm=T)
```

```
## [1] 326.6583
```

Is the same as. . .

```
mean(as.numeric(data_pollution$co2), na.rm=T)
```

```
## [1] 326.6583
```

This can improve the efficiency of a code. In a single command, several functions can be applied in a readable way.

Subset observations

Filter: Extract rows that meet logical criteria.

```
data_continents <- data_pollution_num %>%
  filter(country %in% c("Africa","Europe","Oceania","Asia",
                        "North America","South America"))
```

sample_n: Select a random sample

```
sample_data_pollution <- data_pollution_num %>%
  sample_n(10)
```

Slice: Select rows by index

```
slice_data_pollution <- data_pollution_num %>%
  slice(25:40)
```

distinct: Remove duplicates

```
distinct_data_pollution <- data_pollution_num %>%
  distinct(country,.keep_all = T)
```

Modify variables

select: Selection of columns by name

```
data_continents <- data_continents %>%
  select(country, year, population, co2,cumulative_co2)
```

rename: change the name of a column

```
data_continents <- data_continents %>%
  rename(CO2 = co2)
```

Summarise data (I)

What is the number of observations per continent?

```
data_continents %>%
  dplyr::count(country)
```

```
##          country   n
## 1         Africa 137
## 2           Asia 191
## 3         Europe 271
## 4 North America 236
## 5        Oceania 161
## 6 South America 137
```

Summarise data (II)

summarise: summarise data into single row of values

```
data_continents %>%
  group_by(country) %>%
  summarise(number_obs = n())
```

```
## # A tibble: 6 x 2
##   country       number_obs
##   <chr>              <int>
## 1 Africa               137
## 2 Asia                 191
## 3 Europe               271
## 4 North America        236
## 5 Oceania              161
## 6 South America        137
```

Get basic statistics for some variables

```
data_continents %>%
  filter(year >=2010 & year < 2020) %>%
  summarise(sum_co2 = sum(CO2))
```

```
##    sum_co2
## 1 342167.8
```

```
data_continents %>%
  group_by(country) %>%
  summarise(mean_co2_emis = mean(CO2, use.na=T))
```

```
## # A tibble: 6 x 2
##   country       mean_co2_emis
##   <chr>                 <dbl>
## 1 Africa                 348.
## 2 Asia                    NA
## 3 Europe                1960.
## 4 North America         2025.
## 5 Oceania                130.
## 6 South America          316.
```

```
#%>% mutate(rank = rank(-mean_co2_emis))
```

Create new variables (I)

Mutate: Create a numerical variable. . .

```
data_continents %>%
  mutate(co2_per_capita = CO2/population) %>%
  group_by(country) %>%
  summarise(mean_co2_per_capita = mean(co2_per_capita, na.rm=T)) %>%
  mutate(rank = rank(-mean_co2_per_capita))
```

```
## # A tibble: 6 x 3
##   country       mean_co2_per_capita  rank
##   <chr>                       <dbl> <dbl>
## 1 Africa                0.000000556     6
## 2 Asia                  0.000000926     5
## 3 Europe                0.00000384      3
## 4 North America         0.00000720      1
## 5 Oceania               0.00000548      2
## 6 South America         0.00000118      4
```

Create new variables (II)

with *case_when*: Create a categorical variable. . .

```
data_continents <- data_continents %>%
  mutate(cat_co2 = case_when(
    CO2 < 23 ~ '[0,23[',
    CO2 >= 23 & CO2 < 220 ~ '[23,220[',
    CO2 >= 220 & CO2 < 1385 ~ '[220,1385[',
    CO2 > 1385  ~ '[1385,20609[',
  ))
data_continents %>%
  filter(year==2019)
```

```
##          country year population       CO2 cumulative_co2       cat_co2
## 1         Africa 2019 1308064186  1408.479       46284.70 [1385,20609[
## 2           Asia 2019 4600172830 20608.592      512599.09 [1385,20609[
## 3         Europe 2019  748381407  5430.239      526209.75 [1385,20609[
## 4 North America 2019  587512602  6460.726      472026.81 [1385,20609[
## 5        Oceania 2019   42128048   471.189       20536.73   [220,1385[
## 6 South America 2019  427199423  1065.510       42294.56   [220,1385[
```

Combine data (I)

Join: merge the data but keep only the rows of both data sets.

```
Metadata_Country <- read.csv2('DATA/Metadata_Country.csv', sep=",") %>%  rename

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

Combine data (II)

Anti-join: Keep all rows of the left data set that do not match the right data set.

```
antijoin_pollution_wb_data <- data_pollution_num %>%
  anti_join(Metadata_Country, by = c("iso_code" = "Country_code"))

head(antijoin_pollution_wb_data %>%
  group_by(country) %>%
  count())
```

```
## # A tibble: 6 x 2
## # Groups:   country [6]
##   country                            n
##   <chr>                          <int>
## 1 Africa                           137
## 2 Anguilla                          31
## 3 Antarctica                        41
## 4 Asia                             191
## 5 Asia (excl. China & India)       191
## 6 Bonaire Sint Eustatius and Saba   95
```