

# **R for Beginners**

**Draft II**

Massimiliano Porto\*

October 26, 2017

\*Kobe University



# Contents

<b>1 PART 1</b>	<b>5</b>
1.1 R and RStudio . . . . .	5
1.2 RStudio: Basics . . . . .	5
1.2.1 RStudio Interface . . . . .	5
1.2.2 Launch a new project . . . . .	6
1.2.3 Basic operations . . . . .	8
1.2.3.1 Objects . . . . .	9
1.2.3.2 Vector and Matrix . . . . .	10
1.2.3.3 Logical operators . . . . .	15
1.2.3.4 Flow control . . . . .	16
1.2.3.5 Functions . . . . .	18
1.3 Getting help for R . . . . .	18
1.4 Resources for R . . . . .	19
1.4.1 CRAN Task Views . . . . .	19
1.4.2 Books . . . . .	19
1.4.3 Online Guides for R . . . . .	19
1.4.4 R-Bloggers . . . . .	19
1.4.5 Search . . . . .	19
<b>2 PART 2</b>	<b>21</b>
2.1 Packages . . . . .	21
2.2 R Script & R Markdown . . . . .	21
2.2.1 LaTeX . . . . .	29
2.3 Data management . . . . .	29
2.3.1 Class and structure . . . . .	31
2.3.2 Import dataset . . . . .	32
2.3.3 Simple operations on dataset . . . . .	35
2.3.3.1 Select and rename columns . . . . .	35
2.3.3.2 Transform database long-wide . . . . .	35
2.3.3.3 Rename single columns and drop variables . . . . .	37
2.3.3.4 Operations only on some columns of the database . . . . .	37
2.3.3.5 Subset the database . . . . .	37
2.3.3.6 Merge two databases . . . . .	39
2.4 Dates in R . . . . .	40
2.5 Plot . . . . .	41
2.5.1 Simple plot . . . . .	42
2.5.2 GGPLOT . . . . .	44



# 1 PART 1

## 1.1 R and RStudio

R is a free software environment for statistical computing and graphics.

R Studio is an integrated development environment that makes R easier to use.

## 1.2 RStudio: Basics

### 1.2.1 RStudio Interface

If you open Rstudio you will see a screen like in figure 1.1. It is divided in 4 panes.

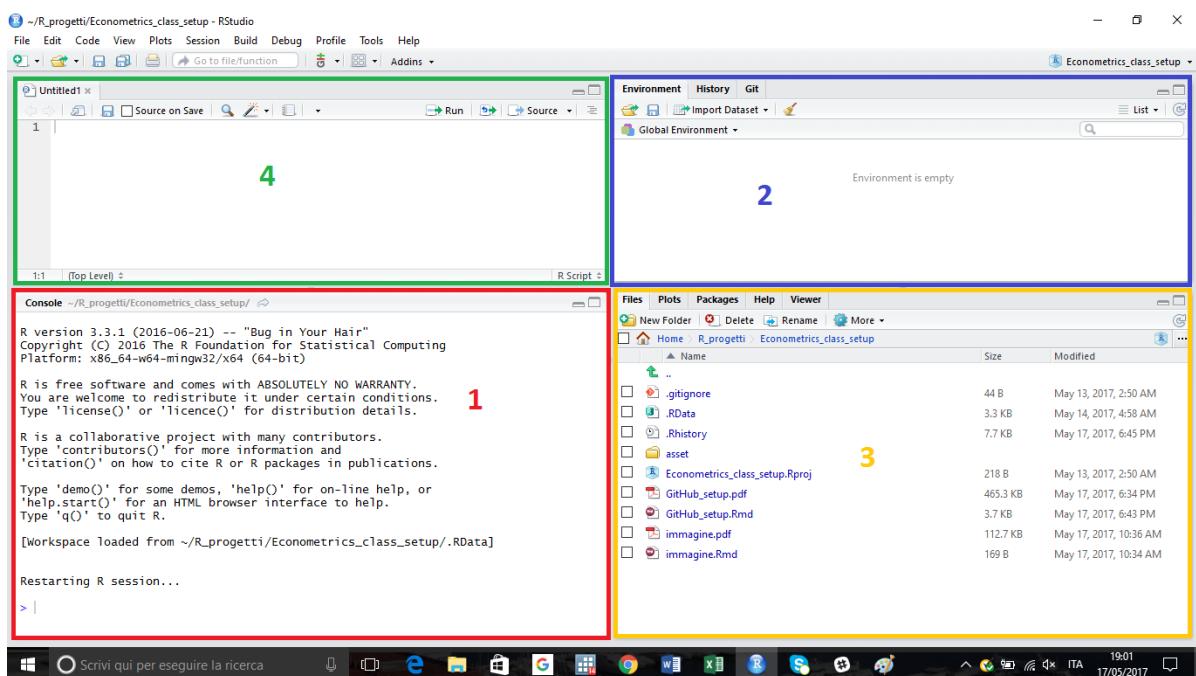


Figure 1.1: RStudio panes

**Console pane:** the **console** is the pane we numbered 1 in the figure 1.1. Here you write your code, called *command* in *R language* because you command your computer something to do.

## 1 PART 1

**Environment pane:** the **environment** pane (2 in figure 1.1) is the place where that stores all the objects you create in R.<sup>1</sup> We will discuss more about objects later.

**Files, plots, packages,.. pane:** the pane number 3 in figure 1.1 is where you find your files, the packages you can install to improve the capabilities of R, where you can visualize the plots you create etc.

**Source pane:** **source** pane (4 in figure 1.1) provides you different ways to write and save your code. Later we will cover **R Script** and **R Markdown** in detail.

However, if you are not satisfied with this visualization you can modify it by clicking on the **toolbar** View > Panes > Pane Layout.

### 1.2.2 Launch a new project

A project is a place to store your work on a particular topic (or project). Create a project called "**Econometrics**" for all of your work in this course.

Follow the procedure as in figures 1.2, 1.3, 1.4.

Click on the R symbol in the top hand right corner, click New Directory > Empty Project and then write the directory name and click create project.

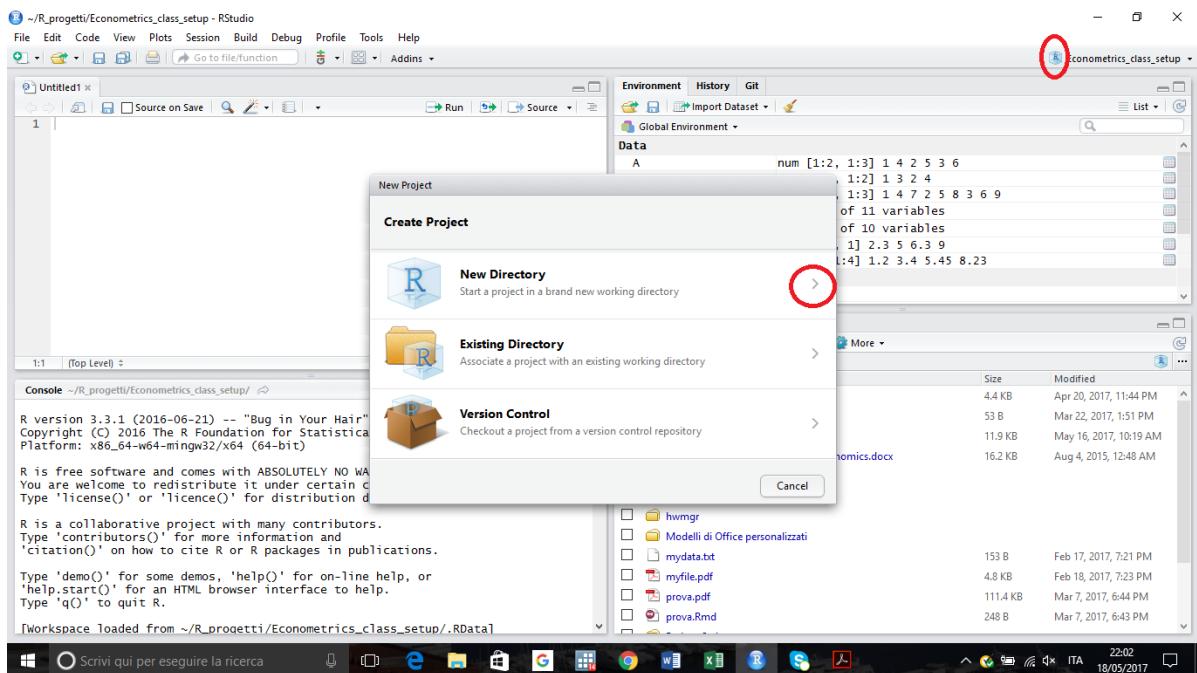


Figure 1.2: Launch a new project

If you want to check if you are in the right directory enter the following in the console pane.

<sup>1</sup>To be precise, the environment pane doesn't show hidden objects. (it's a quite technical issue.)

## 1.2 RStudio: Basics

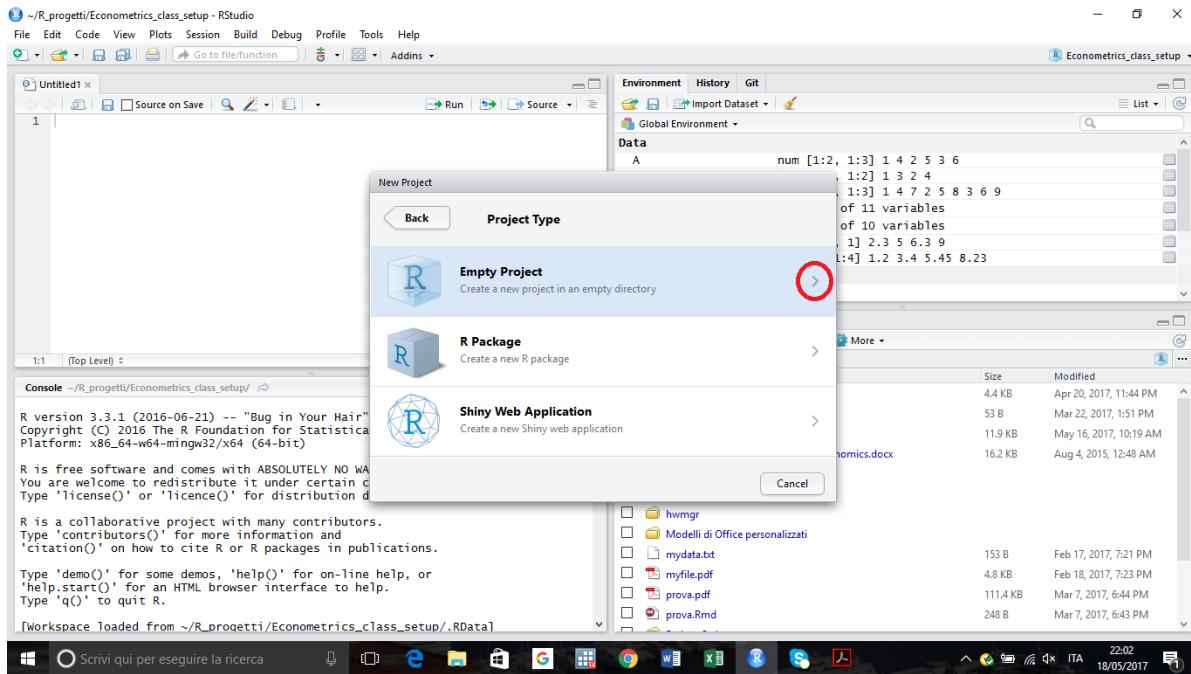


Figure 1.3: Launch a new project

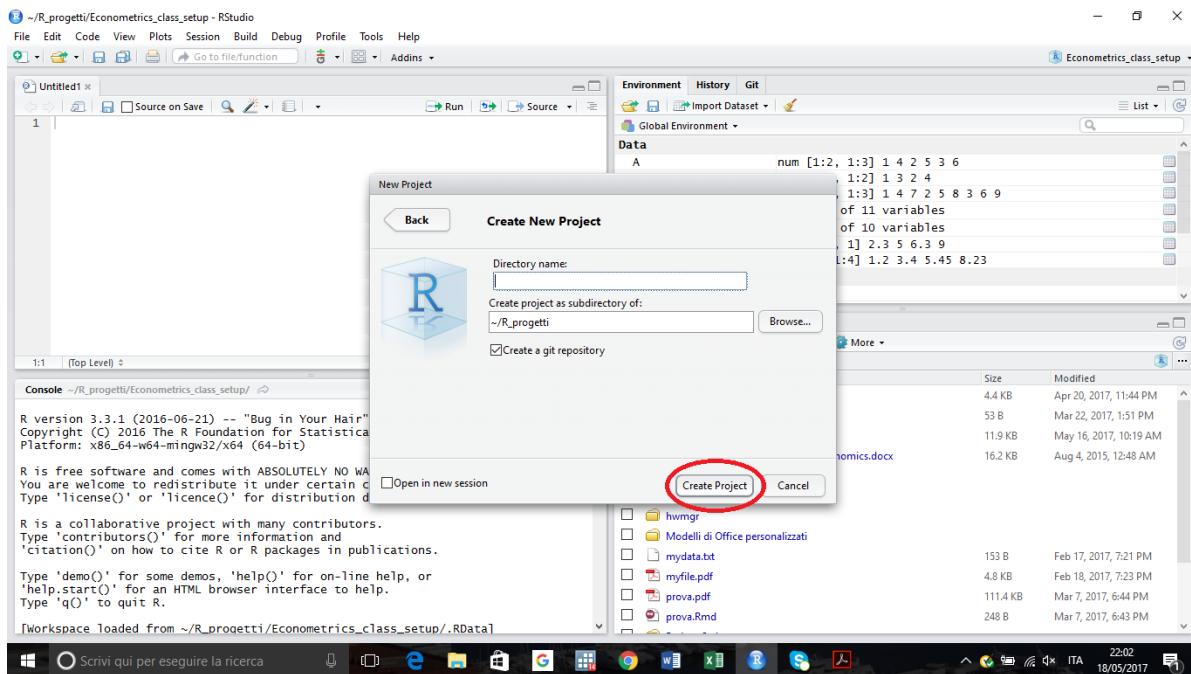


Figure 1.4: Launch a new project

## 1 PART 1

```
getwd()  
## [1] "C:/Users/porto/OneDrive/Documenti/R_progetti/Econometrics_class_setup"  
If you want to change it  
setwd()
```

write the new directory path in the brackets or from **Files, plots, packages...** pane by clicking on **more** as in figure ??.

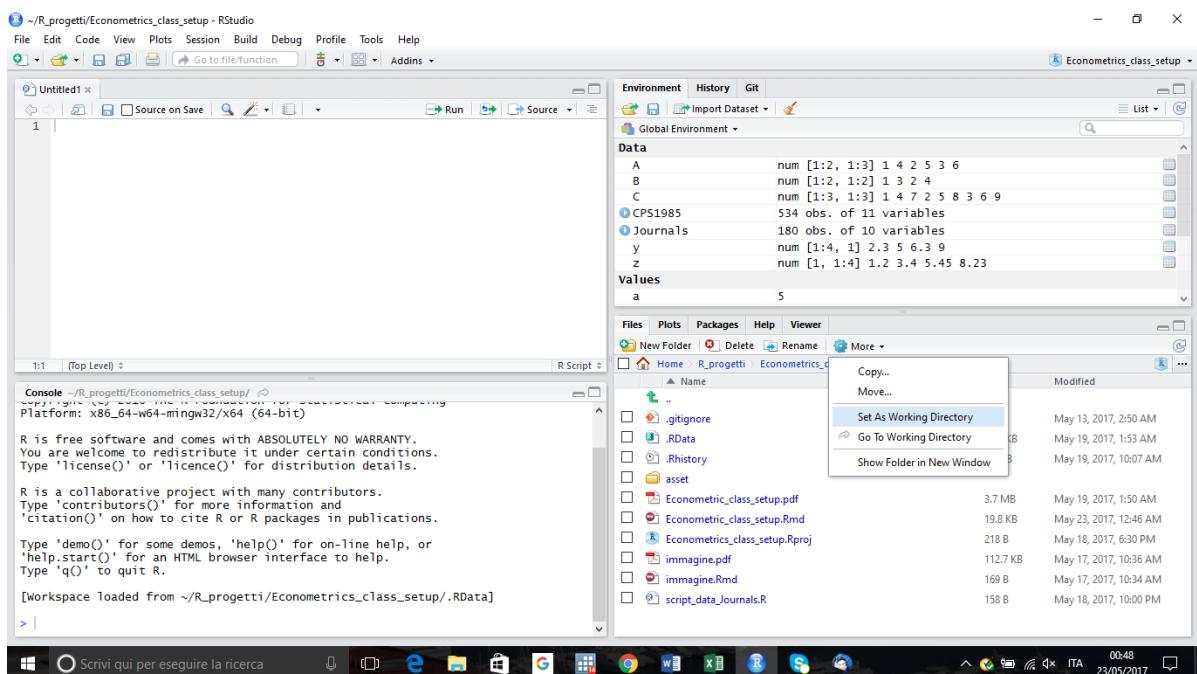


Figure 1.5: Set working directory ??

### 1.2.3 Basic operations

Files in your working directory include:

- **.RData:** Holds the objects etc in your environment;
- **.RHistory:** Holds the history of what you typed in the console;
- Save on exit;
- **.RProfile:** Holds specific setup information for the working directory you are in.
- Did you create it? Check by writing in the console pane `file.exists("~/Rprofile")`. If not, you can create it by writing in the console pane `file.edit("~/Rprofile")`;
- **Setting up R profile:** you can customize your environment for a project. We will discuss how to set up .RProfile in class.

### 1.2.3.1 Objects

Let's start giving some basic commands to R in the **console** pane. Let's treat R as a basic calculator making it compute  $1 + 1$ .

We see that R provides the answer 2 preceded by [1]. Let's skip the meaning of these brackets for the moment.

Let's say that you think to use this operation  $1 + 1$  recursively during your work. R provides a fast and simple way to use this information without the need to give the same inputs again and again. How? By creating an **object**. Objects are values with structure. The name you assign to the **object** stores a value.

For example, let's call the object that stores our former operation  $a$ .<sup>2</sup>

Did you see what happened? The **object** that we created was just added to the **environment pane**. Let's sum  $a + 2$ . What will be the solution? (figure 1.6)

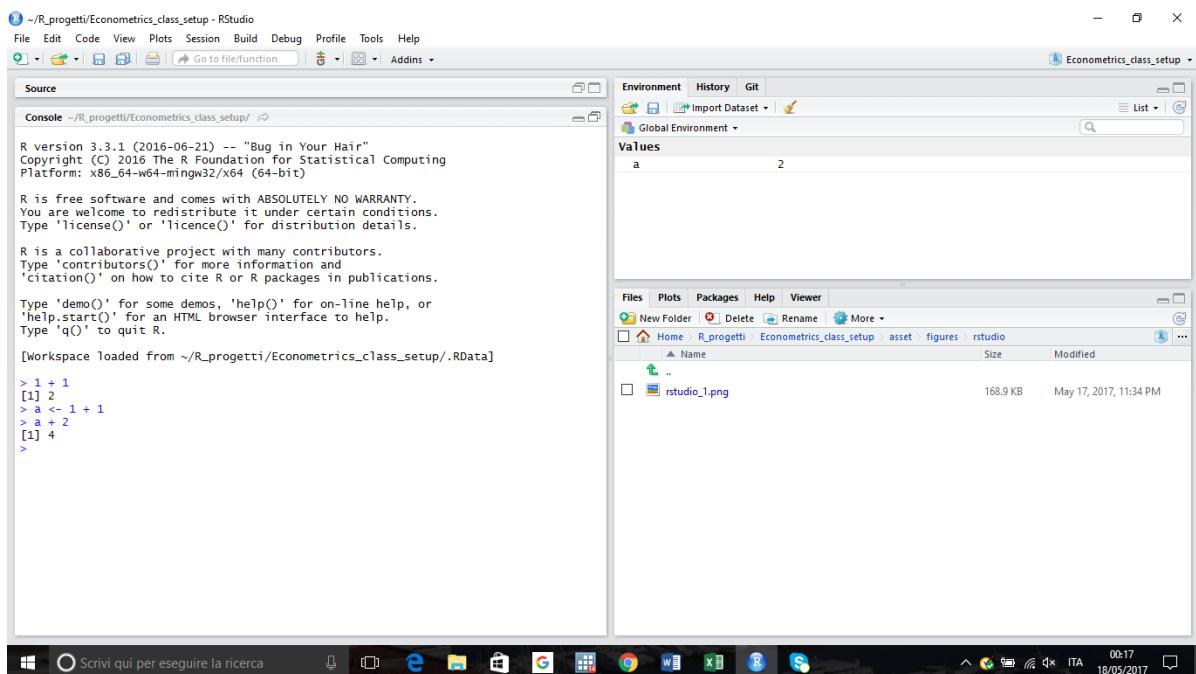


Figure 1.6: Objects

Be careful with **object** names. If you call another **object**  $a$  (for example  $a = 5$ ), R overwrites your previous object called  $a$ . So in the environment pane you will find again only one **object** called  $a$ , but with the new assigned value. (figure 1.7)

Let's examine the meaning of the brackets. Create a new object,  $b$ , that stores all the integers from 0 to 100. We do this using the operator  $::$ . We see that a new object is added at the environment pane. Let's visualize it in the **console** typing the input  $b$ . The command returns the numbers in object  $b$  preceded by bracketed numbers  $[]$ .

<sup>2</sup>The less-than symbol,  $<$ , followed by minus symbol,  $-$ , is the assign operator. In most cases  $=$  has the same meaning but the former symbol that looks like an arrow is what is used in programming language

# 1 PART 1

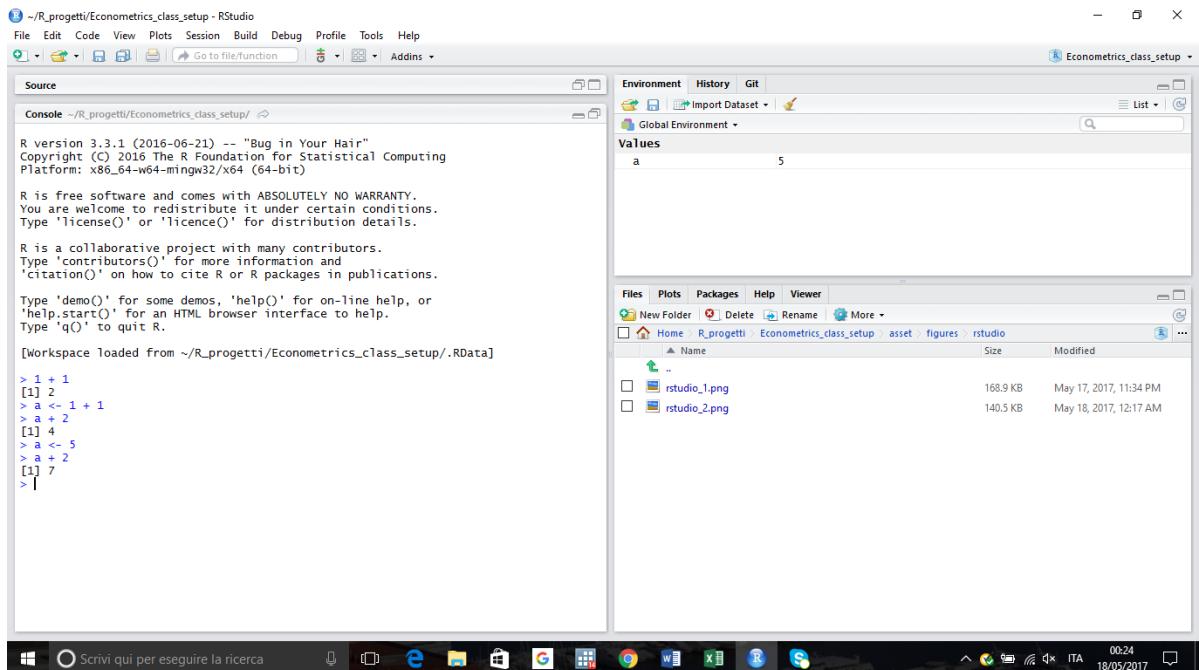


Figure 1.7: Overwriting objects

The bracketed numbers indicate the position of the first number in the row, within the object. This means that 0 is the first number of the series, while 18 is in the nineteenth position and so on (Fig. 1.8).

Now let's compute another operation. Let's multiply  $5 * b$  or we can also compute  $a * b$ . (figure 1.8)

### 1.2.3.2 Vector and Matrix

The former operation is a *scalar by vector* multiplication. Understanding vectors is key to understanding R. For details read **Applied Econometrics with R, Chapter 2**.<sup>3</sup>

Let's tries some basics. We compute a vector using the function `c()`.

```
x <- c(0.5, 3, 4.44, 7)  
## [1] 0.50 3.00 4.44 7.00
```

Other functions to create vectors are `rbind()` and `cbind()`.

```
y <- rbind(2.3, 5, 6.3, 9)  
##      [,1]  
## [1,]  2.3  
## [2,]  5.0  
## [3,]  6.3
```

<sup>3</sup>Since now it is clear how to give command and what it causes we will use less figures in the rest of this document.

## 1.2 RStudio: Basics

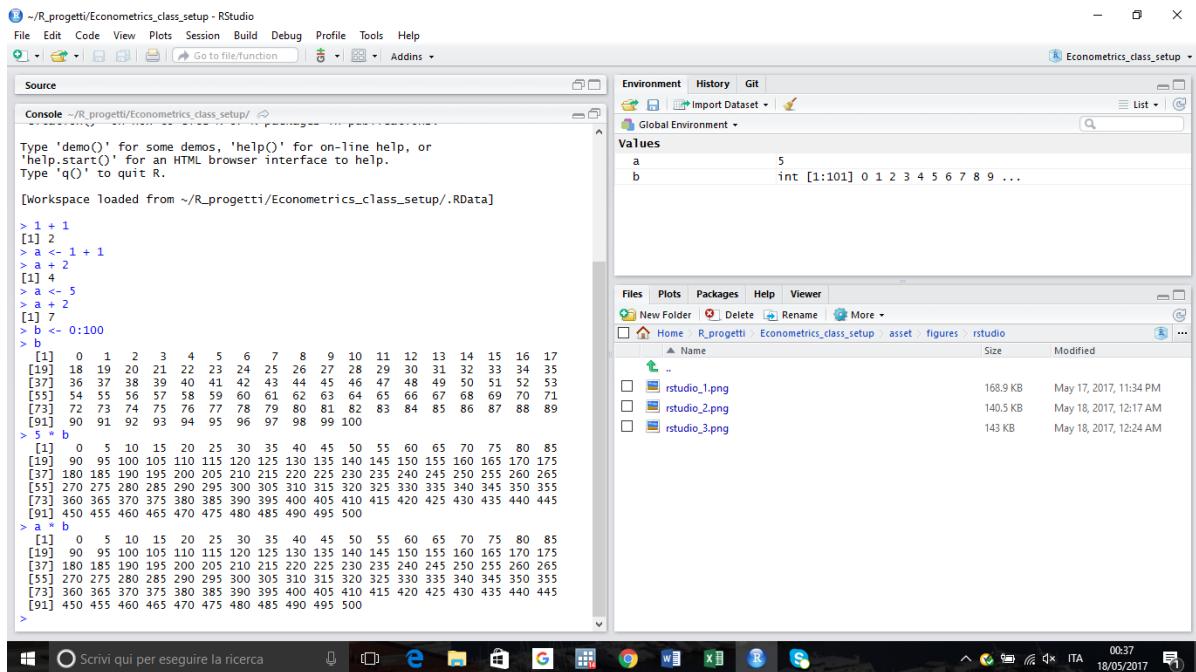


Figure 1.8: Scalar by vector

```

## [4,] 9.0

z <- cbind(1.2, 3.4, 5.45, 8.23)

##      [,1] [,2] [,3] [,4]
## [1,] 1.2  3.4  5.45 8.23

```

The difference with the function `c()` is that this one does not compute a vector in mathematical sense but just creates a list of numbers.<sup>4</sup> We can see the difference by checking the class of the vectors we created.

```

class(x)
## [1] "numeric"

class(y)
## [1] "matrix"

class(z)
## [1] "matrix"

We can compute various operations. For example,

```

```

a + x
## [1]  5.50  8.00  9.44 12.00

```

<sup>4</sup>Not to be confused with function `list()`. For more info write `?list` in the console pane

## 1 PART 1

```
a * y
##      [,1]
## [1,] 11.5
## [2,] 25.0
## [3,] 31.5
## [4,] 45.0
log(z)
##      [,1]     [,2]     [,3]     [,4]
## [1,] 0.1823216 1.223775 1.695616 2.107786
```

where the first operation returns a sum between a scalar and a vector, the second a multiplication between a scalar and a vector and the last operation is the logarithm of the vector.

Now let's build a matrix.

```
A <- matrix(c(
  1, 2, 3,
  4, 5, 6
))

##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
## [4,]    4
## [5,]    5
## [6,]    6
```

How come we didn't get a matrix? Because if we don't indicate the number of row nrow = (and/or the number of column ncol =) we get a column vector.

```
A <- matrix(c(
  1, 2, 3,
  4, 5, 6
), nrow = 2)

##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Still note that the order is not the one we want. How to adjust? Adding byrow = TRUE

```
A <- matrix(c(
  1, 2, 3,
  4, 5, 6
), nrow = 2, byrow = TRUE)

##      [,1] [,2] [,3]
```

```
## [1,]    1    2    3
## [2,]    4    5    6
```

Let's compute  $B$  as a square matrix.

```
B <- matrix(c(
  1, 2,
  3, 4
), nrow = 2, ncol = 2, byrow = TRUE)
```

```
##      [,1] [,2]
```

```
## [1,]    1    2
```

```
## [2,]    3    4
```

Let's now multiply  $a * A$

```
a * A
```

```
##      [,1] [,2] [,3]
```

```
## [1,]    5   10   15
```

```
## [2,]   20   25   30
```

Do you remember what we said at beginning about objects? If we assign the same name R overwrites it. So, why now we can perform our multiplication? Let's have a look at the environment pane (figure 1.9)

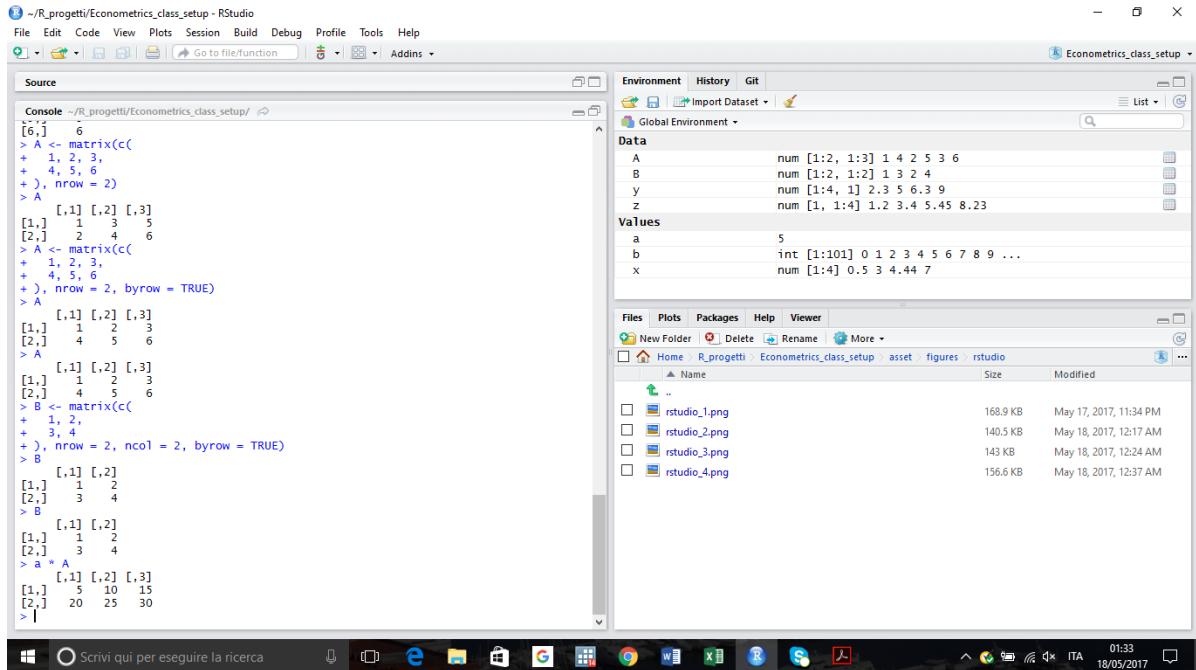


Figure 1.9: Case sensitive:  $a$  &  $A$

You can see that both values  $a$  and  $A$  (but also  $b$  and  $B$ ) are stored. This because R is case sensitive. This means that  $a \neq A$  and  $b \neq B$ . Be careful to this when writing codes.

## 1 PART 1

Have a look at the console pane. If you see how matrices were written you note a + symbol. This symbol means that the code you are writing continues on the following line. However, I anticipate that I didn't write the code in the console pane but in **R Markdown**. We will examine **R Script** and **R Markdown** later in this document.

Let's continue with matrix to see some operations. For example, the matrix multiplication is implemented by the following operator %\*%.

```
B %*% A
```

```
##      [,1] [,2] [,3]
## [1,]     9   12   15
## [2,]    19   26   33
```

We can also compute the determinant

```
det(B)
```

```
## [1] -2
  transpose
```

```
t(B)
```

```
##      [,1] [,2]
## [1,]     1     3
## [2,]     2     4
  inverse
```

```
solve(B)
```

```
##      [,1] [,2]
## [1,] -2.0  1.0
## [2,]  1.5 -0.5
  eigenvalues and eigenvectors
```

```
eigen(B)
```

```
## $values
## [1]  5.3722813 -0.3722813
##
## $vectors
##      [,1]      [,2]
## [1,] -0.4159736 -0.8245648
## [2,] -0.9093767  0.5657675
  and diagonal
```

```
diag(6)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]     1     0     0     0     0     0
## [2,]     0     1     0     0     0     0
## [3,]     0     0     1     0     0     0
## [4,]     0     0     0     1     0     0
```

```
## [5,]    0    0    0    0    1    0
## [6,]    0    0    0    0    0    1
```

Note that if you want 5 on the main diagonal

```
diag(5, 6)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    0    0    0    0    0
## [2,]    0    5    0    0    0    0
## [3,]    0    0    5    0    0    0
## [4,]    0    0    0    5    0    0
## [5,]    0    0    0    0    5    0
## [6,]    0    0    0    0    0    5
```

Note that `rep()` replicates a number in first position how many times you write in the second position.

```
diag(c(1, rep(2, 3), 1))
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    2    0    0    0
## [3,]    0    0    2    0    0
## [4,]    0    0    0    2    0
## [5,]    0    0    0    0    1
```

### 1.2.3.3 Logical operators

R includes logical operators such as `<`, `>`, `<=`, `>=`, `==` (for exact equality) and `!=` (for inequality).

Pay attention to `==` operator.

Let's see an example.

```
3 == 1 + 1 + 1
```

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

But what about `0.3 == 0.1 + 0.1 + 0.1`? Let's try

```
0.3 == 0.1 + 0.1 + 0.1
```

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

Use function `all.equal()` to compare decimal numbers

```
all.equal(0.3, 0.1 + 0.1 + 0.1)
```

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

Moreover, if you want to tie two logical expression you can use `&` for logical "and" and `|` for logical "or".

Some examples

```
2 > 1
```

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

## 1 PART 1

Let's take again our  $x$  vector and check if

```
x > 3 & x <= 7  
## [1] FALSE FALSE TRUE TRUE
```

### 1.2.3.4 Flow control

R allows to implement some actions only if some conditions are met or allows to implement the same action for a certain number of times. It provides control structures such as `if/else`, `for` and `while` loops. Read **Applied Econometrics with R, Chapter 2** for explanation.

Let's see here a simple example based on the previous calculation and compute

```
if(all.equal(0.3, 0.1 +0.1 +0.1)){  
  print("you are right")  
} else{  
  print("you are wrong")  
}
```

```
## [1] "you are right"
```

Let's print the even number of our vector  $b$

```
for(i in b){  
  if(i %% 2 == 0)  
    print(i)  
}  
  
## [1] 0  
## [1] 2  
## [1] 4  
## [1] 6  
## [1] 8  
## [1] 10  
## [1] 12  
## [1] 14  
## [1] 16  
## [1] 18  
## [1] 20  
## [1] 22  
## [1] 24  
## [1] 26  
## [1] 28  
## [1] 30  
## [1] 32  
## [1] 34  
## [1] 36  
## [1] 38
```

```
## [1] 40
## [1] 42
## [1] 44
## [1] 46
## [1] 48
## [1] 50
## [1] 52
## [1] 54
## [1] 56
## [1] 58
## [1] 60
## [1] 62
## [1] 64
## [1] 66
## [1] 68
## [1] 70
## [1] 72
## [1] 74
## [1] 76
## [1] 78
## [1] 80
## [1] 82
## [1] 84
## [1] 86
## [1] 88
## [1] 90
## [1] 92
## [1] 94
## [1] 96
## [1] 98
## [1] 100
```

Let's countdown 10, 9, 8, ... with `while`

```
d <- 10
while(d >= 0){
  cat(d, "\n")
  d <- d - 1
}

## 10
## 9
## 8
## 7
## 6
## 5
```

```
## 4  
## 3  
## 2  
## 1  
## 0
```

### 1.2.3.5 Functions

R allows to write functions that encode different commands avoiding to repeat the same commands.

R also provides a number of built-up functions. A diffuse family of R functions are the `apply()` family tha includes `sapply()`, `tapply()`, `lapply()`.

For details, read **Applied Econometrics with R, Chapter 2** or, for example, type in the **console** pane `?apply.[?"` let to get information about a command in R (you can also use `help()`)]

Let's write our own function for example. Let's say we have a list of matrices but we want only to work with square matrices. We will implement an easy function, that we call `is_square_matrix`, that will give us the result *TRUE* if the matrix is square, *FALSE* otherwise.

```
is_square_matrix <- function(x){  
  if(nrow(x) == ncol(x)){  
    TRUE  
  } else {  
    FALSE  
  }  
}
```

Let's test it with our matrices  $A$ ,  $B$ .

```
is_square_matrix(A)  
## [1] FALSE  
  
is_square_matrix(B)  
## [1] TRUE
```

Given that the building a function is a more advanced topic, make reference to more advanced materials.

## 1.3 Getting help for R

- Function
- `help`, `example`, `demo`, `help.search`
- Internet
- [www.r-project.org](http://www.r-project.org)
- [www.cran.r-project.org](http://www.cran.r-project.org)

- [www.rseek.org](http://www.rseek.org)
- Google
  - filetype: R regression - rebol
  - CRAN regression

## 1.4 Resources for R

### 1.4.1 CRAN Task Views

CRAN Task Views provide information about packages that can be used with R. Packages extend the functionality of R. They provide routines for various types of data manipulation, econometric models, etc. Packages are constantly being developed and updated by R users. Three Task Views are relevant to this course:

- Econometrics: <https://cran.r-project.org/web/views/Econometrics.html>
- Time series: <https://cran.r-project.org/web/views/TimeSeries.html>
- Empirical finance: <https://cran.r-project.org/web/views/Finance.html>

### 1.4.2 Books

Several books have been published on using R for applied econometrics. We will use parts of the following books (provided as handouts in class).

- Heiss, F., "Using R for Introductory Econometrics". Companion site for "Using R for Introductory Econometrics" (includes access to the book online for free) <http://www.urfie.net/index.html>
- Kleiber, C. and A. Zeileis, Applied Econometrics with R, Springer, 2008.

### 1.4.3 Online Guides for R

A comprehensive introduction to R can be found here: <https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf>

A brief and easy introduction to R for econometrics can be found here: [https://mondo.su.se/access/content/user/ma@su.se/Public/R\\_intro.pdf](https://mondo.su.se/access/content/user/ma@su.se/Public/R_intro.pdf)

Other R documentation and guides for various applications contributed by R users can be found here: <https://cran.r-project.org/other-docs.html#english>

### 1.4.4 R-Bloggers

A blog site containing many useful posts on using R. <http://www.r-bloggers.com/>

### 1.4.5 Search

Lots of information is posted online by R users including questions and answers. If you are stuck, search for information that will help you answer your question.



# 2 PART 2

## 2.1 Packages

```
# to import database
library("utils")
library("readr")
library("readxl")

# to plot
library("ggplot2")
library("easyGgplot2")

# for Time Series
library("fBasics")
library("tseries")
library("zoo")
library("xts")
library("timeSeries")

# Data management
library("tidyverse")

# for Date
library("lubridate")

# for creating tables
library("kableExtra")

# for editing and making websites in R Markdown
library("bookdown")
```

## 2.2 R Script & R Markdown

The problem in writing directly the commands in the console pane is that it is difficult to retrieve all the commands we wrote.

So if we want to have a trace of all the commands we write during our work is

## 2 PART 2

better to use **R Script**. We open it clicking on the white square with a cross below File as in figure 2.1.

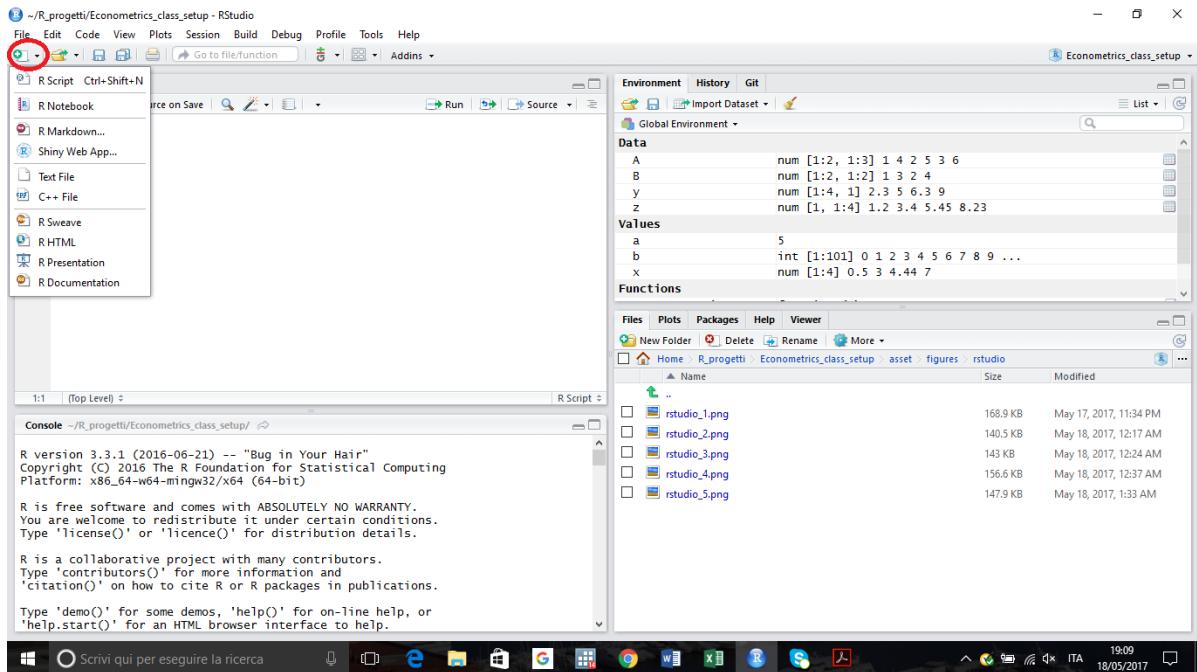


Figure 2.1: Open R Script & R Markdown

As we noted earlier in this section, in the console pane we saw commands written not on the same line but in different lines linked with a +. It was the case of *A* and *B* matrices (see figure 18).

Even if we can write in the console pane in the same way adding + to tell R that our command continues, we can write easily that command in R Script without the need to add + between lines. You can compare the two different writing styles in figure 2.2<sup>1</sup>.

Let's make some examples based on the resources of the textbook **Applied Econometrics with R**, by Kleiber & Zeileis.

To do this we need to load the package with the data.<sup>2</sup>

```
data("CPS1985", package = "AER")
```

Let's use a linear regression model, using the lm() function, to estimate the impact of experience on wage in the dataset CPS1985.

```
cps_lm <- lm(log(wage) ~ experience, data = CPS1985)

summary(cps_lm)
```

<sup>1</sup>Note that even if you can write all the code on the same line, it is convention to split the code in different lines. In this way it is clearer to the reader the main commands of the code

<sup>2</sup>You can install a package by install.packages()

## 2.2 R Script & R Markdown

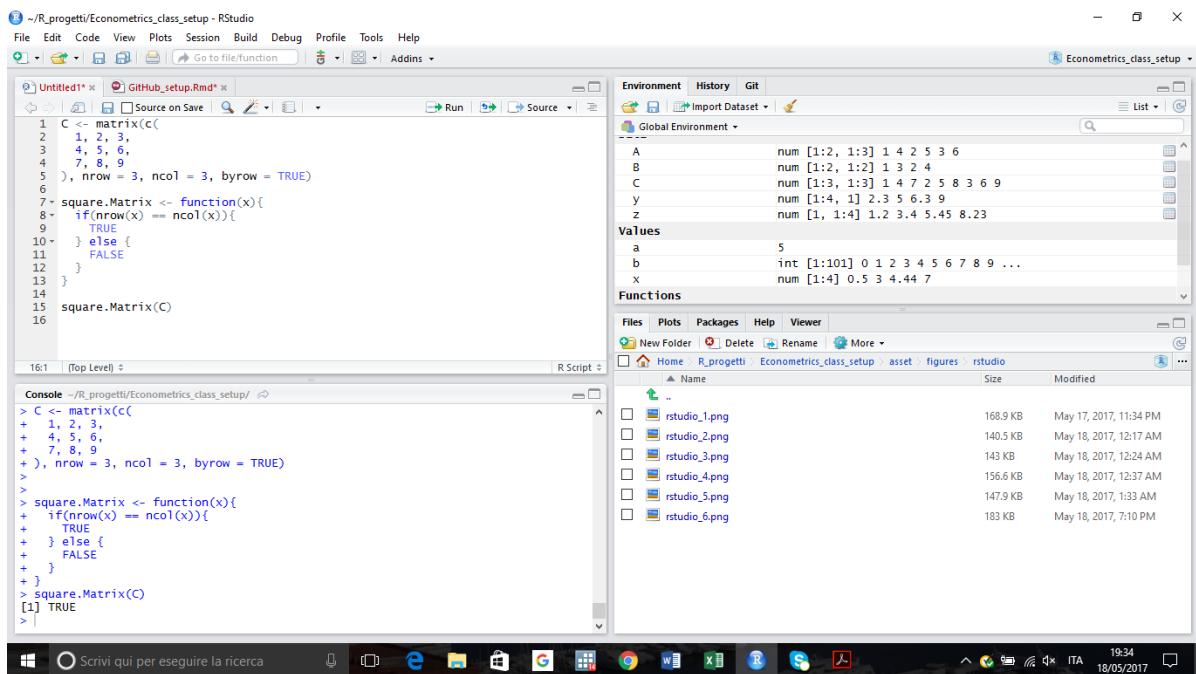


Figure 2.2: Different input in Console and R Script

The command `summary()` returns the usual summary of the coefficients (with estimates, standard errors, test statistics, and p values) as well as the associated R2, along with further information (figure 2.3).

Let's say now that we think that wage is influenced not only by experience but also by education.

So we modify our regression as follows

```

cps_lm <- lm(log(wage) ~ experience + education, data = CPS1985)

summary(cps_lm)

```

If we were probably editing our paper, thesis in Word, perhaps we found easily to copy the commands and the outcome and paste in Word. But now that we modified it, we should delete what we pasted and copy and paste the new outcome in Word.

Let's plot log-wage versus experience (figure 2.4).

Again we think to put it in our paper by copying or saving the image. But now we think perhaps that it would be better to make it fancier and add a title as follows.

```
plot(log(wage) ~ experience, data = CPS1985, pch = 1, col = "blue", main = "Scatterplot 1")
```

## 2 PART 2

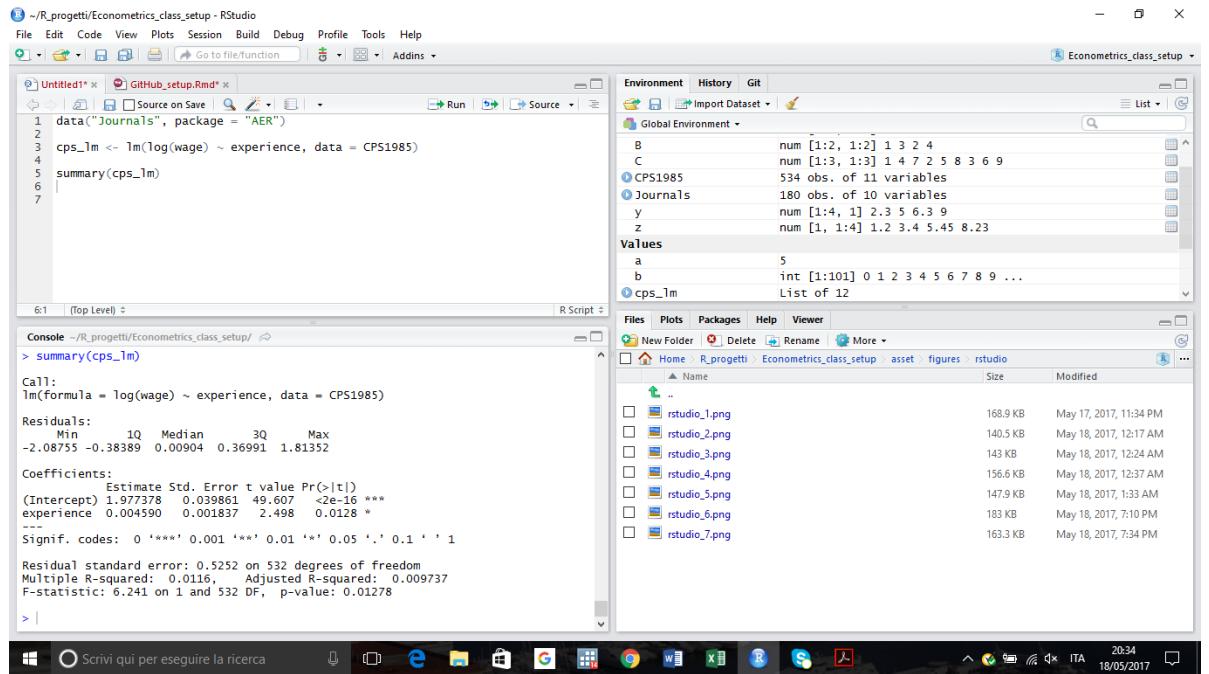


Figure 2.3: Summary of regression

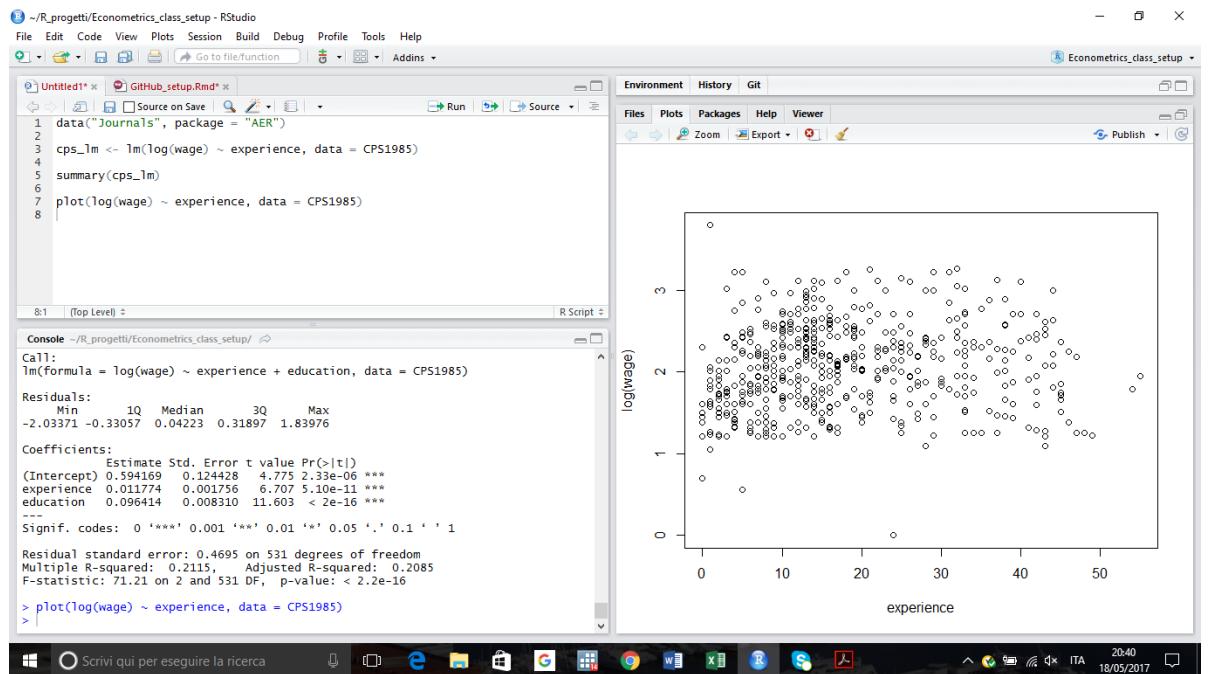
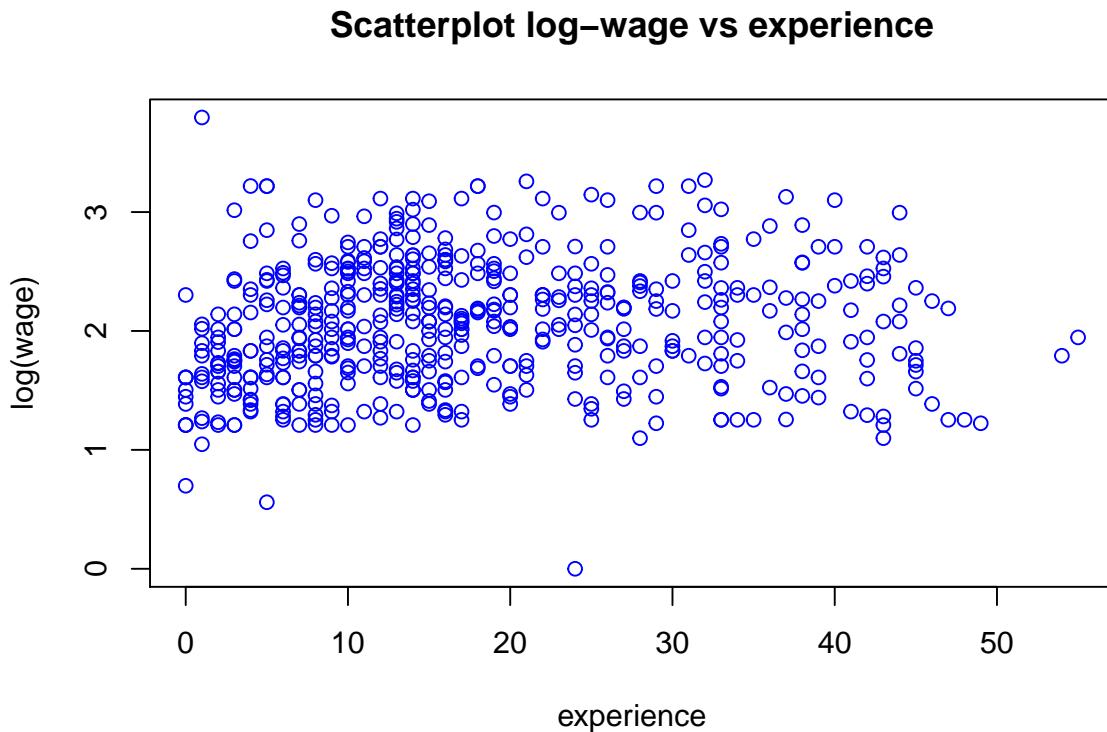


Figure 2.4: Scatterplot of log-wage versus experience



Again, we should change our output again.

In this case, **R Markdown** comes in handy. We open it as showed in figure 19. When we open it, R asks us what output we want (figure 2.5). Anyway this can be changed later.

A window like in figure 2.6 is opened.

After we opened **R Markdown**, it explains its basic characteristics. As you can see there is a grey space in the document

This is the chunk where you can write your code and run it by clicking on the arrow. If you click the arrow in the `{r cars}` example you will see something like this (figure 2.7)

As it is clear, the output is already in your document. What is the good thing? If you change mind about your code you need only to modify it in the chunk and run again. No need to rewrite, copy and paste somewhere else.

**R Markdown** provides various options for displaying the output. For example, you can prevent what you write in the code from being included in your paper.

To start working in **R Markdown** you can delete the examples. To insert by yourself the code chunk just click on *Insert* (figure 2.8). If you click the gear symbol in the code chunk you can have an idea of some options.

This document was edit in **R Markdown**. Let's see some examples of how we edited (figures 2.9 & 2.10).

## 2 PART 2

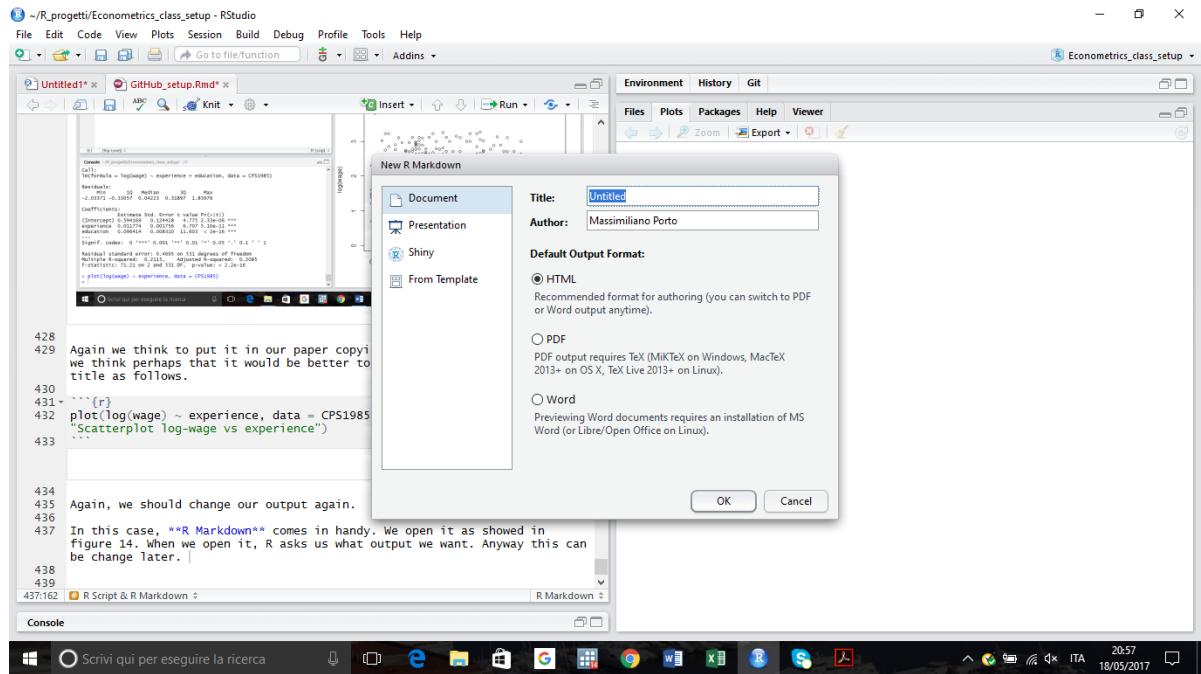


Figure 2.5: Open R Markdown

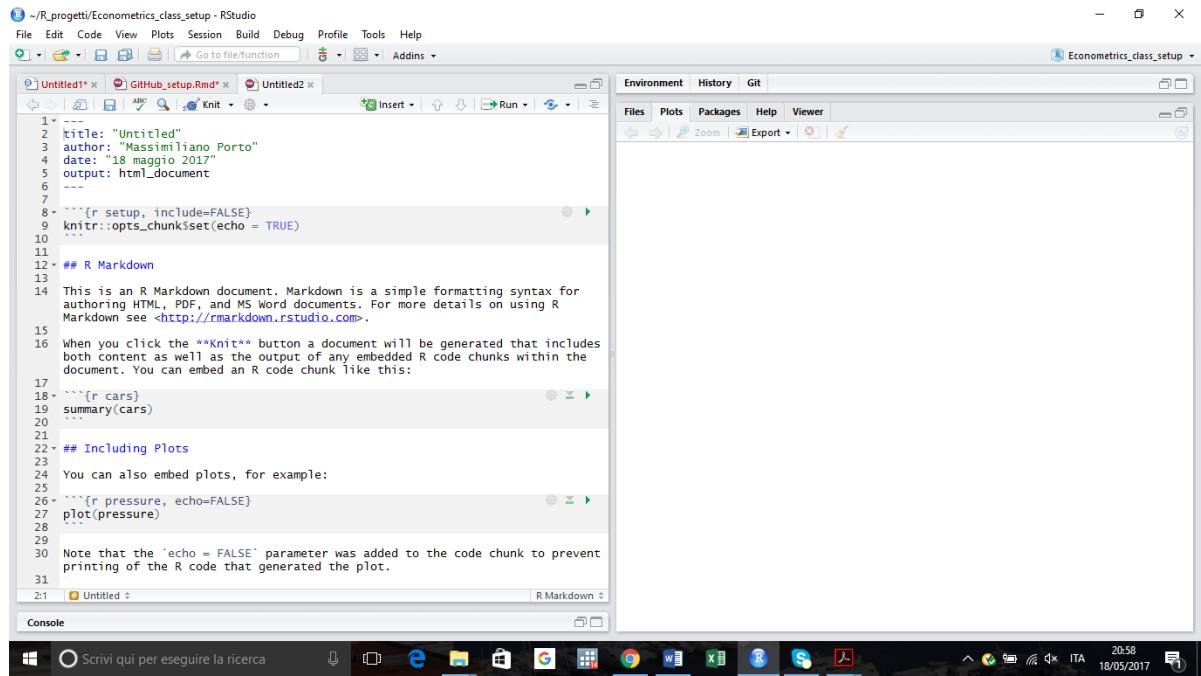


Figure 2.6: Markdown chunk

## 2.2 R Script & R Markdown

The screenshot shows the RStudio interface with the following details:

- Title Bar:** ~/R\_progetti/Econometrics\_class\_setup - RStudio
- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help
- Toolbar:** Go to file/function, Addins
- Code Editor:** Untitled1\* (active), GitHub\_setup.Rmd\*, Untitled2\*
- Code Content:**

```
10
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for
15 authoring HTML, PDF, and MS Word documents. For more details on using R
16 Markdown see <http://rmarkdown.rstudio.com>.
17
18 When you click the **Knit** button a document will be generated that
19 includes both content as well as the output of any embedded R code chunks
20 within the document. You can embed an R code chunk like this:
21
22
23
24 You can also embed plots, for example:
25
26
27
28
29
30 Note that the `echo = FALSE` parameter was added to the code chunk to
31 prevent printing of the R code that generated the plot.
32
33
```
- Output pane:** Shows the output of the R code, including a summary of the 'cars' dataset:

speed	dist
Min. : 4.0	Min. : 2.00
1st Qu.:12.0	1st Qu.: 26.00
Median :15.0	Median : 36.00
Mean :15.4	Mean : 42.98
3rd Qu.:19.0	3rd Qu.: 56.00
Max. :25.0	Max. :120.00
- Environment Tab:** Environment, History, Git
- Files Tab:** Files, Plots, Packages, Help, Viewer
- Bottom Status Bar:** R Markdown

Figure 2.7: Run a chunk

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edi, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Run, Addins.
- Code Editor:** An R Markdown document titled "GitHub\_setup.Rmd". The code includes R Markdown syntax like `## R Markdown` and `###{r cars}`.
- Insert Menu:** A red circle highlights the "Insert" menu icon in the toolbar.
- Knit Button:** A red circle highlights the "Knit" button in the code editor toolbar.
- Output Options:** A dropdown menu is open under the Knit button, showing options: (Use document default), Show output only, Show code and output, Show nothing (run code), and Show nothing (don't run code). The "Show output only" option is selected.
- Environment Tab:** Shows the file structure: Home > R\_progetti > Econometrics\_class\_setup > asset > figures > rstudio.
- Console Tab:** Untitled 1\*.
- Status Bar:** Shows the date and time: 18/05/2017 21:33.

Figure 2.8: Example R Markdown 1

## 2 PART 2

```

313
314 #### Flow control
315
316 R allow to implement some actions only if some conditions are met or allow
317 to implement the same action for a certain number of times. It provides
318 control structures such as ``if/else'', ``for'' and ``while'' loops.
319 Read **Applied Econometrics with R, Chapter 2** for full explanation.
320
321 Let's see here a simple example based on the previous calculation and
322 compute
323
324 ```{r}
325 if(all.equal(0.3, 0.1+0.1)){
326   print("you are right")
327 } else{
328   print("you are wrong")
329 }
330
331
332
333
334
335
336
337
338 Let's print the even number of our vector sbs
339
340 ```{r}
341 for(i in b){
342   if(i%%2 == 0)
343     print(i)
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411

```

Figure 2.9: Example R Markdown 2

```

401 Let's use a linear regression model, using the ```lm()``'' function, to
402 estimate the impact of experience on wage in the dataset *CPS1985*.
403
404 ```{r, eval=FALSE}
405 cps_lm <- lm(log(wage) ~ experience, data = CPS1985)
406
407
408
409 The command ```summary()``'' returns the usual summary of the coefficients
410 (with estimates, standard errors, test statistics, and p values) as well
411 as the associated R2, along with further information (figure 1b).
412
413
414 ![[Summary of regression]](asset/figures/rstudio/rstudio_8.png)
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
255
```

## 2.2.1 LaTeX

Now that we saw how to work in **R Markdown** let's convert the Rmd file in pdf by clicking on **knit** as in figure 2.11.

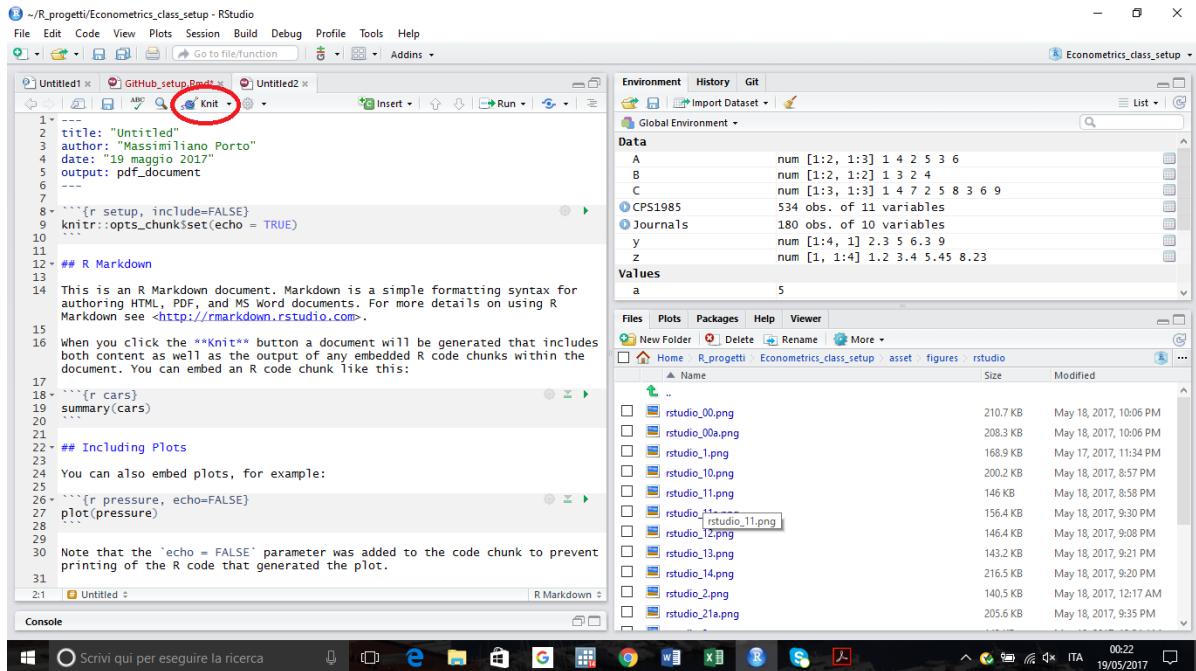


Figure 2.11: Knit

However, before successfully creating your pdf you need to install some packages:

- `bookdown` `install.packages("bookdown")`
- `knitr` `install.packages("knitr")`

Moreover, you need to install LaTeX system. There are different applications, for example:

- TeX Maker (<http://www.xm1math.net/texmaker/>)
- MiKTeX (<https://miktex.org/download>)
- TeX Live (<https://www.tug.org/texlive/>)
- MacTeX for macOS (<https://www.tug.org/mactex/>)

## 2.3 Data management

In this section we will see how to import a dataset in R and simple functions to manage it.

Before starting to work on a dataset, it is important to have a *double look* to the data. You can see your dataset with the function `View()`. You can use `head()` or `tail()` if you want to see only the first or the last entries.

## 2 PART 2

Let's see an example. We will have a look to the dataset *Journals* in the package *AER*.

```
data("Journals", package = "AER")
View(Journals)
head(Journals)

##                                     title
## APEL                  Asian-Pacific Economic Literature
## SAJoEH      South African Journal of Economic History
## CE                      Computational Economics
## MEPiTE MOCT-MOST Economic Policy in Transitional Economics
## JoSE          Journal of Socio-Economics
## LabEc        Labour Economics
##             publisher society price pages charpp citations
## APEL           Blackwell    no   123   440   3822    21
## SAJoEH So Afr ec history assn    no    20   309   1782    22
## CE                 Kluwer    no   443   567   2924    22
## MEPiTE           Kluwer    no   276   520   3234    22
## JoSE            Elsevier    no   295   791   3024    24
## LabEc           Elsevier    no   344   609   2967    24
##         foundingyear subs          field
## APEL           1986    14       General
## SAJoEH        1986    59 Economic History
## CE              1987    17     Specialized
## MEPiTE        1991     2 Area Studies
## JoSE            1972    96 Interdisciplinary
## LabEc          1994    15       Labor

tail(Journals)

##                                     title
## JASA      Journal of the American Statistical Association
## JoFi          Journal of Finance
## QJoE          Quarterly Journal of Economics
## JoPolEc        Journal of Political Economy
## Ecnmt          Econometrica
## AER          American Economic Review
##             publisher society price pages charpp citations
## JASA     Am. Statistical Assn    yes   310   1260   5664    2800
## JoFi     Am. Finance Assn    yes   226   2272   3036    3791
## QJoE          MIT press    no    148   1467   2184    4138
## JoPolEc Univ of Chicago Press    no    159   1669   2640    6697
## Ecnmt          Blackwell    yes   178   1482   2992    7943
## AER          Am Ec Assn    yes     47   1867   3900    8999
##         foundingyear subs          field
```

```
## JASA          1971  487 Econometrics
## JoFi         1945  799 Finance
## QJoE         1886  660 General
## JoPolEc     1892  737 General
## Ecnmt        1932  346 General
## AER          1911 1098 General
```

### 2.3.1 Class and structure

The second look we were talking about is an introspective look to the data through functions `class()` and `str()`.

Through these two functions we can have a better understanding of the type of object we are working with. This is very important because some functions simply don't work with some objects. In this case we have to change their class. We will see some examples later.

Now let's see some examples of `class()`. This function can be applied to any object. For example, let's check the class of some objects we created earlier

```
class(a)
## [1] "numeric"
class(A)
## [1] "matrix"
```

We have a *numeric* and *matrix* class. To put it easily, we have a "number" and a "matrix" object.

But observe the following

```
c <- "3"
class(c)
## [1] "character"
```

It has a *character* class. We see that it is a number, a 3. Let's try what happens if we add `a + c`. We could expect a 8. But

```
a + c
```

we get an error. Again to put it simple, even if the object `c` stores a 3 it has no *numeric* class, i.e. it is not a number.

Since we work with dozens of variables and thousands of observations it is clear that it is fundamental to know what objects we are working with.

Let's go back to dataset. Let's check class for Journals.

```
class(Journals)
## [1] "data.frame"
```

We have a `data.frame` objects that share many of the properties of matrices and lists.

Let's see what information provide the `str()` function.

## 2 PART 2

```
str(Journals)
## 'data.frame': 180 obs. of 10 variables:
## $ title      : chr "Asian-Pacific Economic Literature" "South African Journ...
## $ publisher   : Factor w/ 52 levels "ANU Press","Academic Press",...: 11 45 28...
## $ society     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
## $ price       : int 123 20 443 276 295 344 90 242 226 262 ...
## $ pages        : int 440 309 567 520 791 609 602 665 243 386 ...
## $ charpp      : int 3822 1782 2924 3234 3024 2967 3185 2688 3010 2501 ...
## $ citations   : int 21 22 22 22 24 24 24 27 28 30 ...
## $ foundingyear: int 1986 1986 1987 1991 1972 1994 1995 1968 1987 1949 ...
## $ subs         : int 14 59 17 2 96 15 14 202 46 46 ...
## $ field        : Factor w/ 24 levels "General","Economic History",...: 1 2 3 4 ...
```

It provides the complete structure of the object. We can also select only one variable of the object. For example

```
str(Journals$price)
```

```
##  int [1:180] 123 20 443 276 295 344 90 242 226 262 ...
```

```
str(Journals$society)
```

```
##  Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
```

We see that price is recorded as `int`, i.e. integer, while society as factor, i.e. a category type.

Let's see some other examples with database available on R.

```
data("AirPassengers")
```

```
class(AirPassengers)
```

```
## [1] "ts"
```

```
data("EuStockMarkets")
```

```
class(EuStockMarkets)
```

```
## [1] "mts"    "ts"    "matrix"
```

`AirPassengers` returns a `ts` object, i.e. a time series object, while the class for `EuStockMarkets` says that the object is a multiple time series object.

You can encounter other class types. We will see others later.

For the moment we checked the class of objects that already exist and are available on R. What about the class of the dataset we built?

First let's see how to import it in R.

### 2.3.2 Import dataset

I downloaded Purchasing Power Parity (PPP) data from OECD database. To replicate download from OECD-library the PPP for GDP, year frequency from 1999, for Argentina, Australia, Brazil, Canada, Chile, China, Colombia, Czech Rep., Denmark, EU19, Great Britain, Hungary, India, Indonesia, Israel, Japan, Korea, Norway, New

## 2.3 Data management

Zealand, Poland, Russia, South Africa, Sweden, Switzerland, Turkey or just take it from my GitHub page [https://github.com/massimilianoj/R\\_beginners](https://github.com/massimilianoj/R_beginners).

```
ppp <- read.table("~/R_progetti/Portfolio_management_pmwr/currency_model_dataset/ppp.csv",
  header = TRUE, sep = ",")
```

I used `read.table()` to import it (package "utils"). Note that

- `header = TRUE` means that the first row of the database contains the title of our columns;
- because my file is a csv file (comma separated value) I need to indicate the kind of separator of the file as `sep = ", "`.

There are other functions you can use to import a dataset. For example

```
ppp2 <- read.csv("~/R_progetti/Portfolio_management_pmwr/currency_model_dataset/ppp.csv",
  header = TRUE, sep = ",")
```

Another way is to click on **Import Dataset** button as in figure 2.12

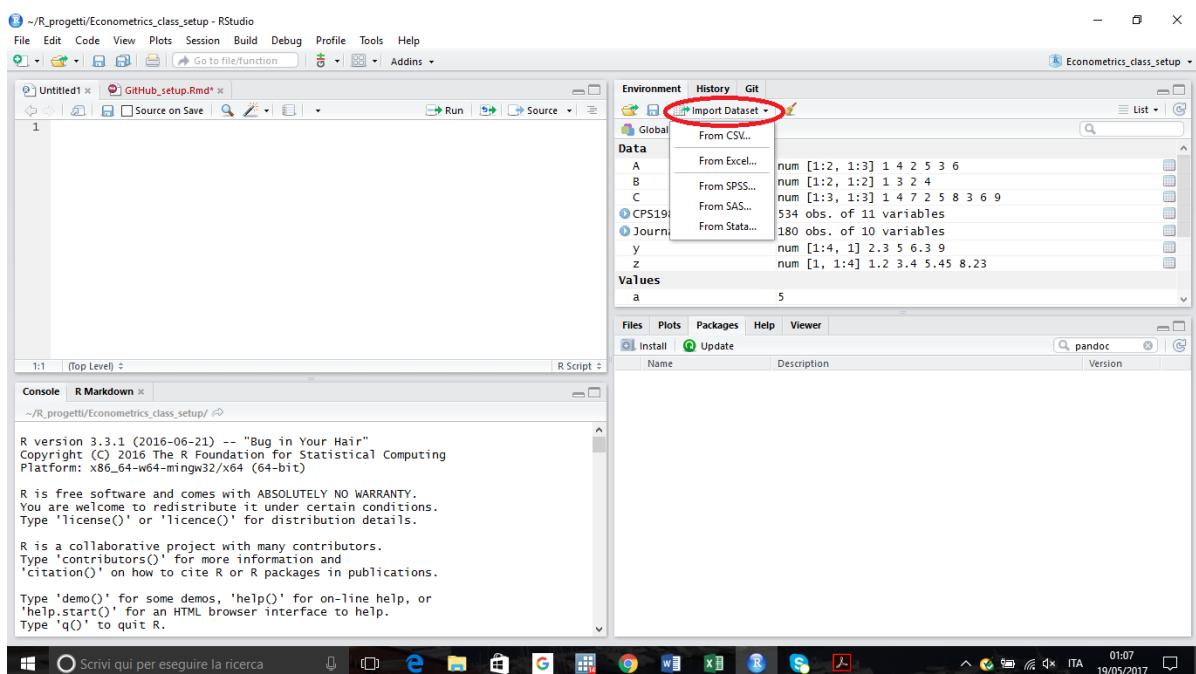


Figure 2.12: Import Dataset

Let's see how to import from a **CSV** file (figures 2.13 & 2.14)

You can see from figure 2.14 that I used another library, `readr` and another function, `read_csv()`. I named this dataset `ppp3`.

The three dataset, at first sight, look the same. Let's check class and structure.

```
class(ppp)
## [1] "data.frame"
```

## 2 PART 2

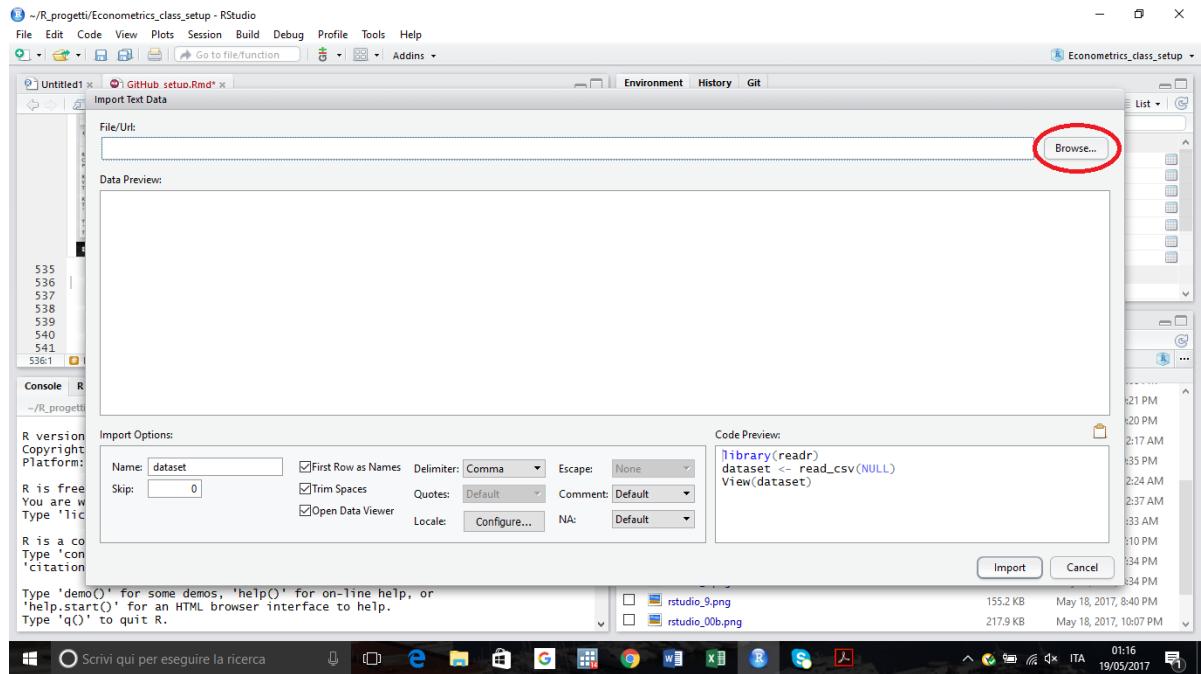


Figure 2.13: Import from CVS file (1)

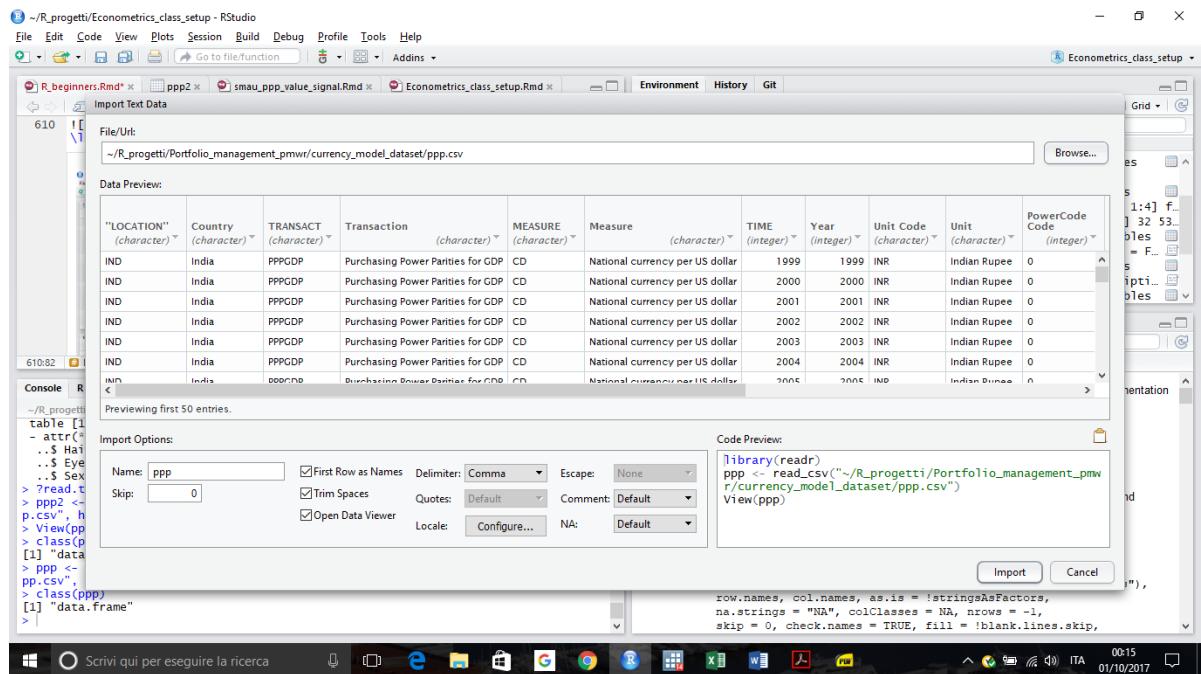


Figure 2.14: Import from a CVS file (2)

```

class(ppp2)
## [1] "data.frame"
class(ppp3)
## [1] "tbl_df"     "tbl"        "data.frame"
The first two databases return a data.frame object, while the third one returns
"tbl_df"     "tbl"        "data.frame" that is a tibble data.frame. For more info
about tibble, write ?tibble in the console pane.

```

From here onwards the example will be provided with ppp database.

### 2.3.3 Simple operations on dataset

#### 2.3.3.1 Select and rename columns

First, let's select the columns of interest and rename them.

```

# select column of interest
ppp <- ppp[, c(1, 8, 15)]
colnames(ppp) <- c("iso_code", "year", "ppp")

```

In the first expression we are saying to R: “select all the rows and columns 1, 8 and 15”.<sup>3</sup> In the second expression we renamed the titles of the columns of our database.

#### 2.3.3.2 Transform database long-wide

Have a look again at the database (Fig. 2.15).

You see that we have a “long” format, while we want it wide, with each country on a different column with its ppp value per year. I will use spread() function (tidyR package) to rearrange it.<sup>4</sup>

```

# transform the dataset wide
ppp_w <- spread(ppp, iso_code, ppp)

```

Let's see the *new* database.

It is what we wanted. But what did we do? In the function spread - the first input is the database we use; - the second input is the key, i.e. the column that will be spread as column title; - the third input is the value, i.e. what we want to put as value under the new columns we created.

In general, a good way to see briefly the structure of a function is to use the function args(). For example

---

<sup>3</sup>If we had written ppp <- ppp[c(1, 8, 15), ] we would have said to R to select only rows 1, 8 and 15 and all the columns.

<sup>4</sup># is used to write comments in the chunk; what follows will not be run by R.

2 PART 2

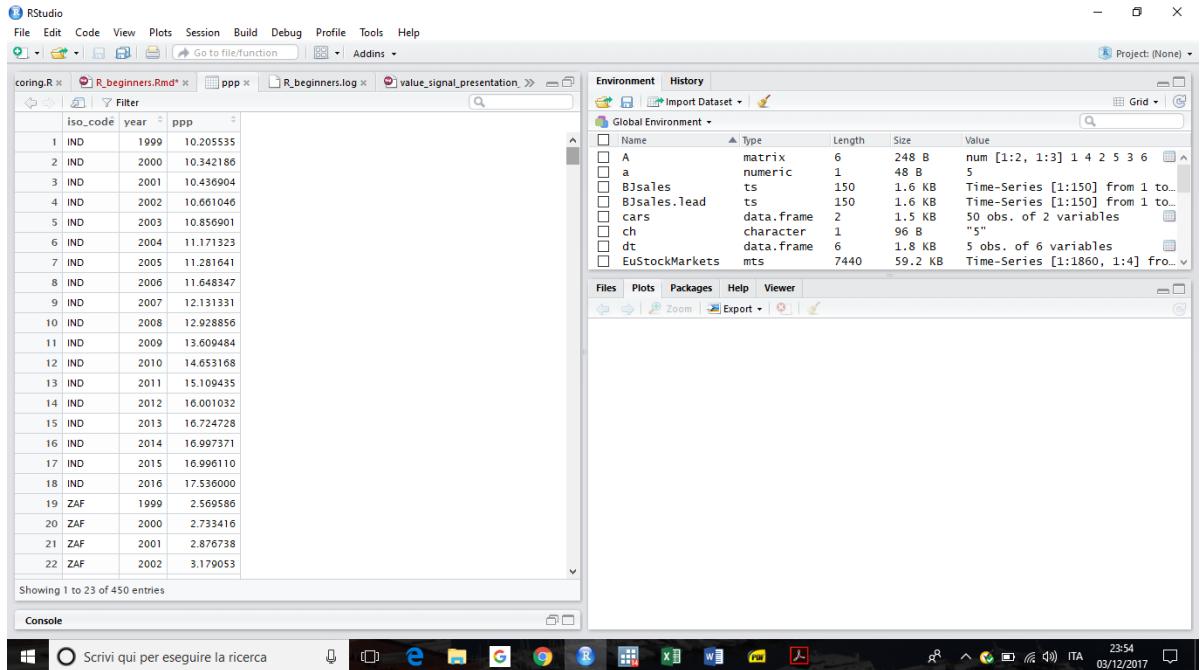


Figure 2.15: Long database

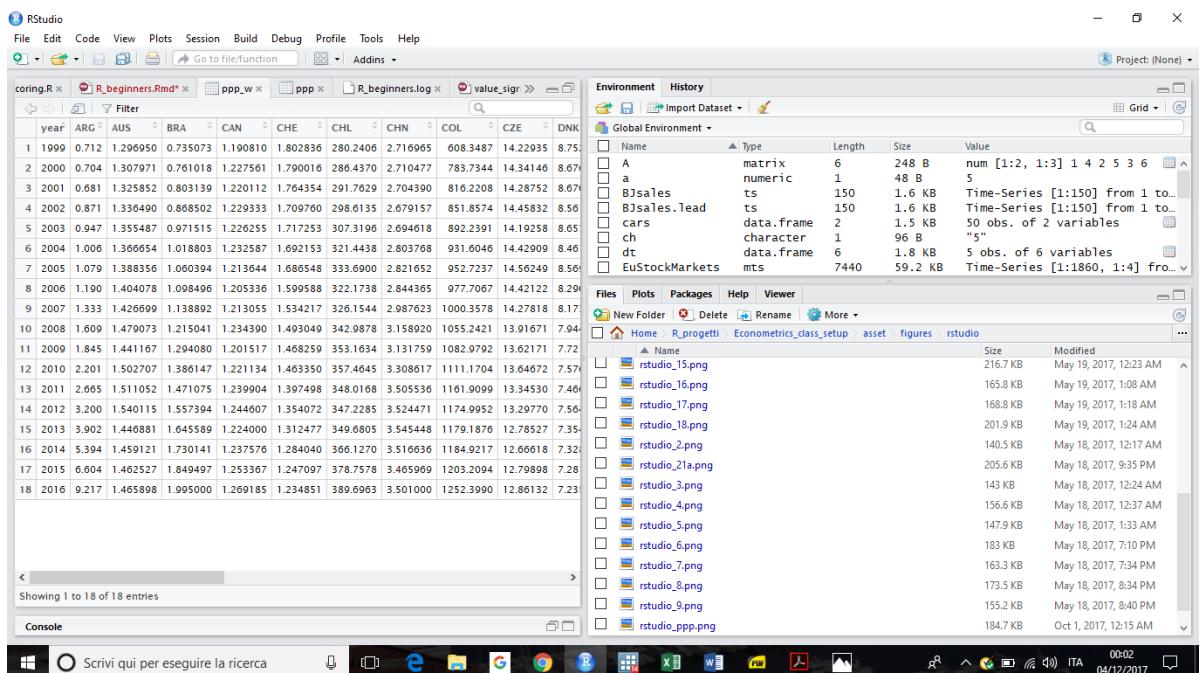


Figure 2.16: Wide database

```
args(spread)

## function (data, key, value, fill = NA, convert = FALSE, drop = TRUE,
##           sep = NULL)
## NULL
```

### 2.3.3.3 Rename single columns and drop variables

Next simple operations are to rename a single column and drop some columns. For example, we want to rename only the column for the eurozone and drop ARG, AUS and NZL from the database. Note that we used a - before the corresponding column number to drop it.

```
# rename column for the eurozone EMU
colnames(ppp_w)[which(colnames(ppp_w) == "EA19")] <- "EMU"

# drop ARG, AUS, NZL
ppp_w <- ppp_w[c(-2, -3, -21)]
```

What about if we want to go from wide to long? Let's do it

```
ppp_l <- gather(ppp_w, iso_code, ppp, BRA:ZAF)
```

Intuitively, to go back we need a column that will contain all the countries, iso\_code and another column, ppp, that will contain all the ppp values per country. BRA:ZAF is the range of column that could be selected also one by one by using c().

### 2.3.3.4 Operations only on some columns of the database

Suppose that we want to make our database relative to EMU ppp. We want to divide all the columns by EMU but not year.

```
# create a new object ppp_to_EMU equal to ppp_w We implement this step
# because we don't want the following modification to affect our ppp_w for
# following examples
ppp_to_EMU <- ppp_w
ppp_to_EMU[, c(2:23)] <- ppp_to_EMU[, c(2:23)]/ppp_to_EMU$EMU
```

You can see that the year column is not affected by the operation (Fig. 2.17)

### 2.3.3.5 Subset the database

Still working on ppp\_to\_EMU. Let's suppose we are interested only in the period between 2007 and 2012. We subset the database using subset() (Fig. 2.18). Note that you can subset a database using other functions such as window() or filter()

```
ppp_EMU_subset <- subset(ppp_to_EMU, year >= "2007" & year <= "2012")
```

## 2 PART 2

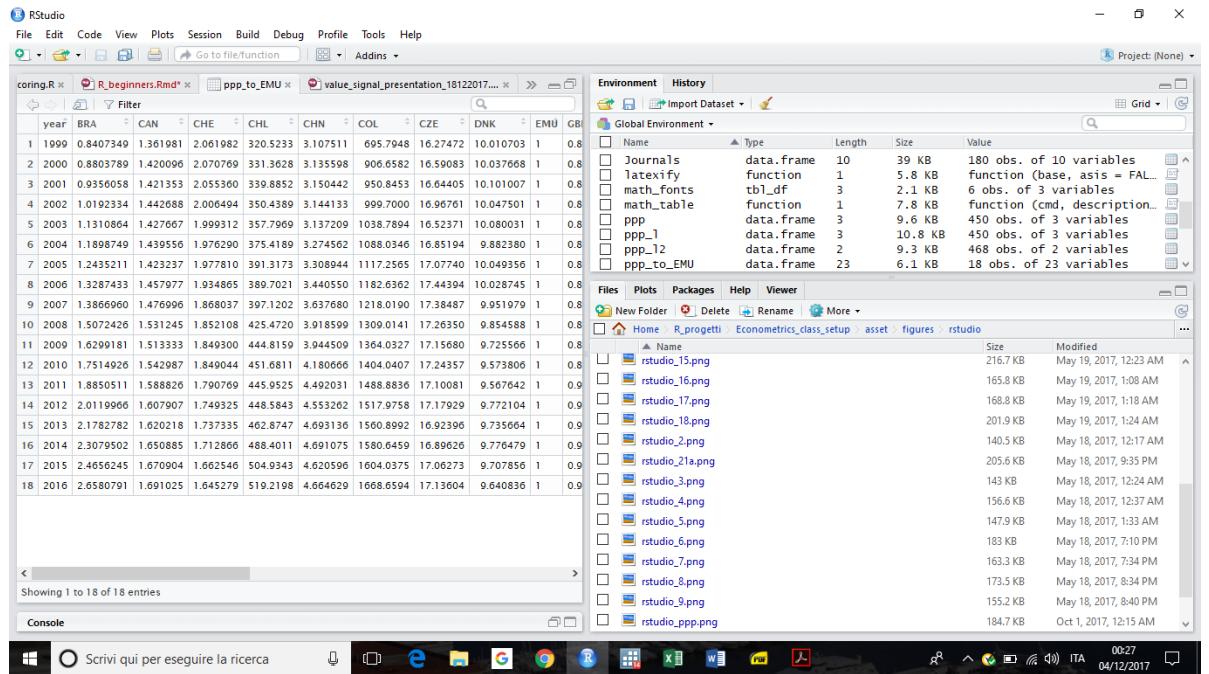


Figure 2.17: Wide database

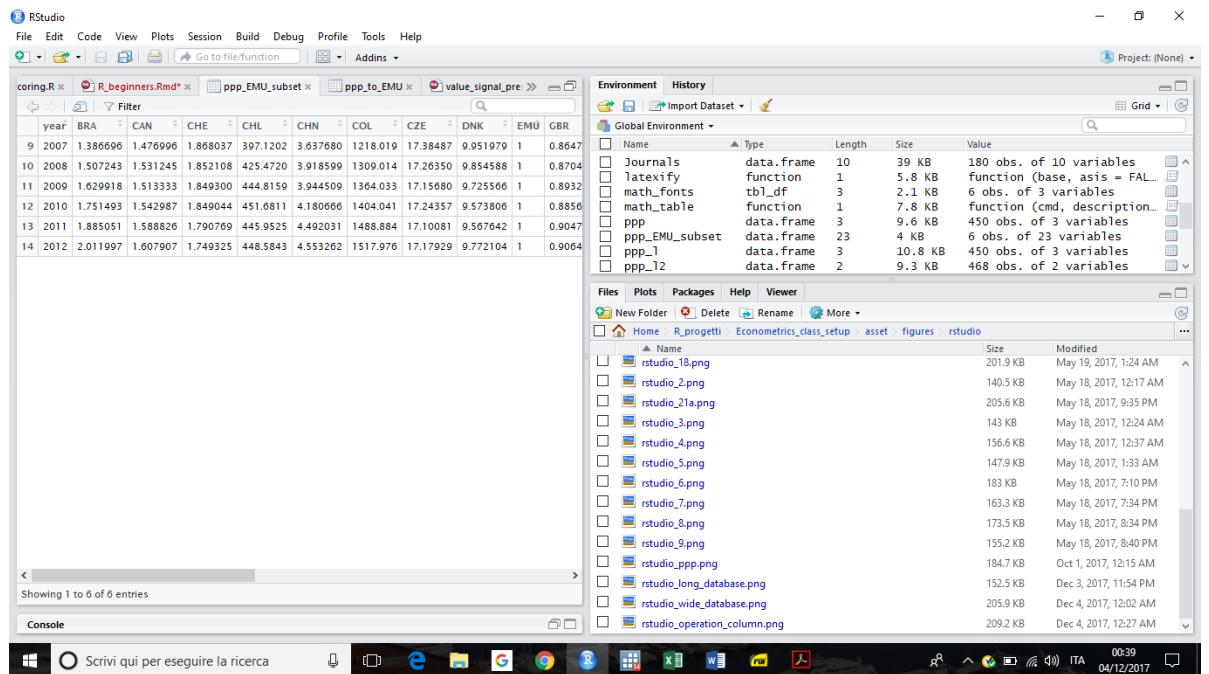


Figure 2.18: Wide database

### 2.3.3.6 Merge two databases

First let's import gdp from OECD and following the previous steps to arrange it.

```
gdp <- read.table("~/R_progetti/Portfolio_management_pmwr/currency_model_dataset/gdp.csv"
  header = TRUE, sep = ",")
```

```
gdp <- gdp[, c(1, 8, 15)]
colnames(gdp) <- c("iso_code", "year", "gdp")
gdp_w <- spread(gdp, iso_code, gdp)
colnames(gdp_w)[which(colnames(gdp_w) == "EA19")] <- "EMU"
gdp_w <- gdp_w[, c(1, 5, 6, 7, 8, 9, 10, 11, 13, 14, 19, 21, 22, 23, 26, 28,
  29, 34, 36, 38, 41, 42, 44)]
gdp_w <- subset(gdp_w, year >= "1999" & year <= "2016")
gdp_l <- gather(gdp_w, iso_code, gdp, BRA:ZAF)
```

Now we have two databases, ppp\_l and gdp\_l, that share same year and iso\_code. We can merge them as follows (Fig. 2.19):

```
merge_data <- merge(ppp_l, gdp_l, by = c("year", "iso_code"))
```

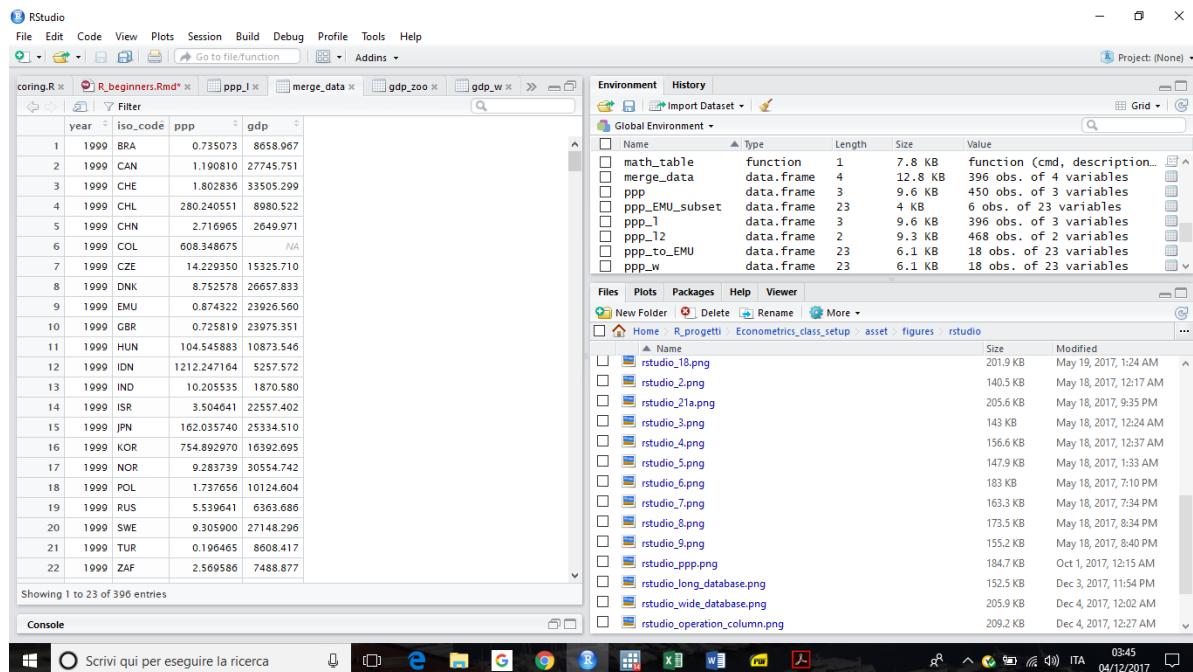


Figure 2.19: Merged database

With this example we conclude this section on how to build a database.

## 2.4 Dates in R

Let's say few words about *dates* in R. Let's use ppp\_w as example. Its frequency is annual. If we check the class for year

```
class(ppp_w$year)
```

```
## [1] "integer"
```

we observe that it is an integer and not a date class. To put it simple, for R a date is composed of year, month and day. We can extract a date from our ppp\_w\$year as follows using zoo package:

```
# create a new column named date
ppp_w$date <- as.Date.yearmon(ppp_w$year)
class(ppp_w$date)

## [1] "Date"
```

Let's make another example with a database with a monthly frequency. Let's import the cpi dataset from the OECD with monthly frequency.

```
cpi <- read.table("~/R_progetti/Portfolio_management_pmwr/currency_model_dataset",
  header = TRUE, sep = ",")
```

```
cpi <- cpi[, c(3,9,17)]
colnames(cpi) <- c("iso_code", "month", "cpi")
class(cpi$month)
```

```
## [1] "factor"
```

In this case we have a factor class for month. Let's make it date and then extract also the quarter.

```
# make month as a date class
# create a new column named quarter
cpi$date <- cpi$month
cpi$month <- as.Date(as.yearmon(cpi$month))
class(cpi$month)

## [1] "Date"

cpi$quarter <- as.yearqtr(cpi$month)
```

The yearmon() returns a date in the default format year-month-day, where day is the beginning of the month. We could decide we want a specific day of the month instead. We can make it manually as follows

```
cpi$date <- paste(cpi$date, "15", sep = "-")
```

Note that in this case we have a character class

```
class(cpi$date)
```

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

that we can easily convert into date having the year-month-day format

```
cpi$date <- as.Date(cpi$date)
class(cpi$date)
```

## [1] "Date"

An advance package to manage dates is `lubridate()`. You can read more about `lubridate()` here <https://cran.r-project.org/web/packages/lubridate/vignettes/lubridate.html>

We conclude this section with a remark on the format of the date:

1. Four digit year is '
2. two digit year is '
3. numeric month is '
4. alphabetic (abbreviated) month is '
5. alphabetic (full) month is '
6. day is '

## 2.5 Plot

A trick to plot all the columns of a database is to make it as zoo class.

For example, let's plot the GDP rom `gdp_w`. First let's make it as zoo. Notice that `zoo` put the column with the dates at the place of the rows numbers (Fig. 2.20)

```
gdp_zoo <- read.zoo(gdp_w)
```

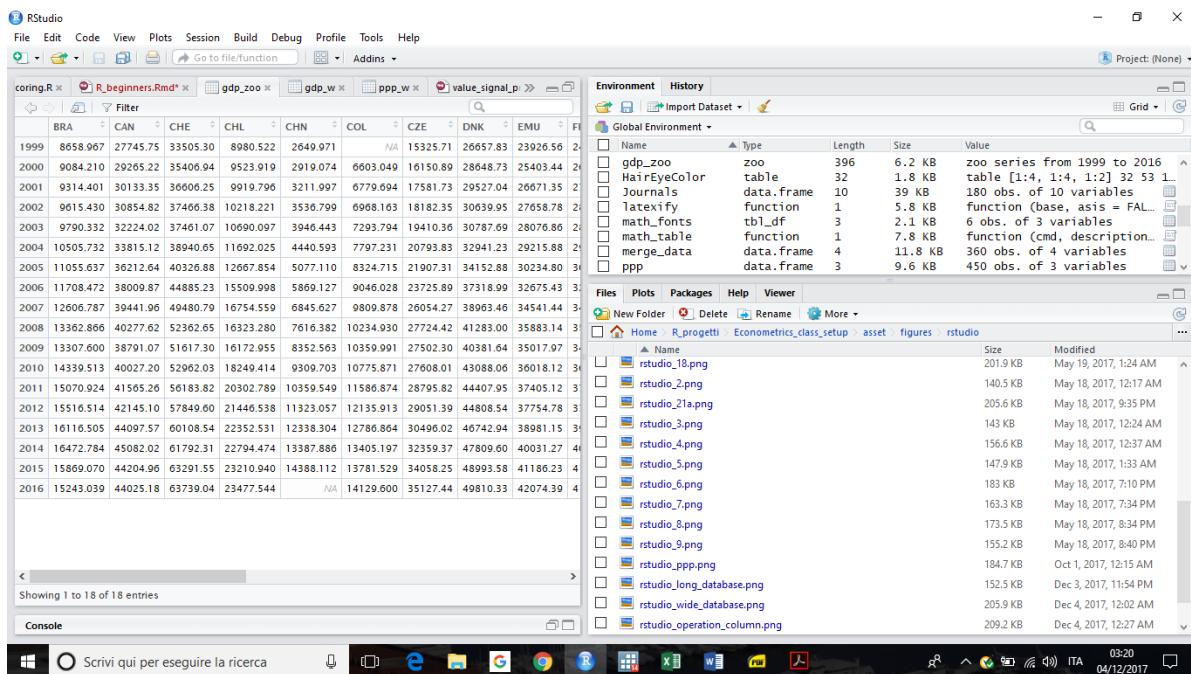


Figure 2.20: Zoo database

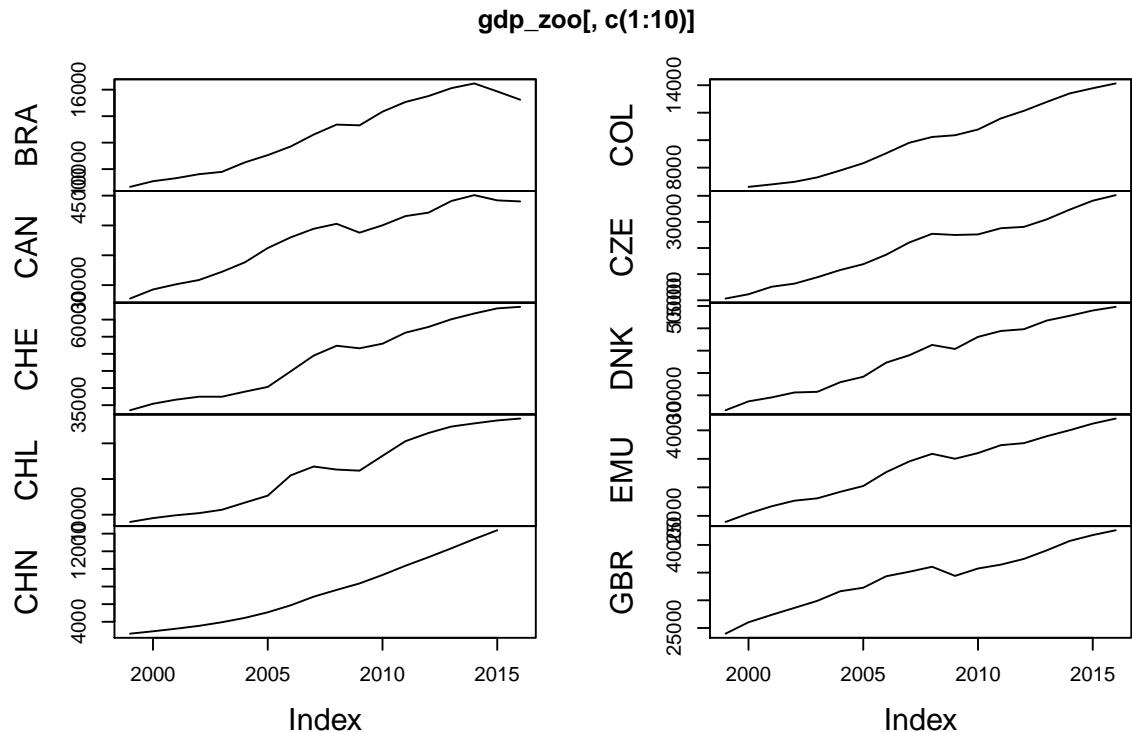
**Remark:** given that we have an annual frequency we didn't transform the year in

## 2 PART 2

Date class because it can be plotted as integer (a constant 1 integer difference).

Second, just plot the database as follows (because we have many countries we just selected the first 10):

```
plot(gdp_zoo[, c(1:10)])
```

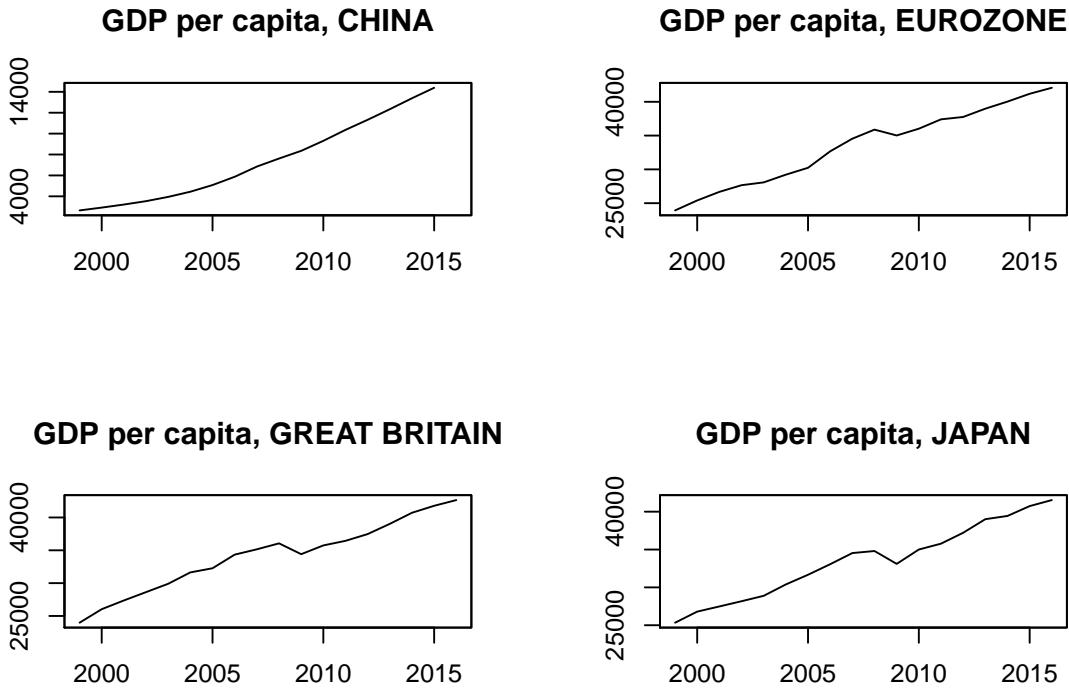


This is a quick way to have a look to the plot. Now let's see how to make nice plots.

### 2.5.1 Simple plot

Let's plot CHN, EMU, GBR and JPN.

```
par(mfrow = c(2, 2))
CHN_plot <- plot(gdp_zoo$CHN, main = "GDP per capita, CHINA", xlab = "", ylab = "")
EMU_plot <- plot(gdp_zoo$EMU, main = "GDP per capita, EUROZONE", xlab = "", ylab = "")
GBR_plot <- plot(gdp_zoo$GBR, main = "GDP per capita, GREAT BRITAIN", xlab = "", ylab = "")
JPN_plot <- plot(gdp_zoo$JPN, main = "GDP per capita, JAPAN", xlab = "", ylab = "")
```

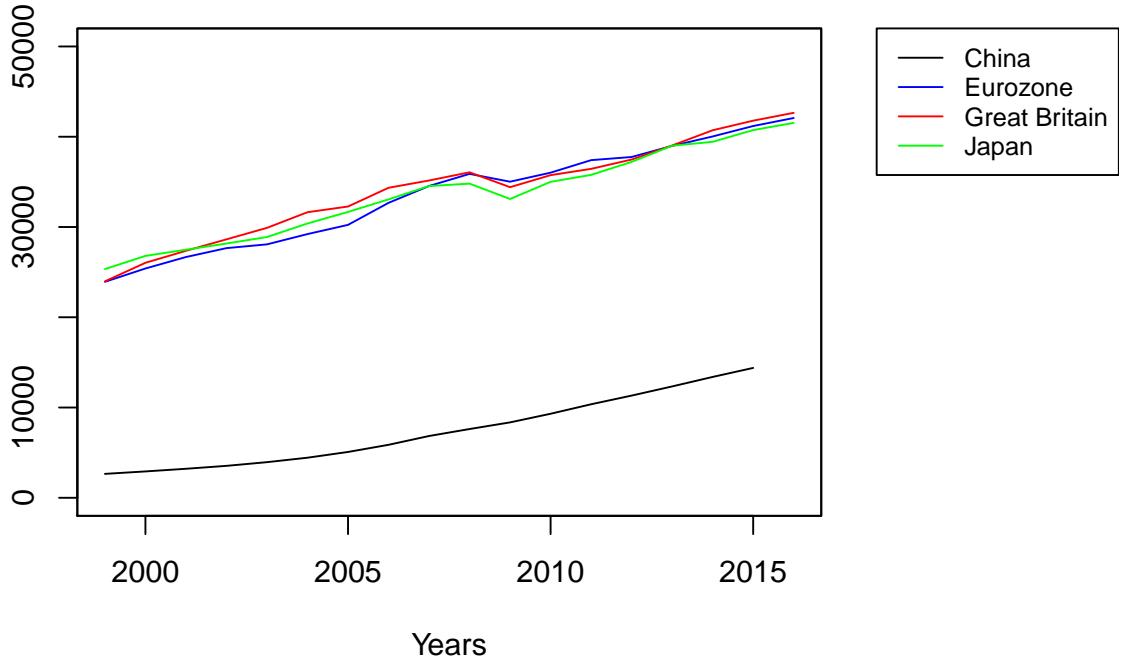


```
par(mfrow = c(1, 1))
```

Note that `par(mfrow = c())` sets the display of the plots. In this case, we said to R to display the plots on two rows and two columns. At the end, we set again the default, one plot; in `main =` we write the title of the plot; we set `xlab = ""`, `ylab = ""` equal to "" to prevent R from creating default title for the axis.

Now let's plot all together.

```
par(xpd = T, mar = par()$mar + c(0, 0, 0, 6))
plot(CHN_plot, type = "l", col = "black", lty = 1, ylim = c(0, 50000), ylab = "", 
     xlab = "Years")
lines(EMU_plot, type = "l", col = "blue", lty = 1)
lines(GBR_plot, type = "l", col = "red", lty = 1)
lines(JPN_plot, type = "l", col = "green", lty = 1)
legend("topright", inset = c(-0.4, 0), legend = c("China", "Eurozone", "Great Britain",
    "Japan"), col = c("black", "blue", "red", "green"), cex = 0.8, xpd = TRUE,
    lty = 1)
```



```
par(mar = c(5, 4, 4, 2) + 0.1)
```

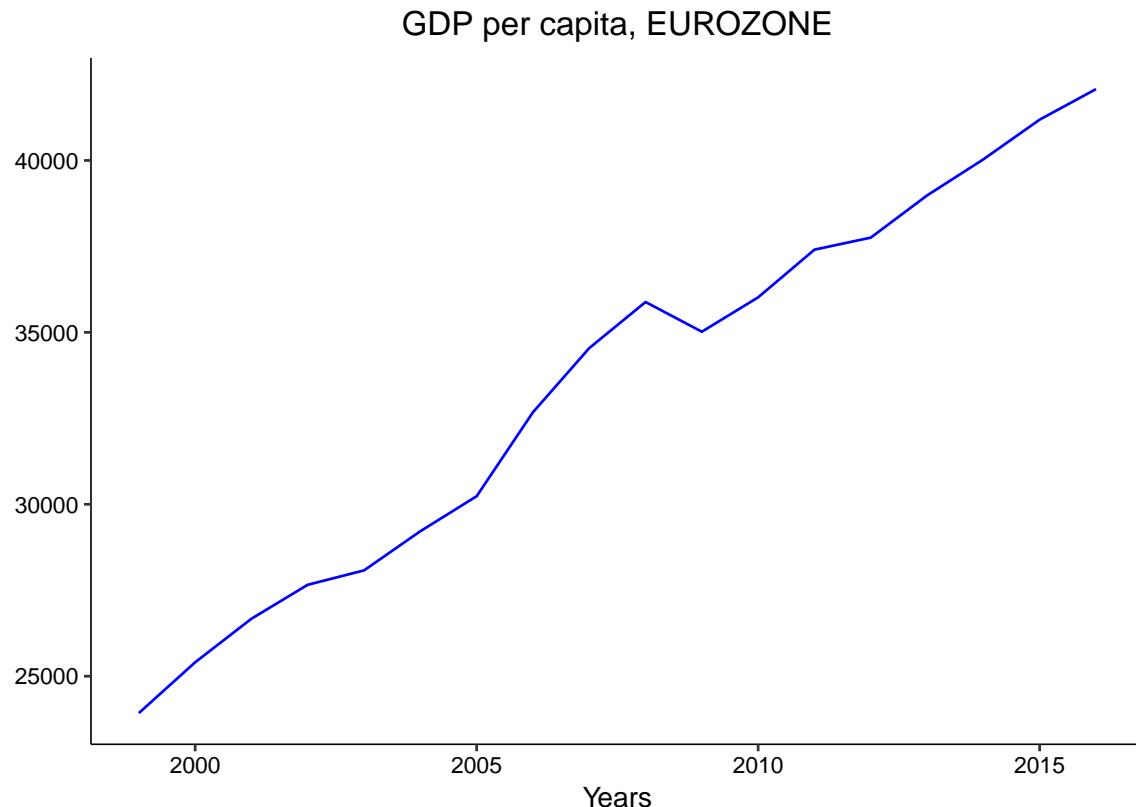
Note that the first and last expression sets the margin to put the legend out of the plot area. `type`, `lty`, `cex` are arguments for making the plot fancier. For details read **Keliber & Zeiles, p. 43.**

## 2.5.2 GGPLOT

Following two examples with GGPLOT.

In the first example we plot the GDP per capita for the Eurozone. First, in the `ggplot` function we set the database, the value for x axis (in this case years) and for y axis (in this case GDP per capita values) in `aes()`. Second, we decide for the the kind of plot, in this case line. All other options make the plot fancier. Try to include and exclude them to see the difference.

```
ggplot(gdp_w, aes(x = gdp_w$year, y = gdp_w$EMU)) + geom_line(size = 0.5, col = 'black') + labs(x = "Years", y = "", title = "GDP per capita, EUROZONE") + theme(plot.title = element_text(size = 16, color = "black"), panel.background = element_rect(fill = "white"), axis.line = element_line(size = 1, color = "black"), axis.text = element_text(colour = "black"), panel.grid = element_rect(linetype = "dashed", color = "black"))
```



In this second example we plot together the GDP per capita for China, Eurozone, Great Britain and Japan as a bar plot. First we make a new dataframe, `gdp_df`, that contains `year_gdp`, a vector that replicates the year 4 times (because of the number of countries) and `country_gdp`, a vector of GDP per capita for each country. `type` replicates each country for the number of years. Second, we put these new data into the `ggplot` function. Now our database is `gdp_df` and values for x axis and y axis are, respectively, `year_gdp` and `country_gdp`. Third, we choose the kind of plot, bar, and add a number of options to make it fancier. Again, try to include and exclude them to see what changes.

```
year_gdp <- rep(gdp_w$year, 4)
CHN_gdp <- gdp_w$CHN
EMU_gdp <- gdp_w$EMU
GBR_gdp <- gdp_w$GBR
JPN_gdp <- gdp_w$JPN
country_gdp <- c(CHN_gdp, EMU_gdp, GBR_gdp, JPN_gdp)
type <- c(rep("China", 18), rep("EUROZONE", 18), rep("Great Britain", 18), rep("Japan", 18))
gdp_df <- data.frame(year_gdp, country_gdp)
graph_gdp <- ggplot(gdp_df, aes(x = year_gdp, y = country_gdp)) + geom_bar(stat = "identity")
aes(fill = type), position = position_dodge()) + scale_fill_brewer(palette = "Paired")
theme_classic() + xlab("") + ylab("US dollars") + ggtitle("GDP per capita") +
```

## 2 PART 2

