Tutorial 1: Getting Started with R and R Studio

Code

Ana Morales-Gomez

2020-01-29

1. Introduction

This is the first tutorial of the workshop. This is for you to familiarise yourself with R, R Studio and the console.

2. Installing R and R Studio

The PCs in the university cluster have R and R studio already installed, so this task is for installing these software packages **in your own computer or laptop**. You can skip this section if you already have R and R studio installed.

We will be using R Studio, but R studio can only work if R is already installed, so we need to have both software installed.

2.1. Installing R

R is free and can be installed in Windows, Mac, Linux. Go to the R project website (https://www.r-project.org/). Once there click on **Download R** and select a CRAN Mirror. For the UK you can choose between the University of Bristol Imperial College London, either of them would do the job.

Now, follow the instructions and you will have R installed in your own computer.

2.2. Installing R Studio

Go to the R Studio website (https://rstudio.com/products/rstudio/download/), choose the free version and download the file according to your operating system.

Once you have installed both R and R Studio, you don’t need to interact at all with R. You can do everything you need from within R Studio.

3.Getting started with R Studio

3.1. Working directory

Set working directory (ideally where the data are saved)

```
setwd("copy/the/Path/to/your/folder/here")
```

For example, in my PC it will look like this (it won't work in your PC because the path is different)

```
setwd("C:/Users/Anita/UKDS/NCRM workshop")
```

If you are unsure where your current working directory is, you can use the function `getwd()`, which literally means: "get working directory". The working directory will be printed in your console.

```
getwd()
```

3.2. Installing packages

Installing packages can be done by coding or clicking on the "Tools" tab of the R Studio bar. You only need to install packages once (unless you change computer).

The code for installing packages is as follows:

```
#one time instalation
install.packages('tidyverse')
install.packages('haven')
install.packages("foreign")

# You can also install more than one package at a time, adding a letter "c" f
or concatenate before the names of the packages

install.packages(c("tidyverse", "haven"))
```

3.3. Loading packages

Installing packages is the first step, but we still need to add another step to actually be able to use them. Unfortunately, this has to be done each time you start a new session in R or R Studio.

Every time that you start a new data analysis project, you need to load the packages that you are going to use. You don't need to add them all at once (sometimes we realise that we need a package that we haven't loaded into the session), but it is a good practice to start your script calling the packages that you want to use.

For this we use the function `library` as we are actually calling the library of our computer to use a package, as we do with books.

```
# Loading packages (load the packages in every session!)

library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------- tidyvers
e 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ---------------------------------------------- tidyverse_conf
licts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(haven)
library(foreign)
```

3.4. Data types and structures

Creating variables (vectors) of length 1, using the assign arrow

```
a <- 3
b <- 5
c <- 9
```

We can perform basic operations with variables

```
b*c
```

```
## [1] 45
```

We can assign the result to a new variable called ‘d’

```
d <- b*c/a
```

and then, see the value of the variable d

```
d
```

```
## [1] 15
```

4. Loading/Importing data into R

The function we use to import data into R will depend on the format in which the data is. Data in `.txt` or `.csv` can be installed using base R functions (don’t need to use any package), but data in Stata, SPSS or SAS format, to name a few, need a special package.

Today we will be using the package readr (https://cran.r-project.org/web/packages/readr/readr.pdf) which is part of the `tidyverse` family, to import delimited files, such as txt and csv or excel files.

In the next practical we will use the package haven (https://cran.r-project.org/web/packages/haven/haven.pdf) to import a Stata data file.

Each of these packages has a specific function to read a specific dataset. You can check the package documentation (linked above) or a simple Google search to get your answer.

In this example we used the function `read_csv` of the package 'readr' to read a .csv file. The data we will be loading is open data from the Police crime record for Manchester. This data can be accessed on the police.uk (https://data.police.uk/data/) website.

```
gmpolice <- read_csv("data/ManchesterPolice.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   `Crime ID` = col_character(),
##   Month = col_character(),
##   `Reported by` = col_character(),
##   `Falls within` = col_character(),
##   Longitude = col_double(),
##   Latitude = col_double(),
##   Location = col_character(),
##   LSOA_code = col_character(),
##   LSOA_name = col_character(),
##   Crime_type = col_character(),
##   `Last outcome category` = col_character(),
##   Context = col_logical()
## )
```

Alternative you can use the click option from the "Files" tab on the lower-right side of the screen.

5. Pipes

Pipes are meant to make the coding easy to write and read. It writes the code following a logical set of instructions. This is an example of how the pipes would work. I'll show more examples during the exercise.

If you want to have a look at more examples on how to use pipes, you can check chapter 18 of the free book R for data Science (https://r4ds.had.co.nz/pipes.html)

```
group_summary<- gmpolice %>%    # use the dataset called 'gmpolice'
  group_by(Crime_type) %>%      # Now group it by the variable 'Crime_type' o
f
  summarise(crime_count = n())   #Create an aggregated variable: crime_count
```