E401: Machine Learning for Economic Data

First Steps with R

Fall 2023

08/24/2023

In this tutorial session, we will start to get familiar with the statistical software package R. The main purpose of this handout is to get a first glimpse at what R can do and to make sure everything works smoothly on your computer. We will get back to many features in more detail over the course of the semester.

Please answer the following questions. I recommend working in groups of two or three students. A useful reference on essential R commands can be found here:

https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf.

Often however, you'll find an answer fastest by simply googling your problem or searching on Stackoverflow (see below).

Part 0: Installation of R & RStudio

This part is not necessary if you are working in any of IU's computer labs. R is a collection of powerful programs for statistical analysis. RStudio is merely an editor/GUI that facilitates interacting with R. You can download R for free for almost any operating system from this website: http://ftp.ussg.iu.edu/CRAN/ or any other mirror of CRAN (Comprehensive R Archive Network). RStudio can be downloaded for free from this website: https://www.rstudio.com/products/rstudio/download/.

Part 1: R & RStudio

1. Familiarize yourself with RStudio's user interface. The most important windows are the console, the environment (or workspace) window and the R-script editor. Discuss with your partner what each of these windows contains.
2. There are two ways to execute R code: Either by typing commands directly into the console or by writing and executing an R-script. While the console is great for experimenting with commands, we generally prefer to collect all commands in a script (or several scripts). Why?
3. Create a new R-script (the equivalent of a do-file in Stata or a .py-script in Python). Scripts usually consists of two types of lines: Actual commands and comments that

are ignored by the computer but explain the code to humans. Find out how to write comments in your R-script. It is strongly recommended to structure and explain your code using comments. Insert just a single command into your R script, such as `print("Hello World!")`, save the script and execute it. What's the difference between the Run and the Source command?

4. There are several ways you can execute R code in scripts. If you want to execute only a single line, you can move the cursor there and hit `Ctrl/Cmd + Enter`. Notice that the cursor will automatically move to the next line afterwards so that this is a convenient way for executing code lines step-by-step. If you want to execute the full script, you hit `Ctrl/Cmd + Shift + S`.
5. One of the drawbacks of using open source software is that many features are only implemented via additional packages that we have to download first before we can use them. Note that you only have to install them once, not every time you use them! Throughout the course, we will need quite a few packages. You can install them by typing the following command into the command line:

```
install.packages(c("tidyverse","nycflights13"), dependencies=TRUE)
```

Note that after you have installed a package, you will have to load the package before you can use it. While you have to install the package only once on your machine, you will have to load the package every time you start R. The most effective way of telling R that you want to use a specific package in a program is to simply include the `library(name-of-package)` in the first lines of your script. For example, if you want to use the `tidyverse` package, you can type:

```
library(tidyverse)
```

Try to understand the structure of this command and discuss its single elements with your partners. A handy feature of RStudio is auto-completion. To auto-complete a command use the tab key and the arrow keys to navigate the auto-complete popup. When typing the commands for the exercises below, experiment with this feature to see whether you like it.

Part 2: First steps with R

1. Create two 500×1 vectors of random numbers (`rnorm()`) and store them in your workspace (`<-`). What is the difference between the commands `rnorm()` and `runif()`?
2. What does the assignment operator `<-` do? Could you have used `=` instead? Most R programmers recommend using `<-` instead of `=`. Using `=` instead of `<-` is likely to cause confusion in more advanced R programs, especially when working with other programmers. Therefore, I recommend using `<-` throughout your R code. RStudio has a convenient shortcut for typing the assignment operator: `Alt + -`. It automatically surrounds `<-` with spaces which is good formatting practice. Can you figure out some of the differences between `<-` and `=`?
3. Plot the two vectors. (`plot()`)
4. Many commands that you may be familiar with from your operating system work in R as well, for example you might want to list all objects in your workspace (`ls`) or remove

certain objects from your workspace (`rm()`).
5. Before proceeding to the next question, clear your workspace.

Part 3: Basic manipulations of vectors & numbers

1. Create a 5×1 vector x of arbitrary numbers. Pick any numbers you like. (`c()`) We will use the `c()`-command a lot to construct vectors. Try to figure out how this command works.
2. Construct a vector y that stacks $-x$, 0 and x vertically. What will be the dimension of y?
3. Most math operators will work as you would expect: $+, -, *, /$, $log, exp, sin, cos, sqrt$, $min, max, range, length$, $sum, prod$, $mean, var, sort$. Experiment with these commands and figure out what exactly they do.
4. Will R perform the following operation: `z <- 2*x+y+1`? If yes, what will be the result?
5. Clear your workspace using the graphical user interface of RStudio.

Part 4: Generating sequences of numbers and logical vectors

1. Create a vector that contains the sequence of the first 15 natural numbers (`:` or `seq()`). `seq()` is one of many *functions* that R provides to automate many routine tasks. The general syntax for using functions is

```
function_name(arg1 = val1, arg2 = val2, ...)
```

where `arg` are arguments that a function requires and `val` are the values you assign to an argument when making a specific function call. 2. Is there any difference between `2*1:15` and `(2*1):15`? 3. What is a logical vector? Create a random vector and let R find all elements of this vector that are larger than 1.5. 4. Now find all the elements that are exactly equal to 1.2. Are you surprised by the result? 5. Can you construct a sequence of numbers in reverse order, i.e., the first element being the largest?

Part 5: Arrays & matrices

1. Generate a 100×1 vector x of random numbers. By default R arranges all objects in vector form. Moreover, R does not distinguish between column and row vectors.
2. Transform the 100×1 vector x into a 10×10 matrix. How does R rearrange the single elements? (`dim()`)
3. Instead of generating a matrix in two steps, you can use the `matrix()`-command directly. `matrix()` takes two main arguments: a vector of data and a vector of matrix dimensions. Generate the same matrix from the previous question using the `matrix()` command.
4. How can you extract entire columns, entire rows, single elements or a range of rows/columns from a matrix? (`[]`)
5. Compute the transpose of one of the matrices you have created. (`t()`)
6. Now construct 2 random square matrices. Compute both the mathematical matrix product and the component-wise product. (`*` and `%*%` - which is which?)

7. Clear your workspace. One convenient command to clear the entire workspace is `rm(list=ls())`.

Part 6: The Working Directory and R projects

1. R needs to interact with files on your computer in two ways. First, we typically need to load data for our analysis a file saved on your computer. Second, after having run the analysis, we need to save the results in the form of tables of graphs. When reading or writing files from/to your computer you need to tell R where to do find and create files. By default, R loads and saves files only from your working directory.
2. Find out what R's current working directory is. (`getwd()`)
3. Typically, you need to change the working directory in order to load or save files to a specific folder. You can change your working directory using

```
setwd("path-to-yourworkingdirectory")
```

where you provide the path for your working directory in quotes. All commands that you are familiar with from navigating in a command line should work, for example you can move up in your file directory by typing `setwd("../")`.

4. Download the file `vgr.csv` from Canvas. Navigate your RStudio working directory to the location of your download and import the data into R using the command `read.csv()` or `read_csv()` (from the tidyverse). How many observations does the data set contain?
5. Once you start working more seriously with data, you will most likely have to organize and coordinate many different scripts and data from the outside world and potentially share your coding work with others. This can make working with different working directories a pain, for example path names look very different on Windows than on Mac/Linux. A convenient way for avoiding many problems is by organizing your scripts in *projects*. You can create a new project by clicking on `File > New project`. Give your project a name and select a directory to save it. Now your working directory will be set automatically to the folder where your project is saved. Projects make it easy to work with relative path names and find any files that your scripts generate. However, there are many alternative different ways of organizing your workflow and your files. I encourage to experiment with Rprojects throughout the semester and see for yourself whether they work well for your projects.

Part 7: If you get stuck...

1. When doing data analysis and coding, it is very common to get stuck with a specific problem. These days, it is very likely that somebody else had a very similar problem before. There are numerous websites out there where programmers can ask these questions and get help. As a first step, you can try googling your problem or the error message that you received from R.
2. If googling does not help, there is a very high probability, that you will find help on more specialized programer's' platforms like www.stackoverflow.com. The community

there is very welcoming and supportive. So if you cannot find the answer to your question, don't be afraid to post a question yourself. However, chances of getting a satisfactory answer are much higher if you keep in mind some general guidelines, as for example presented https://stackoverflow.com/help/how-to-ask If you want to learn more about R and its community, visit http://www.r-bloggers.com which is the most popular meta blog on anything related to R programming.