Lab No. 1 (part1) — R Introduction for SGPE

Advanced Time Series Econometrics Labs

Ping Wu (ping.wu@strath.ac.uk (mailto:ping.wu@strath.ac.uk))

Welcome to R!

Why R?

- Licensing
- Availability of add on software (packages)
- Very easy to adapt to your needs. You can look at existing code, change it and reuse it.
- Ease of connecting to other (faster) software like fortran or C
- Who uses R?
    - Very active and growing community in science and business.
    - data scientists use R
    - who in business uses R? (https://www.quora.com/Which-companies-use-R)

Getting R and RStudio

- Click here for www.r-project.org and download R for your OS (https://www.r-project.org/)
- Click here for www.rstudio.org to download RStudio for your OS (https://posit.co/downloads/)

Start your R!

Part I: Working Directory and Workspace

The current working directory is displayed by RStudio within the title region of the Console. Or you can ask R for the current working directory location using the command:

```
getwd()
```

We can create a new folder for the current project we are working on as follows

```
dir.create("Lab1test")
```

```
## Warning in dir.create("Lab1test"): 'Lab1test' already exists
```

If you want to change the current working directory.

```
setwd("Lab1test")
```

In order to provide a nice folder structure it is reasonable to employ the project with subfolders for your data, images, … To navigate between subfolders, relative or absolute paths can be used. It is best practice to have the user running the script begin in a consistent directory on their machine and then use relative file paths from that directory to access files.

Let us create two subfolders in our working directory for R code and data

```
dir.create("code")
```

```
## Warning in dir.create("code"): 'code' already exists
```

```
dir.create("data")
```

```
## Warning in dir.create("data"): 'data' already exists
```

When creating new folders, use lowercase letters in the folder names: Windows ignores upper/lowercase but not Mac, which may lead to broken code, especially when using the same code on different computers.

Relative paths start with a “.” at your working directory. To navigate into a folder use “/Foldername”, to get out of a folder use another “.”

- “.” is the current folder
- “..” is one level above the current folder

Example:

```
setwd("./code")
getwd()
```

But for now: let’s stay in our Lab1test folder as working directory. “..” changes back in the folder above the current folder

```
setwd("..")
getwd()
```

The workspace is your current R working environment and includes any user-defined objects (vectors, matrices, data frames, lists, functions). At the end of an R session, the user can save an image of the current workspace that is automatically reloaded the next time R is started

Let’s create a simple scalar object

```
a <- 5
```

save the workspace to the file .RData in the cwd (current working directory)

```
save.image()
```

we can also specify a certain name for the .RData file

```
save.image(file="dummy.RData")
```

load a workspace into the current session, if you don't specify the path, the cwd is assumed

```
load("dummy.RData")
```

save specific objects (not the whole workspace) to a file, if you don't specify the path, the wd is assumed

```
save(a,file="myfile.RData")
```

load a workspace into the current session.if you don't specify the path, the cwd is assumed

```
load("myfile.RData")
```

If you want to remove elements from the environment

```
rm(a)
```

If you want to remove all objects currently in the working directory

```
rm(list=ls())
```

Part II: Packages

Packages are collections of R functions, data, and compiled code in a well-defined format. The directory where packages are stored is called the library. R comes with a standard set of packages. Others are available for download and installation. Once installed, they have to be loaded into the session to be used.

```
.libPaths() # get library location
library()   # see all packages installed
search()    # see packages currently loaded
```

Adding Packages: You can expand the types of analyses you do be adding other packages. A complete list of contributed packages is available from CRAN.

Follow these steps:

- Download and install a package (you only need to do this once).
- To use the package, invoke the library(package) command to load it into the current session. (You need to do this once in each session, unless you customize your environment to automatically load it each time.)

On MS Windows: use the console to install packages, e.g.

```
# install.packages("dplyr")  # install the package dplyr (install only if necessar
y)
# library(dplyr)             # load it in the current session
```

For Time Series:

- zoo - Provides the most popular format for saving time series objects in R.
- xts - Very flexible tools for manipulating time series data sets.

Part III: Basic Commands

You can use R as a calculator:

```
3 + 3         #addition; code works regardless of the spaces, but it is recommended
to use them
3 - 5         #subtraction
3 * 5         #multiplication
3 / 5         #division

3 ^ 5         #exponentiation
sqrt(81)      #square root
243 ^ (1/5)   #a-th root
sin(pi/2)     #sine
cos(0)        #cosine
tan(0)        #tangent
log(1)        #natural logarithm
exp(1)        #e^x

ceiling(1.2)  #next higher integer
floor(1.2)    #next lower integer
abs(-1)       #absolute value
round(2.45,1) #rounds given number of digits
```

Some useful R functions

```
x <- seq(1:10)  # sequentially generate from 1 to 10 (1,2,3,...,10)

mean(x)         # arithmetic mean
sd(x)             # standard deviation
var(x)          # variance
median(x)         # median
quantile(x)     # quantiles (quantiles on default: min, 25%, median, 75%, max)
range(x)          # range
sum(x)          # sum
min(x)          # minimum
max(x)          # maximum
```

Part IV: Vector and Matrix

Create a vecotr or matrix

```
a <- c(1,2,5.3,6,-2,4)               # numeric vector
b <- matrix(1:20, nrow=5,ncol=4)     # generates 5 x 4 numeric matrix (from 1 to 20)
```

Identify rows, columns or elements using subscripts

```
a[c(2,4)]   # 2nd and 4th elements of vector
b[,4]       # 4th column of matrix
b[3,]       # 3rd row of matrix
b[4,2]      # element in the 4th row and 2nd column
b[2:4,1:3]  # rows 2,3,4 of columns 1,2,3
```

Matrix addition

```
c <- matrix(51:70, nrow=5,ncol=4)    # generates 5 x 4 numeric matrix (from 51 to 70)
e <- b + c
```

Getting help()

- if you want help about a function, type help(fname)
- if you don't know which function you are looking for, try `??fname`

Learning R if you know Matlab/Stata/other

- R for Stata Users (https://www.matthieugomez.com/statar/)
- R for Matlab Users (https://cran.r-project.org/doc/contrib/Hiebeler-matlabR.pdf)
- ChatGPT