Financial Econometrics 2022/2023
Exercise set 1

1 Notes

- Due date: November 7^{th}, 2021, 23:59 (CET).
- Send your solutions through Microsoft Teams. Start a one-to-one private chat with the instructor and attach a *single* .txt file, name it as `GX_E1_ECXXX_ECXXX_ECXXX`, where `GX` is the group number, `ECXXXX` is the student number (there are up to four `ECXXX` strings depending on how many students are working together). Use only the student numbers of those who, within your group, worked on the exercises.
- Fill up the `Groups_and_grades` excel file in teams to make up the groups.
- If you work in a group, please designate *one* person to send the solutions and open only *one* chat for submitting the assignment. Please *avoid* changing the groups in the following exercise sets,. and *reuse* the same chat for the following submissions.
- The document is supposed to contain, answers, codes, and possible comments, for each question. Make a .txt file with the codes and comments that runs smoothly in the R console (by copy-pasting the *whole* content in R there the code should run all the way till the end), see the file *Example.txt* and copy-paste its content in R.
- Additionally write your student number(s) on the top of the document. You can work in groups of 2,3,4 students or individually.
- If you would like to discuss the exercises and get some feedback before the due date you can come to office 2.23 (second floor) on Fridays from 10 to 11. No booking, if at some point I am not in the office just wait some minutes, I'll be shortly back.
- Last updated on 2022-10-21 at 09:04:17 (UT).

2 Exercises

Total points: 1.0.

1. Install packages, import .csv, make a dataframe.
   (i) Get from Yahoo finance the historical prices for the Apple Inc. stock (AAPL), download the data from January 1st 2019 to December 31st 2020. (ii) Import the corresponding .csv file in R. There are different ways of doing it, the standard approach is to use the functions `read.csv` or `read.table`. (iii) Try importing the data twice, by using both the functions. Make sure you understand what you are doing, as there are no analyses without data: now, in future courses, for your thesis, and perhaps at work you will have to use `read.csv`, `read.table` again and again.

   Once you imported the data in some variable, (iv) check that this variable is a dataframe object, otherwise convert it to an R `data.frame` object. Dataframes are the standard way to store tabular data and there are many useful functions that apply to dataframes to edit and access their data: in the vast majority of situations you want to work with dataframes, so check that what you just imported is indeed a dataframe object.

   In exercise 2 you are asked to remove the dates from the dataframe: do not use time in importing the date column as dates (quite difficult!), if you get strings, numbers or whatever is just fine right now.

   (iv) Have a look at the first ten rows of the imported dataframe by using the function `head`. (iv) Call the `attach` function on the dataframe. What does it do? Explain.
   Hints: (a) you have to either provide the full path to the .csv file (tedious), or make sure the working directory coincides with the folder where the file is (easier). In this regard, use the commands `getwd` and `setwd`, also bear in mind that when providing a path, you have to invert the slashes (at least in Windows - in MacOS I have no idea). What in Windows reads like `C:\Windows\...` in R becomes `C:/Windows/...`.(b) before calling `read.csv` and `read.table`, check the function documentation (type `?read.csv`, and `?read.table`): do you need to specify any values for the optional arguments? Yes... for instance make sure you tell R that the first row of the data is just the header (i.e. columns' labels), and what is the "field separator character" for `read.table`? (c) you can comment on your code by inserting `#`: R executes the following code line after encountering a `#` symbol.
   0.2 points.

2. Some operations on rows and columns.
   Access your dataframe, and (i) make sure that the data is sorted in ascending time order (if not, sort the original .csv file e.g. in excel), (ii) remove the column containing dates (many way of doing it), (iii) keep only the first 252 rows, (iv) create a variable `p` containing adjusted closing prices.
   0.1 points.

3. Plot and logreturns.
   (i) Plot the time-series `p`, (ii) compute logreturns in a neat way, (iii) compute returns' volatility (annualized).
   0.1 points.

4. Multi-panel plot.
   (i) Make a 2-rows by 2-columns multi-panel plot. On the top row plot the logreturns and squared logreturns, on the bottom rows plot the two corresponding autocorrelation function (ACF) plots.
   0.1 points.

5. Summary statistics and higher moments.
   (i) Print summary statistics for the logreturns. Compute their skewness and kurtosis, (ii) with the `moments` package and (iii) by implementing the formulae by yourself (slides 101-102, Chapter 1). The values obtained by the two methods *must* match for the exercise to be correct. (iv) Provide a comment.
   Hint: (a) in R type `?summary`. (b) To install a package use `install.packages` or do it manually from the "Packages" menu. (iii) Once the package is installed you have to load it, use the function `library`. (iv) you can use functions `mean`,`var`, and `sd`.
   0.3 points.

6. Normality test.
   (i) Perform the Jarque-Brera test on the logreturns. (ii) Provide a comment on the p-value and test results.
   Hint: (a) need to install a package for this, e.g. `tseries`.
   0.1 points.

7. Normal approximation.
   (i) If you really want to approximate the distribution of the logreturns with a normal distribution, what would the mean and variance parameters be?
   0.1 points.