Financial Econometrics 2023/2024
Exercise set 1

1 Notes

- Due date: Monday, November 13th, 2024, 23:59 (CET): 3 weeks of time.
  - 3 weeks is an exception: for the holidays break on November 1-5 I am out of office and can't help, thus I add an extra week.
  - The deadline is strict, no exceptions.
- 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`, where `X` is the group number (e.g., `G4_E1.txt` is the name for group 4).
- 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 please *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. This means that if I copy-paste the *whole* content from your .txt file in the R console, *the code should run all the way with no errors*, see the file *Example.txt* and copy-paste its content in R. Makes sure your code runs.
- Additionally, write your student names, surnames and number(s) on the top of the document as a comment. You can work in groups of 2-5 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 during regular office hours (please book), or ask to arrange a different time.
- Avoid sending lines and lines of code through Teams: rely on office hours to get help.
  - As the deadline approaches, I get more and more request for help: then I end up being fully booked and don't have time for all. Plan the work well ahead!
  - Be considerate, do not expect replies on the weekend/late at night. Note that the deadline is on Monday: you don't wanna work on the assignment the day before, as I won't be around for helping.
- In the tasks, name the variables exactly as indicated! so that when, e.g., I see `r` I have a reference and understand what you are trying to do. Write your code in a way that is understandable: e.g., you would name a variable that contains prices (returns) as `p` (`r`) and not `h` (`u`).
- Last updated on 2023-11-13 at 09:38:49 (UT).

2 Exercises

Total points: 1.0.

1. Install packages, import .csv, make a dataframe. 0.2 points.
   This is the most challenging exercise, it's after all the first time you use R! Use Google as much as you want to get it done properly. It's important that you solve this task first as it loads the data required for all the other exercises! If you are badly stuck, come and talk to me: if you fail this first tasks, you fail the entire assignment.

   (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. Save the downloaded file with the name `AAPL.csv` (*same name for all*, otherwise I have to rename the file every time to check your code).

   (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`. Assign the data to a variable `z` (*same name for all*).

   (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: what is the code to check for a variable to be a `data.frame`?

   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.

   (v) 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.

2. Some operations on rows and columns. 0.1 points.
   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` *Same name for all* containing adjusted closing prices.

3. Plot and logreturns. 0.1 points.

   (i) Plot the time-series `p` (don't worry about the x-axis indexes, labels and graphics)

   (ii) compute logreturns in a neat way, and assign them to a variable `r` *same name for all*

   (iii) compute returns' volatility (annualized).

   Hint: it's a good idea to talk to some other group and compare your logreturns, as the following tasks are about analyses on logreturns.

4. Multi-panel plot. 0.1 points.
   Make a 2-rows by 2-columns multi-panel plot.

   (i) On the top row plot the logreturns and squared logreturns,

   (ii) On the bottom rows plot the two corresponding autocorrelation function (ACF) plots.

5. Summary statistics and higher moments. 0.3 points.

   (i) Print a table of summary statistics for the logreturns

   Compute their skewness and kurtosis,

   (ii) with e.g. the `moments` package and

   (iii) by implementing the formulas by yourself (slides in Chapter 1)

   (iv) Provide a comment (on all the computed moments) based on what you know on stylized facts

   The values obtained by the two methods *must* match for the exercise to be correct.
   Hint: (a) in R type `?summary`. (b) To install a package use `install.packages` or do it manually from the "Packages" menu. (c) Once the package is installed you have to load it, use the function `library`. (d) you can use functions `mean`, `var`, and `sd`. (e) the formulas for the skewness and kurtosis involve the sample variance: some packages compute it by using $(N-1)$ in the denominator, some others compute the sample variance by dividing by N, this could lead to a difference between what you manually compute and what the package does.

6. Normality test. 0.1 points.

   - (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`.

7. Normal approximation. 0.1 points.

   - (i) If you want to approximate the distribution of the logreturns with a normal distribution, what would its mean and variance parameters be?

   Hint: the answer is *simple*. If you end up fitting distributions/using fancy packages/complex functions or have a very complex answer... sorry, you are on the wrong way.