Assignment 1

Philippe van der Beck

philippe.vanderbeck@epfl.ch

Problem 1

Suppose that your statistical model for the evolution of a share price is given by the geometric brownian motion (we assume that the stock does not pay any dividend),

$$\frac{dS_t}{S_t} = \mu dt + \sigma dz_t, \tag{1}$$

where $\mu = 0.06$ and $\sigma = 0.20$.

(a) Import NumPy. NumPy is the fundamental package for scientific computing with Python. (*Hint: Use the command* `import numpy as np`.)

(b) Import the matplotlib.pyplot module, which contains functions that allow you to generate many kinds of plots. (*Hint: Use the command* `import matplotlib.pyplot as plt`.)

(c) Solve the SDE defined by (1), and use the solution to simulate a path for the share price with daily observations over a horizon of 10 years. Assume there are 365 trading days in a year, so you should obtain a path of 3650 observations. (*Hint: You can store the share price realizations in a "list", which is created through the command:* `list = [1,2,3,5]`. *Functions in the NumPy package are accessed using "np.function". For example, the square root of 2 is computed using the command:* `np.sqrt(2)`. *Finally, note that you can generate a sample from a univariate normal distribution of mean 0 and variance 1 using the function:* `np.random.randn`.)

(d) Plot the path of daily share prices. (*Hint: Some functions that you might find useful are:* `plt.plot`, `plt.legend`, `plt.show`.)

(e) Compute daily continuously compounded returns. (*Hint: If* $R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$ *is the simple return, then the continuously compounded return is* $r_t = \log(1 + R_t)$.)

(f) Plot the time-series of daily log-returns.

(g) Compute both annualized mean and standard deviation of log-returns, and print the result on the screen. (*Hint: You can use the function* `print`.)

Problem 2

What are the estimates of standard error of the mean and volatility? Put another way, as we roll through the data, or simulate new time series of the same length, what are the estimates of variation in the mean and volatility? How sensitive are the mean and standard deviation estimates to the sampling frequency? In this and the next exercise, we will use pandas to do this efficiently.

(a) Import pandas. pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. (*Hint: Use the command* `import pandas as pd`.)

(b) Use your solution to (1) to simulate daily share prices for the period from January 1$^{\text{st}}$, 1950 to December 31$^{\text{st}}$, 2020. However, instead of storing the data in a `list`, create a `series` using pandas. To scale the mean and variance in the discretized solution to (1), assume as before that there are 365 trading days in a year.[1] (*Hint: To create the series, use the command* `pd.Series(data,index)`, *where* `index` *contains all calendar days in the period and can be generated using* `pd.period_range(start,end,freq='D')`.)

(c) How many observations has your sample? Plot the path of daily share prices.

(d) Create a new time series which contains the monthly average share price. In other words, the horizon still covers the region January 1$^{\text{st}}$, 1950 to December 31$^{\text{st}}$, 2020. But now the time series has a monthly frequency, and the n$^{\text{th}}$ observation is the arithmetic average of the daily prices in the n$^{\text{th}}$ month of your original time series. (*Hint: If* `S` *is a pandas series, then the method* `S.resample(rule).how` *might be useful here.* `rule` *is the string representing target conversion, e.g., 'D' (calendar day frequency), 'W' (weekly frequency), 'M' (month end frequency), etc.* `rule` *is the string representing the method for down- or re-sampling. It defaults to 'mean', but other options include 'first', 'last', 'var', etc.*)

(e) Plot the path of monthly share prices.

[1] Thus, we abstract from weekends and holidays.

Problem 3

(a) From your simulated data of daily prices, generate a time-series of daily log-returns, a time-series of weekly log-returns, and a time-series of monthly log-returns. (*Hint: If* `S` *is a* pandas *series, then the method* `S.resample(rule).how` *might be useful here.*)

(b) For each frequency, compute summary statistics. (*Hint: If* `S` *is a* pandas *series, then the method* `S.describe()` *might be useful here.*)

(c) For each frequency, compute the annualized mean and standard deviation of log-returns, and print the result on the screen.

(d) For each frequency, plot the time-series of the annualized mean and std deviation of log-returns estimated using a rolling one-year window. (*Hint: If* `S` *is a* pandas *series, then the method* `S.rolling` *might be useful here.*)

(e) How does the historical variation in the annualized mean and standard deviation estimates compare across sampling frequencies?

(f) Group your *daily* simulated data into non-overlapping "bins", where each bin covers a period of one year. Since we are not considering weekends or holidays, each bin should contain 365 days. Then, estimate the mean and variance of log-returns using each bin (one at a time) as your sample. For the sample variance, use the "simple" formula of Merton (1980). In this way, you can obtain a time-series of annualized mean and variance estimates. You can then compute the mean and variance of this time-series.

Repeat this experiment with your *monthly* simulated data. That is, group your monthly simulated data into non-overlapping "bins", where each bin covers a period of one year. (How many observations does each bin contain?). Etc.

For both frequencies (daily and monthly), what is the mean and variance of your mean estimator? What about the mean and variance of your variance estimator? How do these estimates compare to the theoretical moments of the estimators which we derived in class? (*Hint:* `S.resample(rule).how` *is a really cool method!*)

Problem 4

In this exercise, we repeat the analysis carried in Problem 3 but using *real* data instead of *simulated* data! You will have to get the data from the WRDS database which we will use more extensively later during the course. After having created an account at https:

//wrds-web.wharton.upenn.edu/wrds/, you have to install the packages *wrds* and *psycopg2* by starting the anaconda prompt window and typing

`pip install wrds`

and

`pip install psycopg2`, respectively. If you are using WRDS and Python for the first time, you will have to run the following code (replacing "joe" with your WRDS username):

```
import wrds
db = wrds.Connection(wrds username=?joe?)
db.create pgpass file()
```

You will be asked to enter your login details while running these lines. After that, you have created a pgpass file on your computer and will not be asked to enter your login details after running *db = wrds.Connection(wrds username='joe')*. You can now start working with the data.

(a) Run the following code:

```
import wrds
db = wrds.Connection(wrds username='joe')
```

(b) Download daily closing stock returns[2] for the period starting on January 1st, 2001 and ending on December 31st, 2020 of the following companies: Apple, Goldman Sachs, Microsoft, Procter and Gamble, and General Electric. (*Hint: Use the command:* `aapl=db.raw sql("select date, ret from crsp.dsf where permco in (7) and date>='2001-01-01' and date<='2020-12-31'")` to get the data for Apple (which has the CRSP's company number 7). GS, MSFT, PG and GE have the permco's 35048, 8048, 21446 and 20792, respectively. You can find details on how to access data from WRDS at https://wrds-www.wharton.upenn.edu/pages/support/programming-wrds/programming-python/querying-wrds-data-python/.)

(c) Save the data panel to a .cvs file using the command `DataFrame.to csv()`.

(d) We can now read the HDF5 file with the saved Yahoo Finance data using the command `pd.read csv()`.

[2]Prices in the CRSP database do not contain dividends and are not adjusted for stock splits etc. In order to get meaningful data on returns you have to download the variable denoted by "ret."

(e) Repeat the analysis of Problem 3 (a) to (d) using this new time-series `S`. As before, use the sampling frequencies: monthly, weekly, daily.

(f) How does the variation in the mean and standard deviation estimates compare across sampling frequencies? How does the pattern you observe compare to the pattern you observed with the simulated data? If you observe a similar pattern, why? If you observe some differences, what could be the reason behind this?

(g) How do the estimates change as we go through the COVID Crisis window? Do you observe other crises through estimations?