Project Econometrics 2021

Frankfurt School of Finance and Management

Jan Vecer

Due Date: June 7, 2021 (midnight). Later submissions result in a grade subtraction.

Group Project: This is a group project with a group of a maximal size two. An individual project (a group of one) is possible.

Output: A single file: a Jupyter notebook that runs with Python. The Jupyter notebook should be fully executable. The commands and the results should come with detailed comments. The file name should be STUDENTID.ipynb, where STUDENTID is your FS student IDs if you choose to submit it as a group of one project. The file name should be STUDENTID1_STUDENTID2.ipynb for a group of two project. The files should be uploaded to a designated area in the campus.fs.de website.

Project Description: Analysis of data of stocks in SP500. Your tasks are the following.

1. Get the data. First, you should set your individual first and the last date for your data analysis. This is done by

```
import numpy as np
import datetime
from datetime import timedelta

np.random.seed(STUDENTID)

start_date = datetime.date(2020, 4, 1) - timedelta(days = np.random.randint(20))
start_date.strftime("%Y-%m-%d")

end_date = datetime.date(2021, 4, 1) + timedelta(days = np.random.randint(20))
end_date.strftime("%Y-%m-%d")
```

for a group of one project and

```
import numpy as np
import datetime
from datetime import timedelta

np.random.seed(STUDENTID1)

start_date = datetime.date(2020, 4, 1) - timedelta(days = np.random.randint(20))
start_date.strftime("%Y-%m-%d")

np.random.seed(STUDENTID2)

end_date = datetime.date(2021, 4, 1) + timedelta(days = np.random.randint(20))
end_date.strftime("%Y-%m-%d")
```

for a group of two project.

For getting SP500 data, the default suggested source is Yahoo, but feel free to use any other alternative sources, such as Bloomberg. For the Yahoo data option, get the current list of SP500 components from Wikipedia:

```
table = pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')[0]
tickers = table['Symbol'].tolist()
```

As the tickers from Wikipedia use '.' rather than '-' that is used in Yahoo for a couple of stocks, the eventual dots need to be replaced"

```
for i in range(len(tickers)):
    tickers[i] = tickers[i].replace(".","-")
```

Add the index SP500 itself to the data set:

```
tickers.append('^GSPC')
```

The actual download of data for all SP500 component stocks is done using a single command (after calling the respective library pandas_datareader):

```
from pandas_datareader import data
sp500 = data.DataReader(tickers, 'yahoo', start_date, end_date)['Adj Close']
```

Note that this procedure takes about 15 minutes. It is advisable to modify the DataReader command and download the data individually for each ticker using a 'for' loop and print the progress of the downloading procedure. Mark the stocks that were added to the index after April 2020 as their data may come with some missing values. As a part of the exercise, save the corresponding table in a local sp500.csv file. For your own project, you should download the data only once and your subsequent work should only read the existing sp500.csv file as your data source.

2. This part is to generate nice graphic representation of the prices and familiarize you with the Python plotting library `bokeh`. First, select 9 random stocks from SP500. This can be done by calling

```
a = np.random.choice(len(sp500.columns), 9, replace=False)
```

so that the stock data are in

```
sp500[tickers[a[i]]]
```

for i in $0, \ldots, 8$.

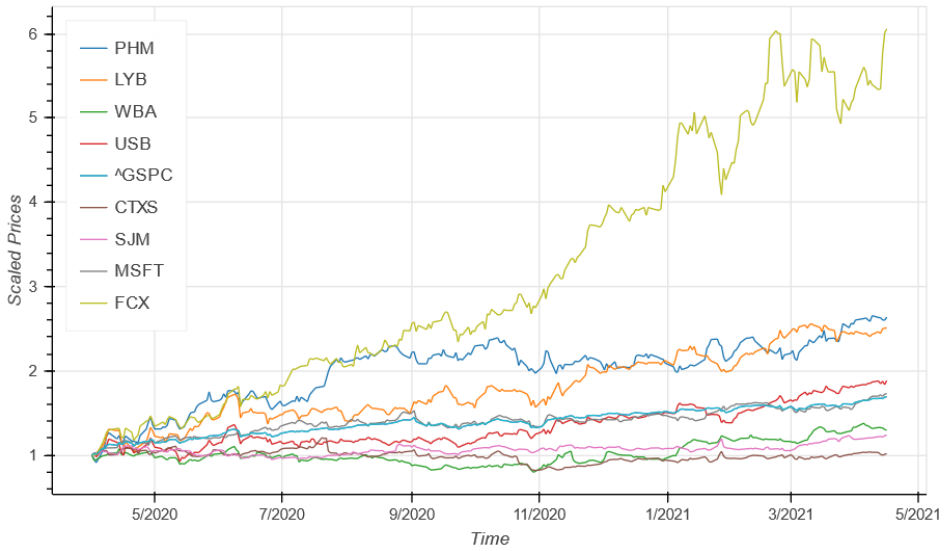

Figure 1: Plot of the selected stocks.

Add the index itself. Plot the time series of the selected stocks on a single interactive `bokeh` graph, with functions pan, zoom in, zoom out, reset, hover and crosshair and with the option to hide and show the selected series. This is done by calling

`p.legend.click_policy = "hide"`

The prices need to be normalized so that their scale is proportional. Normalize to 1 as the starting point. The resulting graph should look like Figure 1. The hover function should give the exact date, the ticker and the price (normalized is good enough). Learn the `bokeh` library to the point that you are able to generate similar graphs with different plotting options for your own future reference.

3. Perform a basic mean - variance analysis. From the table of the adjusted closing prices, create a table representing log returns. Avoid the use of `for` loops. Instead, apply vector operations (such as `log, diff, apply`). Compute the averages and standard deviations for each stock. Properly treat any missing values. Plot a frequency histogram for the averages with a density fit. Add a fit based on normal distribution. Do the same for the standard deviations, but use a fit to gamma distribution. Summarize the results. Plot an interactive graph (ideally using `bokeh` library) on (x, y) axis with points representing the standard deviation x and the average y for each stock. The resulting graph should be as clear as possible by using the proper size of plotting points together with the legend corresponding to the ticker. The points should come with different colors depending on the stock sector as defined by Global Industry Classification Standard that appears in the downloaded Wikipedia table. Compute the confidence ellipse (as a function of a standard deviation) and plot it together with the points. Add the SP500 index as a category on its own (without the confidence ellipse). The interactive graph should hide and show the points with the legends and the confidence ellipse on a click with the GICS category. The resulting graph should look something like Figure 2.

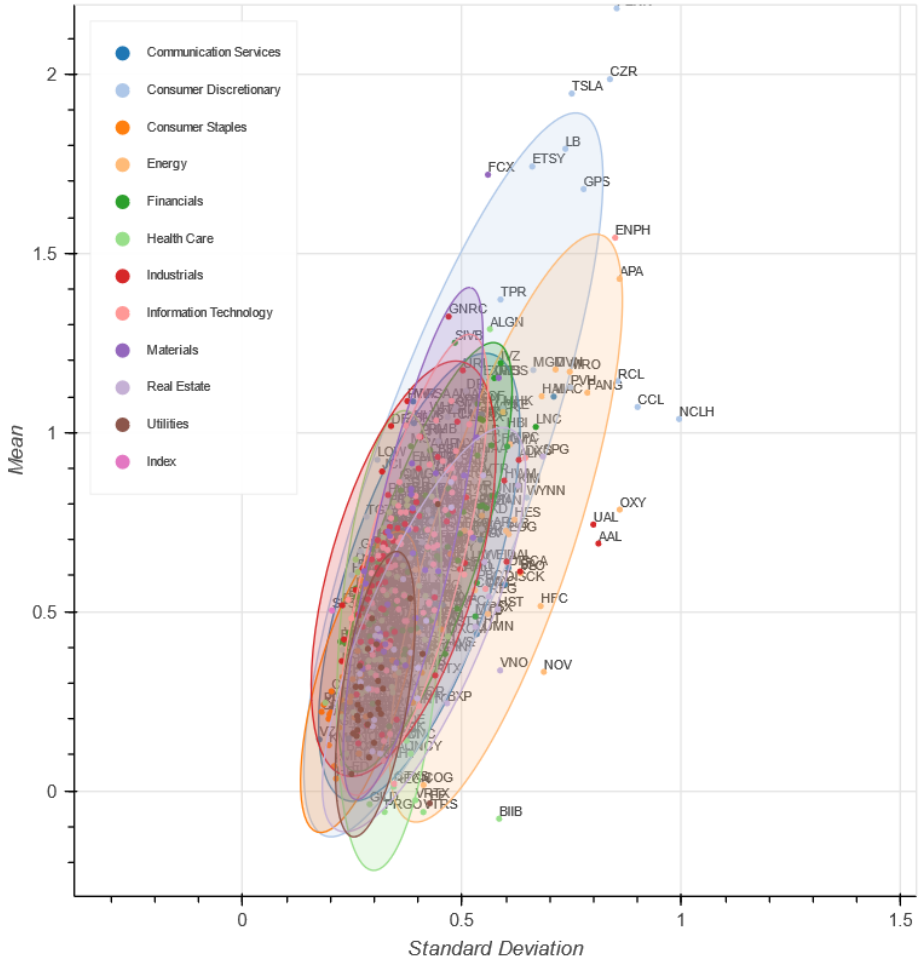

Figure 2: Mean and variance analysis for SP500 stocks.

4. Perform a basic CAPM analysis for each stock in your SP500 data. For each stock, run regression `(y ~ x)`, where y is a vector of returns of a given stock and x is a vector of returns of a given

benchmark, in our case the SP500 index. This means in particular that you need to get the values of the index itself (ticker ^GSPC) and compute its returns. Store the resulting coefficients representing α and β for each stock together with the P-value corresponding to the coefficient α. Plot the histogram of the vector of betas with the density fit and the fit based on a normal distribution. Identify (= show them in your Jupyter notebook) stocks at both extremes, the ones with the smallest and the largest betas. Plot the histogram of the vector of alphas with the density fit and with the fit based on normal distribution. Identify the stocks with statistically significant alphas (from the corresponding P-values). Here, the word "identify" means that you also add a detailed discussion, possibly with some ex post and ex ante investor advice. Obviously, picking low beta stocks is critical in the situation when the stock market is crashing, so it is good to find the individual stocks and industries that satisfy this condition. Stocks with significant alphas have either over or under performed the index, again it is good to identify them.

For the two group project, one student should address the alpha part of the analysis and one student should address the beta part of the analysis and assign who is responsible for each part.

Lastly, choose one stock that you find interesting, and plot the joint returns of the stock and the index together with the corresponding regression line.

5. Plot the histogram of log returns of the SP500 index and fit it to the normal density and the Student-t density (using `fit` function from `scipy` library) in the same graph. List the fitted parameters for both densities.