Lab 3

Introduction to Python Programming for Economics & Finance

Richard Foltyn
University of Glasgow

May 25, 2023

Contents

Daily returns of US stock market indices

In this lab, we examine how the three major US stock market indices performed this year using data from Yahoo! Finance.

1 Download data from Yahoo! Finance

Use the `yfinance` library and its `download()` function to obtain the time series of daily observations for the S&P 500, the Dow Jones Industrial Average (DJIA) and the NASDAQ Composite indices. Restrict the sample to the period from 2023-01-01 to 2023-04-30 and keep only the closing price stored in column `Close`.

Hint: The corresponding ticker symbols are `^GSPC`, `^DJI`, `^IXIC`, respectively.

Rename the DataFrame columns to `'SP500'`, `'Dow Jones'` and `'NASDAQ'` using the `rename()` method.

Hint: You may need to first install `yfinance` as follows (in particular when running on Google Colab):

[1]:
```
# Uncomment this to install yfinance in your Python environment
#! pip install yfinance
```

Note: If you cannot import `yfinance`, e.g., because you are using the Jupyter Lite environment, you can load the data from a the CSV file located in the same directory as this notebook as follows:

```python
import pandas as pd

# Uncomment this to use file in local directory
DATA_PATH = '.'

# Uncomment this to load data directly from GitHub
# DATA_PATH = 'https://raw.githubusercontent.com/richardfoltyn/python-intro-PGR/main/labs'

data = pd.read_csv(
    f'{DATA_PATH}/US-stock-indices.csv',
    index_col='Date',
    parse_dates=True
)
```

2 Plot daily closing prices

Plot the three time series (one for each index) in a single graph. Label all axes and make sure your graph contains a legend.

Hint: You can directly use the `DataFrame.plot()` method implemented in `pandas`.

3 Plot normalised daily closing prices

The graph you created in the previous sub-question is not well-suited to illustrate how each index developed in 2023 since the indices are reported on vastly different scales (the S&P500 appears to be an almost flat line).

To get a better idea about how each index fared in 2023 relative to its value at the beginning of the year, normalize each index by its value on the first trading day in 2023 (which was 2023-01-03). Plot the resulting normalised indices.

4 Compute and plot the daily returns

For each index, compute the daily returns, i.e., the relative change vs. the previous closing price in percent.

Create a plot of the daily returns for all indices.

5 Plot the distributions of daily returns

Compute the average daily returns and the volatility (standard deviation) for each index. Create a histogram of daily returns for each index using 25 bins (i.e., create a figure with 3 panels). For each histogram, add the density of a normal distribution that has the same mean and variance.

Hint: You can either use `DataFrame.hist()` to plot the histogram, or Matplotlib's `hist()` function. In either case, you should add `density=True` such that the histogram is appropriately rescaled and comparable to the normal density.

Hint: Use the `pdf()` method of the `scipy.stats.norm` class to compute the normal density.

6 Compute and plot the pairwise correlations

Compute the pairwise correlations between the daily returns of each index pair. Create a 3-by-3 graph where each panel contains a bivariate scatter plot of the daily returns of one index vs. another.

Hint: You can use the function `scatter_matrix()` to accomplish this task. Alternatively, you can create a figure with 3-by-3 subplots using Matplotlib's `subplots()`, iterate over all rows and columns and add a `scatter()` plot to each `axes` object.