Statistics from Stock Data

In this lab we will load stock data into a Pandas Dataframe and calculate some statistics on it. We will be working with stock data from Google, Apple, and Amazon. All the stock data was downloaded from yahoo finance in CSV format. In your workspace you should have a file named GOOG.csv containing the Google stock data, a file named AAPL.csv containing the Apple stock data, and a file named AMZN.csv containing the Amazon stock data. All the files contain 7 columns of data:

Date Open High Low Close Adj_Close Volume

We will start by reading in any of the above CSV files into a DataFrame and see what the data looks like.

In [ ]:
```
# We import pandas into Python


# We read in a stock data data file into a data frame and see what it looks lik
e


# We display the first 5 rows of the DataFrame
```

We clearly see that the Dataframe is has automatically labeled the row indices using integers and has labeled the columns of the DataFrame using the names of the columns in the CSV files.

To Do

You will now load the stock data from Google, Apple, and Amazon into separte DataFrames. However, for each stock data you will only be interested in loading the `Date` and `Adj Close` columns into the Dataframe. In addtion, you want to use the `Date` column as your row index. Finally, you want the DataFrame to recognize the dates as actual dates (year/month/day) and not as strings. For each stock, you can accomplish all theses things in just one line of code by using the appropiate keywords in the `pd.read_csv()` function. Here are a few hints:

- Use the `index_col` keyword to indicate which column you want to use as an index. For example `index_col = ['Open']`
- Set the `parse_dates` keyword equal to `True` to convert the Dates into real dates of the form year/month/day
- Use the `usecols` keyword to select which columns you want to load into the DataFrame. For example `usecols = ['Open', 'High']`

Fill in the code below:

In [ ]:
```
# We load the Google stock data into a DataFrame
google_stock =

# We load the Apple stock data into a DataFrame
apple_stock =

# We load the Amazon stock data into a DataFrame
amazon_stock =
```

You can check that you have loaded the data correctly by displaying the head of the DataFrames.

In [ ]:
```
# We display the google_stock DataFrame
```

You will now join the three DataFrames above to create a single new DataFrame that contains all the `Adj Close` for all the stocks. Let's start by creating an empty DataFrame that has as row indices calendar days between `2000-01-01` and `2016-12-31`. We will use the `pd.date_range()` function to create the calendar dates first and then we will create a DataFrame that uses those dates as row indices:

In [ ]:
```
# We create calendar dates between '2000-01-01' and  '2016-12-31'
dates = pd.date_range('2000-01-01', '2016-12-31')

# We create and empty DataFrame that uses the above dates as indices
all_stocks = pd.DataFrame(index = dates)
```

To Do

You will now join the the individual DataFrames, `google_stock`, `apple_stock`, and `amazon_stock`, to the `all_stocks` DataFrame. However, before you do this, it is necessary that you change the name of the columns in each of the three dataframes. This is because the column labels in the `all_stocks`dataframe must be unique. Since all the columns in the individual dataframes have the same name, `Adj Close`, we must change them to the stock name before joining them. In the space below change the column label `Adj Close` of each individual dataframe to the name of the corresponding stock. You can do this by using the `pd.DataFrame.rename()` function.

In [ ]:
```
# Change the Adj Close column label to Google
google_stock =

# Change the Adj Close column label to Apple
apple_stock =

# Change the Adj Close column label to Amazon
amazon_stock =
```

You can check that the column labels have been changed correctly by displaying the datadrames

In [ ]:
```
# We display the google_stock DataFrame
```

In [ ]:
```
# We display the apple_stock DataFrame
```

In [ ]:
```
# We display the amazon_stock DataFrame
```

Now that we have unique column labels, we can join the individual DataFrames to the `all_stocks`DataFrame. For this we will use the `dataframe.join()` function. The function `dataframe1.join(dataframe2)` joins `dataframe1` with `dataframe2`. We will join each dataframe one by one to the `all_stocks` dataframe. Fill in the code below to join the dataframes, the first join has been made for you:

In [ ]:
```
# We join the Google stock to all_stocks
all_stocks = all_stocks.join(google_stock)

# We join the Apple stock to all_stocks
all_stocks =

# We join the Amazon stock to all_stocks
all_stocks =
```

You can check that the dataframes have been joined correctly by displaying the `all_stocks` dataframe

In [ ]:
```
# We display the all_stocks DataFrame
```

To Do

Before we proceed to get some statistics on the stock data, let's first check that we don't have any *NaN*values. In the space below check if there are any *NaN* values in the `all_stocks` dataframe. If there are any, remove any rows that have *NaN* values:

In [ ]:
```
# Check if there are any NaN values in the all_stocks dataframe
```

In [ ]:
```
# Remove any rows that contain NaN values
```

You can check that the *NaN* values have been eliminated by displaying the `all_stocks` dataframe

In [ ]:
```
# Check if there are any NaN values in the all_stocks dataframe
```

Display the `all_stocks` dataframe and verify that there are no *NaN* values

In [ ]:
```
# We display the all_stocks DataFrame
```

Now that you have eliminated any *NaN* values we can now calculate some basic statistics on the stock prices. Fill in the code below

In [ ]:
```
# Print the average stock price for each stock


# Print the median stock price for each stock


# Print the standard deviation of the stock price for each stock


# Print the correlation between stocks

```

We will now look at how we can compute some rolling statistics, also known as moving statistics. We can calculate for example the rolling mean (moving average) of the Google stock price by using the Pandas `dataframe.rolling().mean()` method. The `dataframe.rolling(N).mean()` calculates the rolling mean over an `N`-day window. In other words, we can take a look at the average stock price every `N` days using the above method. Fill in the code below to calculate the average stock price every 150 days for Google stock

In [ ]:
```
# We compute the rolling mean using a 150-Day window for Google stock
rollingMean =
```

We can also visualize the rolling mean by plotting the data in our dataframe. In the following lessons you will learn how to use **Matplotlib** to visualize data. For now I will just import matplotlib and plot the Google stock data on top of the rolling mean. You can play around by changing the rolling mean window and see how the plot changes.

In [ ]:
```
%matplotlib inline

# We import matplotlib into Python
import matplotlib.pyplot as plt

# We plot the Google stock data
plt.plot(all_stocks['Google'])

# We plot the rolling mean ontop of our Google stock data
plt.plot(rollingMean)
plt.legend(['Google Stock Price', 'Rolling Mean'])
plt.show()
```

In [ ]:
```

```