Quantitative Investment Solutions Coding Assignment
Spring 2020
Due: March 7th, 2020 11:59 PM

You must complete all three questions. You may submit your completed solutions as three Python files (one for each question) or a link to a Github repo containing the three files. Please make an effort to write clean, readable code and comment as you feel necessary so you are able to explain your code.

<u>Question 1 - Metrics</u>

Quant is in charge of developing and maintaining the SSMIF Risk Screen. You are going to create four functions in Python to calculate some of the metrics we use. You are only allowed to use these Python packages/modules:

- Pandas
- Pandas_datareader
- Python's built-in statistics package
- Python's built-in Math
- Python's built-in Datetime

Your task:

1. Create a function called Daily_Returns that has one parameter, a pandas dataframe. The function should return a list of daily returns calculated from the adjusted closes. The dataframe is guaranteed to have a column called "Adj Close".

2. Create a function called Monthly_VaR that has two parameters, a ticker and a confidence level with a default value of 0.05. The function should return the monthly value at risk for a given ticker and confidence level. Within the function, you should use pandas_datareader to download stock data from Yahoo, starting from 1/1/2019 and ending on 12/31/2019. The ticker will always be a string and the confidence level will always be a double. For more information about VaR, you can read this article.

3. Create a function called Monthly_CVaR that has two parameters, a ticker and a confidence level with a default value of 0.05. The function should return the monthly value at risk for a given ticker and confidence level. Within the function, you should use pandas_datareader to download stock data from Yahoo, starting from 1/1/2019 and ending on 12/31/2019. The ticker will always be a string and the confidence level will always be a double. For more information about CVaR, you can read this article.

4. Create a function called Monthly_Volitility that has one parameter, a ticker. The function should return the monthly volatility for a given ticker. Within the function, you should use pandas_datareader to download stock data from Yahoo, starting from 1/1/2019 and ending on 12/31/2019. The ticker will always be a string. For more information about Volatility, you can read this article.

Note: Monthly_VaR, Monthly_CVaR, and Monthly_Volatility should all use your Daily_Returns function.

Question 2 - SQLite

This next question is a continuation of the previous one, however, you are going to use SQLite. This is a Python built-in package. Don't worry if you've never heard of it, it's fairly straight forward to Google.

To make it a little easier, this is the code you want to use to connect to the database and to create the table you are going to use for this question:

```
import sqlite3
conn = sqlite3.connect('SSMIF.db')
c = conn.cursor()
c.execute("""CREATE TABLE "Stock_Data" (
                "Timestamp" INTEGER NOT NULL,
                  "Open" DECIMAL(10, 2),
                  "High" DECIMAL(10, 2),
                  "Low"  DECIMAL(10, 2),
                  "Close" DECIMAL(10, 2),
                  "Adj_Close" DECIMAL(10, 2)
              );""")
conn.commit()
conn.close()
```

If the database does not exist, it will be created upon connection.

Timestamp, Open, High, Low, Close, and Adj_Close are the names of the columns in the table. If you notice, they are also the same columns in a dataframe returned from pandas_datareader. The only difference is the Timestamp column. This column is just the datetime index from the dataframe converted to a timestamp using the datetime module.

For this question you are limited to these packages/modules:

- pandas_datareader
- Python's built-in Sqlite3
- Python's built-in Datetime
- Python's built-in Math

Your task:

1. Create a function called Fill_Table that has one parameter, a ticker. This ticker will always be a string. Using pandas_datareader, download stock data for a given ticker from Yahoo starting from 1/1/2019 and ending on 12/31/2019. Take this data, iterate over the rows, and insert the row data into the table. One row in the dataframe equals one row in the table. Remember to convert the index to a timestamp.
    a. DB Browser for SQLite is a great piece of software to easily view an SQLite database. It may be useful to have for this question to make sure your table is populated correctly.

2. Next what you're going to want to do is modify your Daily_Returns function from the previous question. The function should still have one parameter, but it should be a list of adjusted close values. It should still return a list of daily returns.

3. Finally, you are going to want to modify your Monthly_VaR function from the previous question. It should only have one parameter, a confidence level with a default value of 0.05. Change the function to select the adjusted close values from the table. This function cannot use pandas_datareader.

Question 3 - Logic Question

Write a function, sum_ssmif, which takes a nested list as an input. This nested list will always consist of one outer list which contains one or more inner lists.

For Example: [[1,2,3,4], [1,2,3], [3,2,1,0, -1]] This example input has 3 inner-lists.

Your task is to compute and return the sum of all of the numbers, but there are a few things to check for:

a. If the index of an inner-list is even, you must *double* all values starting from (and including) the **first** occurrence of the number 9 in that list all the way to (and including) the **next** occurrence of the number 6 in the remainder of that list.

    1. If the list index is even:
    2. [1, 2, **9, 4, 6**, 3] should be valued as [1+2+18+8+12+ 3].
    3. [**9, 5, 6,** 9, 5, 6] should be valued as [18+10+12+9+5+6].

b. If the index of an inner list is odd, you must *triple* all the values starting from (and including) the **first** occurrence of the number 7 in that list all the way to (and including) the **next** occurrence of the number 4 in the remainder of that list.
    1. If the list index is odd:
    2. [1, 2, **7, 2, 2, 4**, 1] should be valued as [1+2+21+6+6+12+1].
    3. [**7, 1, 4,** 7, 1, 4] should be valued as [21+3+12+7+1+4].

c. Then, once you have a single list of values which were obtained from each inner-list, you must *ignore* any values in that list starting from (and including) the **first** occurrence of the number 4 all the way to (and including) the **next** occurrence of the number 5 in the remainder of the list.

d. If there is no end marker (6, 4, 5) after a start marker (9, 7, 4), you must return the normal sum of the list.
    1. [1, 2, **9, 4, 3**] should be valued as [1+2+9+4+3]
    2. [1, 2, **7, 2, 2, 1**] should be valued as [1+2+7+2+2+1]
    3. [41, **4, 6, 38**] should be valued as [41+4+6+38]

e. * It is important to keep all lists in the order given.

Example Test Case:

a. Input: [[1, 2, 3, **9, 2, 6**, 1], [1, 3], [1, 2, 3], [**7, 1, 4**, 2], [1, 2, 2]]
b. Simplifies to: [1+2+3+18+4+12+1, 1+3, 1+2+3, 21+3+12+2, 1+2+2]
c. Simplifies to: [41, **4, 6, 38, 5**]
d. Simplifies to: [41, 0, 0, 0, 0]
e. Correct output: 41.

If you have any questions or need clarification, feel free to reach out.

scaratoz@stevens.edu

Good luck!