Paige Rosynek

CS 3860 041

Dr. Magana

10.20.2022

**Lab 5 – Problem Solving with Data Analysis**

**Step 1: Portfolio Table**

The screenshot below shows the first 10 rows of the portfolio table; there are a total of 250 rows in the table. The SQL queries used to calculate the cumulative return for each stock is included at the top of the following page; I hard coded the start date for all the calculations for simplicity. Each of the UPDATE queries has two aliases for the portfolio table to avoid RW errors. Note: the screenshots below are of the same table (portfolio) but I included separate screenshots for readability.

A picture containing table

Description automatically generated

Table

Description automatically generatedA picture containing table

Description automatically generated

UPDATE portfolio p1, portfolio p2

SET p1.CELG\_cumulative\_return = p1.CELG\_adjusted\_close / (

    SELECT p2.CELG\_adjusted\_close

    WHERE p2.date='2016-10-06'

);

UPDATE portfolio p1, portfolio p2

SET p1.FB\_cumulative\_return = p1.FB\_adjusted\_close / (

    SELECT p2.FB\_adjusted\_close

    WHERE p2.date='2016-10-06'

);

UPDATE portfolio p1, portfolio p2

SET p1.GOOG\_cumulative\_return = p1.GOOG\_adjusted\_close / (

    SELECT p2.GOOG\_adjusted\_close

    WHERE p2.date='2016-10-06'

);

UPDATE portfolio p1, portfolio p2

SET p1.NVDA\_cumulative\_return = p1.NVDA\_adjusted\_close / (

    SELECT p2.NVDA\_adjusted\_close

    WHERE p2.date='2016-10-06'

);

UPDATE portfolio p1, portfolio p2

SET p1.SPY\_cumulative\_return = p1.SPY\_adjusted\_close / (

    SELECT p2.SPY\_adjusted\_close

    WHERE p2.date='2016-10-06'

);

**Step 2: Write a portfolio simulation function**

The code for the simulation function is included in the **lab05\_rosynek.py** file. The simulation function loads the entire portfolio table into a pandas Dataframe to allow for calculations to be made outside of SQL. It also allows for all changes to be committed all at once to update the table using less commits.

**Simulation 1**

**Input:**

Stocks = ['GOOG', 'CELG', 'NVDA', 'FB']

Allocation = [0.3, 0.3, 0.2, 0.2]

Text

Description automatically generated**Output: full table in ./lab5 results/weights03\_03\_02\_02.csv**

Table

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated with medium confidence

The first 10 rows and calculation results of the simulation function, using the inputs above, are shown above. It can be observed from the screenshots of the portfolio table, that the value of each of the stocks on the first day is equal to each of their respective weights because the cumulative return for the first date is always 1 and the assumed first-day value of the portfolio is $1. Based on the equation for calculating a stock’s daily value: weight of stock × portfolio value × daily cumulative return for stock, it can be observed that as the daily cumulative return for a stock decreases, the value also decreases. This is shown in the portfolio table above. The value for the portfolio reflects the cumulative return of the portfolio and the value of SPY reflects the cumulative return of SPY. The sharpe ratio calculated for the entire portfolio was approximately **22.876**. This means that with a portfolio with 30% of assets in GOOG, 30% in CELG, 20% in NVDA, and 20% in FB the ratio of the measure of excess return per unit of deviation relative to SPY is 22.876 dollars. The results of the simulation function produced the following values:

**STD of portfolio daily return =** 0.1283

**AVG of portfolio daily return =** 0.1856

**Overall portfolio cumulative return =** 0.5879

**Sharpe ratio =** 22.8761

**Simulation 2: Find a better sharpe ratio**

**Weights tried for stocks = ['GOOG', 'CELG', 'NVDA', 'FB']**

stock\_allocation = [0.25, 0.25, 0.4, 0.1]

sharpe ratio = 24.559035867825763

Text

Description automatically generated**Results: full table in ./lab5 results/weights025\_025\_04\_01.csv**

Table

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenceIn order to find a set of stock allocation that produced a better sharpe value, I arbitrarily chose weights. The first set of weights I tried was [0.25, 0.25, 0.4, 0.1], for the stocks: ['GOOG', 'CELG', 'NVDA', 'FB'], which produced a sharpe ratio of **24.5590** which is greater than the sharpe ratio from the previous section. Therefore, these weights produced a higher return with less risk than the original weights. The results of the simulation function with the defined weights produced the following values:

**STD of portfolio daily return =** 0.2072

**AVG of portfolio daily return =** 0.3219

**Overall portfolio cumulative return =** 0.8568

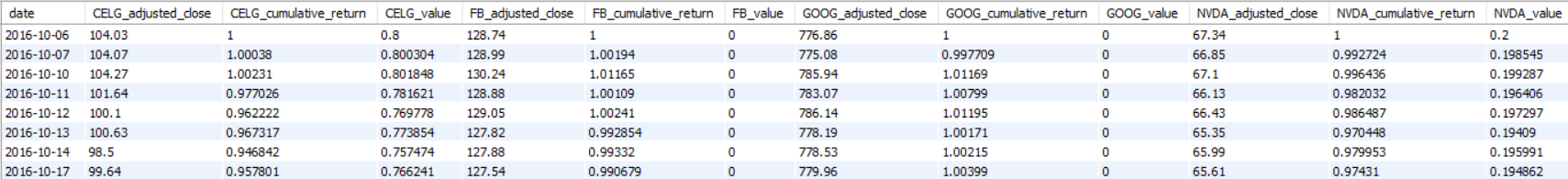
**Sharpe ratio =** 24.5590

**Step 3: Optimal sharpe ratio & stock weights**

**Optimal stock allocation:**

Stocks = ['GOOG', 'CELG', 'NVDA', 'FB']

allocation = [0.0, 0.8, 0.2, 0.0]

Table

Description automatically generatedText

Description automatically generated**Output: full table in ./lab5 results/max\_sharpe**

In order to find the set of stock allocations that maximizes the sharpe ratio of the portfolio, I looped through every permutation of the values from 0-1 with a step of 0.1 and stored the weights that summed to zero. Then for each of the test weights I ran the simulation function to find the optimal set of weights. The optimal weights were [0.0, 0.8, 0.2, 0.0] for ['GOOG', 'CELG', 'NVDA', 'FB'] and produced a maximum sharpe ratio of **26.8521**. Since the sharpe value is maximized, the following values are also maximized:

**STD of portfolio daily return =** 0.1294

**AVG of portfolio daily return =** 0.2197

**Overall portfolio cumulative return =** 0.6561

**Sharpe ratio =** 26.8521