**Group F**

**Lan Zhou, Linh Katie Cao, Nha Hoang**

**ISGB-79AA-001 – Advanced Python for Financial Programming – Summer 2019**

**Final Project Report**

From a financial perspective, our group wants to create an investment portfolio among popular stocks that has the highest risk-adjusted returns. Besides, we would also like to apply and incorporate technical methods of getting the data, creating a database and querying the data that we learn throughout the course to this project.

**Overview**

Our project seeks to find the portfolio with the optimal [Sharpe ratio](https://www.investopedia.com/terms/s/sharperatio.asp) using stocks from the S&P 500. To analyze the stocks, we will use one-year historical daily data scraped from Yahoo Finance. The project will make use of SQL tables as we will store the data within a PostgreSQL server hosted by Heroku. Once the data has been successfully loaded, we will download the data back from Heroku to perform analysis on it.

**Step 1: Getting the Data**

Our first step was to web scrape the Yahoo Finance website for historical data for the S&P 500. We get a list of the S&P 500 by using Beautiful Soup to get the ticker information from Wikipedia. Then, we use this list to download the adjusted close price for the tickers using the Yahoo Finance REST API and the Python Requests library. Once we had the data, we created a data frame that has the adjusted close as the columns and the date as the rows. This data frame containing all adjusted close price will then be uploaded to the Heroku PostgreSQL server.

**Step 2: Filtering tickers**

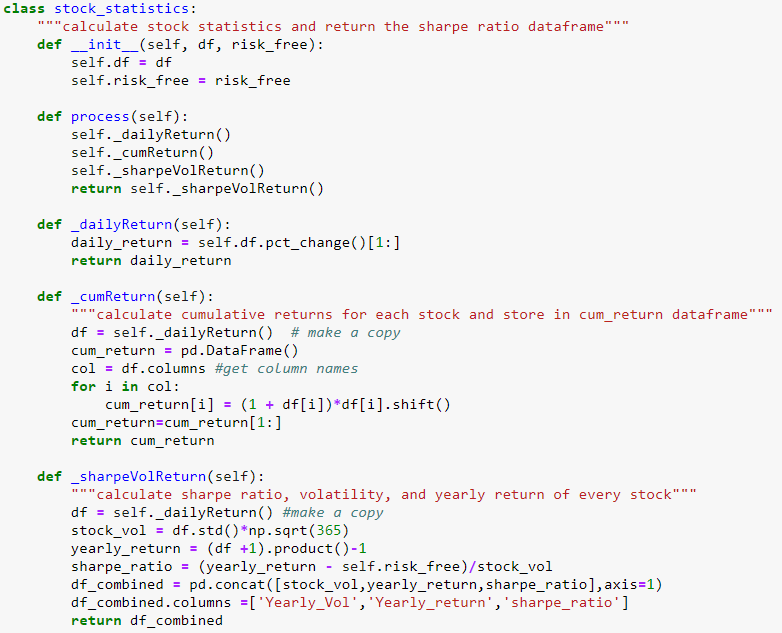
Creating a matrix of 500 x 500 would be a high computational cost. Therefore, we picked three methods to filter “better” stocks as followed. Filter 1 is the top 10 stocks with the highest average daily returns. Filter 2 is 10 stocks with the smallest standard deviation of daily returns, assuming a lower standard deviation of daily returns equals less volatility. A common criticism of the sharpe ratio is that it only works with stocks whose returns are normally distributed. Hence, we came up with Filter 3 which is to filter stocks with skewness in range of -0.8 to 0.8 and kurtosis -3.0 to 3.0 (Those are the classified parameters for a normal distribution). With Filter 3 we’re left with 232 stocks. However, for the time constraint of this project, we are only going to use Filter 1 and Filter 2 for further analysis. Another option would be to apply Filter 3 first and then apply either Filter 1 or Filter 2.

**Step 3: Performing analysis and measurements**

We then pulled all the data from Heroku Postgres server and saved it as a CSV file that we could easily convert into a data frame to perform further analysis. In this step, we constructed a class called stock\_statistics to calculate the yearly return and volatility of each stock, the cumulative return of the timeframe provided, and the sharpe ratio for each stock based on the excess return and volatility. We also constructed a class to visualize the results.

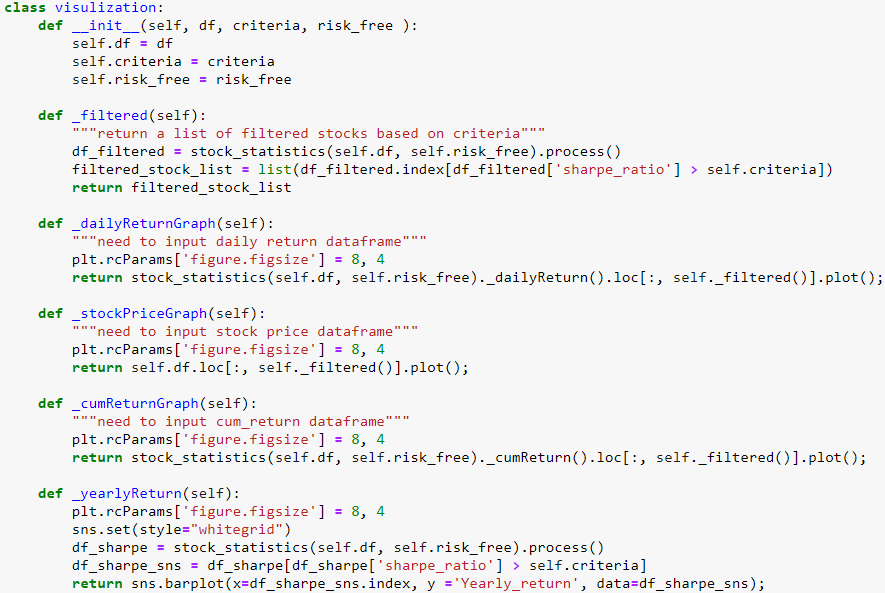
Since our data’s time frame is exactly one year, we took the time series daily return (which was calculated using pct\_change()), and compounded it for the entire year and minus one to get the geometric annualized return. We then calculated the standard deviation and annulized it by multiplying the square root of the length of the data frame (which is 251) to get the annualized volatility. Finally, we calculated the sharpe ratio by using the excess return, which is the yearly return minus the risk free rate (T-bill rate of 0.0212 on 6/22) , and divided by the volatility of each stock.

We calculated the cumulative return to see how stocks performed over a specific time period. It’s calculated by taking the daily return plus one and multiplies the daily return of the previous day until the end of the period. We later visualized the cumulative return and found out that it’s pretty flat year over year (6/2018 - 6/2019) since equity has a significant dip at the end of 2018 and it’s finally recovered after six months.

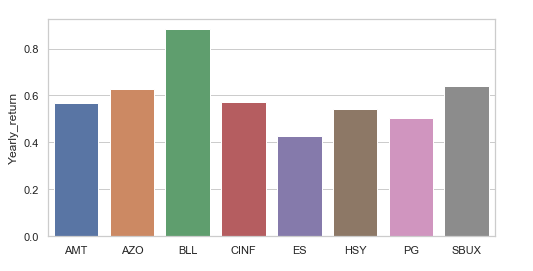


In the visualization class, we graphed the statistics based on our sharpe ratio criteria. The libraries used include matplotlib and seaborn. We first defined a function to filter stocks with sharpe ratio to the level or range that we desire (i.e. over 2 or between 1.5 and 3). We then built the function to graph stocks’ daily return, price movement, cumulative return and yearly return over that period of time.

Eventually, the user just needs to input the sharpe ratio criteria and update the risk free rate, and the class functions can be called to calculate and visualize the results.

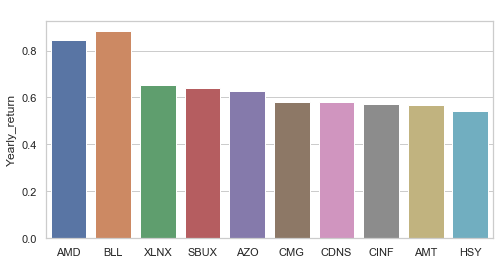


Yearly return for highest Sharpe Ratio stocks:

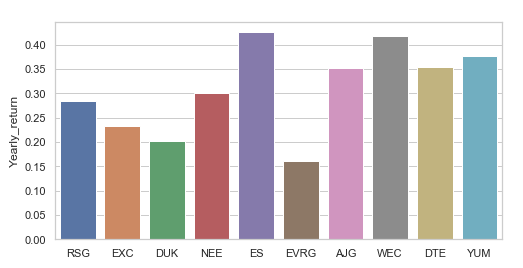


In addition to using the highest Sharpe ratio as a filter, we also created filters for the highest mean return and the lowest standard deviations. These were calculated using the pandas built-in mean() and std() functions. We used our visualization class that we created earlier to show plots for daily return, stock price, cumulative return, and yearly return:

Yearly return for highest mean returns:



Yearly returns for lowest standard deviation of returns:



**Step 4: Portfolio Construction**

For each potential portfolio, we picked the top 4 for each criterion and performed analysis to find the optimal weights that maximize the Sharpe ratio. To find the optimal weights, multiple portfolios with random weights were generated. The return and volatility for these portfolios were charted and the shape formed from the plot is the efficient frontier. On the efficient frontier exists a portfolio with the maximum return over volatility ratio, which is the definition of the Sharpe ratio.

We will use a hypothetical investment of $10,000 for each portfolio to compare its performance during the month of June 2019.

Our Sharpe Ratio filter gave us the following weights:

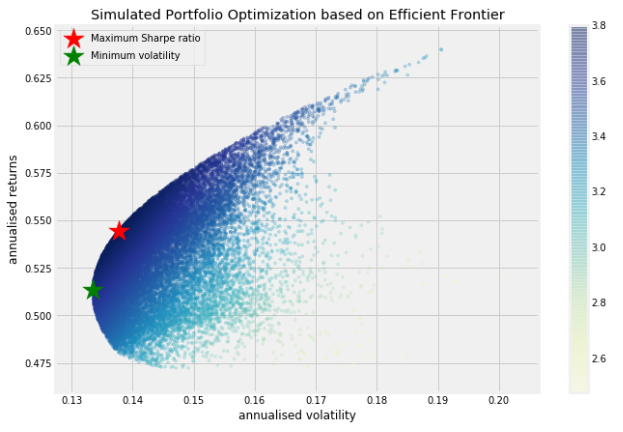
AMT - 32.55

BLL - 34.49

CINF - 16.29

SBUX - 16.66

If we actually allocated according to these weights to create a portfolio using a $10,000 investment, we would have a final balance of $10,699 for the month of June, a 6.99% return investment.



Our return mean filter gave us the following weights:

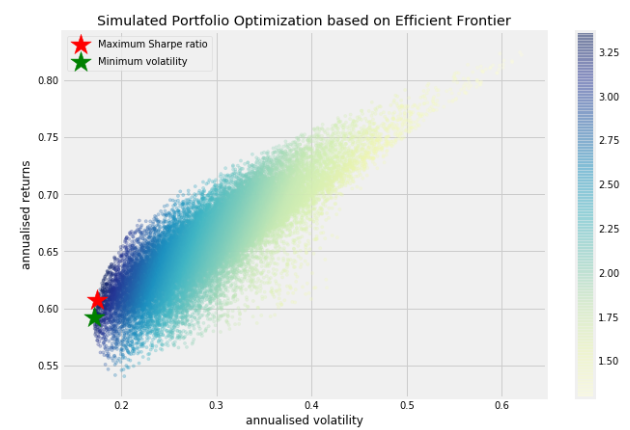
AMD - 0.24

BLL - 59.99

XLNX - 2.96

SBUX - 36.82

This method would have a final balance of $11,264 over the month of June, a 12.64% return investment.



Our standard deviation filter gave us the following weights:

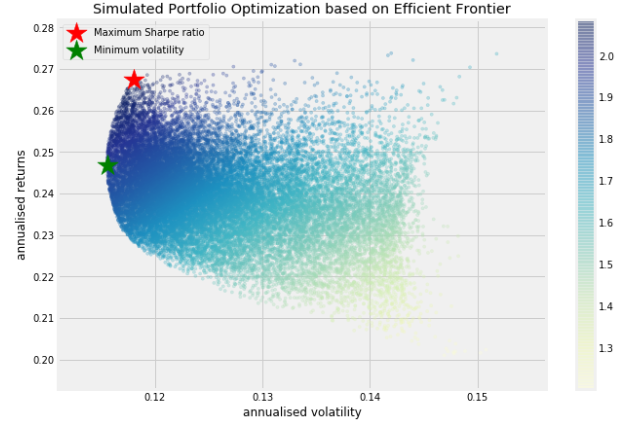
RSG - 55.33

EXC - 0.99

DUK - 0.16

NEE - 43.52

This method would have given us a $10,305 final balance, with a return of 3.05%



**Conclusion:**

**Out of the three portfolios, the highest average daily return filter gave the best performance over the month of June**. The stocks with the highest means can be very volatile, but optimizing the Sharpe ratio means that we are using diversification to minimize the effects of the individual stocks’ volatility. We would assume that the standard deviation filter would give us a lower return, but will most likely have the lowest portfolio volatility as well. The Sharpe ratio filter will most likely give us a portfolio that has better return than the standard deviation filter and lower volatility than the return mean filter. To fully draw conclusions for the dataset, we would want to continue our analysis for a year to see how the volatility and returns are affected.