Mike Chen
1006274693
December 19th, 2021

PHY407 Final Report

Trading algorithms using different computational techniques

Written in the "lab" style

Financial Background:

The stock market refers to the group of venues where the buying and selling of shares of publicly held companies occurs. This is primarily done as a means to make money since the value of shares of any given company changes based on information that is released. For example, if a company reveals that they have grown a large amount in the past year, the stock will (generally) increase to reflect this. If a company goes bankrupt the share prices will go to 0.

It is a well established fact that a well-diversified portfolio generates the best returns on average. Stocks such as SPY which track the S&P 500 (a collection of stocks of the 500 biggest companies across many sectors) are commonly known to be sound investments for investors.

Figure 1: A graph of the performance of the stock SPY over time

In fact, the vast majority of fund managers (people who are hired to manage other people's money) do not beat the S&P 500. As such the S&P 500 has become the benchmark to beat for any computer algorithm, though few succeed. We will be examining 3 different algorithms and comparing their performance to SPY.

Datasets relevant to report:

407 csv files in the format (Stock name).csv. Each one contains the price of the stock each day for a year. Downloaded through the use of a list of stock names and a program that is included in the files.

Computational Background:

General Algorithm background:

For all algorithms the general structure is the same:

1. A portfolio, this is a data structure which records the number of each stock the algorithm has.
2. A method to determine which stocks in the portfolio should be sold.
3. A method to determine which stocks in the stock market should be bought

There are also general functions which must be created:

1. A method to retrieve the stock price at a given time.
2. A method to update a portfolio given a list of stocks to sell
3. A method to update a portfolio given a list of stocks to buy
4. A method to determine the value of the portfolio at any given time

Random Number Algorithm:

There is an apocryphal tale of investors in Wall Street throwing darts at a stock list and beating the S&P 500. Even in the modern day there are hamsters which are successfully outperforming the best investors in the world. Because of this many people would say investing is up to luck. We will test this saying by implementing a random algorithm. That is, an algorithm which decides whether to buy or sell a stock on a coin flip. The algorithm will generate a random number using uniform distribution, and decide whether to buy or sell a stock depending on the result.

Linear Regression Algorithm:

Linear regression is a commonly used tool in the field of finance. It is the act of taking data points and modelling a linear relationship that approximates the trend of the data. The function used to model the relationship between any given stocks price and time will be:

$$y = ax + b \quad (1)$$

Where y is the stock price and x is the time in days. The variables a and b will be fitted using the function curve_fit(). By taking the “a” value we can observe the general trend of the stock and thus determine whether to buy or sell the stock. As such, a Linear Regression algorithm will buy the stock when the “a” value is positive and thus trending up and sell when it is negative and thus trending down.

Monte Carlo Simulation and Covariance Matrix Algorithm

The Monte Carlo Simulation and Covariance Matrix algorithm (from now on shortened to Monte Carlo) is another commonly used tool in the field of finance[1]. It is primarily used to assess risk in investments and model which portfolios are optimal for any given situation[2]. It combines matrix operations with Monte Carlo simulations. The Monte Carlo algorithm has multiple steps which are the following:

1. Choose a list of stocks[4]
2. Get the price history of each stock[4]
3. Calculate the geometric daily returns of each stock[4]
4. Use Monte Carlo simulation to generate many iterations of the portfolio[4]
5. Calculate expected return of each portfolio[4]
6. Calculate the risk of each portfolio by using a Covariance matrix[4]
7. Calculate the Sharpe ratio to determine which portfolio is optimal and invest based on that portfolio[4]

Steps 1 and 2 are self explanatory. To do step 3, use the formula:

$$Return = Log(\frac{Today's\ price}{Yesterday's\ price}) \quad (2)^4$$

To do step 4, partition the cash in your portfolio randomly into the same number of groups as the number of stocks in the list of stocks. This should be done with uniform randomness.

To do step 5, first calculate the mean of the returns for each stock with the formula:

$$Asset\ expected\ return = \frac{Sum(Returns)}{Total\ number\ of\ returns} \quad (3)^4$$

Then calculate the expected return with the formula:

$$Expected\ portfolio\ return = Sum(Weight * Asset\ expected\ return)\ for\ each\ asset \quad (4)$$

To do step 6, you must first create a covariance matrix. This is done with the following steps:

1. Calculate the average price of each stock[4]
2. Create a matrix where each column of the matrix is the prices of a stock over time[4]
3. Subtract each corresponding column of the matrix with the average price of each stock[4]
4. Multiply the resulting matrix by its transpose to get the covariance matrix[4]

You can then calculate the standard deviation of each portfolio with the following formula:

$$Standard\ deviation = \sqrt{weights\ vector\ *\ covariance\ matrix\ *\ Transposed\ weights\ vector} \quad (5)$$

To do step 7, you use the following formula:

$$Sharpe\ ratio = \frac{Portfolio\ expected\ return}{Standard\ Deviation} \quad (6)^3$$

In theory, the portfolio with the highest Sharpe ratio should be the one which has the largest possible returns at the smallest possible risk.

Questions

Important: All questions are ran at the same time using the file PortfolioManager

Note: All portfolios should start off with $100,000.00 USD.

1. Starting off and Random numbers in Investing

Implement the Random algorithm and track its performance over a 6 month period of time, also implement the basic functions which will be used for all future algorithms (name it PortfolioFunctions) and the program which controls all the algorithms (name it PortfolioManager).

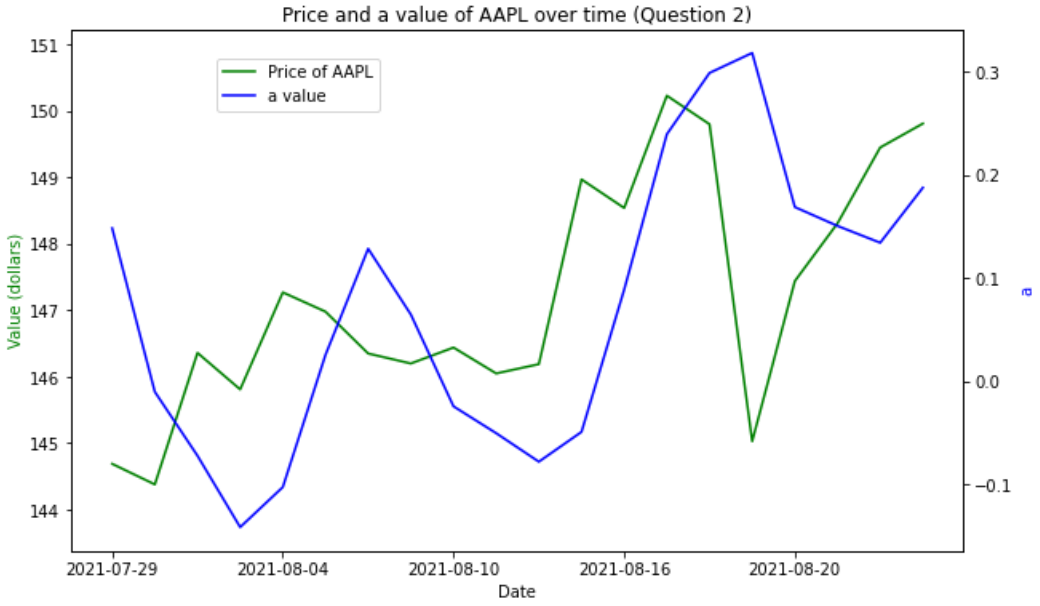
Nothing to hand in yet, save the data for question 4

2. Linear Regression in Investing

a) Use linear regression (equation 1) on the 6 months of price history on a stock to get the "a" value. Is this a good predictor of price movement of the stock? Plot the "a" value and the price on the same plot and compare.

Post plots of the "a" value and the price being on the same plot for a stock of your choosing. Comment on the result.

Figure 2.1: Price and "a" value of AAPL stock over time



As can be seen in figure 2.1, the "a" value tracks the price movement of the stock. However, it always trails behind it and does not seem to be a good predictor of the future price of the stock. In other words, past price movement does not indicate future price movement.

b) Implement linear regression into the trading algorithm and use the "a" value to determine whether to buy or sell. Vary the thresholds of "a" needed to buy and sell stocks, what is a good performing threshold? Did you notice a general pattern with the thresholds and how well the algorithm performed?
Don't hand in anything for this question yet, save the data for question 4. Comment on the result.

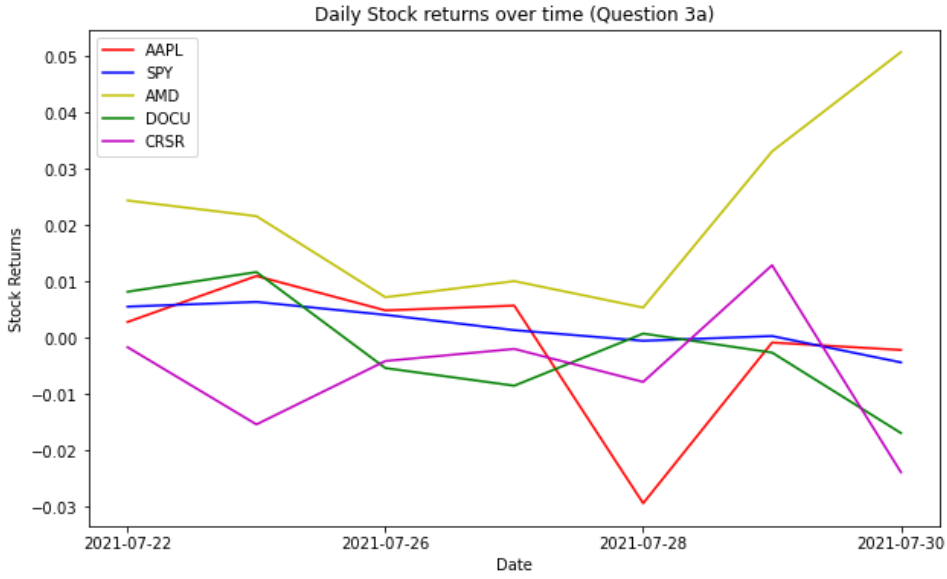
Note: There can be many answers, but the person should explain their reasoning for why their threshold makes sense

A threshold that was found to perform well was by having the algorithm buy shares if the "a" value was above 1, and sell shares if it was below -1. This is likely because high buy and sell thresholds makes it so that the algorithm only acts on extremely strong trends instead of minor fluctuations. In general, it seemed like the magnitude of the thresholds was positively correlated with the algorithm's performance.

3. Monte Carlo simulation and Matrix Operations in Investing.

a) Select 5 stocks of your choosing to build your portfolio with, then calculate the daily returns of these stocks using equation 2. Do this for 7 days and then use this data to plot the daily returns of the stocks on a single plot.
Post plots of the stocks returns over time.

Figure 3.1: Geometric daily stock returns of different stocks over time



b) Use the Monte Carlo simulation method to generate 10000 portfolio's with a uniform random distribution of cash in the stocks. Plot the value of a few of the portfolios over

time. Then simulate the expected returns of those portfolios using equation 3. Then plot the same portfolios alongside their expected returns. Does the expected returns match up well with what actually occurred?

Post plots of the portfolios value over time, and then the same plot except with expected returns as well.

Figure 3.2: Different Portfolio Values over time using Monte Carlo Simulation Method

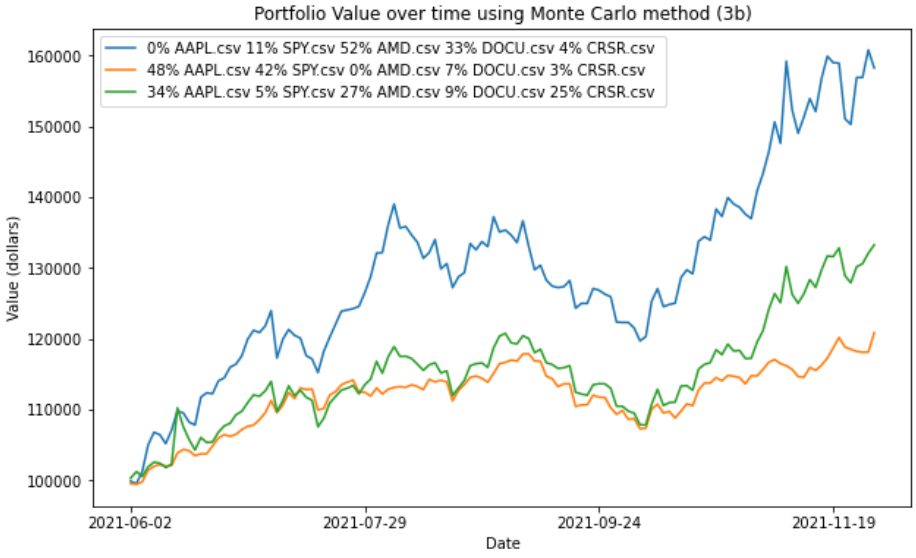
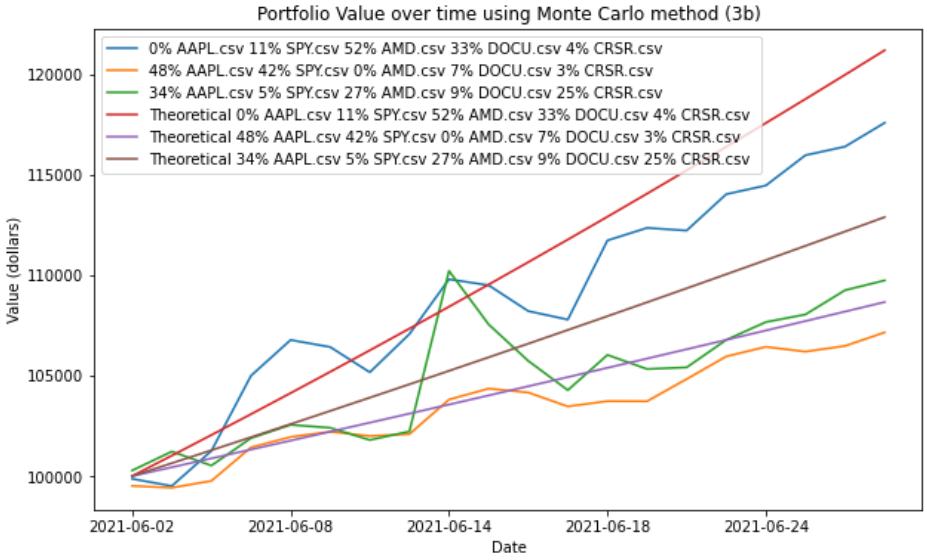


Figure 3.3: Different Portfolio Values over time along with their Theoretical returns



The theoretical returns of each portfolio initially track the real portfolios closely but drift far apart as time increases. This supports the claim in the question 2.a) that past stock performance does not have a strong correlation with future performance.

c) Expected return is not the only factor that the monte carlo method needs to consider. Another factor is volatility. Calculate the volatility of each portfolio by performing step 6. Then calculate the sharpe ratio using equation 6. Calculate the sharpe ratio of all the

portfolios, then plot all of the portfolios where the y axis is the expected return and the x axis is the risk. Make sure to make the following portfolios especially noticeable:

1. Portfolio with highest Sharpe Ratio
2. Portfolio with lowest Sharpe Ratio
3. Portfolio with lowest standard deviation
4. Portfolio with cash evenly distributed between the stocks

Then plot the above portfolios actual returns and comment on how the portfolio's performed. Which portfolio performed the best? The worst? Comment on whether it went according to your expectations.

Post a plot of the portfolio's value over time and comment on the results.

Figure 3.4: Portfolios generated with Monte Carlo simulation and plotted with risk-return

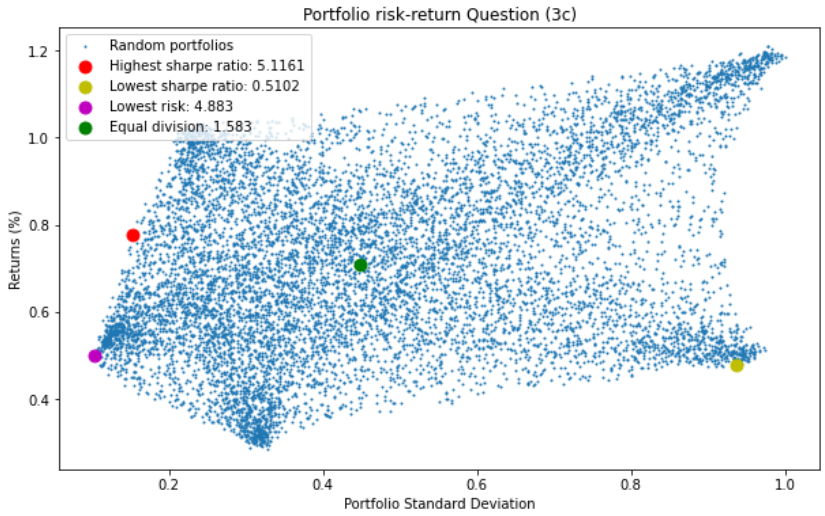
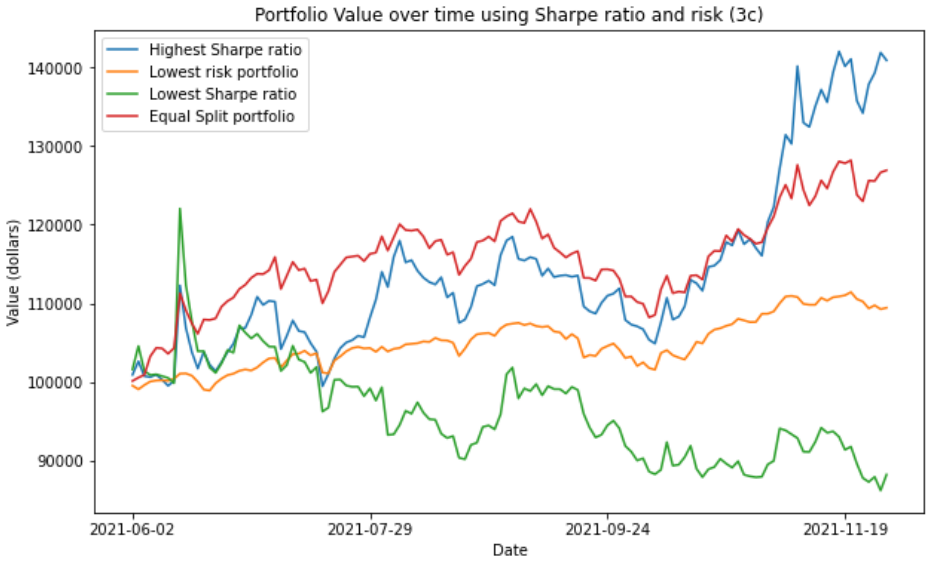


Figure 3.5: Portfolio value over time where the portfolios were selected via risk factors and Sharpe ratio.

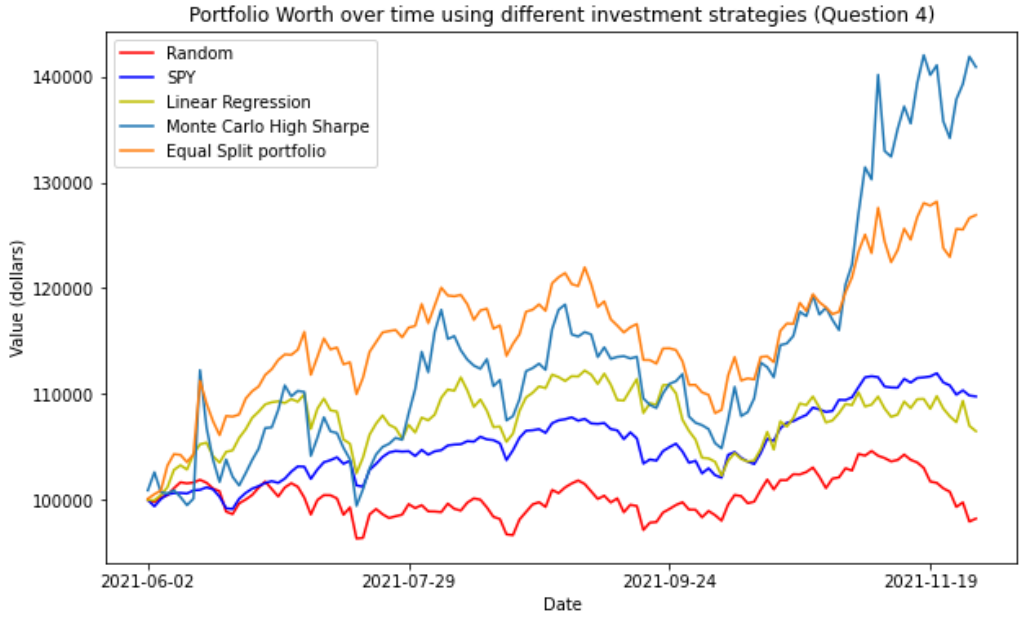


As expected, the portfolio with the highest Sharpe ratio performed the best. However the equal split portfolio followed the highest Sharpe ratio portfolio closely and only slightly lost to it. This shows how a well diversified portfolio tends to do well. The lowest sharpe ratio portfolio performed the worst as expected. This makes sense since its risk-return ratio was found to be the worst. The lowest risk portfolio remained fairly stable which is to be expected, without taking risk there is little chance to profit.

4. Plot Analysis

Plot the value of the portfolios of the random, linear regression and Monte Carlo algorithms over time. For the Monte Carlo algorithm use the portfolio with the highest Sharpe ratio and the one with cash evenly distributed. Also plot the SPY base case (a portfolio fully invented in SPY stock). Comment on the performance of each of the algorithms relative to each other. Does your plot support what was claimed in the financial background section?

Figure 4.1: Portfolio values over time using the investment algorithms of previous questions



From figure 4.1, we can see that the Monte Carlo High Sharpe ratio portfolio significantly outperformed the other algorithms. This supports the use of the Monte Carlo algorithm as an effective way to assess risk and invest in the optimal portfolio. The second and third best portfolios were the well diversified one and the SPY base case portfolio. This shows that a well diversified portfolio that tracks the market is still an extremely effective option which tends to beat other investment methods. The linear regression algorithm performed slightly worse than SPY which makes sense since it tended to chase the stocks with high returns such as the ones in SPY, effectively making it a worse SPY. The random portfolio performed the worst, actually losing money. This makes sense since the algorithm is random, and as such has no way of actually figuring out how to make money.

5. Error and efficiency Analysis

Several very strong assumptions were made while this lab was performed, name some of them and explain why they may cause the results to be different from what you obtained in this lab. Also discuss any other possible sources of error. Also explain the main bottlenecks in efficiency and give a rough estimate using big O notation. Additionally discuss how you know your results are bug free.

There are several factors for why the values of the portfolios may be off by a small margin. The first is slippage. For every buy order there must be a seller of the stock, slippage refers to the concept that the price you offer to buy a stock at may not be filled even if the stock is valued at that price. This may cause the price you buy or sell the stock at to be slightly off. Another factor that needs to be considered is commission fees, which are the fees you pay each time you make an order. The commission fee is dependent on the platform but it means that in general, the profit you make on any given trade is less than what you may expect. However, given the commission fees are usually small ($5 per trade is considered high), at the portfolio values we are using the commission fees are mostly negligible since they occupy a tiny percentage of the value of the trades. The final factor is the price of the given stocks, the algorithms took one price point for each day for simplicity, but the price of a stock fluctuates significantly every day. As such if the algorithm takes too long to execute, the stock may change in price enough to make the algorithm meaningless. Thankfully the program executes fairly quickly, so this is not a significant issue unless the stock is extremely volatile.

Floating point operations are a significant point of concern. As more floating point operations are executed, the error may propagate to a significant point. Thankfully there were ways to diminish the effect it has. In the SPY base case and the Random algorithm, no floating point numbers were used. In the Linear regression algorithm where the margin of error needs to be small, the only case where floating operations occur is when the data is being linearly regressed. The Monte Carlo algorithm has a significantly higher tolerance for error since the portfolio's which are found to be the best are usually significantly better. 6 floating point operations are done which correspond with steps 2-7 of the algorithm. While this is significant, the highest Sharpe values which are compared at the end would all be good portfolios and their expected returns would all be almost the same.

The speed at which the code runs is important for an algorithm's success. The code which runs all the algorithms and generates all the graphs takes 101.66 ± 3 seconds to run. This is a fairly reasonable time since it is processing multiple portfolios and simulating a large period of time of over 6 months. The efficiency for the more complicated algorithms is mainly bottlenecked by the number of stocks which must be searched through, the number of days which must be simulated and the number of possibilities which must be traversed. Below is a table giving the O(n) values for each algorithm:

Figure 5.1: Table showing the runtimes for each algorithm with big O notation

Algorithm name:	O(n):
SPY base case	O(n)
Random Algorithm	$O(n^2)$
Linear Regression Algorithm	$O(n^4)$
Monte Carlo Algorithm	$O(n^5)$

Given that data processing must always occur, the complexity of an algorithm means that the runtime can quickly ramp up and there is little that can be done to make the speed faster. However runtime can be improved by curating the list of stocks so that the number of data points which must be traversed can be decreased. Memory is also an issue with algorithms, with a large amount of stocks the number of possible iterations of a portfolio can quickly spiral out of control. This can be curated by culling the portfolios which are too similar to each other and removing old portfolios which are no longer relevant as well as curating the stock list.

Any errors or bugs in an algorithm would be devastating and as such must be checked carefully. The first check is done by plotting each step of the algorithm like what was done in question 3 for the Monte Carlo algorithm to make sure the numbers and plots of each step are reasonable. Since each step seems to have reasonable numbers, we can be fairly certain that all the steps culminating in the end result are correct. To check to see if the end result is correct, we can implement a simple base case. This is done by first putting in a SPY base case. This is a portfolio which is entirely invested in SPY. In other words it is a check of the algorithm against the market. If an algorithm performs extremely poorly or well relative to the base case, we can be fairly certain that there is an error somewhere. Another base case was implemented in the Monte Carlo algorithm by putting in an equal split portfolio. We know that the equal split portfolio should perform significantly better than the portfolio with the lowest Sharpe ratio and the lowest risk since they should underperform the market. We also know that it should usually be worse than the portfolio with a high Sharpe ratio. In our case, we can clearly see that this is the case. As such, since we have verified that every step is correct and the end result is correct, we can be confident that the algorithm is accurate. Another form of bug testing implemented is by testing edge cases. This is done by putting in extremely high and low numbers in different parameters and seeing if the algorithm can handle it. Another edge case that was tested was by passing in faulty or missing data. Companies can go bankrupt and thus have no entry for value. These were all accounted for by the code. As such we can be reasonably sure that the algorithms can handle most cases.

References

1. Art B. Owen. "Monte Carlo Theory, Methods and Examples, Chapter 1 End Notes," Page 10. Stanford University. Accessed May 11, 2020.
2. Massachusetts Institute of Technology. "Explained: Monte Carlo Simulations." May 11, 2020.
3. Farhad Malik. Towards Data Science. “Best Investment Portfolio Via Monte-Carlo Simulation In Python.” Aug 18, 2019
4. Varun Divakar. “Calculating The Covariance Matrix And Portfolio Variance.” Dec 27, 2018