IEOR 231: Stock Simulation and Portfolio Optimization

Irene Zheng
Jiaqing Li
Shichen Wu
Yichao Dai

May 12, 2023

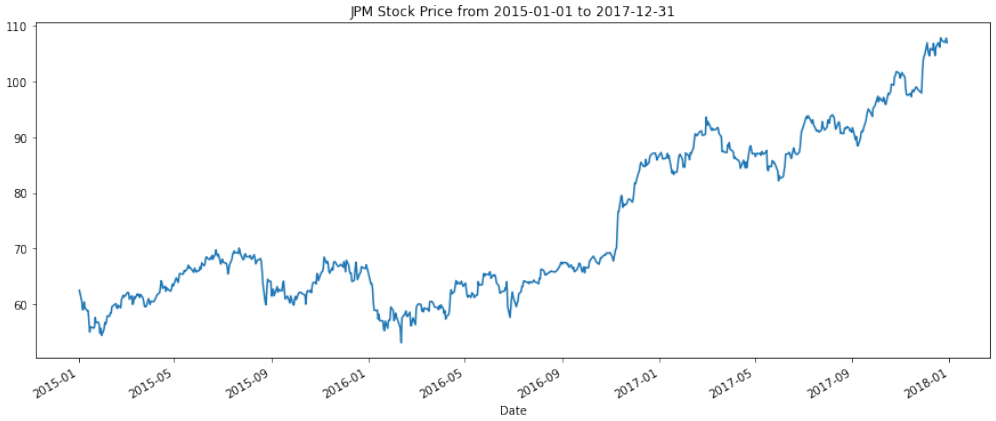
1 Introduction

In this project, we aim at predicting the stock prices and determining the optimal portfolio of stocks based on Monte Carlo Simulations, using analytic techniques such as Brownian Motion, Reinforcement Learning etc. Portfolio optimization is the process of creating a portfolio of assets such that your investments have maximum return and minimum risk. We will design a financial system that can both predict the stock prices and optimize the stock portfolios as long as users input the time and target stocks.

One challenge that we are facing when optimizing the portfolio using RL is how to precisely simulate the actual trading environment since the market is continuously changing from time to time. Besides, when defining the trading actions, instead of mimicking buy, sell and hold, is there any other more efficient way to construct the actions. In terms of Monte Carlo Simulation Method, another challenge is we need to make a tradeoff between simulation efficiency and accuracy. Increasing sample size and variance reduction may increase the simulation accuracy, however, overall complexity may exponentially increase. Consequently, we plan to apply Quasi Monte Carlo Simulation by choosing a deterministic sequence of points to tackle this challenge. Please refer to our code on **github.com** for more details

2 Stock Price Prediction

Monte Carlo simulations are a type of computational algorithm that use random sampling to model complex systems or processes that involves probabilistic or random variables. During the simulation process, they run an enormous amount of trials with different random numbers generated from an underlying distribution for the uncertain variables. We intend to use Monte Carlo simulations to support our stock data prediction and portfolio optimization. We selected 4 stocks and downloaded data from '2015-01-01' to '2017-12-31' from Yahoo Finance: JPMorgan Chase, Citi, Bank of America, and Wells Fargo. We first monitor the price fluctuation like the plot below for each stock.



2.1 Define the Model

We use Brownian Motion to estimate the return. It is a stochastic process that could model the random movement of particles in a fluid. Brownian Motion has two components, drift and volatility. Drift is the direction that rates of returns have had in the past. It is the expected return of the stock. We calculate the drift using

$$\text{Drift} = u - \frac{1}{2} \cdot \text{var} \tag{1}$$

Volatility is the historical volatility multiplied by a random standard normal variable.

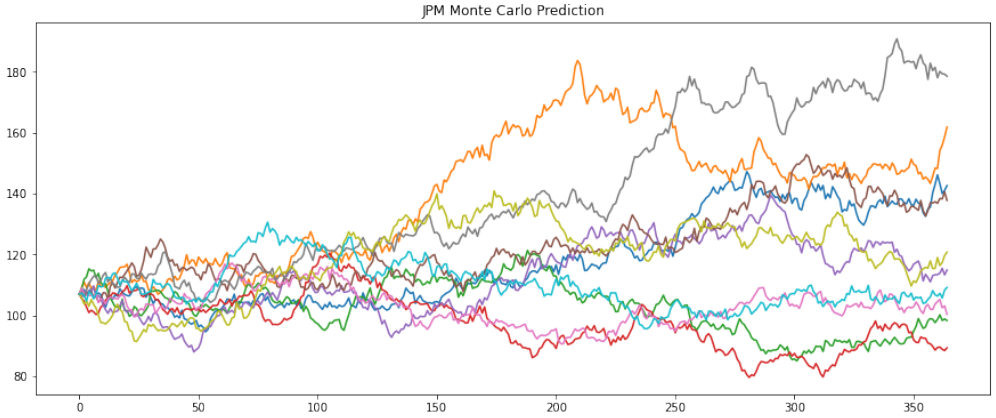
$$\text{Volatility} = \text{Std.Dev} \cdot Z(\text{Rand}(0,1)) \tag{2}$$

Therefore, the asset pricing equation is

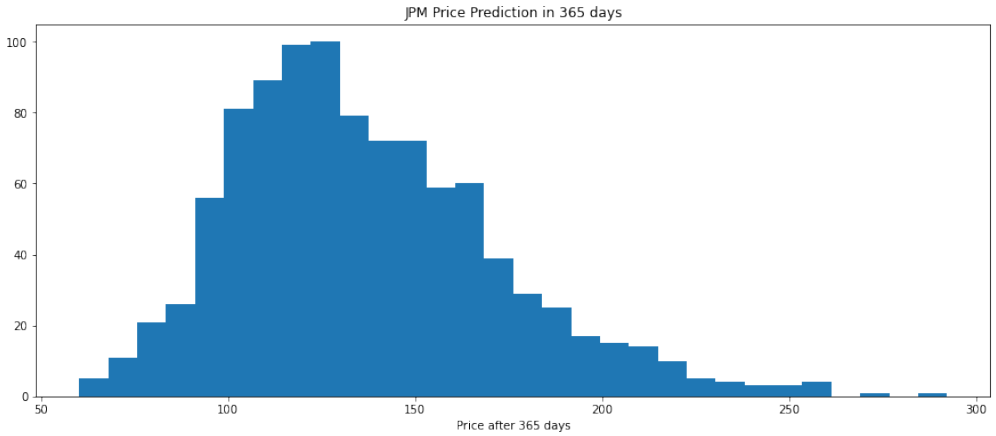
$$\text{PriceToday} = \text{PriceYesterday} \cdot e^{\text{drift}+\text{Volatility}} \tag{3}$$

2.2 Generate Random Scenarios

We then generate a large number of random scenarios for the possible future stock price paths based on the model parameters. Each path will be a series of simulated stck prices over time. We set up 1000 iterations and are interested to see the stock price after 1 year (365 days).



The figure above shows 10 of the 1000 trials of JPMorgan's predicted stock prices using Monte Carlo prediction for the next 365 days.



The figure above presents JPMorgan's predicted stock prices distribution after 365 days.

2.3 Calculate Evaluation Metrics

We evaluate the simulation results using mean absolute error (MAE) and mean squared error (MSE) by comparing the predicted stock price with the actual price. By definition, MAE is a measure of the average magnitude of the errors between the predicted and actual values. It is calculated as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

MSE is a measure of the average squared differences between the predicted and actual values. It is calculated as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

We then were able to obtain the baseline MAE and MSE.

3 Simulation Optimization

The simulation of stock usually has a lot of uncertainty, which comes from random variables outside the data. However, the simulation results can be optimized by certain methods. At a high level, we can generally increase the complexity of simulation models by increasing their complexity, but increasing complexity also leads to increased computational requirements and decreased interpret ability. In this section, we are going to talk about ways that can optimize the stock simulation result evaluated by Mean absolute error (MAE) and Mean squared error (MSE). First, we will use the variance reduction method, which is a technique used to reduce the variance or uncertainty in the simulation output, in order to improve the accuracy and efficiency of the optimization process. Second, we will use Markov chain Monte Carlo (MCMC), a widely used method for optimization simulation, to efficiently sample from complex probability distributions, for further optimization.

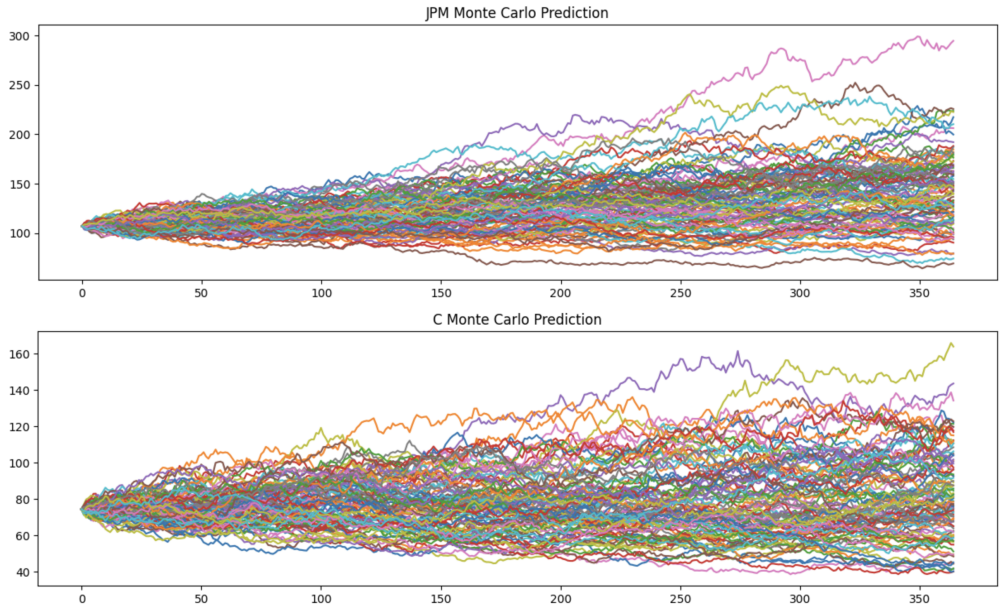
Last but not the least, we are going to apply Quasi-Monte Carlo. The QMC method is an alternative to the Monte Carlo method and is especially useful when the integrand is smooth and the dimensions of data are complex. All analysis are performed based on the optimization result of previous 4 stock data (JPM, Citi, WFC, BAC).

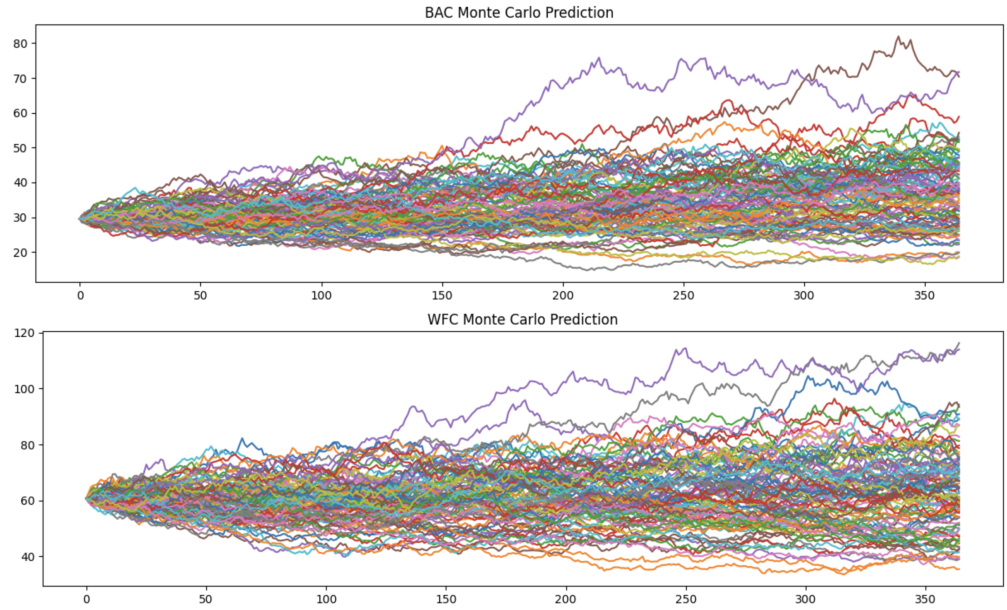
3.1 Variance Reduction

There are several common variance reduction methods used in simulation optimization, including Antithetic Variate Method, Control Variate Method, and Importance Sampling Method. All these Methods are aimed to reduce the variance of the simulation output, however, different method may involve in different sampling method; as a result, different level of complexity are introduced to the models.

3.1.1 Antithetic Variate Method

The Method involves simulating from both the original data and a mirrored version of the data. For the stock data, we generate two sets of random variables (in this case, daily returns) for each time interval, and using the average of the two sets to estimate the expected value. By simulating the mirrored version of the data, the variance of the estimate can be reduced since the two sets are actually from the same system as the origin data does, as a result, they will cancel out some of the random fluctuations. The following 4 plots show the 100 simulation result by applying the AVM. We can see the variance of first 100 days of JPM definitely converge to a certain level, however,the last 100 days are more variant than the baseline model. Also, according to the table of the comparison, we can see that JPM are strongly affected by the introduced variables.



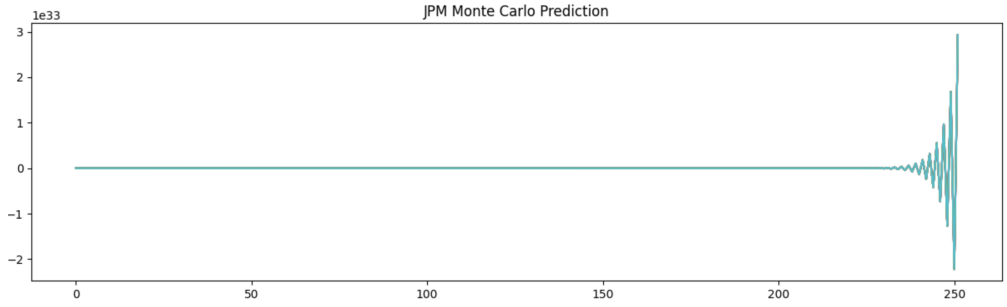


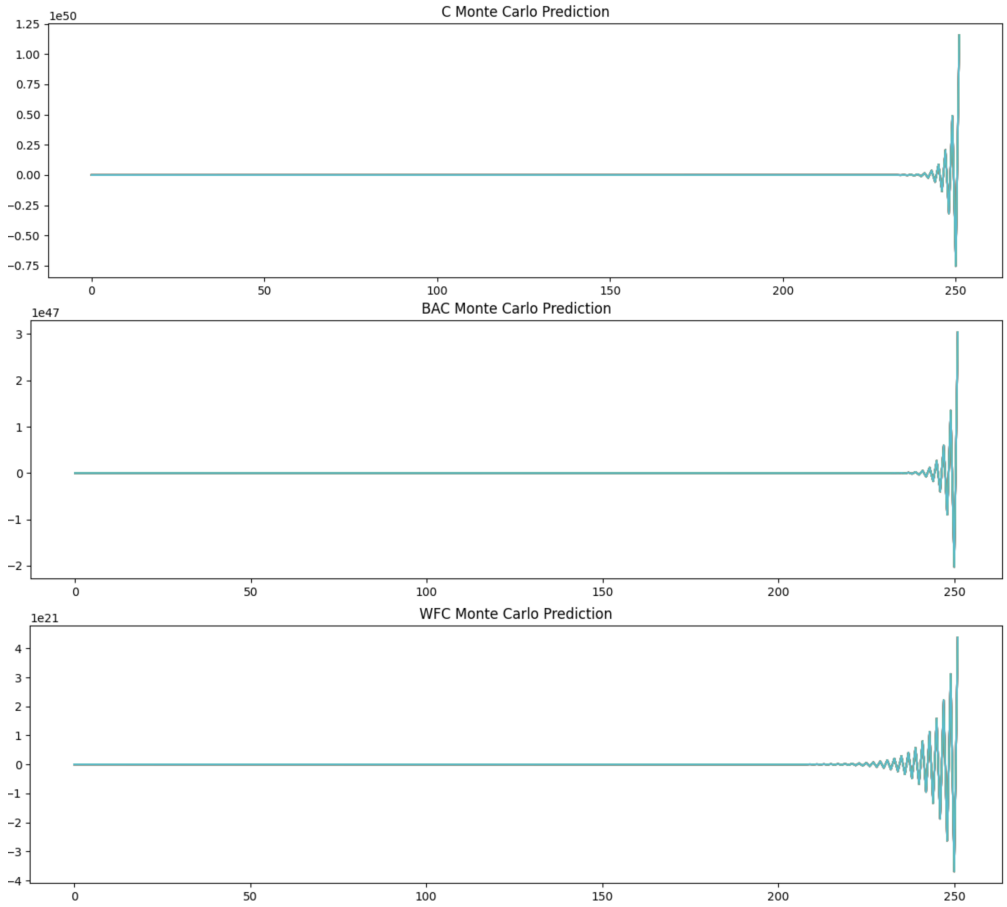
However, if we look at the other 3 plots, although the trends are not significant, the variance are actually reduced due to its symmetric distribution features. This is because AVM has advantages to apply if the data inputs have a symmetric distribution around zero. Carefully looking into the data, we can say the Data input of JPM are slightly left skewed, however, the average return of other 3 stocks are more closed to zero. That is also why AMC are show positive effect on Citi and BAC stock.

Stock Name	Baseline-MAE	AVM-MAE	AVM-MSE
JPM	32.21	38.30	2479.09
Citi	24.59	16.31	310.95
BAC	10.38	9.92	169.54
WFC	14.21	15.02	404.41

3.1.2 Control Variate Method

Control Variate Method (CVM) is another method to reduce the variance of simulation. It introduce an auxiliary variable that is correlated with the output variable of interest. Different from the AVM, CVM try to use the variable outside of the original data system. For particular stock data, the auxiliary variable can be S&P 500 index or Doe Jones' Index. By using the auxiliary variable to control and limit the output, the variance can be reduced to a certain level.



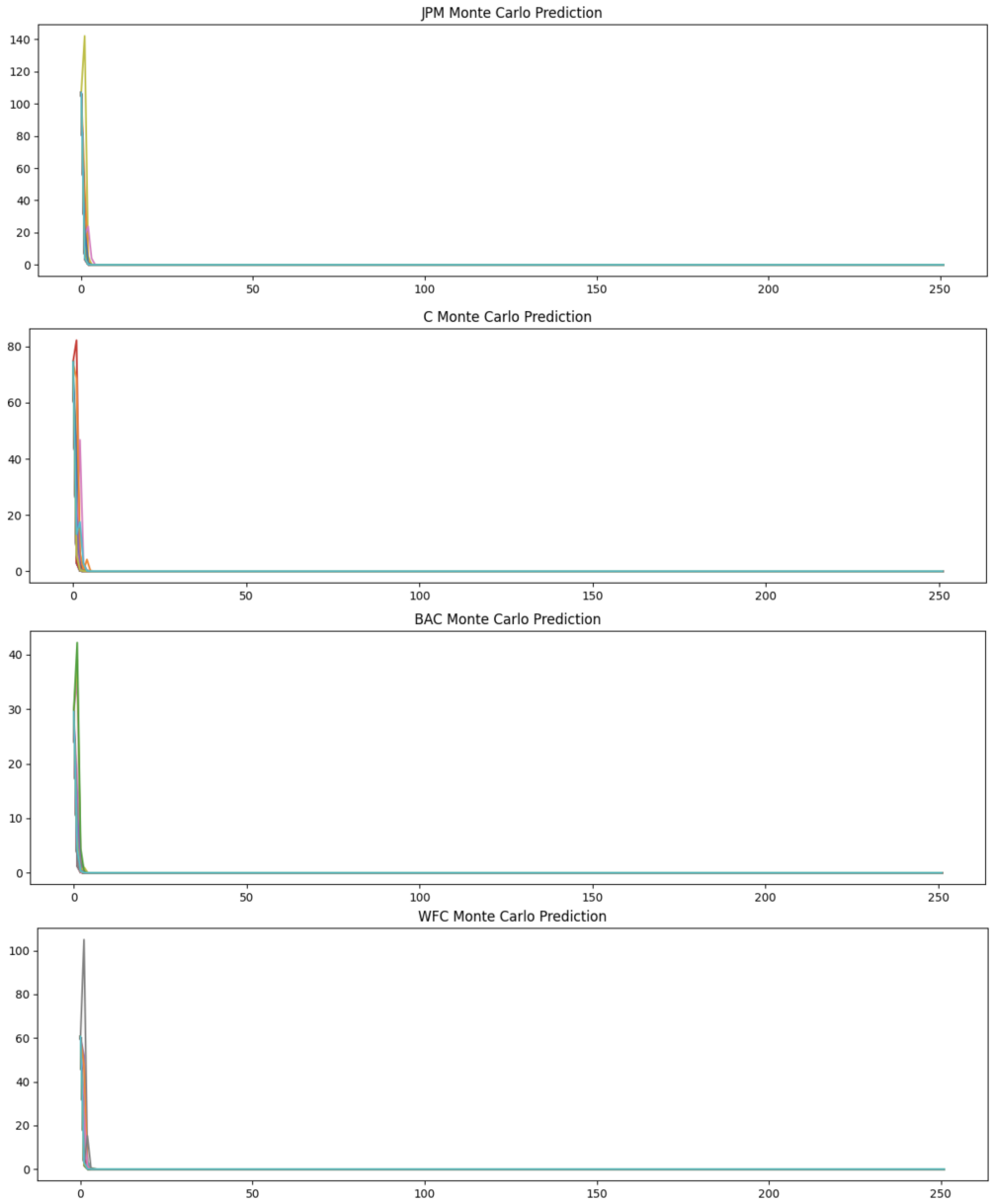


However, this method is the worst method in all our optimization method. It is particular difficult to select the right auxiliary variable. In our project, we select the S&P 500 index as the auxiliary variable, but it seems that the last 50 data are extremely fluctuated. and the effectiveness of the method depends on the quality of the chosen variable. Moreover, the selection of auxiliary variable should be correlated with the output variable of interest. It is hard to determine that which variable has this correlation, which make the entire optimization less effective and accurate.

Stock Name	Baseline-MAE	CVM-MAE	CVM-MSE
JPM	32.21	Worse	Worse
Citi	24.59	Worse	Worse
BAC	10.38	Worse	Worse
WFC	14.21	Worse	Worse

3.1.3 Importance Sampling Method

Importance Sampling Method involves sampling from a probability distribution that is different from the original distribution used in the simulation. It is similar to the CVM, which require extra system outside the original data. The careful-selected distribution may reduce the variance of simulation to a certain level. Here are the 4 plots show the simulations result by introducing the log-normal distribution random variables.

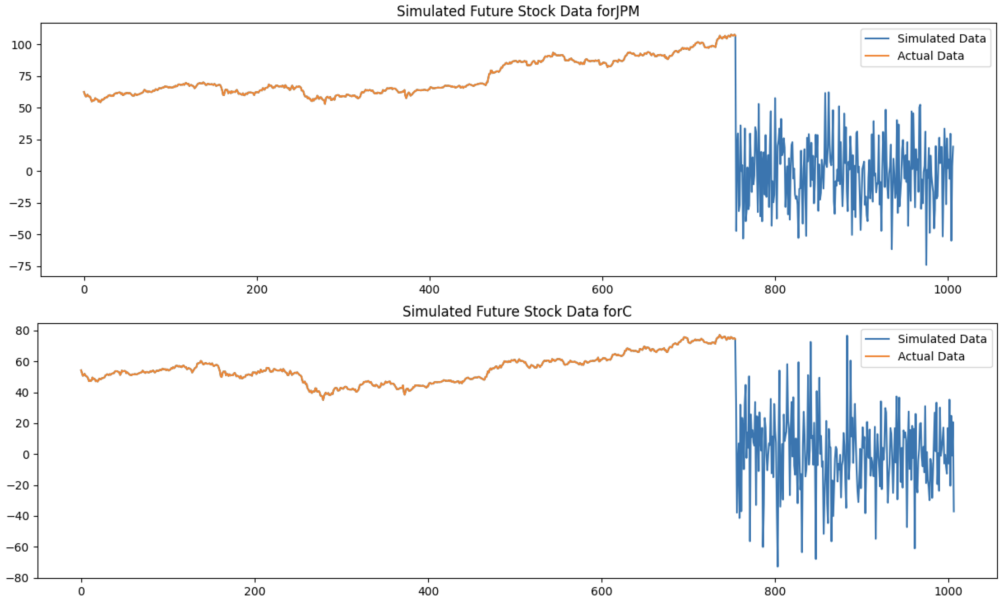


However, this method is also similar to the worst method (CVM). It is also particular difficult to select the proper distribution to offset some variance of output data. Although the log-normal distribution make sense to align the the distribution of stock price, however, it is not an ideal importance function, because it does not matter that stock price should follow the log-normal distribution. Accoring to the plots, all the simulates tend to arrive 0 at the end, suggesting that log-normal distribution is not a good importance function to offset the variance. The Importance Sampling Method (ISM) is a useful variance reduction technique for estimating rare events, but its effectiveness depends on the choice of importance function.

Stock Name	Baseline-MAE	ISM-MAE	ISM-MSE
JPM	32.21	110.78	12294.24
Citi	24.59	69.53	4865.42
BAC	10.38	29.72	886.91
WFC	14.21	55.31	3077.51

3.2 Markov-chain Monte Carlo

In the previous section, we have seen that the variance reduction method are constrained by the sampling method and extra variable we choose, which somehow hard to simulate the stock data. Markov-chain Monte Carlo (MCMC) simulation is particularly suitable for optimization problems where the objective function is difficult to directly evaluate or optimize, and where traditional optimization methods may not work well. MCMC simulations are particularly useful in Bayesian statistics, where the goal is to estimate the posterior distribution of model parameters given stock data. Here shows 4 plots that showing the simulation results of MCMC. Although MCMC allow us to indirectly sample from a posterior distribution by constructing a Markov chain that converging, the simulation are also extremely fluctuated. It is worth mentioning that although the simulation results are very volatile, they have shown a positive impact on Citi and WFC stocks, especially for WFC stock data, the MAE is decreased from 14.21 to 3.52, which show a huge optimization gap. However, for JPM and BAC, although the accuracy is reducing, but the effect is slight. Compared to the previous traditional model, MCMC simulation show a overall great performance on simulating the time-series data. Although the overall performance are great, the efficiency of the simulation has been decreased sharply. MCMC apparently take more time to determine the posterior distribution.



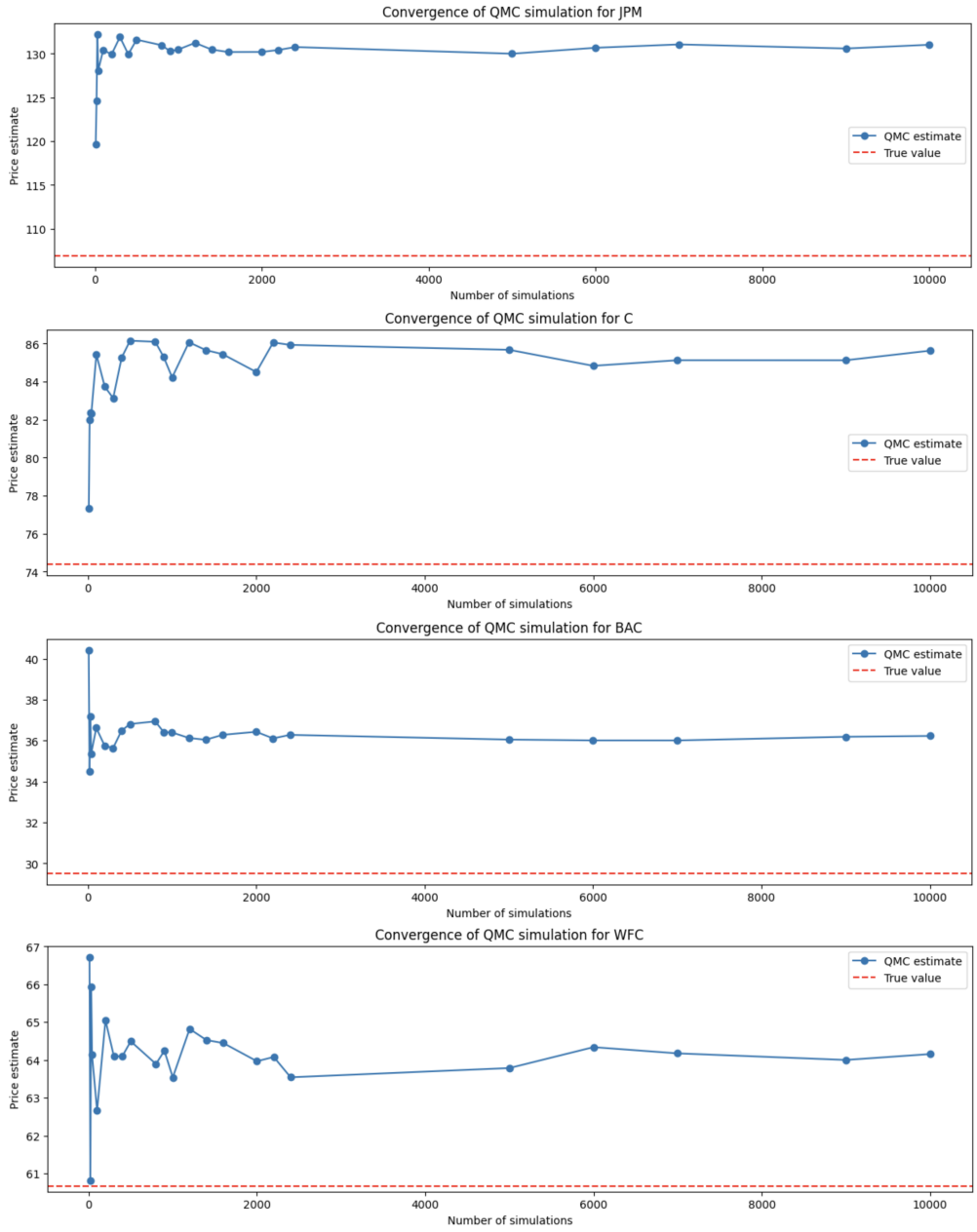
The following table show the detailed comparsion between the MCMC simulation and basedline simulation. Although the MCMC simulation allow us to retrieve effective information from complex time-series data, it is important to choose a suitable proposal distribution for moving Markov chains from one state to another. In our MCMC proposal, we are using default Gaussian Proposal Distribution directly from Python EMECC Module. Using Gaussian Proposal Distribution can be tuned to achieve an appropriate balance between exploration and exploitation of the target distribution. However, there are also some other proposal distribution such as Multivariate Proposal Distribution and Adaptive Proposal Distribution, although these two proposal distribution can be useful when the target distribution has correlations among the parameters, but it may cause the model difficult to determine a priori. Overall, MCMC simulation provide a best practice to simulate complex data, but there is a significant trade of between the accuracy and effectiveness.

Stock Name	Baseline-MAE	MCMC-MAE	MCMC-MSE
JPM	32.21	46.97	2253.45
Citi	24.59	16.31	310.95
BAC	10.38	13.14	178.51
WFC	14.21	3.52	22.08

3.3 Quasi-Monte Carlo method

Qausi-Monte Carlo (QMC) simulation provide an alternative way to standard Monte Carlo Simlation, it provide some optimization on low-discrepancy sequences, also known as quasi-random sequences, which using deterministic sequences that are designed to be evenly distributed throughout the sample space. Here provides 4 plots that show the simulation convergence result of QMC simulation. JPM simulation shows a great decrease in MAE from 32.21 to 18.56. From the convergence plot, we can see that at a certain range, increasing the number of simulation can converge the simulation result to the actual value. If the target distribution is irregular or highly nonlinear, QMC simulations may not achieve significant efficiency gains over standard Monte Carlo simula-

tions, this effect could be extremely obvious in the time-series data because these data cannot be predicted by normal linear models.



The following table shows the details comparison between the baseline simulation and the QMC simulation. Although the QMC give a great series of simulation of JPM stock data, however, the other three companies' stock are not simulated well. Despite its advantages, QMC simulation also has some limitations on the regularity of the target distribution.

Stock Name	Baseline-MAE	QMC-MAE	QMC-MSE
JPM	32.21	18.56	558.68
Citi	24.59	38.04	1773.74
BAC	10.38	76.66	5993.00
WFC	14.21	52.80	2949.66

3.4 Simulation Optimization Summary

In previous sections, we discussed different ways to optimize the simulation, and the table below lists the best ways to optimize the results for each company. In general, if we know the best optimization method for a certain company, we can simulate the stock data of different companies by choosing different optimization methods.

Stock Name	Baseline-MAE	Best MAE	Best MSE
JPM	32.21	18.56 (QMC)	558.68 (QMC)
Citi	24.59	16.31 (MCMC)	310.95 (MCMC)
BAC	10.38	9.92 (AMC)	169.54 (AMC)
WFC	14.21	3.52 (MCMC)	22.08 (MCMC)

However, in principle, due to the complexity of the simulation and the number of stocks, we may not be able to take different simulation methods for different companies. Of the approaches we've discussed, Markov-chain Monte Carlo (MCMC) is the one that shows the most potential. For the methods in Variance Reduction sessions, simulation result are heavily depends on the chosen variable, and in reality, these variable are hard to find and retrieve. MCMC simulation allows us to retrieve thess correlated and complex information through the Markov-Chain transformation. Although the selection of different proposal distribution may also affect the MCMC's result, it is easy to determine compared to choosing outside variables among millions of factors. Quasi-Monte Carlo method actually is not a suitable method to simulate the time-series data with significant non-linear feature. Consequently, MCMC is the overall accuracy and effective way to further optimize the MC simulation result.

4 Portfolio Optimization

Portfolio optimization is the process of creating a portfolio of assets, for which your investment has the maximum return and minimum risk. In this part, we design a system which could return optimal portfolio weights as long as users input their target stock and time. We consider a portfolio made up of 4 banking stocks, and optimize their weights to achieve maximum expected return for a given level of volatility.

4.1 Monte Carlo Simulation

As we mentioned before, Monte Carlo Simulations is a technique used to understand the impact of risk and uncertainty when making a decision. Simply put, a Monte Carlo simulation runs an enormous amount of trials with different random numbers generated from an underlying distribution for the uncertain variables.

In this section, we will dive into how to optimize portfolios using a Monte Carlo simulation. This section consists of several steps, including data collection, Allocation, Monte Carlo simulation and portfolio evaluation.

4.1.1 Data Collection

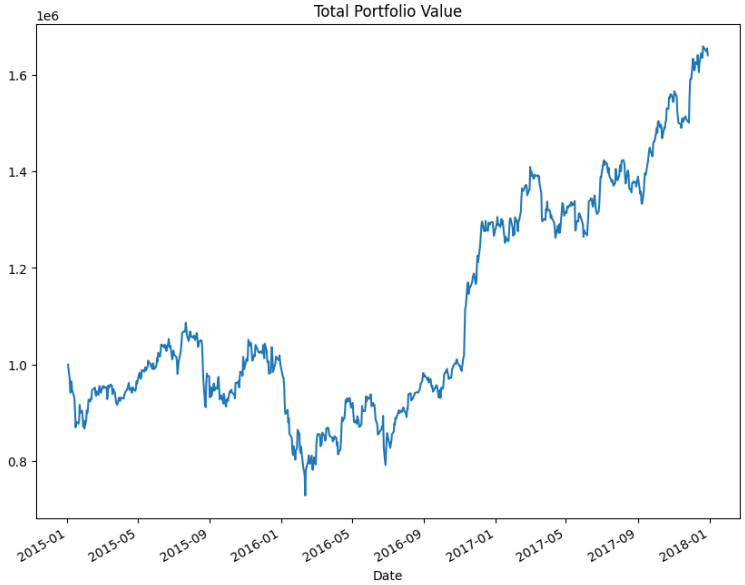
We used the Yahoo Finance API to collect historical data from Jan 1, 2015 to Dec 31, 2017 on the assets in the portfolio, such as their daily closing prices, over a specified time period. We collected data on four stocks: JP Morgan(JPM), Citi(C), Bank of America(BAC) and Wells Fargo(WFC). Next, we will normalize the prices by taking the adjusted close price at a particular date and dividing it by the very initial adj. close price.

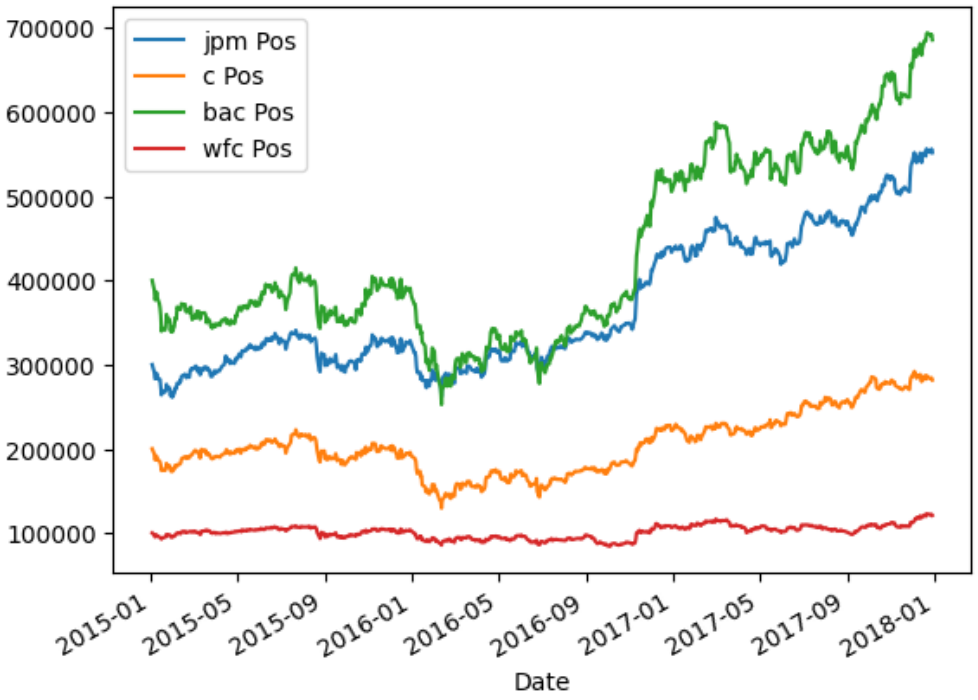
4.1.2 Allocation

Next, we assign an array of random weights for the purpose of calculation. These weights will represent the percentage allocation of investments between these four stocks. They must add up to 1. Let's assume we had the following allocations for our total portfolio:

- 30% in JP Morgan Chase
- 20% in Citi
- 40% in Bank of America
- 10% in Wells Fargo

Assume we invested a million dollars in this portfolio. Hence, (0.3 x 1 million dollars) is allocated in JP Morgan on day 1. So, this becomes our Position Value for JPM on day 1. At the very next day, since JPM went down the Position Value has now turned to a little less. We can expand this idea to other stocks and create a larger dataframe containing position values of each stock and also the total portfolio value at the end of each day.





4.1.3 Monte Carlo Simulation

Now we have to choose what is the optimal amount of money to put in each of these stocks. Portfolio optimization is finding the optimal values of weights that maximizes expected returns while minimizing the risk (standard deviation).
To decide the optimal portfolio, we use the Sharpe Ratio as our metric. The Sharpe ratio of a portfolio is the ratio of the expected excess return of the portfolio to the portfolio's volatility. It is a measure for calculating risk-adjusted return, and has become the industry standard for such calculations.

$$Sharpe\ Ratio = \frac{E(R_p) - E(R_f)}{\sigma_p},$$

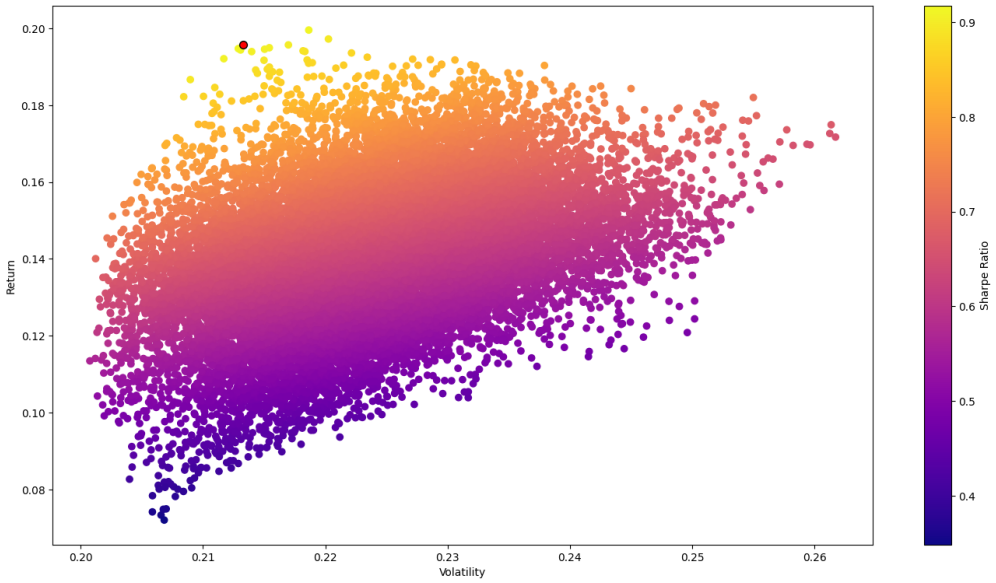
where R_p is return of portfolio, R_f is risk-free rate, and σ_p is standard deviation of the portfolio's excess return. Since the risk-free rate in the US is very low, we will assume a risk-free rate of 0.
The total expected return and variance for a portfolio are given by:

$$E(R_p) = \sum_i \omega_i E(R_i),$$

$$\sigma_p^2 = \sum_i w_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} w_i w_j \sigma_i \sigma_j \rho_{ij},$$

where ω_i is the weight of each stock in the portfolio, σ_i is the standard deviation of each stock price. We used Monte Carlo simulation to generate future returns for each asset in the portfolio based on their historical data. We generated 20,000 random samples of future returns for each asset and calculated the expected returns and standard deviation of the portfolio for each sample. In the

figure below, we plot all the portfolio combination runs on a graph and point out the maximum sharpe ratio. The maximum sharpe ratio (0.918) is represented by the red dot with a return of around 0.195 and volatility around 0.213, which means the allocation of best portfolios.



4.1.4 Portfolio Evaluation

We tested the optimized portfolio against a baseline portfolio to determine if the model is effective in optimizing the portfolio. The baseline portfolio was constructed by equally weighting the assets in the portfolio. Having the allocation of the best portfolios, we calculated the performance metrics of the whole year of 2018 and evaluated the optimized portfolio by comparing the expected rate of return to that of the baseline portfolio. By comparing the results of baseline and optimal portfolio, we could see that although both have a loss since all the 4 stock prices had dropped during 2018, the loss of optimal portfolio (-0.089) is still much lower than that of the baseline (-0.186). This indicates the Monte-Carlo method does help us to choose the better portfolio weights.

Baseline	MC Simulation
-0.186	**-0.089**

4.2 Reinforcement Learning

4.2.1 An Overview of Reinforcement Learning

Reinforcement Learning (RL) [6] is one of the basic machine learning paradigms in which an agent learns from the interactions with the environment through trial-and-error. Specifically, any RL problem can be formalized as a 5-tuple Markov Decision Process (S, A, P, R, γ), Where S is a set of states of the environment; A is a sequence of potential actions an agent can select; P is the transition function which denotes the probability of transitioning to state s' and receiving reward r at time t, based on the current state s and action a. It is formulated as followed:

$$P(s_{t+1} = s', r|s, a) = Pr(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a), \tag{6}$$

where $\forall s, s' \in S, \forall a \in A, \forall r \in R$; R is the reward function: $S \times A \to \mathbb{R}$. The expected reward for each pair of state and action is defined below:

$$r(s,a) = \mathbb{E}[r_t | S_{t-1} = s, A_{t-1} = a] = \sum_{r \in R} r \sum_{s' \in S} P(S', r | s, a), \tag{7}$$

where $\forall s s' \in S, \forall a \in A, \forall r \in R; \gamma \in [0,1)$ is the discount rate that determines how much immediate rewards are considered to estimate the future rewards

The objective of RL is to maximize the cumulative future expected reward, which is denoted as return G_t. The discounted return is formalized as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma_k R_{t+k+1}, \tag{8}$$

where $\gamma \in [0,1)$. If $\gamma = 0$, the agent is myopic because it only concerns the immediate reward at the next step. A policy $\pi(a|s)$ maps from states to the probabilities of possible actions to be selected. It defines the distribution of probability that $A_t = a$ if $S_t = s$. The agent's policy will change during its interaction with the environment.

The state-value functionv_π is the expected return starting from a state s and following a policy π afterwards, which is defined by:

$$v_\pi(s) = \underset{\pi}{\mathbb{E}}[G_t | S_t = s] = \underset{\pi}{\mathbb{E}}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s], \forall s \in S \tag{9}$$

The action-value function $q_\pi(s,a)$ is similarly defined as the expected return that begins from states and takes action a under the policy π:

$$q_\pi(s,a) = \underset{\pi}{\mathbb{E}}[G_t | S_t = s, A_t = a] = \underset{\pi}{\mathbb{E}}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a], \forall s \in S, \forall a \in A. \tag{10}$$

4.2.2 Deep Q-Networks

The Deep Q-Networks (DQN) was first proposed in [7], which combines RL with deep convolutional neural networks to estimate optimal action-value function,

$$q^*(s,a) = max_\pi E[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a, \pi], \tag{11}$$

where the maximum target Q value is achieved at policy $\pi = P(a|s)$, after performing the action (a) and making an observation (s). Formally, the value function is redefined as $Q(s,a;\theta_i)$ where $_i$ is a parameter that represents the weights of DQN at iteration i. To improve the stability of the Q-network, they used a mechanism inspired by biology, namely the experience reply, to randomize the data. It helps to smooth the unstable changes in data distribution and removes correlations in the observation. Besides, to reduce the target correlations, the adjusted Q values have been updated periodically in each iteration. Specifically, the experiences of an agent have been denoted

as $e_t = (s_t, a_t, r_t, s_{t+1})$ at timestep t and stored in a dataset $D_t = e_1, ..., e_t$. During the learning process, the updates of experience follows a uniform distribution $(s, a, r, s')U(D)$. At the iteration i, the update follows the loss function as:

$$L_i(\theta_i) = \mathop{\mathbb{E}}_{(s,a,r,s')\ U(D)}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-))^2], \tag{12}$$

where θ_i^- is the parameter to calculate target for iteration i and is only updated with the parameter θ_i in a fixed time steps.

4.2.3 Portfolio Optimization under RL Framework

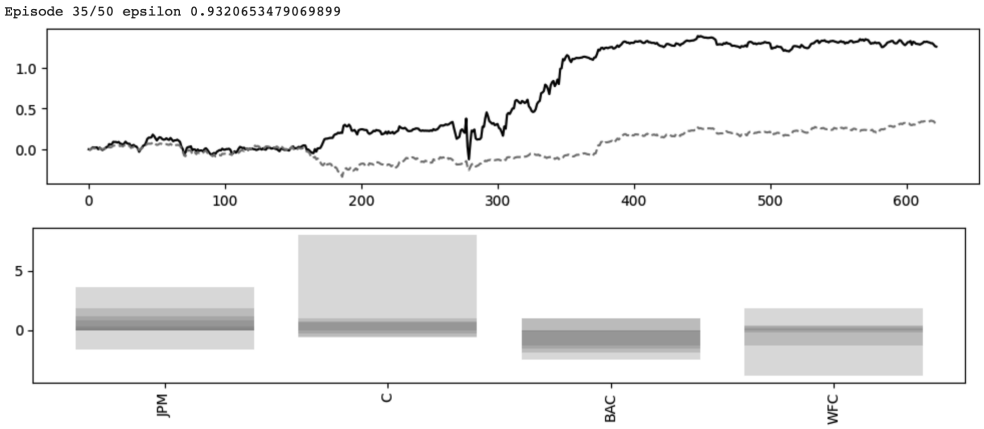
We analyzed the problem of Portfolio Optimization based on Reinforcement Learning (RL) framework. We defined our problem as: Given the histories a batch of assets, how could we train a policy to build a portfolio to maximize the likelihood of the return. Specifically, we define a RL framework to the portfolio optimization problem as follows. Firstly, the environment is the historical price data of the batch of asses. Secondly, the state refers to as the correlation matrix of the instruments in a specific time window, which perfectly reflects the relationships between each asset. Thirdly, the action is to select or rebalance the weights of each asset in a portfolio. The state-action value, Q generated by DQN Model will be converted into weights. Finally, we calculate reward following the equation of sharpe ratio, a measure of the excess return per unit of risk. Also, the agent is simply a portfolio manager who assigns a weight to each stock at each timestep based on its current observation.

4.2.4 Experiments

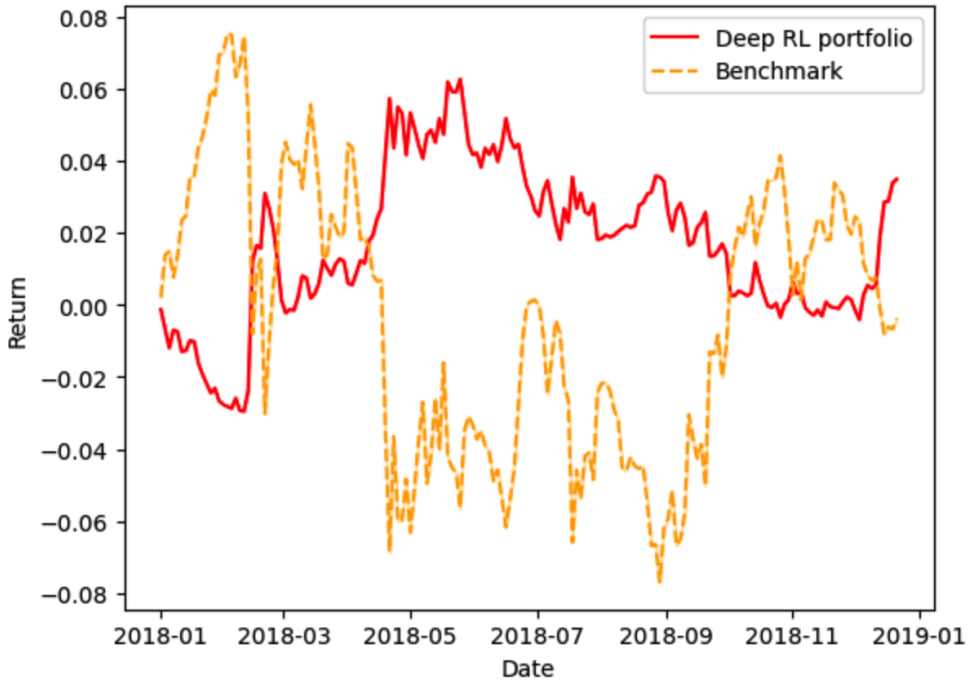
We selected 4 bank stocks from yahoo finance module, namely JPMorgan, Citi Bank, Bank of American and Wells Forgo and used their historical data as our environment. Specifically we used the data from '2015-01-01' to '2017-12-31' as the train dataset and the data from '2018-01-01' to '2019-01-01' as the test dataset. We train the DQN model to select an optimal policy and apply it to test data to predict the final cumulative profit following a sequence actions of investment. The training hyper parameters are listed as following:

Episode count	50	Window size	180
Batch size	32	Rebalance period	90
Alpha	0.8	Gamma	0.95
Epsilon min	0.01	Epsilon	1
Dropout rate	0.65	Optimizer	Adam

To ensure statistical significance, we calculated 5 seeds to estimate our model's performance and calculate the average return at each time step. Also, we visualize the variance by the shading area in the graph below, which gives a rough idea of its general performance and stability. To compare with the Monte Carlo simulation, we restrict the actions to only buying and sitting, without selling. The baseline algorithm is to assign equal weight to each asset. Please refer to the table below for the final results. In general, our deep RL algorithm outperforms the baseline and is also superior to Monte Carlo Simulation.



The training result in 35th episode



Testing Performance

As we can see in the table below, in the setting with three actions, namely buy, sell and sit, the final cumulative reward for Deep RL is positive and larger than any other algorithms including baseline. Also, our Deep RL model under the setting without short outperform the Monte Carlo simulation and the baseline algorithm.

Baseline	Baseline(shorts)	MC Simulation	Deep RL	Deep RL (shorts)
-0.186	-0.004	-0.089	-0.002	**0.035**

5 Conclusion

In this project, our group wishes to predict the stock price in 2018 and determine the optimal portfolio. We first select JPMorgan Chase, Citi, Bank of America, and Wells Fargo as our prediction targets and downloaded their historical data from Yahoo Finance from the time range '2015-01-01' to '2017-12-31'. By applying Monte Carlo simulations, we calculated the predicted price using Brownian Motion. Then, we use several techniques such as variance reduction, Markov-Chain Monte Carlo Simulation, and Quasi-Monte Carlo simulation to optimize stock simulations. We find that different techniques may increase the result's accuracy to a certain level, and none of these techniques are perfect. MCMC is the overall accurate and effective way to further optimize the MC simulation result. For portfolio optimization, we use Monte Carlo simulation and the reinforcement learning algorithm to optimize the weights of each stock in the portfolio and compare their results. Both two methods give better results than the baseline, and the return rate of the reinforcement learning algorithm is higher. Thus, we conclude that both our prediction and optimization models contribute to competitive performance, and have the potential to be used in the stock prediction and investment market.

References

[1] Magnusson, A., Punt, A. E., Hilborn, R. (2013). Measuring uncertainty in fisheries stock assessment: the delta method, bootstrap, and MCMC. Fish and Fisheries, 14(3), 325-342.

[2] Ramström, Alexander. "Pricing of European and Asian options with Monte Carlo simulations: Variance reduction and low-discrepancy techniques." (2017).

[3] Qiang, Z., Guo, L., Guiding, G. (2013). Variance reduction techniques of importance sampling Monte Carlo methods for pricing options. Journal of Mathematical Finance, 2013.

[4] Dang, D. M., Jackson, K. R., Mohammadi, M. (2015). Dimension and variance reduction for Monte Carlo methods for high-dimensional models in finance. Applied Mathematical Finance, 22(6), 522-552.

[5] Jourdain, B. (2009). Adaptive variance reduction techniques in finance. Advanced Financial Modelling, 8, 205.

[6] R. S. Sutton and A. G. Barto (2020). Reinforcement Learning: An Introduction, Cambridge: The MIT Press.

[7] Mnih et al. (2015). Human-level control through deep reinforcement learning, Nature, vol. 518, 2015.

[8] Elias, M. (2020). Monte Carlo Simulations for Stock Price Predictions [Python], Analytics Vidhya.