Monte Carlo Simulation

Rupesh Poudel

Seminar Paper:
Numerical Introductory Course

Lecturer: Prof. Dr. Brenda López Cabrera

March 7, 2022

Abstract

Monte Carlo Simulation uses random numbers to perform repeated sampling of a complex process. This paper generates uniformly distributed sequence of random numbers using Linear Congruential Generator and Lagged Fibonacci Generator, and converts the uniformly distributed random sequence of numbers into the standard normally distributed random sequence of numbers by Box-Müller Method, Inversion method and the Marsaglia method. These random sequence of numbers, along with the default random numbers generated from Python's Numpy library will be used to carry out the Monte Carlo Simulation to predict the price of the stock of the Apple Inc. 6 months into the future. Sensitivity of the simulation with respect to the change in random number generator and the number of trials are reported along with the computational time of the simulation. Mean Squared Error and Confidence Intervals are also reported. The Kernel Density Estimators is drawn for each variation.

Contents

1 Motivation

In a financial market, any information about the value a stock price of a particular company prevailing in the future is highly sought after. Investors can buy shares, even with high leverage, if the shares are expected to rise in the future, and offload their position in addition with short selling if the shares are expected to fall. Complex financial derivatives like futures and options, whose payoffs are based on the future price of the underlying, can allow investors to gain the profits with relatively less initial investment. Different private and institutional investors have their own future price prediction model, which they use to pick stocks to buy. The prediction power of the model determines the success of an investor. It is therefore important to find a suitable model that outperforms the market and hopefully the competitors. This paper will discuss about Monte Carlo Simulation, a statistical method of repeated sampling and aggregation, as an initial estimate for such prediction.

Monte Carlo Simulation is a tool that can be used to solve problems numerically when it is too complex to be solved analytically. It can solve higher dimensional problems whose numerical evaluation is computationally engaging and tedious. Furthermore, when the underlying input for the estimation, forecasting, or some decision process is stochastic in nature, Monte Carlo Simulation can provide with a range of choice for that input.

Scientists working for the "Manhattan Project", namely Stanislaw Ulam, John von Neumann, and Nicholas Metropolis, are credited for the invention of Monte Carlo Simulation. It all started when Ulam wished to calculate the probability of him winning any hand of Solitaire, a popular card game, analytically. Unable to find a solution, Ulam hypothesised whether playing, or equivalently simulating the game for sufficiently large number of trials and counting the number of wins would be a reasonable alternative, when combinatorics involved in the game is too complex. The same approach could be replicated in numerous other sectors, including nuclear physics, where they had to "solve the problem of neutron diffusion in fissionable material." Neumann, after being suggested the approach, wrote a detailed, computer execution-able set of instructions for neutron diffusion be executed in ENIAC, a first generation computer. Metropolis suggested the name "Monte Carlo", referencing to Ulam's uncle and his gambling tendencies and frequent visits to the casinos of Monte Carlo, Monaco. Metropolis (1987) outlines the aforementioned origin story and formalises the method of Monte Carlo Simulation.

Uniformly distributed (pseudo) random numbers are generated using an algorithm, in our previous story, the middle square digits. These random numbers are then converted to a suitable distribution that best matches the problem under consideration. The inverse of the cumulative distribution function of our desired distribution is used for the conversion. These transformed random numbers are used to carry out instances of deterministic computation. Appendix 1 shows an example of a coin toss experiment which can serve as a primer for the way in which Monte Carlo Simulation works, and Appendix 2 outlines the middle square digits method of generation of random sequence of numbers.

2 Relation to existing literature or methods

Monte Carlo Simulation draws upon the Bernouli's law of large numbers. When an experiment is conducted multiple times, as the number of times the experiment is performed increases, the mean of the results from those experiments will converge to the true (theoretical) mean. A fair coin tossed countably infinite times will see the probability of each of its outcome {Heads or Tails} approach 0.5. A fair six-sided dice will witness the probability of each of its side to face upwards after a spin converge to 1/6.
Estember and Maraña (2016) use Monte Carlo Simulation to model future stock prices and compare the Geometric Brownian Motion (GBM) vs the Artificial Neural Network (ANN) method. They show GBM method to be more effective in predicting the future stock price movement. This paper will also follow suit, and simulate price paths of trials using a given mean and volatility of returns.

3 Application for Finance and Statistics

3.1 Application for Statistics

Statistics is among the many fields where the use cases of the Monte Carlo Simulation is plentiful. Monte Carlo Simulation can be used to provide an efficient random estimate for the Hessian matrix. Yu (2016) uses automatic differentiation under Monte Carlo setting to compute estimate of the Hessian Matrix. The Hessian matrix can be used in Fisher Scoring, and subsequently the Fisher Information matrix. Das, Spall, & Ghanem (2010) show the use of resampling algorithm, a Monte Carlo technique, for computing Fisher Information Matrix. The analytical determination of the Fisher Information Matrix in non linear models is difficult because of the intractable higher dimension integration and modelling requirement.
Similarly, Bayesian Statistics is making use of Monte Carlo Simulation. Chen, Shao, Ibrahim (2012) develop advanced Monte Carlo methods by using posterior distribution's sample to compute posterior qualities, namely: marginal posterior densities, marginal likelihood, Bayesian Credible Intervals and so on. Monte Carlo Simulation is used in statistical interference as well. Hypothesis testing, comparison of variance of two estimators, evaluating higher dimension integrals and parametric bootstraps are some uses identified in Gentle (2009, Chapter 11).

3.2 Application for Finance

Hull (2018) suggests Monte Carlo Simulation for option pricing. Path dependent options such as Asian options , and options with many stochastic variables can use Monte Carlo Simulation. Hull(2018, pg 511) also suggests Monte Carlo Simulation to calculate the dollar change in portfolio in a day, or VaR. The value of derivative such as options can be calculated by sampling the random path of an underlying in a risk neutral world, calculating the payoff go the option, and run sufficiently high number of trials so that average of the sample payoff can be calculated to get the expected payoff in the risk neutral world. The option is now discounted expected payoff at the risk free rate. (Pg. 469). The performance of stop loss hedging can be seen by using Monte Carlo Simulation.

In addition, an investor can analyse the default risk of a company or credit. The cash flow analysis of an uncertain project can be carried out. Instead of a point estimate, a confidence interval can be computed which guides the investors to make informed decisions. Furthermore, the evolution of a stock price, and consequently the investment portfolio of an investor can be modelled. This paper will carry out an analysis on the evaluation of the Apple share price in the coming sections.

4 Formalization of the method

The paper starts with a very generic description of a stock price process as listed below:

$$dS_t = \mu S_t dt + \sigma S_t dZ_t$$

To make sure that the stock process is a martingale, and that the next value in the sequence is equal to present value in terms of conditional expectation. The price process, without taking into account the cost of carry, changes into the following.

$$dS_t = rS_t dt + \sigma S_t dZ_t$$

This was the same process represented in Black, Scholes, Merton (1973). From here on, I will include the cost of carry. Apple paid 88 cents of dividend in 2021 where its ending price was $ 158.58. This is a rate of 0.55 percent. Treasury bill of 6 month maturity was trading for 0.6 percent in March 1, 2022. So, to assume the cost of carry similar to a risk free rate is not a far fetched assumption. The Geometric Brownian Motion now becomes:

$$dS_t = (r - \delta)S_t dt + \sigma S_t dZ_t = \sigma S_t dZ_t. \;\; for \;\; r = \delta$$

$$dS_t / S_t = \sigma dZ_t$$

Assuming that, for the purpose of valuation of the stock, the only change in the future are driven by the innovation terms σ, we can simplify the prediction. Also, the diffusion term, or the volatility of the process is assigned the value of the historical volatility of log returns. Then random numbers which follow standard normal distribution are generated and then transformed into random numbers with the aforementioned historical volatility and 0 mean. To predict N days into the future, N random numbers that adhere to zero mean and the historical volatility are drawn. The result of which will guide the daily movement of the stock each day for N days. Plotting the value of the stock from initial price to the price prevailing in the day N into the future will plot one price path for 1 trial of the simulation.
Performing the simulation for M times will lead to M prices for day N. As random sequences of numbers were generated for each price path, they will lead to M different prices for day N value of the Apple share. These M different prices for day N value are stored in a different list, or a price series, to observe its behavior. One can compute the mean of the price series, the minimum and the maximum value of the price series. A Kernel Density Estimator, similar to a histogram, can be plotted to see how the final day price series are lined up in a plot. From there, one can not only see the point estimate for the future value of the share, but also the range of the values.

In the section below, I will discuss about random numbers. I explained where and how the random number was used. The following section explains why random numbers are needed in Monte Carlo Simulation.

4.1 Random Numbers

Random numbers, or more accurately, a random sequence of numbers, are essential for Monte Carlo Simulation. In this paper, random numbers are required to dictate where the stocks are supposed to move on any given day for any given trial. Without sufficiently randomized sequence of numbers, the estimates of the inputs will be biased. Biased inputs will lead to less trustworthy outputs. To get unbiased estimates, different attempts have been made to construct algorithms that produce uniformly distributed and independent random numbers. Random numbers must adhere to some conditions for it to be usable. Each numbers in the range must be equally likely, the random numbers must be continuous and not discrete. The mean and variance must neither be too high or too low. Similarly, there should not be significant auto-correlation between numbers.There must be no seeming pattern between numbers. A period of a sequence of random number reflects the number of terms in the sequence before the sequence repeats. An appropriate PRNG must consist of very high period.
For the purpose of this paper, once I generate the uniformly distributed random sequence of numbers, I test for their randomness. I plot the scatterplot of two consecutive random numbers to check whether there is a pattern. The points in scatterplot which is all over the graph shows that there is no apparent pattern. I also compute the variance covariance matrix of the random numbers to check whether there is an association in the random sequence of numbers generated from different algorithm/transformation. A kernel density estimator is also plotted to show the pdf of the estimated random sequence of numbers, be it uniform or standard normal.

4.2 Pros and Cons of the Method

Pros: Deterministically complex problems can be solved numerically. As was with the coin toss, nuclear fission, solitaire, and stock price estimation, Monte Carlo method uses random number based numerical approach to provide estimate for an answer. For non- invertible functions, Monte Carlo can use reverse search to plot the graph of the inverse when possible. One can easily visualise at which domain of the inputs the function becomes unbounded, and where the function can be tamed. Monte Carlo method is a powerful tool for solving any problem with uncertainty involved. When random numbers with an appropriate distribution is used as an input to replace the uncertain factors, the solution has reasonable accuracy with surprisingly low variance.
Cons: Considerable computational time and resources are consumed by Monte Carlo Simulation. Reverse search to invert a non-invertible function will make the computer evaluate the function under consideration for a given range of inputs, The evaluation will most likely be discrete with spacing of the input predetermined by the user. As it is normal for Monte Carlo simulation to involve more than a million simulation. This leads to a same set of computation being performed N (number of simulation) times. Each simulation replicates a process. This process itself will include some computation M. So, carrying M

computations N times means MN computation, which takes processing speed away from the computer. Sometimes, a computer cannot sustain request to perform simulation citing low available resources and the device crashes. The choice of programming language, the efficiency of algorithm, the choice of storage of data, the allocation of bits for storage of calculations all play a part in determining the computational time. Moreover, If our input or parameters are garbage, our output will be garbage too. This includes the domain of input and the underlying probability distribution.
Monte Carlo method does not factor in irrationality and behaviour aspects of say, finance and economics very well. Monte Carlo methods do not accurately predict tail events. I did state that the evolution of the stock price is determined solely by innovation terms. But, major shocks to the industry, such as a possible war breakout, and supply chain issues like global chip shortage, has to be specifically modelled and Monte Carlo Method might need to incorporate jumps. The model I will use is a basic model which has major drawback, and is far from reality, specially when reality itself is deviating from the norm on a daily basis. Use of random numbers and Monte Carlo simulation itself, that can have extreme maximum and extreme minimum values, does replicate tail event in some of its simulation. However, this extreme is a result of experimental design of Monte Carlo Simulation as a statistical tool and is found even when no tail event occurs in the real life.

5 Data Description, Simulation Settings, Testing Results

5.1 Data Description

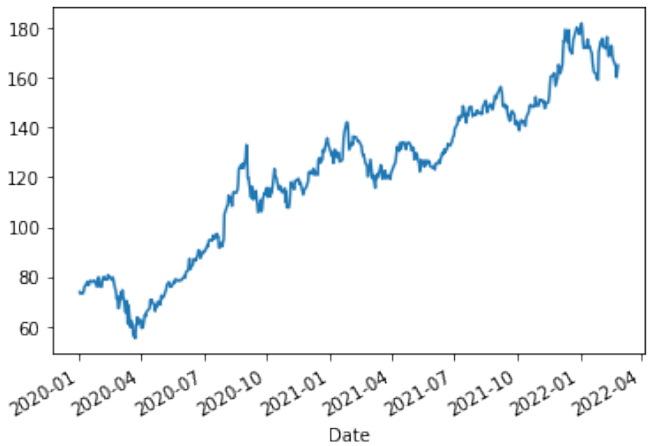

Figure 1: Share Price History of Apple Inc.

I am using Monte Carlo Simulation to predict the future share price of Apple Inc. I wish to concentrate the analysis on the COVID-era. I set the timeframe for the data to "2020-1-1" to "2022-02-28". I chose the python library "yfinance" to download the data. The library "yfinance" allows the user to extract the

share price history of Apple Inc. from the Yahoo Finance database. Apple trades on NASDAQ under the ticker AAPL. I specified the ticker symbol and the timeframe of the price series to download the data of Apple Share under 6 headings: "Open", "High", "Low", "Close", "Adj Close", "Volume". I took "Adj Close", short for "Adjusted Close" and plotted the progression of the adjusted closing price with respect to time. Figure 1 is that plot. Looking at the figure, Apple has seen a rapid post-COVID boom. The market crash brought about by COVID appears to be a very small price fall in comparison to the rise thereafter.
S_t/S_{t-1} gives the return of the price series. Here S_t is the price of the stock at time t, and S_{t-1} is the price of the stock a day before. I will call this a return series. From this returns series, the natural logarithm of stock returns from day $t-1$ to t is calculated. The paper assumes that the natural logarithm of the stock follows normal distribution with given mean and standard deviation. Figure 2 below gave a mean and standard deviation of 0.0014804427509803103 and 0.023306304733618895 respectively. To assume a mean of 0 is then reasonable. This standard deviation of "log returns", the natural logarithm of daily stock returns, is around 2.33 percent. From the visual inspection of Figure 2, the log returns seem to have heteroskedastic error terms. They are not stationary. Non-parametric models or variants of GARCH are appropriate to model this type of series. I choose to assume the volatility of 2.33 percent to persist in the next 6 months as well. Six months equals 0.5 * 252 = 126 trading days.

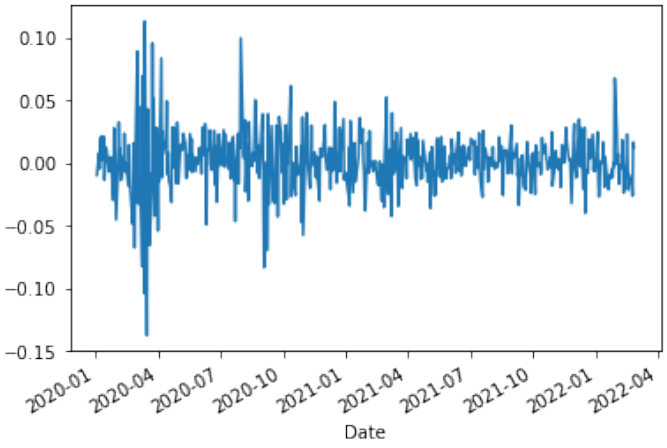

Figure 2: Returns History of Apple Inc.

Linear Congruential Generator, Lagged Fibonacci Generator, and Numpy's default pseudo uniform random number generator generates uniformly distributed random sequence of numbers. These sequence is thereafter transformed into standard normal distribution using the Box-Müller Method, Inversion method, and Marsaglia method, details of which is found in Franke, Härdle, Hafner (2019, Chapter 6). These standard normal random numbers are converted into normally distributed random numbers with the mean of 0 and a standard deviation of 2.33 percent. This way, I fulfilled one criteria of Monte Carlo simulation outlined by Metropolis (1987), i.e. convert uniformly distributed random

number into a suitable distribution that best matches the problem under consideration. The suitable distribution for log returns is normal distribution with mean around zero and standard deviation of around 2.33 percent.

5.2 Simulation Setting

For the initial run, a matrix of size **[126,100000]** consisting of normally distributed random numbers with the desired mean and dispersion is generated using Numpy's default random number generator. As I wish to predict the price path for 126 trading days ahead and run the Monte Carlo Simulation 100,000 times, the matrix has that specific dimension. For comparability and results reproduction, the matrix is stored under a separate variable so that the same random sequence of numbers can be used to check the sensibility of the final results on a reduced number of simulations. In case of other random numbers, an array of 12.6 million normally distributed random sequence of numbers with the same mean and standard deviation are generated. The array is then reshaped into a matrix of size **[126,100000]**. Different set of matrices representing different pseudo random number generators and consequently different conversion methods will follow the same procedure.

5.3 Testing Results

As stated above, 100 thousand simulation for 126 trading days was carried out. The following figure

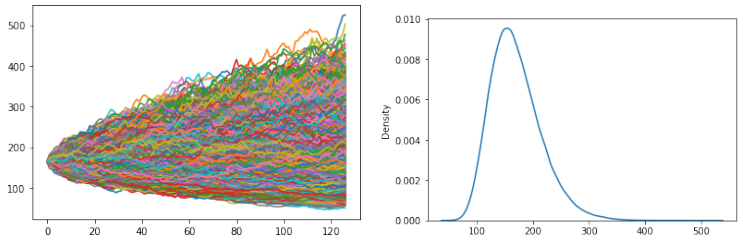

Figure 3: 100,000 simulation for 126 days and its KDE plot

164.85

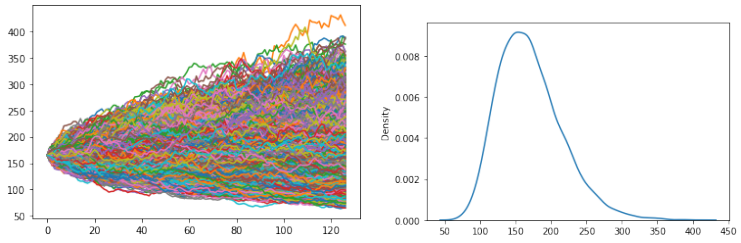

Figure 4: 10,000 simulation for 126 days and its KDE plot

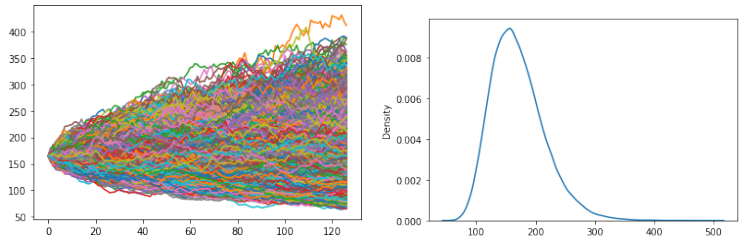

Figure 5: Different PRNG, 100,000 simulations and its KDE plot

6 Results of Empirical Analysis

(Baddeley & Hitch 1974, p. 48).

7 Conclusion

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

8 Appendix

8.1 Appendix 1: The Coin Toss Experiment

Let there be two players: A B. A and B will participate in a coin toss game where a fair coin will be tossed 100 times. The outcome of each tosses will be recorded. Any "Heads" will entitle A a balance transfer of 1 from B and "Tails" will endow B with 1 from the bank balance of A. Alternatively, the winnings can be recorded and a single transfer to offset the payables can be carried out after 100 tosses. This is a setup of an ordinary random walk with two nodes. Franke, Härdle, & Hafner (2019, Chapter 4) outlines the ordinary random walk in the following way:

$$X_t = X_0 + \sum_{k=1}^{t} Z_t, \quad t = 1, 2, 3...$$

where $X_0, Z_1, Z_2....$ are independent. $P(Z_k = 1) = p$, and $P(Z_k = -1) = 1 - p$ for all k . There is a source of uncertainty in this problem. The value of at the end of each coin toss will not be known beforehand. Monte Carlo will make use of random number it generates for each trial to suggest the value for Z_t. As the heads and tails are equally likely, any symmetric distribution is a suitable distribution, provided that the experimental design is unbiased towards either outcome. Using the uniformly generated random numbers for example, one can

fairly allocate the random numbers if one allocates even numbers for head and odd numbers for tail when choosing for integers between 0 and 100. One can achieve the same results by allocating the numbers greater or equal to 0.50 as heads and the remaining half as tails when choosing the numbers in range [0,1). Once the heads and tails are determined by the use of uniformly distributed random numbers, can be computed. The coin toss experiment was settled without tossing a single coin. Such is the use of Monte Carlo Simulation.

8.2 Appendix 2: Middle Square Digits Algorithm

John von Neumann, in his first Monte Carlo code ever written, proposed the middle square digits algorithm for random number generation. In this algorithm, an arbitrary number of N digits is chosen. A 4 digit number 5555 for illustration purposes will show the workings. The initial number from which a random number generator starts to generate subsequent random numbers is called the seed of the algorithm. Starting with the same seed will help replicate the sequence of pseudo random numbers for future use. The square of 5555 is 30858025. This is an eight digit number. The middle 4 digits, 8580 is the new random number and the process is repeated for the generation of subsequent random numbers.

8.3 Appendix 3: Some external links

US Interest rates https://www.federalreserve.gov/releases/h15/
edftests
anderson darling kolmogotov smirnov

References

Baddeley & Hitch 1974
BADDELEY, Alan D.; HITCH, Graham: Working memory. In: *Psychology of learning and motivation* (1974), pages 47–89

References

Baumgärtner, A., Binder, K., Hansen, J. P., Kalos, M. H., Kehr, K., Landau, D. P., & Weis, J. J. (2013). Applications of the Monte Carlo method in statistical physics (Vol.

Chen, M. H., Shao, Q. M., & Ibrahim, J. G. (2012). Monte Carlo methods in Bayesian computa

Das, S., Spall, J. C., & Ghanem, R. (2010). Efficient Monte Carlo computation of Fisher in

Eckhardt, R., Ulam, S., & Von Neumann, J. (1987). the Monte Carlo method. Los Alamos Scien

Estember, R. D., & Maraña, M. J. R. (2016, March). Forecasting of stock prices using Brown

Franke, J., Härdle, W.K., & Hafner, C.M. (2019). Statistics of Financial Markets: An Intro

Gentle, J. E. (2009). Monte carlo methods for statistical inference. In Computational Stat

Harrison, R. L. (2010, January). Introduction to monte carlo simulation. In AIP conference

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistribut

Metropolis, N. (1987). The beginning of the Monte Carlo method. Los Alamos Science, 15(584

Riffenburgh, R. H. (2006). Methods You Might Meet, But Not Every Day. Statistics in Medici

Van Groenendaal, W. J., & Kleijnen, J. P. (1997). On the assessment of economic risk: fact

Yu, X. (2016). Efficient Computation of Hessian Matrices in a Monte Carlo Setting using Au