# Shiny App Challenge

Confidence Interval for Sharpe Ratio using

Monte Carlo Simulations on Stock returns

Jonathan Mac
Cornell University'20
Applied Statistics, Finance
2/4/2020

**Assumptions**

1) Monte Carlo simulation follows a Brownian Motion which models random movement of stock prices:

- (today's price) = (yesterday's price) * e^(r)
- where r is the periodic daily return

2) Due to the Central Limit Theorem, Brownian Motion models the random movement of future periodic daily returns (ln(today/yest)) as normally distributed with mean (as drift), and standard deviation (as historical standard deviation of periodic daily returns)

3) r has two components: drift (overall constant driving force; Expected daily rate of return with greatest odds of occurring)and shock(a random component)

- r = drift + shock
- drift: u - var/2
    - Where u = ln(daily returns) and var = var(u).
    - For standard Monte Carlo, I use a volatility eroded historical mean.
    - In words, u is natural log of daily returns, and var is the variance of those natural log returns
- shock=: sd*qnorm(runif(1))
    - In words, shock is a random component(returns # of sds) that makes simulated returns normally distributed.

**Logic of project**

1) First, generate theoretical stock prices:
    a. I generated random stock prices for random entity called Company A over 3650 days (10 years) by assuming its IPO stock price = $100, and future daily return had mean = .0002 and standard deviation = .002.

2) Secondly, run Monte Carlo Simulation to create n sets (simulations) of future data:
    a. Solve for drift and shock value for periodic daily return
    b. For each simulation:
        i. Using periodic daily return and two user inputs (number of returns needed + and frequency), simulate future daily values
        ii. Based on user input frequency, calculate returns (daily, monthly, or yearly) using geometric means
        iii. Calculate annualized return and annualized volatility
    c. Repeat steps 1-4 for n = 100 times, and record results for each iteration
    d. Obtain a Confidence Interval with user input confidence level

i. Adjust critical values for a two-tailed test because we are interested in finding a spread from a mean here

e. Graph the future stock prices with varying colors for each simulation

**Analysis of Shiny App Results**

From figure 1 (a screenshot below of my Shiny App), users can adjust the number of returns they desire to see, frequency, and confidence level. This simulation was possible by and relied heavily on the assumption that Brownian Motion modeled future periodic returns as normally distributed. In a realistic finance setting, I would have made assumptions looser and enforced stricter criteria due to kurtosis and skewness of real stock data.

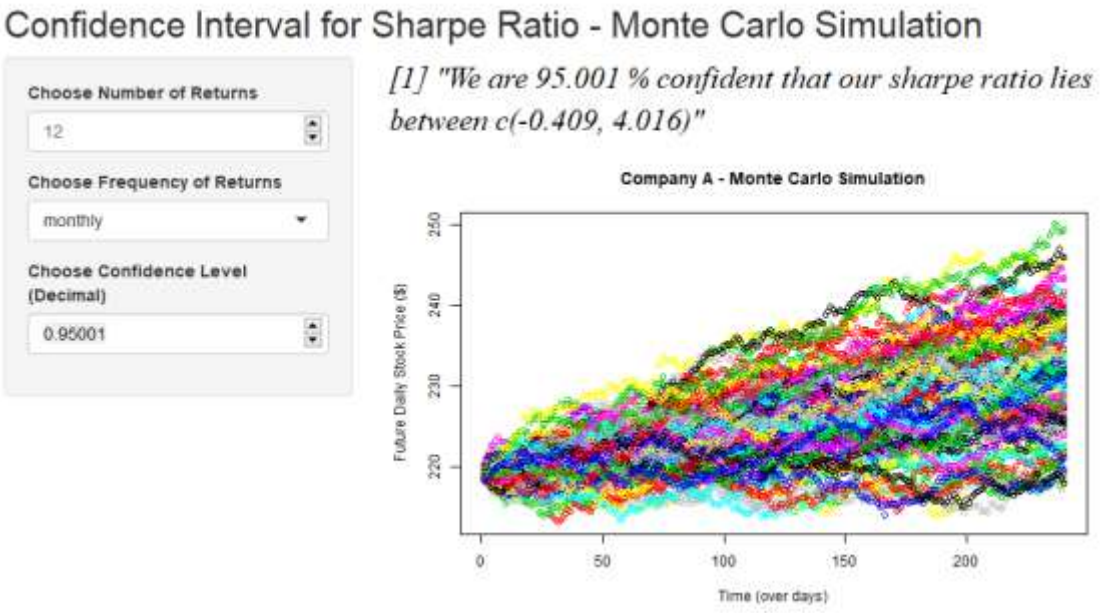


***Figure 1***

I analyzed the tradeoffs of the user inputs by varying them. Holding number of returns and frequency of returns constant, I observed that decreasing the confidence level made our confidence interval for my Sharpe Ratio much narrower. Statistically, this makes sense because decreasing our confidence interval also decreases our critical values (which represent the area under the Probability Density Function of a Normal Bell Curve, as seen in figure 2 on the next page). Smaller critical values make the spread (which equals t.critical.value * standard deviation) smaller, which narrows the confidence interval (mean – spread, mean + spread).

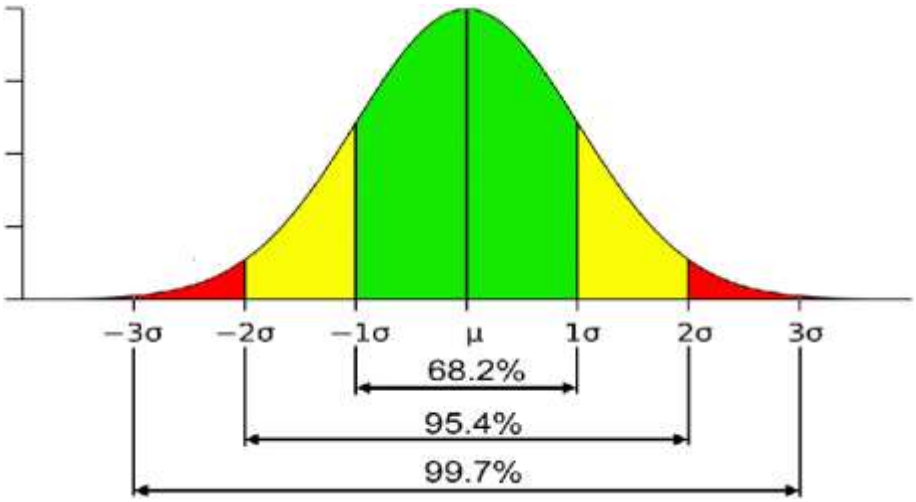


***Figure 2***

The vice versa, which I also observed, also makes sense. Increasing our confidence level also increases our critical values and spread, which widens our confidence interval. Intuitively both observations also make sense. If we had to be highly confident (ex. 99.99% confident) about a confidence interval, then it would be wider because it's nearly almost impossible to be that accurate, so we'd give a large range. Alternatively, if we could be minimally confident (ex. .0001% confident), then we could give a narrower range because it's natural to not be very confident that the true value lies between an extremely small interval, which is unlikely to be accurate.

Lastly, I analyzed the impact of changing the number of returns and frequency of returns while holding a confidence level constant. I observed that the less data (smaller number of returns or lower frequency) generated from my Monte Carlo Simulation, the wider my confidence interval became. Intuitively, this makes sense because with sparse data, it is difficult to come to any substantial solution, so the unpredictability remains high. This is the reason I set the number of simulations n = 100 which is a generally agreed upon sample size number by most statisticians for results to be significant and trustworthy.