Comparing Monte Carlo Simulations and ARIMA Models for Cryptocurrency Price Prediction

Statistical Computing Final Project

Kamaljeet Singh

Abstract

Due to its high volatility and decentralized nature, cryptocurrency prediction has always been a challenge. The crypto market is extremely volatile and prone to a variety of unknown causes, including shifts in market sentiment, regulatory changes, geopolitical events, and technological advancements. This makes it challenging to anticipate the price of cryptocurrencies. In this project, I want to forecast three cryptocurrencies price (Bitcoin, Ethereum, and XRP) using Monte Carlo simulations and the ARIMA (Autoregressive Integrated Moving Average) model. Then, I want to compare the two methods in terms of their prediction accuracy. To compare the prediction accuracy, I will use Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). I will compare the two methods for predictions for the next 7, 15, and 30 days. The project will use historical data of cryptocurrencies prices to train both Monte Carlo simulations and ARIMA models. The results of this project may be helpful for cryptocurrency traders and investors who want to make wise judgments based on price forecasts.

1. Background

Although the origin of crypto can be tracked in 2009 with the launch of Bitcoin, but it captured the public's attention in late 2017, when its price reached close to 20,000 USD. Since then, a large number of other crypto currencies have been introduced by developers across the globe. Due to their volatile nature, crypto investing can be very profitable or equally detrimental to one's financial situation. In order to take a calculated decision to invest in crypto, people have been used a wide range of algorithms including Monte Carlo Simulations, Machine Learning and Deep learning algorithms. In this project, I will use Monte Carlo Simulations and ARIMA (Autoregressive Integrated Moving Average) model to predict the price of three different cryptocurrencies and compare the two methods in terms

of their prediction accuracy. Monte Carlo simulations uses random walk to predict future prices while ARIMA model predicts using the previous values of the crypto price.

The data is downloaded from the website marketwatch for Bitcoin, Ethereum and XRP. The data for each of the coin consists of date, opening price, closing price, average and high price of a day. However, for my analysis I have only used closing price of the day and I will be predicting the closing price of a day.

2. Methodology

2.1 Monte Carlo Simulations

Monte Carlo simulation is a computational technique that uses random sampling to model and analyze complex systems or problems. In Monte Carlo simulations, a model of a system or process is created, and random inputs or parameters are generated to represent the uncertainty or variability in the system. The model is then run many times with different random inputs, generating a distribution of possible outcomes or results. By analyzing the distribution of outcomes, the estimate of the likelihood of different scenarios or events are calculated.
In price prediction problems, Monte Carlo technique is used to understand the impact of risk and uncertainty before making a decision. In our problem of predicting a crypto currency price, the first problem is to identify the underlying model. I will use the Black-Scholes model, which is is a mathematical model used to estimate the price of European-style options. The model was developed by Fischer Black and Myron Scholes in 1973 and is widely used in finance to calculate the theoretical value of options based on various inputs such as the current stock price, the strike price, time to expiration, interest rates, and volatility. The model assumes that the price of the underlying asset follows a geometric Brownian motion. The Black-Scholes model is a key tool in options trading and risk management.

The Black-Scholes Model is given as

$$P_t = P_0 e^{\sigma W_t + (\mu - \sigma^2/2)t}$$

$$P_t = P_{t-1} e^{\sigma Z + (\mu - \sigma^2/2)}$$

where μ is Drift which can be defined as the direction that rates of returns have had in the past.
and $\sigma Z + (\mu - \sigma^2/2)$ is the Volatility, which accounts for the random variation.
Where P_t : Price at time t (In our case,since we are dealing with daily data, t represents day) P_{t-1} : Price at time t-1

The model can be further simplified as

$$P_t = P_{t-1} * e^r$$

Where

$$r = \sigma Z + (\mu - \sigma^2/2)$$

is the Parameter of our model that is estimated using Monte Carlo Simulations.
and

$$Z \sim N(0, 1)$$

To predict the daily return we will use the Brownian Motion which is a Stochastic process used for modelling random behavior over time.

Assessing changes in a variable over time can be a complex task. To facilitate this, one commonly used approach is calculating the "percent change", which involves subtracting one value from another and then dividing the result by a chosen reference quantity. For instance, to determine the percent change between the quantity of a variable from a particular time period and the quantity from the previous period, one can use the following formula:

$$percentage\ change = \frac{Y_t - Y_{t-1}}{Y_{t-1}}$$

Regrettably, the "percent change" method is not symmetrical, meaning that computing the percentage change from a prior period to a subsequent period is not equivalent to calculating the percentage change in the opposite direction, i.e., from the subsequent period to the prior period. In other words, the direction in which the calculation is performed affects the result obtained, which is a limitation of this method.
An alternative method for measuring changes over time is through "log differences". Instead of using percent change, this method involves subtracting the logarithm of one quantity from the logarithm of another. The benefit of this approach is that it produces symmetrical calculations that are consistent in both directions, going forward and backward. The log difference calculated in one direction is the same as the additive inverse of the log difference in the opposite direction. This symmetrical property is an advantage over percent change, which can produce different results depending on the direction of the calculation.

2.2 ARIMA (Autoregressive Integrated Moving Average):

ARIMA model is a class of linear models that forecasts future values using historical data. ARIMA stands for Autoregressive Integrated Moving Average, each of which technique contributes to the final forecast. Theoretically, ARIMA models are the most versatile class of forecasting models for time series that may be made "stationary" by differencing (if necessary). Simply said, stationarity refers to observations that are independent of time. There are three components of ARIMA:

2.2.1 Autoregressive(AR)

In an autoregression model, we make predictions for a certain variable by combining its previous values in a linear manner. The name "autoregression" implies that we are performing a regression of the variable against itself, i.e., we use past values of the variable as predictors to estimate future values. When we refer to an autoregression model of order p, it means

that we are considering p number of previous values of the variable to make our predictions for the future. The formula for an autoregression model of order p can be expressed as:

$$Y_t = \theta_0 + \theta_1 Y_{t-1} + \theta_2 Y_{t-2} ... + \theta_p Y_{t-p}$$

The equation that the present value of the response variable is determined by a linear combination of its past p values. The coefficients of this combination are calculated during the model fitting process. There are various techniques to identify the most suitable value of p, and one of them involves analyzing the Autocorrelation and Partial Autocorrelation plots. These methods help in determining the optimal values for p to achieve the best performance of the model.

2.2.2 Integrated (I)

The term "Integrated" refers to the process of differencing the data in order to make it stationary. To determine the level of differencing required, the data can be tested for stationarity using the Dickey-Fuller test, and different differencing factors can be experimented with. A differencing factor of d=1 implies taking the difference between the current observation and the previous observation (i.e., $Y_t - Y_{t-1}$).

2.2.3 Moving Average (MA)

Moving average models are different from regression models as they use past errors in forecasting future values instead of using past values of the variable of interest. The equation for a moving average model is denoted by MA(q) and can be written as follows:

$$Y_t = \beta_0 + \beta_1 e_{t-1} + \beta_2 e_{t-2} + \beta_3 e_{t-3} + \ldots + \beta_q e_{t-q}$$

In the above equation, e represents the deviations or errors between the model and the actual target variable.

The final equation of the ARIMA(p,d,q) equation is written as:

$$Y_t = C_0 + \theta_1 Y_{t-1}^d + \theta_2 Y_{t-2}^d ... + \theta_p Y_{t-p}^d + \beta_1 e_{t-1} + \beta_2 e_{t-2} + \beta_3 e_{t-3} + \ldots + \beta_q e_{t-q} + e_t$$

Where Y_i^d represents the response variable after it has undergone d rounds of differencing.

To fit an ARIMA model I have used *auto.arima*() function in R, which give you the best parameters for the ARIMA Model.

2.3 Prediction Accuracy

In order to check the prediction accuracy of the two methods, I have used two measures given as:

2.3.1 Root Mean Square Error

Root Mean Squared Error, which is a commonly used metric to evaluate the accuracy of a prediction model. It measures the difference between the predicted values of a model and the actual values, and is calculated by taking the square root of the mean of the squared differences between predicted and actual values. RMSE is a measure of the deviation of the errors between predicted and actual values, and a lower RMSE indicates a better fit between the predicted values and the actual values.

$$RMSE = \sqrt{\frac{\sum(actual - predicted)^2}{n}}$$

2.3.1 Mean Absolute Percentage Error

MAPE measures the percentage difference between the predicted values of a model and the actual values. It is calculated by taking the mean of the absolute percentage differences between the predicted and actual values, where the absolute percentage difference is the absolute value of the difference between the predicted and actual values, divided by the actual value, and multiplied by 100.

$$MAPE = \sum \frac{|actual - predicted|}{|actual|} * 100/n$$

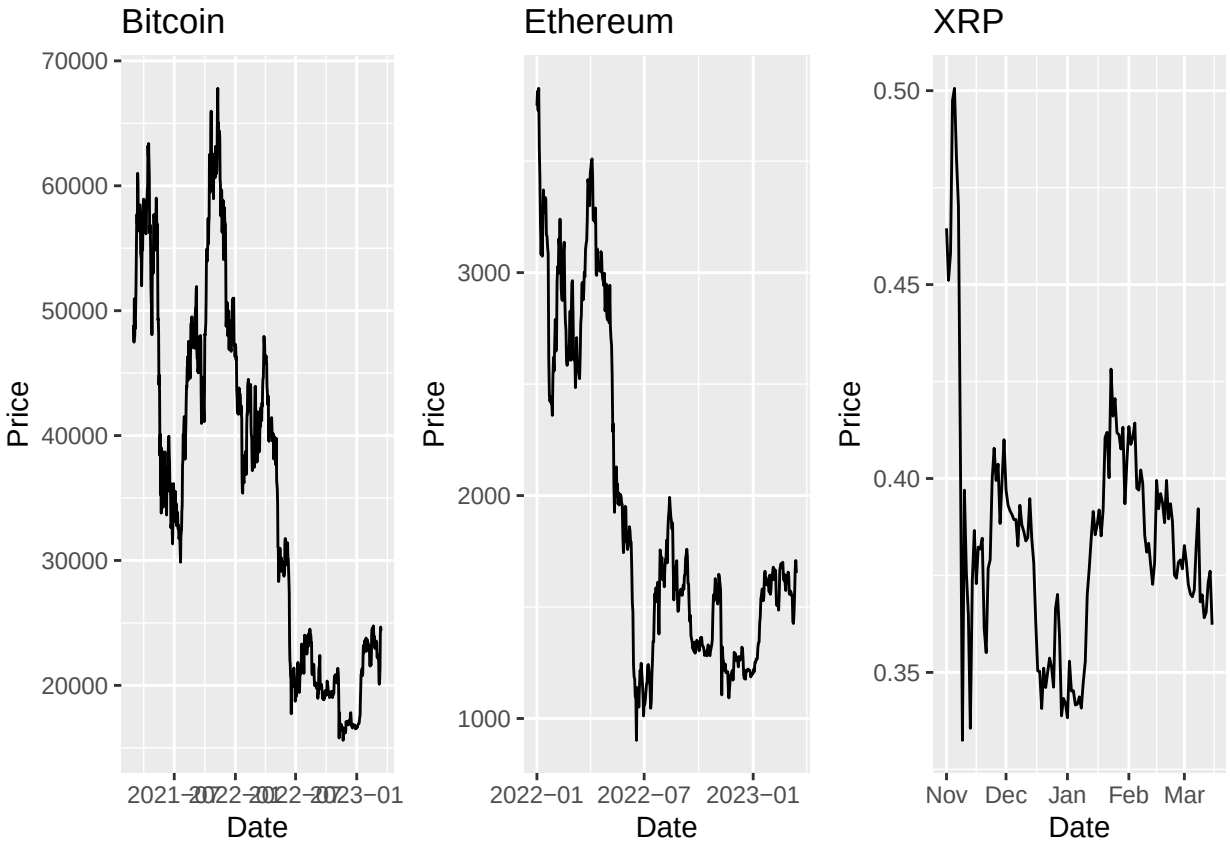
However, for comparison purpose, I haven't multiplied the mean of the relative differences with 100.

3. Results

In order to ensure that my comparison results are not affected by the length of the data used for each coin, I have utilized different lengths of data for each coin. I have used a randomized approach where I selected different subsets of data for each coin with varying lengths. Specifically, I have used Bitcoin data from March 1, 2021 to March 15, 2023, Ethereum data from January 1, 2022 to March 15, 2023, and XRP data from November 1, 2022 to March 15, 2023. By using different lengths of data for each coin, I will be able to obtain more reliable comparison results that are less dependent on the length of the data. This approach also helps to find that if using more data is helpful or it produce biased results by ignoring the recent trends. Overall, using a randomized approach with different lengths of data for each coinis a useful technique to minimize any potential biases and obtain more reliable results in my comparison between Monte Carlo simulationand ARIMA.

For comparison, I have used data from March 16, 2023 to April 14, 2023 as my testing data.

Here is a plot for the three coin's past performance



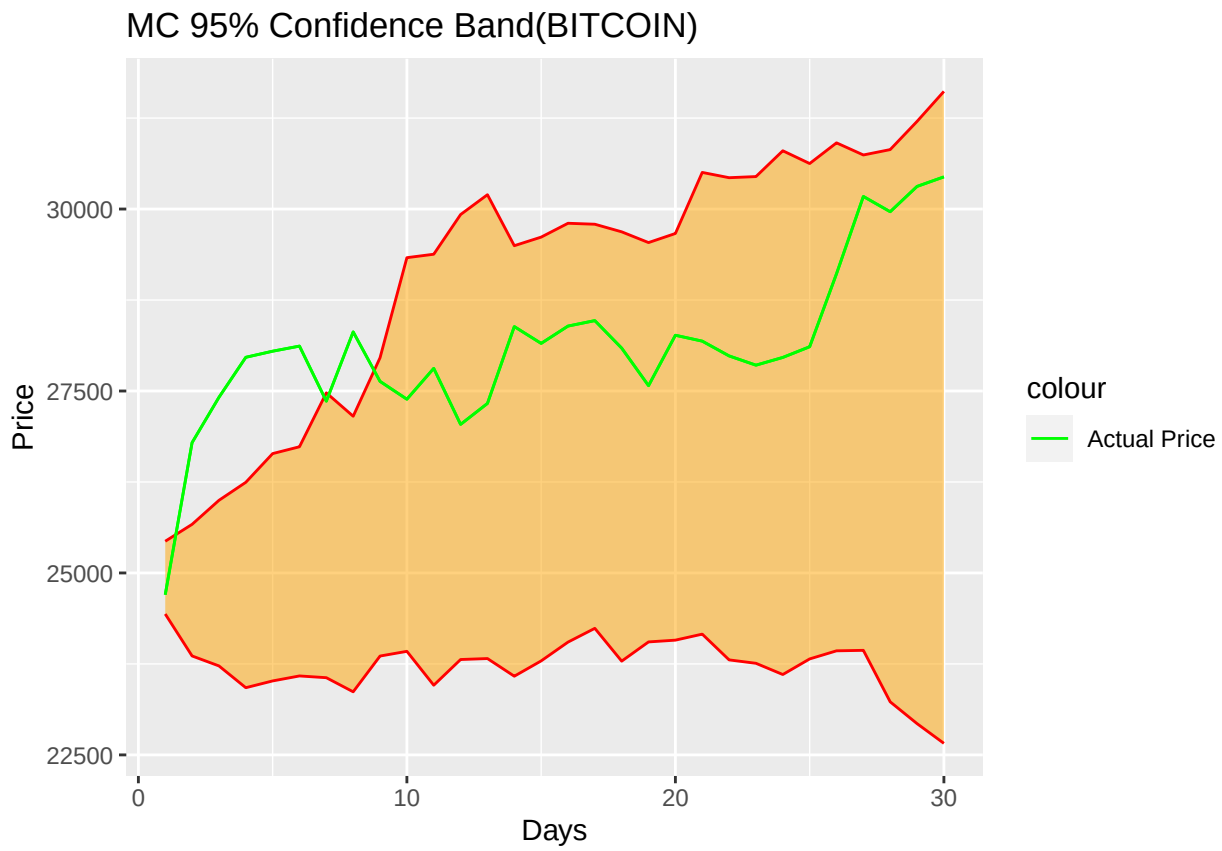
It is very clear from the graphs of Bitcoin and Ethereum that the overall trend for the price is negative. However, upon closer analysis of the graph, it is observed that the recent trend in the price is positive. Unfortunately, the factor μ in $\sigma Z + (\mu - \sigma^2/2)$ gives equal weightage to all the price values and does not recognize the latest trend. This can potentially affect our predictions and lead to incorrect results. To address this issue, I have added an adjustment factor μ_{af} to the mean μ, which is equal to the mean of the log differences of the last 15 days. Specifically,

$$\mu = \mu + \mu_{af}$$

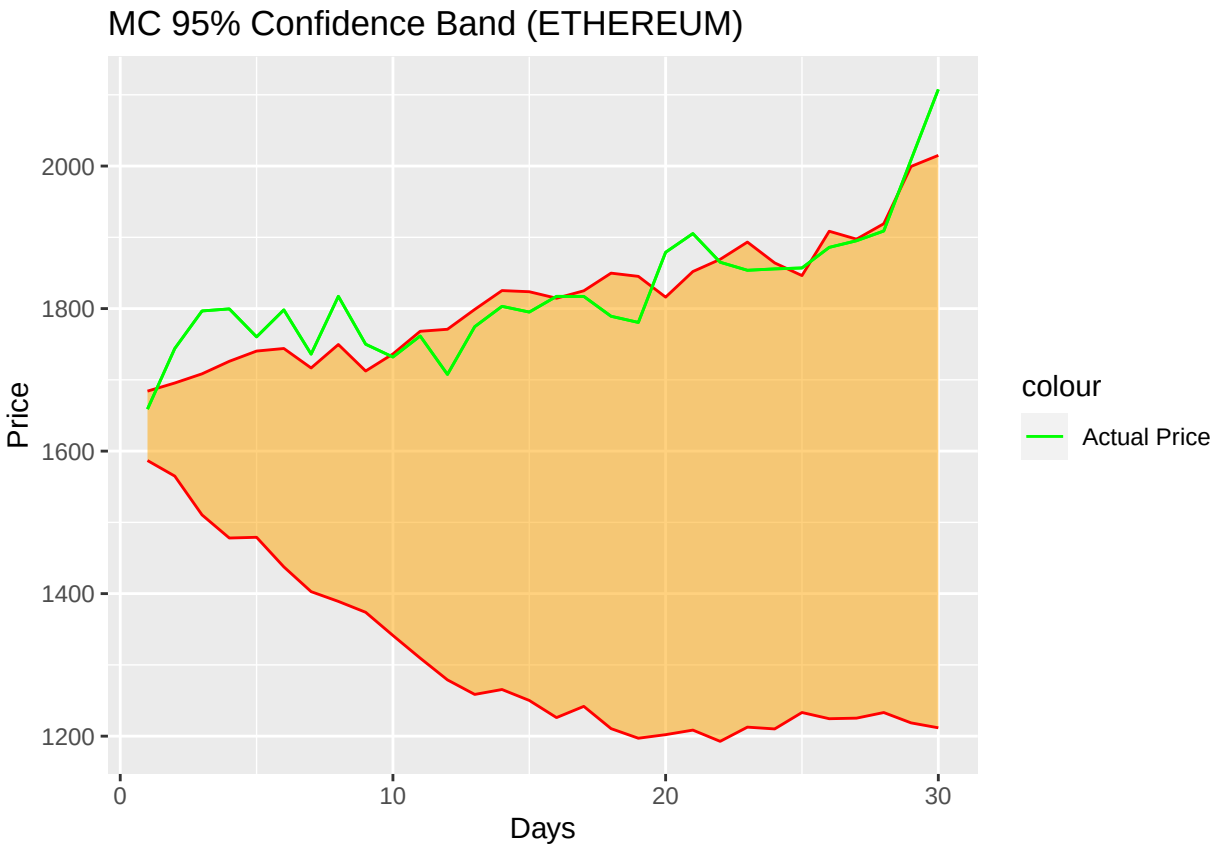
where μ_{af} represents the mean of the log differences of the last 15 days.

The results from the monte carlo simulations are given in below plots:

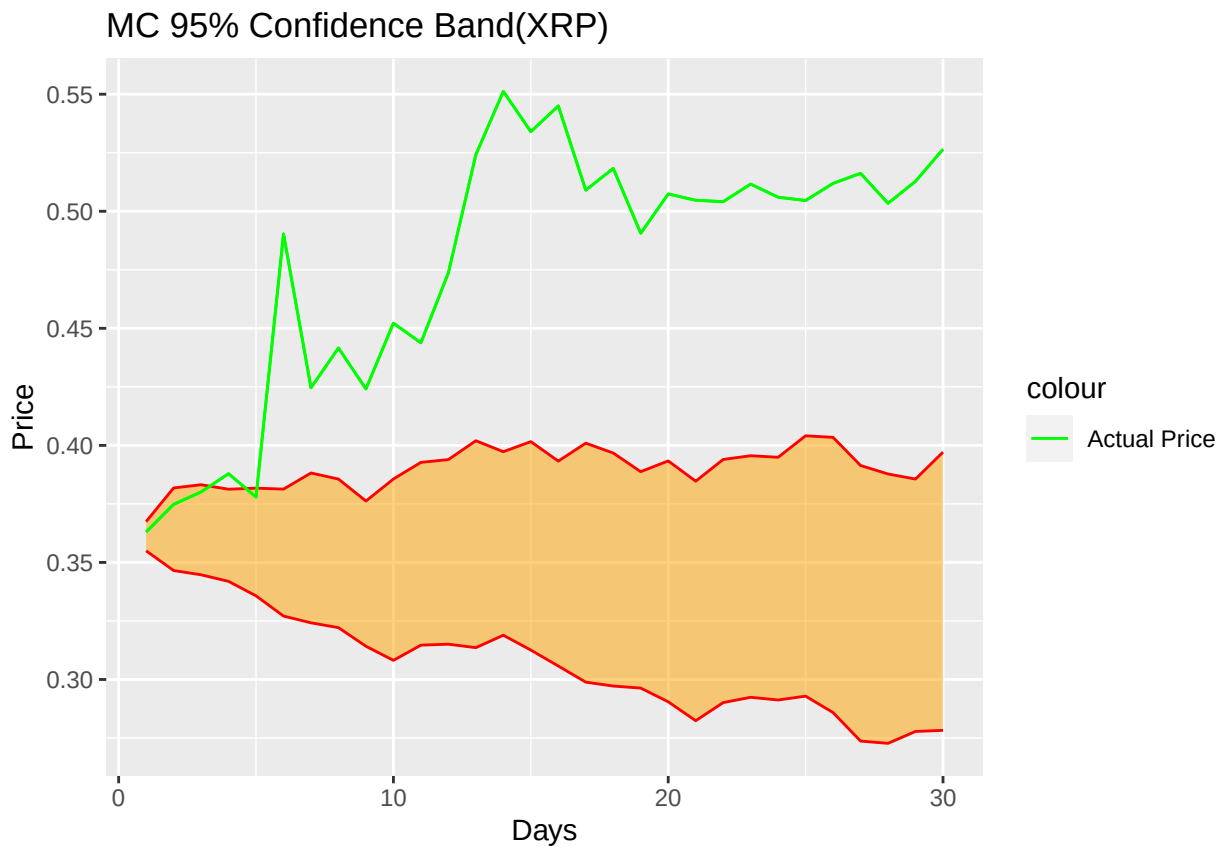
Bitcoin


MC 95% Confidence Band(BITCOIN)
Price
30000
27500
25000
22500
0
10
20
30
Days
colour
Actual Price

Ethereum


MC 95% Confidence Band (ETHEREUM)
Price
2000
1800
1600
1400
1200
0
10
20
30
Days
colour
Actual Price

XRP



ARIMA

I have used *auto.arima*() function to identify the best parameters for the ARIMA model. Here are the model suggested by *auto.arima*() and its corresponding parameters for different cryptocurrencies:

Bitcoin

```
## Series: bitcoin$Close
## ARIMA(2,1,3)
##
## Coefficients:
##           ar1      ar2     ma1     ma2      ma3
##       -1.0803  -0.7110  1.0657  0.6655  -0.0837
## s.e.   0.1687   0.1279  0.1689  0.1376   0.0383
##
## sigma^2 = 1867341:  log likelihood = -6424.91
## AIC=12861.81   AICc=12861.93   BIC=12889.49
```

Ethereum

```
## Series: ethereum$Close
## ARIMA(1,1,1)
##
## Coefficients:
##           ar1     ma1
##       -0.8513  0.8930
## s.e.   0.1356  0.1057
##
## sigma^2 = 7058:  log likelihood = -2561.27
## AIC=5128.54   AICc=5128.6   BIC=5140.79
```

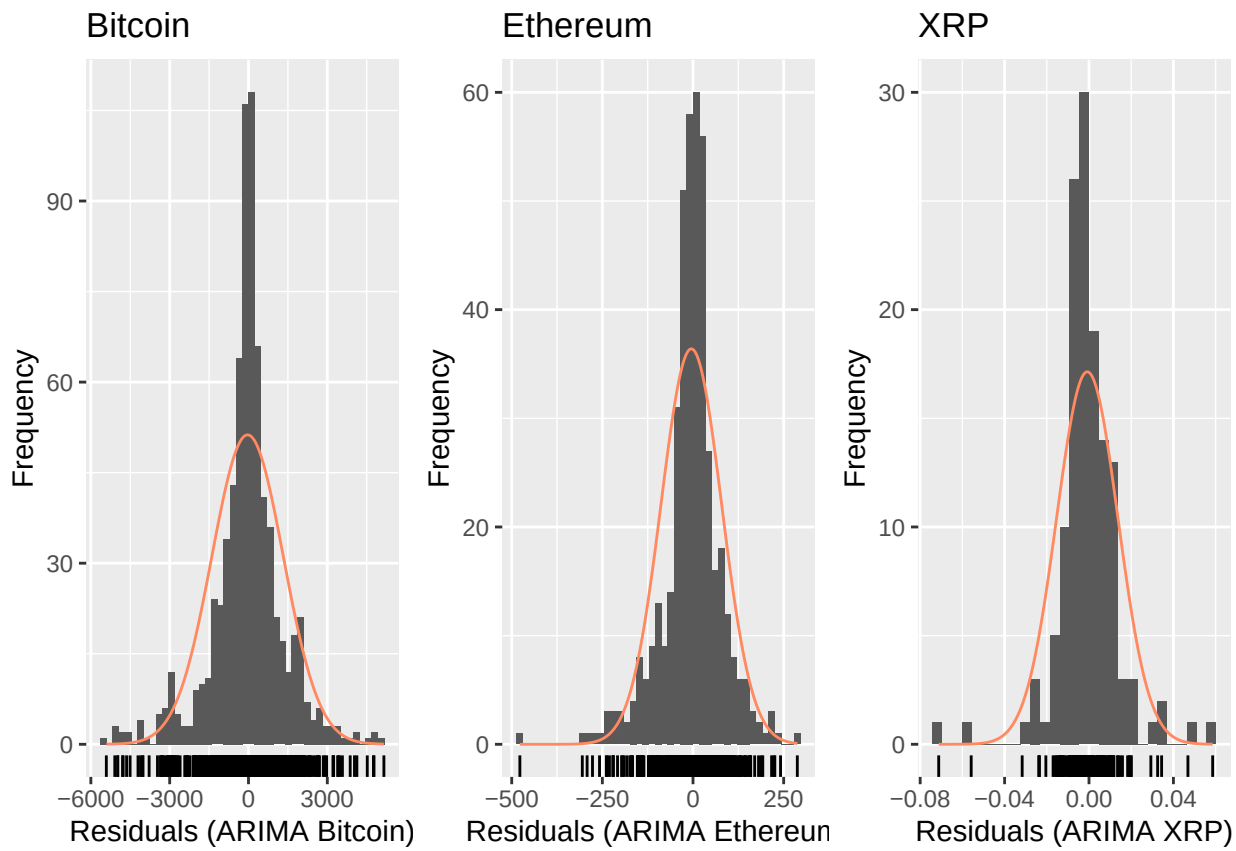
XRP

```
## Series: XRP$Close
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.8934  0.3879
## s.e.  0.0411  0.0111
##
## sigma^2 = 0.0002145:  log likelihood = 378.84
## AIC=-751.67   AICc=-751.49   BIC=-742.96
```

```
## Warning in arima(ethereum$Close, order = c(1, 1, 1)): possible convergence
## problem: optim gave code = 1
```

ARIMA Diagnostics

For checking the residuals of our fitteed ARIMA model's, let's first check the histogram's of the residuals.



After fitting the ARIMA model's for all the three cryptocurrencies, I tested the residuals for the presence of autocorrelation in the residuals. For this, I have used Box-Pierce test which is commonly used after fitting an ARIMA model to a time series data to test for the presence of autocorrelation in the residuals. The null hypothesis is that the residuals are uncorrelated (i.e., independent).

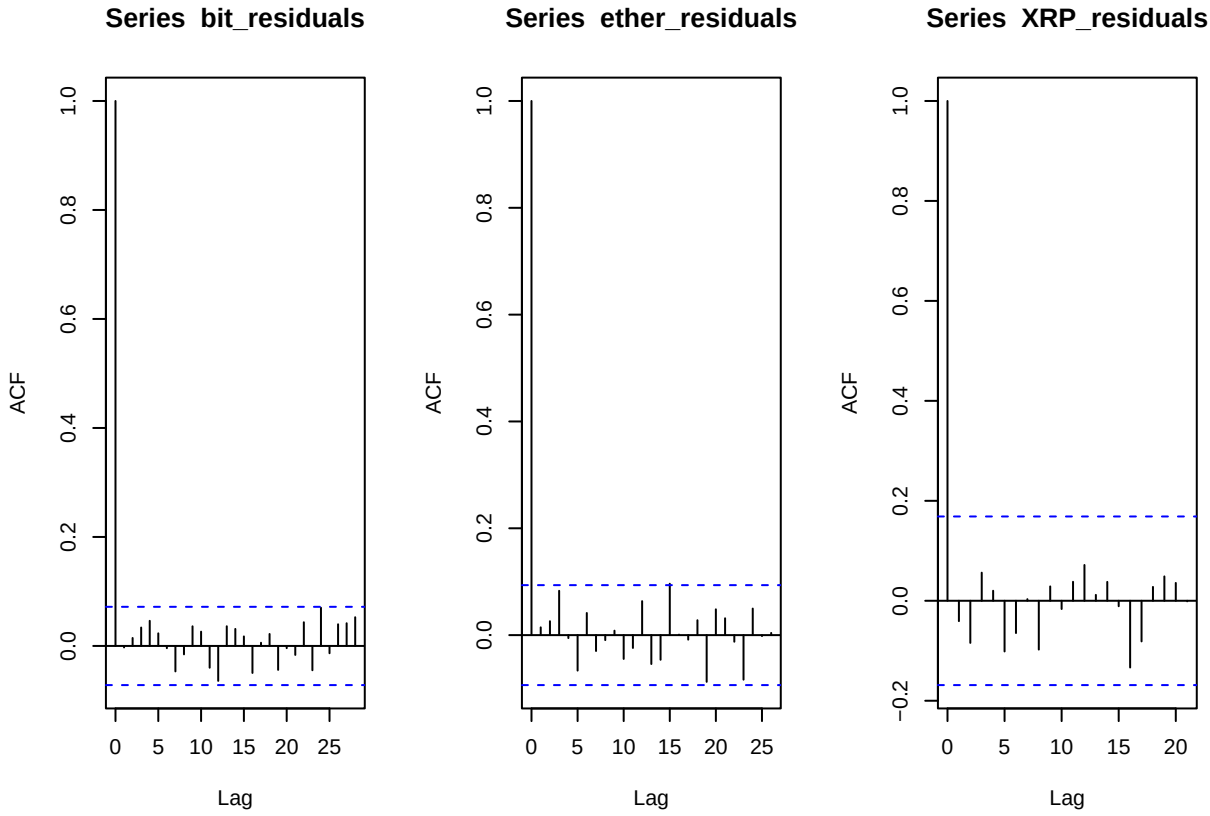
```
##
##  Box-Pierce test
##
## data:  bit_residuals
## X-squared = 0.0054855, df = 1, p-value = 0.941

##
##  Box-Pierce test
##
## data:  ether_residuals
## X-squared = 7.4522, df = 10, p-value = 0.6822

##
##  Box-Pierce test
##
## data:  XRP_residuals
## X-squared = 5.0527, df = 10, p-value = 0.8876
```

Clearly all the p-values are significant, hence our Null Hypothesis of no correlation is not rejected.

Moreover, I have also used Autocorrelation function (ACF) plot of the residuals, which is one of the most common tools used to diagnose whether the residuals of the ARIMA model exhibit any significant autocorrelation.



All the diagnostics test does not indicate any problem in pur fitted models. Hence we are can use our fitted ARIMA models for prediction.

Here are our final tables for comparison of RMSE and MAPE for diiferent methods and coins :

Table 1: BITCOIN (RMSE)

	7 Days	15 Days	30 Days
MC	2432.724	2031.055	1956.914
ARIMA	3064.540	3272.544	3927.643

Table 2: BITCOIN (MAPE)

	7 Days	15 Days	30 Days
MC	0.0812253	0.0634170	0.0597347
ARIMA	0.1038511	0.1139331	0.1327997

Table 3: ETHEREUM (RMSE)

	7 Days	15 Days	30 Days
MC	162.2853	200.4100	277.8059
ARIMA	115.4852	119.0731	192.5444

Table 4: ETHEREUM (MAPE)

	7 Days	15 Days	30 Days
MC	0.0851129	0.1080472	0.1420519
ARIMA	0.0596046	0.0629345	0.0921181

Table 5: XRP (RMSE)

	7 Days	15 Days	30 Days
MC	0.0593284	0.1059103	0.1420259
ARIMA	0.0479756	0.0862169	0.1081385

Table 6: XRP (MAPE)

	7 Days	15 Days	30 Days
MC	0.0904391	0.1802912	0.2563991
ARIMA	0.0651492	0.1377536	0.1915770

4. Conclusion

It is observed that the results for RMSE and MAPE are heading in same direction for the two methods. Hence the final conclusion table can be summarized in one table is given as

Table 7: Final Results

	7 Days	15 Days	30 Days	Data Length
Bitcoin	Monte Carlo	Monte Carlo	Monte Carlo	745
Ethereum	ARIMA	ARIMA	ARIMA	439
XRP	ARIMA	ARIMA	ARIMA	135

Based on the final result table, it is observed that the performance of the two methods varied across the different cryptocurrencies. For Bitcoin, Monte Carlo Simulation method appeared to be a better fit for the data as compared to the ARIMA model. However, for Ethereum and XRP, ARIMA is observed to perform better. Therefore, it can be concluded that the choice of method may depend on the specific characteristics of the data being analyzed. It is important to carefully evaluate the performance of different methods for each individual case in order to make informed decisions about the most appropriate approach. The main points of the results can be summarized as below:

- 1. Monte Carlo Simulation performs better if we have data of very large length as compared to ARIMA.
- 2. The results of the Monte Carlo Simulation also depends upon the adjustment factor (Volatility factor), which should be chosen carefully.
- 3. The predictions from ARIMA appears to be converging to the mean, which may not be a good estimates for a long term prediction.

Reference

1. Effectivness of Monte Carlo Simulation and ARIMA Model in Predicting Stock Prices by Almira Arnaut-Berilo, Azra Zaimović, Nedzmija Turbo-Merdan.
2. Monte Carlo Simulations for Stock Price Predictions
3. How to Use Monte Carlo Simulation With GBM
4. Stock price prediction using ARIMA Model