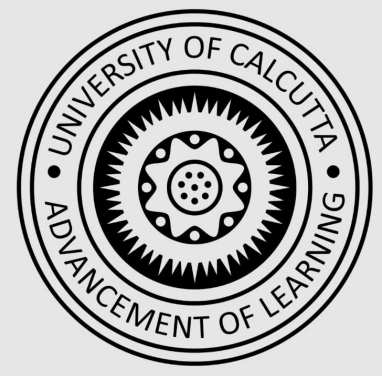
Time Series Analysis of Cryptocurrencies

STAT 421 : PROJECT REPORT

Rishiraj Kundu

A project submitted for the degree of Master of Science
Statistics



University Roll Number : C91/STS/201019
University Registration Number : 133-1111-0100-20
Supervisors
Dr. Gaurangadeb Chattopadhyay and Dr. Bhaswati Ganguly

Contents

1 Introduction

Cryptocurrency is any form of currency that exists digitally or virtually and uses cryptography to secure transactions. Cryptocurrencies don't have a central issuing or regulating authority, instead using a decentralized system to record transactions and use new units.
Cryptocurrency is a digital payment system that don't rely on banks to verify transactions. It is a peer-to-peer system that can enable anyone any where to send and receive payments. Instead of being physical money carried around and exchanged in real world, cryptocurrency payments exist purely as digital entries to an online data base describing specific transactions. When one transfer cryptocurrency funds, the transactions are record in a public ledger. Cryptocurrency is stored in digital wallets.
Cryptocurrency received its name as it uses encryption to verify transactions. This means advanced coding is involved in storing and transmitting cryptocurrency data between wallets and public ledger. The aim of encryption is to provide security and safety.
Cryptos run on a distributed public ledger called block chain, a record of all transactions updated and held by currency holders. Units of cryptocurrency are created through a process called mining, which involves using computed power to solve complicated mathematical problems that generate coins. Users can also buy the currency from broker, then store and spend them using cryptographic wallets. There are thousands of crypto currencies, two of the best known are Bitcoin and Ethereum.
Bitcoin was founded in 2009, it was the first cryptocurrency and is still the most commonly traded. The currency was developed by Satoshi Nakamoto-widely believed to be a pseudonym for an individual or group of people whose precise identity remains unknown. Ethereum developed in 2015, Ethereum is a block chain platform with its own cryptocurrency called Ethereum. It is the most popular cryptocurrency after Bitcoin

2 Data description and source of data

2.1 Data description

For this study I have taken 5 cryptocurrencies namely Bitcoin, Ethereum, Tether, Binance Coin and USD coin. For each coin, data is available on time stamp, opening price of that day, closing price of that day, highest price achieved on that day, lowest price achieved on that day, time of day at which the highest price was achieved and time at which lowest price was

achieved. Two sets of data were used for each of 5 cryptocurrencies, namely training data and testing data. For Bitcoin, training data is available from 29^{th} April 2013 till 21^{st} February 2022, for Ethereum it is available from 8^{th} August 2015, for Tether it is available from 26^{th} February 2015, for Binance coin it is available from 26^{th} July 2015 and for USD coin it is available from 9th October 2019 till 21^{st} February 2022. Data for next 68 days are used to compare the predictions from the models.

2.2 Data source

Data is collected using a library called "crypto2" in R programming language. This library retrieves crypto currency information and historical prices as well as information on exchanges they are listed on. All data is scraped from https://coinmarketcap.com via their web api. The functions used to extract data can be updated anytime to get current data on the cryptocurrencies.

3 Objectives of this project

- Aim of this project is to analyse the daily closing prices of each of the 5 cryptocurrencies through appropriate time series plots.
- Then the next goal is to model the daily closing price data using suitable time series models. For this project I have restricted myself to ARIMA models and Exponential smoothing models. After fitting, the models will be compared on training and testing data for each cryptocurrency.
- After that I have used two different versions of GARCH method to model the volatility of daily returns data. (Daily returns data can be calculated from closing prices)

4 Method of Analysis : ARIMA

4.1 General form of ARIMA

$$\phi(B)\nabla^d Z_t = \theta(B)\epsilon_t$$

where,

$$\phi(B) = 1 - \phi_1(B) - \phi_2(B^2) - ... - \phi_p(B^p)$$
$$\theta(B) = 1 + \theta_1(B) + \theta_2(B^2) + ... + \theta_q(B^q)$$

$\phi(B)$ is called the autoregressive operator, $\phi(B)\nabla^d$ is called the generalized autoregressive operator, $\theta(B)$ is called moving average operator.
It is assumed that $\epsilon_t \sim WN(0, \sigma^2)$

4.2 Identification of orders of AR and MA processes

An ARIMA model has 3 parameters namely p, d, q. ARIMA model are applied to non-stationary data. Parameter "d" represents the number of times the series must be differenced in order to get a stationary series. After finding the appropriate degree of differencing for the data, we try to determine a suitable choice for the orders "p" and "q" for the autoregressive and moving average operator. If the autocorrelation function of an autoregressive process of order "p" tails off, its partial auto correlation function has a cut off after lag "p". Conversely, the auto correlation function of a moving average process of order "q" has a cut off after lag "q", while its partial auto correlation function tails off. If both the auto correlation and partial auto correlation tail off, a mixed process is suggested. In general, autoregressive (moving average) behaviour as measured by the autocorrelation function, tends to follow moving average (auto regressive) behaviour as measured by the partial auto correlation function. For example, the auto correlation function of a first-order AR process decays exponentially, while partial auto correlation function cuts of after the first lag. Correspondingly, for a first-order MA process, the auto correlation function cuts off after the first lag. The orders of auto regressive and moving average processes are "p" and "q", when the PACF and ACF values tail off after lag "p" and "q" respectively. Often seen while dealing the data, the ACF and PACF plots shows significant spikes at irregular intervals till very high lags. In those cases, a suitable metric of accuracy or a model evaluating criterion is selected and models are fitted for different combination of "p" and "q". The pair providing best results is the selected for the final model.

4.3 Stationarity of Series

Before checking auto correlation function (ACF) and partial auto correlation (PACF) plots we must check if the data is stationary. If not, the series needs to be differenced and then again ACF and PACF plots need to be checked. To check the stationarity of a series we use Augmented Dicky fuller test (ADF test), which tests upon the null hypothesis that the series is non-stationary. If p-value of the ADF test is less than 0.01, the null hypothesis

is rejected and we conclude series is stationary.

4.4 Model evaluation and performance

In order to evaluate the model fitting and compare with other models Akaike information criterion (AIC) has been used. In order to ensure that the residuals don't have any identifiable pattern and $e_t \sim WN(0, \sigma^2)$ Ljung-Box test is used. In order to have an idea about the fit and performance of the model we use measures like root mean squared error, mean absolute error, mean percentage error.

5 Graphical comparison of market capitalization

Market cap is used as an indicator of dominance and popularity of cryptocurrencies. In general, higher the market cap of a cryptocurrency, the more dominant it is considered to be in the market.

$$MarketCap = Price \times CirculatingSupply$$

5.1 Bitcoin and Ethereum market capitalization

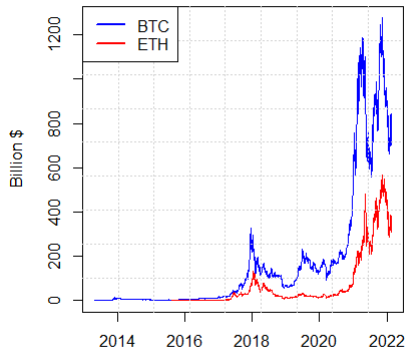


Figure 1: BTC vs ETH market cap

From the above graph we can understand why Bitcoin and Ethereum are the 2 most popular cryptocurrencies. Till the last quarter of 2017 we don't

see much movement either of the coins. At the beginning of 2018, both the coins show signs of popularity as their market caps hit 251 billion and 121 billion dollars respectively. At the beginning of 2019 we can see market cap of bitcoin dipped to 64 billion dollars (74% fall in 1 year), Ethereum was around 13 billion dollars in the beginning of 2019 (71% fall in 1 year). Another significant spike can be seen for bitcoin at end of 2019. In April 2021 market cap of Bitcoin was around 1117 billion dollars (673% growth in 1 year). Similarly, Ethereum grew about 2076% in a year reaching around 458 billion dollars in May 2021. Both the coins reached their all time high (for this data set) on 8th November 2021, with market caps 1274 billion dollar and 569 billion dollars respectively. Over all 2021 has been a record year for crypto market, this was achieved after major institutional investors and notable financial companies began to support the cryptocurrency earlier in 2021. Companies including Tesla, Square and MicroStrategy started to use their balance sheets to buy bitcoin, the NFT market also saw great highs, this also promoted crypto exchanges a lot. El Salvador passed a new law in June to adopt bitcoin as a legal tender. This law allows bitcoin to be used as payment for goods and taxes in El Salvador and exchanges were not subjected capital gains tax. There are many more reasons that contributed immensely to growth of crypto market in 2021 which are more technical and beyond the scope of this study.

5.2 Tether, Binance Coin and USD coin market cap

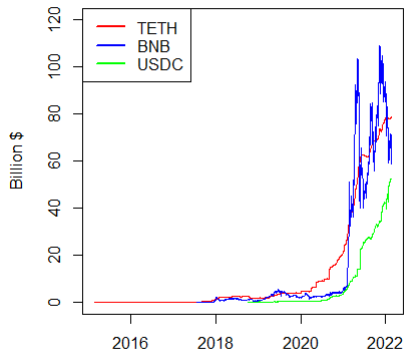


Figure 2: TETH, BNB, USDC market cap

It can bee seen not much has happened till beginning of 2020. After that the market cap has grown exponentially in 2021. USD coin and tether shows us a relatively smooth growth curve compared to binance coin. For this data set, Tether reached its all time highest on 2nd February 2022 (79 billion dollars), binance coin reached its all-time highest on 8^{th} November 2021 (109 billion dollars) and USD coin reached its all time highest on 19^{th} February 2022 (57 billion dollars).

6 Analysis and results of ARIMA models

The analysis is done on daily closing price of cryptocurrencies for last 730 days. Thus, we have 730 data points for each of coins.

6.1 Bitcoin

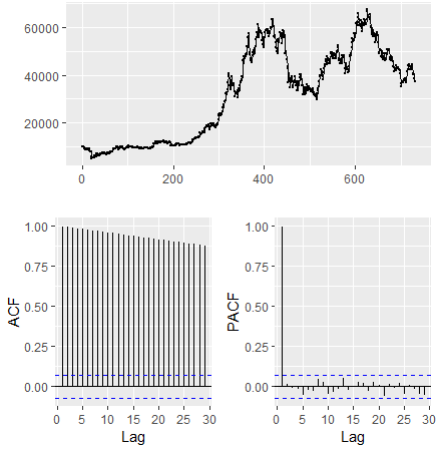


Figure 3: BTC closing price

From visual inspection, it can be seen there is a kind of upward trend in data. Now by performing Augmented Dicky Fuller test we get p-value of 0.8332. Thus, we cannot reject null hypothesis and this time series is non stationary.
Thus, we need to further difference the data to an appropriately differenced ARIMA model. The differenced series of order 1 and its respective ACF and PACF are given below.

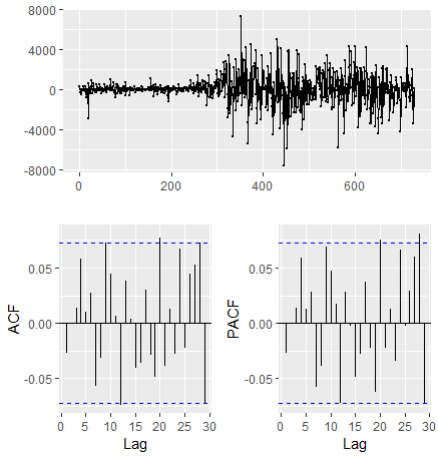


Figure 4: First difference of BTC closing price

Now, this differenced series seems to be stationary by visual inspection. P-value of ADF test on differenced series is less than 0.01. Thus, we reject null hypothesis and conclude the series to be stationary. We can see some significant spikes even up-to lag order 28. Thus, we cannot identify order of AR and MA process by inspecting ACF and PACF plots.
Thus a 5 by 5 grid was taken and in $(i, j)^{th}$ cell AIC from ARIMA(i,1,j) is stored. The combination giving minimum AIC was chosen as final model.
For Bitcoin data ARIMA(3,1,2) gives minimum AIC of 12668.39.

6.1.1 Model parameters for ARIMA on Bitcoin data

Table 1: Bitcoin data

	AR 1	AR 2	AR 3	MA 1	MA 2	AIC	Log Likelihood	σ^2
Estimate	0.832	-0.974	-0.007	-0.869	1	12668.39	-6328.2	1969756
(SE)	-0.037	-0.031	-0.037	-0.004	-0.006			

6.1.2 Residual analysis for ARIMA on Bitcoin data

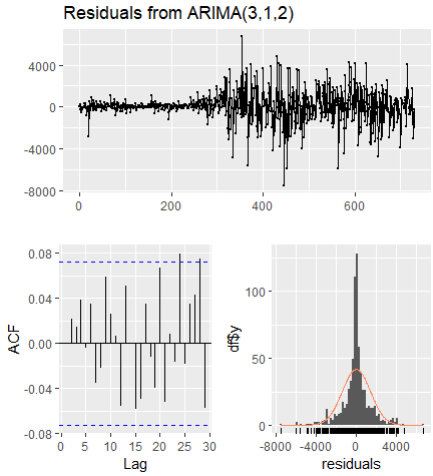


Figure 5: Residuals of ARIMA(3, 1, 2) on BTC data

After model fitting Ljung-Box test was applied to residuals of the model to check if the residuals contain auto correlation. P-value of the test is 0.2316, thus, we cannot reject null hypothesis. Hence randomness of residuals is certain. From the histogram we can also see the residuals are centered around 0, with heavier tails than normal distribution. This information will be used in later part of this project where volatility modelling has been done.

6.1.3 Fitted and Predicted values

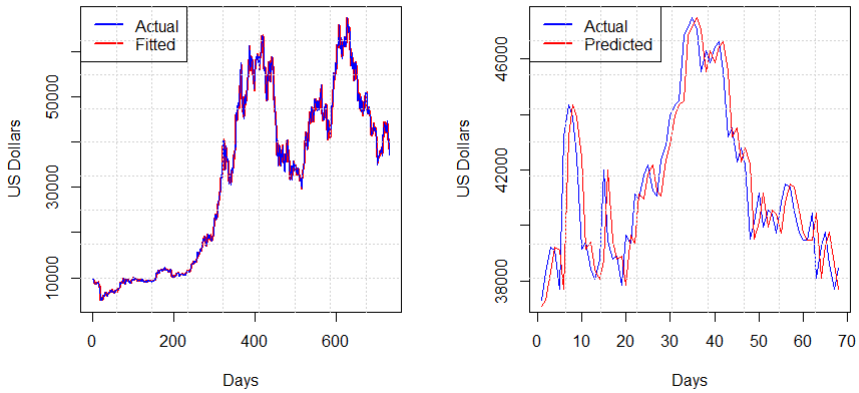


Figure 6: Actual vs Fitted(left) and Actual vs Predicted(right) for BTC data

Table 2: Bitcoin data

	RMSE	MAE	MAPE
Training data	1402.52	884.47	2.72
Testing data	1347.055	981.33	2.38

From the above table it is clear that model fits the data very well, mean absolute percentage error is 2.38% which means the model predictions are off by about 2.38%.

6.2 Ethereum

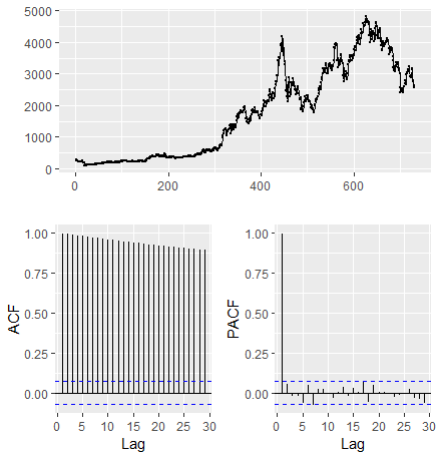


Figure 7: ETH closing price

From visual inspection, it can be seen there is a kind of upward trend in data. Now by performing Augmented Dicky Fuller test we get p-value of 0.4982. Thus, we cannot reject null hypothesis and this time series is non stationary.
Thus, we need to further difference the data to an appropriately differenced ARIMA model. The differenced series of order 1 and its respective ACF and PACF are given below.

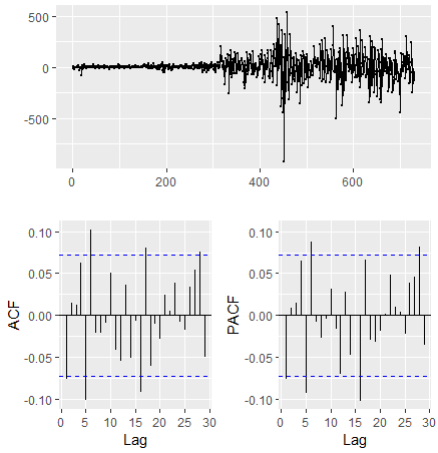


Figure 8: First differnce of ETH closing price

Now, this differenced series seems to be stationary by visual inspection. P-value of ADF test on differenced series is less than 0.01. Thus, we reject null hypothesis and conclude the series to be stationary. We can see some significant spikes even up-to lag order 30. Thus, we cannot identify order of AR and MA process by inspecting ACF and PACF plots.

Thus a 5 by 5 grid was taken and in $(i, j)^{th}$ cell AIC from ARIMA (i, 1, j) is stored. The combination giving minimum AIC was chosen as final model. For Ethereum data ARIMA (5, 1, 5) gives minimum AIC of 8937.248.

6.2.1 Model parameters for ARIMA on Ethereum data

Table 3: Ethereum data

AR1	-1.37 (0.10)	MA1	1.32 (0.11)
AR2	-0.44 (0.11)	MA2	0.38 (0.12)
AR3	0.62 (0.07)	MA3	-0.62 (0.09)
AR4	1.33 (0.12)	MA4	-1.24 (0.13)
AR5	0.85(0.06)	MA5	-0.83 (0.05)
AIC: 8937.25			
Log likelihood: -4457.62			
σ^2 : 11775			

6.2.2 Residual Analysis for ARIMA on Ethereum data

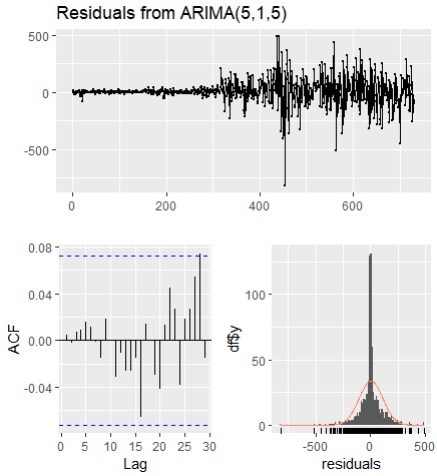


Figure 9: Residuals of ARIMA(5, 1, 5) on ETH data

After model fitting Ljung-Box test was applied to residuals of the model to check if the residuals contain auto correlation. P-value of the test is 0.545, thus, we cannot reject null hypothesis. Hence randomness of residuals is certain. From the histogram we can also see the residuals are centered around 0, with heavier tails than normal distribution. This information will be used in later part of this project where volatility modelling has been done.

6.2.3 Fitted and Predicted values

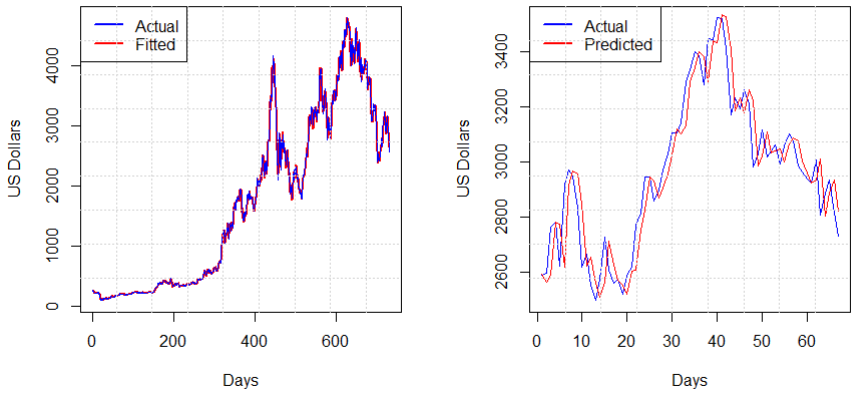


Figure 10: Actual vs Fitted(left) and Actual vs Predicted(right) for ETH data

Table 4: Training and testing errors

	RMSE	MAE	MAPE
Training data	108.43	64.40	3.70
Testing data	101.22	77.61	2.65

From the above table it is clear that model fits the data very well, mean absolute percentage error is 2.65% which means the model predictions are off by about 2.65%.

6.3 Tether

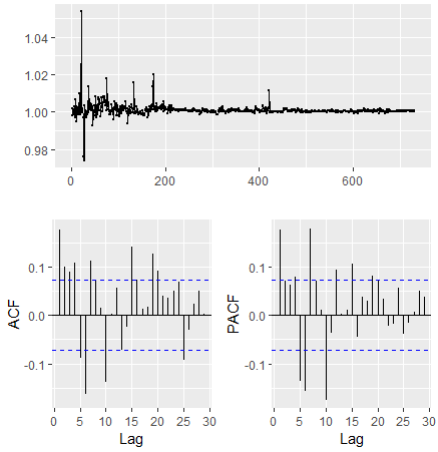


Figure 11: TETH daily closing price

From visual inspection, it can be seen there is no trend in data. Now by performing Augmented Dicky Fuller test we get p-value of 0.3755. Thus, we cannot reject null hypothesis and this time series is non stationary.
Thus, we need to further difference the data to an appropriately differenced ARIMA model. The differenced series of order 1 and its respective ACF and PACF are given below.

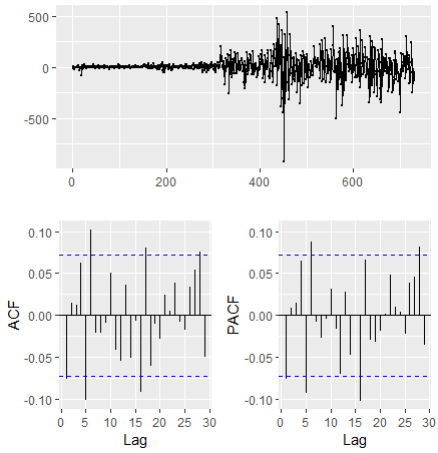


Figure 12: First difference of TETH daily closing price

Now, this differenced series seems to be stationary by visual inspection. P-value of ADF test on differenced series is less than 0.01. Thus, we reject null hypothesis and conclude the series to be stationary. We can see some significant spikes even up-to lag order 28. Thus, we cannot identify order of AR and MA process by inspecting ACF and PACF plots.

Thus, using similar method ARIMA(4, 1, 5) was found to be the best model, giving AIC of -6448.378.

6.3.1 Model parameters for ARIMA on Tether data

Table 5: Tether data

AR1	1.03 (0.13)	MA1	-1.93 (0.16)
AR2	-1.69 (0.14)	MA2	2.68 (0.33)
AR3	1.03 (0.12)	MA3	-2.61 (0.36)
AR4	-0.7149 (0.10)	MA4	1.60 (0.29)
		MA5	-0.71 (0.10)
AIC: -6448.38			
Log likelihood: 3234.19			
σ^2 : 8.23×10^{-6}			

6.3.2 Residual analysis for ARIMA on Tether data

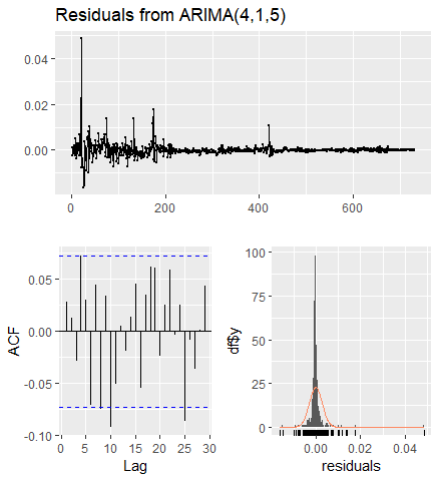


Figure 13: Residuals of ARIMA(4, 1, 5) on TETH data

After model fitting Ljung-Box test was applied to residuals of the model to check if the residuals contain auto correlation. P-value of the test is 2.210^{-5}, thus, we reject null hypothesis. Hence residuals are not random and contains significant amount of auto correlation. From the histogram we can also see the residuals are positively skewed.

6.3.3 Fitted and Predicted values

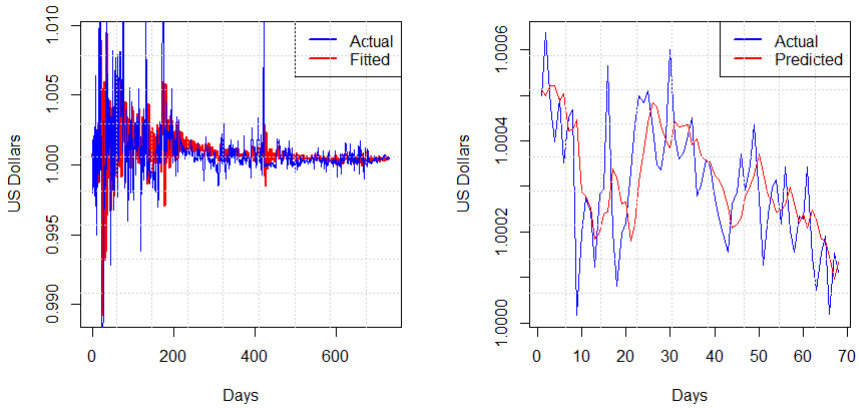


Figure 14: Actual vs Fitted(left) and Actual vs Predicted(right) for TETH data

Table 6: Training and testing errors

	RMSE	MAE	MAPE
Training data	0.0028	0.00121	0.121
Testing data	0.00012		0.0087

All though Ljung Box test suggests the residuals are auto correlated. From the above measures it can be seen that model fits the data very well. In case of future predictions MAPE is 0.0087, indicating model predictions are off by 0.0087

6.4 Binance Coin

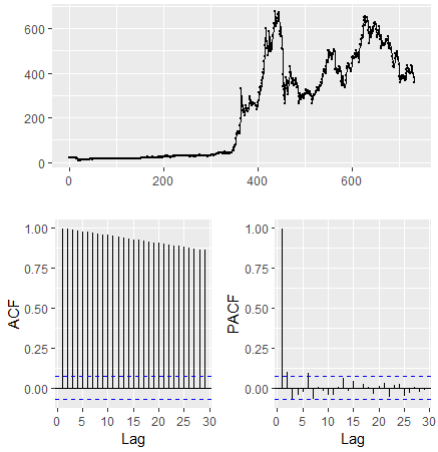


Figure 15: BNB daily closing price

From visual inspection, it can be seen there is a kind of upward trend in data. Now by performing Augmented Dicky Fuller test we get p-value of 0.4276. Thus, we cannot reject null hypothesis and this time series is non stationary.
Thus, we need to further difference the data to an appropriately differenced ARIMA model. The differenced series of order 1 and its respective ACF and PACF are given below.

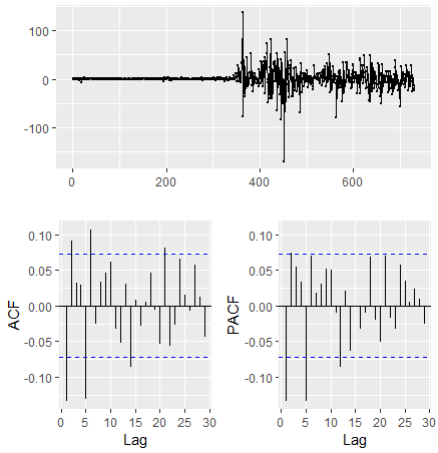


Figure 16: First differnce of BNB daily closing price

Now, this differenced series seems to be stationary by visual inspection. P-value of ADF test on differenced series is less than 0.01. Thus, we reject null hypothesis and conclude the series to be stationary. We can see some significant spikes in ACF plot even up-to lag order 25. Thus, we cannot identify order of MA process by inspecting ACF . In PACF plot last significant spike is seen at lag 5. Proceeding in usual method optimal parameters are obtained for BNB coin. It was found that ARIMA (4, 1, 3) is the best model in terms of minimum AIC. AIC was found to be 6319.38.

6.4.1 Model parameters for ARIMA on BNB data

Table 7: BNB data

AR1	-1.14(0.16)	MA1	1.02(0.16)
AR2	-0.82(0.24)	MA2	0.80(0.23)
AR3	-0.27(0.27)	MA3	0.35(0.24)
AR4	0.12(0.06)		
AIC: 6319.38			
Log likelihood: -3151.69			
σ^2: 329.2			

6.4.2 Residual analysis for ARIMA on BNB data

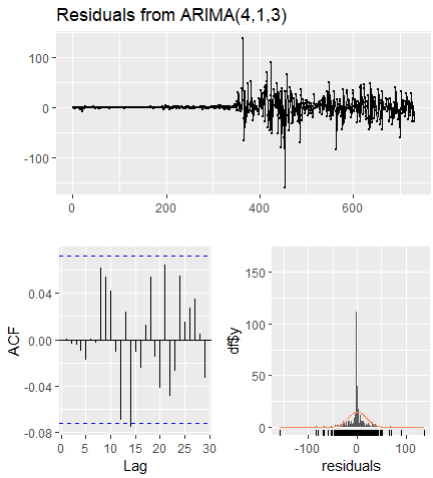


Figure 17: Residuals of ARIMA(4, 1, 3) on BNB data

After model fitting Ljung-Box test was applied to residuals of the model to check if the residuals contain auto correlation. P-value of the test is 0.08263, thus, we cannot reject null hypothesis. Hence randomness of residuals is certain. From the histogram we can also see the residuals are centered around 0, with flat tails.

6.4.3 Fitted and Predicted values

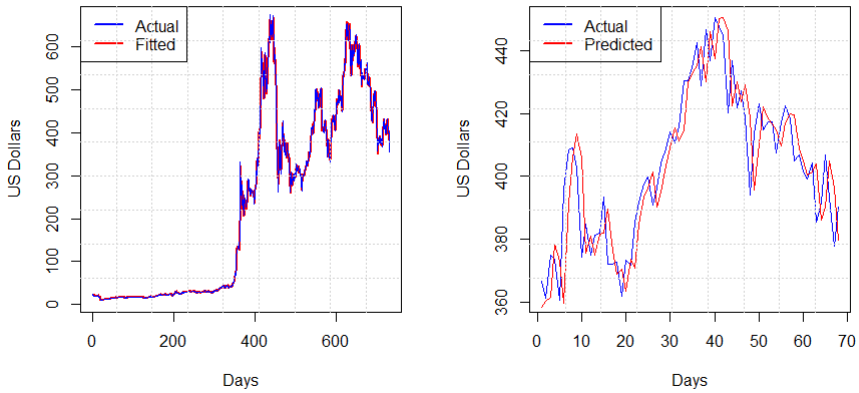


Figure 18: Actual vs Fitted(left) and Actual vs Predicted(right) for BNB data

Table 8: Training and testing errors

	RMSE	MAE	MAPE
Training data	18.13	9.51	3.96
Testing data	11.78	9.28	2.31

6.5 USD Coin

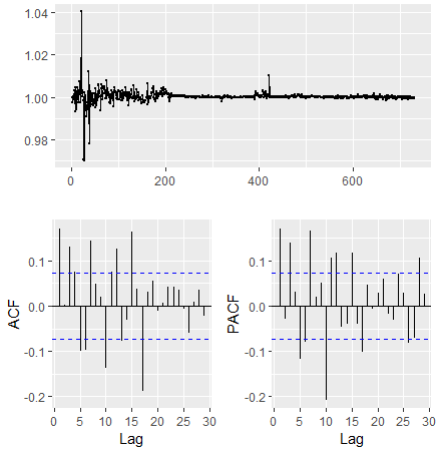


Figure 19: USDC daily closing price

From visual inspection, it can be seen there is no trend in data. Now by performing Augmented Dicky Fuller test we get p-value less than 0.01. Thus, we reject null hypothesis and this time series is stationary.
Since we can see some significant spikes even upto higher order lags, a similar method as used and ARIMA(5, 0, 5) was found to be the best model, giving AIC of -6618.123.

6.5.1 Model parameters for ARIMA on USDC data

Table 9: USDC data

AR1	-1.47	MA1	1.67
AR2	-1.12	MA2	1.48
AR3	-0.98	MA3	1.49
AR4	-0.09	MA4	0.64
AR5	0.42	MA5	-0.16
AIC : -6618.12			
Log Likelihood : 3321.06			
σ^2 : 6.6×10^{-6}			

6.5.2 Residual analysis for ARIMA on USDC data

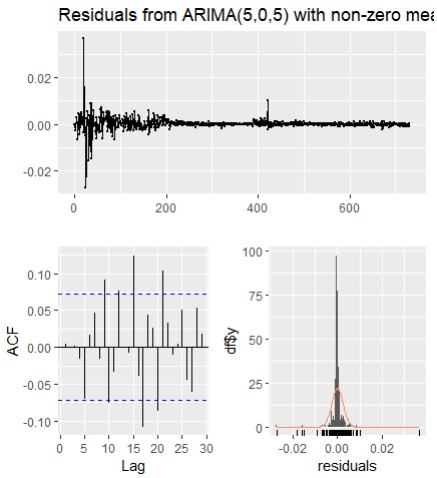


Figure 20: Residuals of ARIMA(5, 0, 5) on USDC data

After model fitting Ljung-Box test was applied to residuals of the model to check if the residuals contain auto correlation. P-value of the test is 6.9610^{-7}, thus, we reject null hypothesis. Hence residuals are not random and contains significant amount of auto correlation.

6.5.3 Fitted and Predicted values

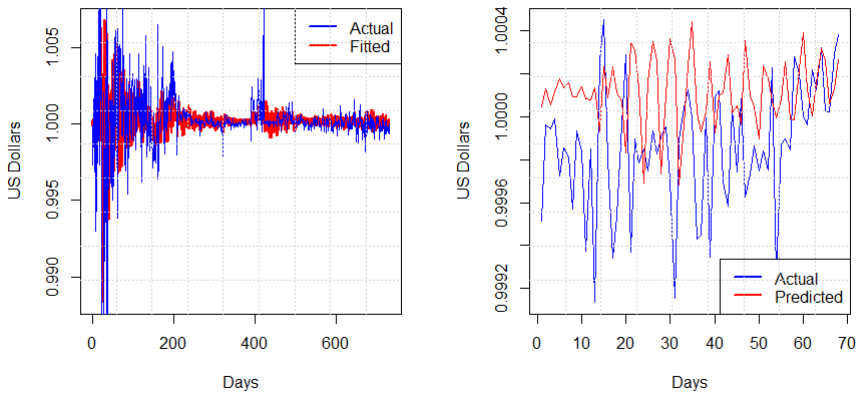


Figure 21: Actual vs Fitted(left) and Actual vs Predicted(right) for USDC data

Table 10: Training and testing errors

	RMSE	MAE	MAPE
Training data	0.002564919	0.001079315	0.1079088
Testing data	0.000429051	0.000341062	0.03411728

From the above table it is clear that model fits the data very well, mean absolute percentage error is 0.034% which means the model predictions are off by about 0.034%.

7 Method of Analysis : Exponential Smoothing

Exponential smoothing is a time series forecasting method for univariate data. Exponential smoothing forecasting methods are similar in that a prediction is a weighted sum of past observations, but the model explicitly uses an exponentially decreasing weight for past observations.

7.1 Simple Exponential Smoothing

The simplest of the exponentially smoothing methods is naturally called simple exponential smoothing(SES). This method is suitable for forecasting data with no clear trend or seasonal pattern.
Predicted X^p_{t+1} is a linear combination of past values.

$$X^p_{t+1} = C_0 X_t + C_1 X_{t-1} + ...$$

As the coefficients decrease exponentially, $C_j = \alpha(1-\alpha)^j, j = 0, 1, 2...; 0 < \alpha < 1$, Thus, $X^p_{t+1} = \alpha X_t + (1-\alpha)X^p_t$ The predictor is obtained as a convex combination of the current observed value and the current predicted value. α is selected using trial and error method to get the best fit.

7.2 Exponential Smoothing with Trend

$$X_{t+h} = m_t + hT_t$$

where, m_t is the mean, T_t is trend at t. Now, using exponential smoothing it is updated as,

$$m^p_t = \alpha X_t + (1-\alpha)(m^p_{t-1} + T^p_{t-1})$$

and

$$T^p_t = \beta(m^p_t - m^p_{t-1}) + (1-\beta)T^p_{t-1}$$

where, $0 < \alpha, \beta < 1$.
Thus,

$$X^p_{n+h} = m^p_n + hT^p_n$$

7.3 Results and findings

7.3.1 Model parameters

Table 11: Exponential smoothing model parameters

	α	β	AIC
Bitcoin	0.9704	0.0075	15444.87
Ethereum	0.9261	10^{-4}	11716.58
Tether	0.0418	10^{-4}	-3606.579
Binance Coin	0.8864	10^{-4}	9092.26
USD coin	10^{-4}		-3755.424

7.3.2 Forecasting

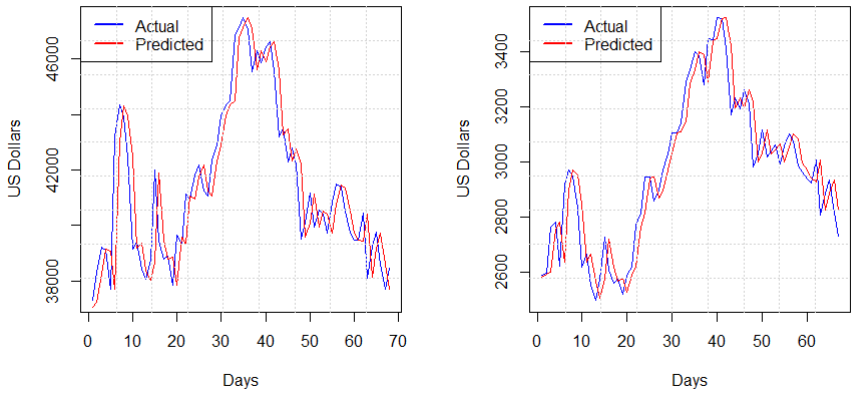


Figure 22: Bitcoin and Ethereum

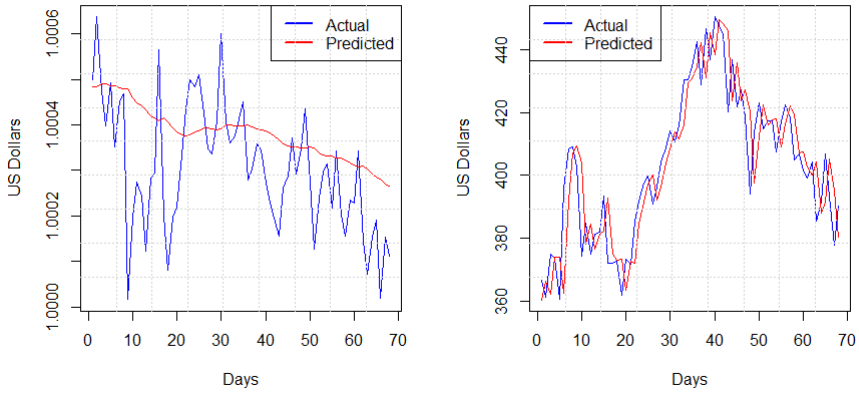


Figure 23: Tether and BNB

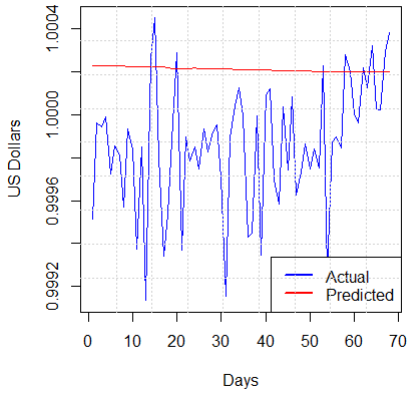


Figure 24: USD coin

Table 12: Training and testing errors for exponential smoothing model

		RMSE	MAE	MAPE
Bitcoin	Training	1422.4850	895.9467	2.8075
	Testing	1352.4870	982.0187	2.3827
Ethereum	Training	111.0558	65.3136	3.7993
	Testing	100.5336	77.8656	2.6693
Tether	Training	0.0031	0.0012	0.1150
	Testing	0.0001	0.0001	0.0112
Binance Coin	Training	18.4490	9.6203	4.1855
	Testing	11.4959	9.0460	2.2602
USD coin	Training	0.0028	0.0010	0.1040
	Testing	0.0005	0.0004	0.0378

From the above table it can be seen fitting is fairly good for all the cryptocurrencies as the mean absolute percent error is less than 5% for the training data. As in case of bitcoin MAPE on testing data is 2.8% indicating that predictions are off by only 2.8%. Similarly in case of ethereum, tether, BNB coin, USD coin the predictions are off by 2.67%, 0.01%, 2.26%, 0.04% respectively.

8 Comparison of ARIMA model and Exponential smoothing models

8.1 Bitcoin

AIC value for ARIMA model is 12668.39 which is significantly less than AIC value of ES model. So clearly ARIMA model gives a better fit to the data, also training errors are lower in case of ARIMA model again indicating a better fit. For testing data RMSE and MAE is also lower for ARIMA model, however mean absolute percentage error is almost same at 2.38%.

8.2 Ethereum

AIC value for ARIMA model is 8937.248 which is significantly less than AIC value of ES model. So clearly ARIMA model gives a better fit to the data, also training errors are lower in case of ARIMA model again indicating a better fit. For testing data RMSE and MAE and MAPE is almost same for the 2 models.

8.3 Tether

AIC value for ARIMA model is -6448.378 which is significantly less than AIC value of ES model. In case of testing data ARIMA model performs significantly better than ES model.

8.4 Binance Coin

AIC value for ARIMA model is 6319.38 which is significantly less than AIC value of ES model. In case of training data and testing data model errors were found to be more or less similar.

8.5 USD coin

AIC value for ARIMA model is -6618.12 which is significantly less than AIC value of ES model. In case of training data and testing data model errors were found to be more or less similar.

9 Volatility modelling using daily returns data

9.1 Data

In this study, closing price of 5 cryptocurrencies were selected because it reflected all the activities of them for each trading day. For this part of study the data has been modified to get the daily returns data. For bitcoin, ethereum, tether, binance coin, usd coin daily returns data is available for 3220, 2389, 2547, 1671, 1231 trading days respectively.
If P_t and P_{t-1} represents the current day and previous day price then the return series calculated as

$$R_t = \frac{P_t - P_{t-1}}{P_t}$$

9.1.1 Correlation between daily returns

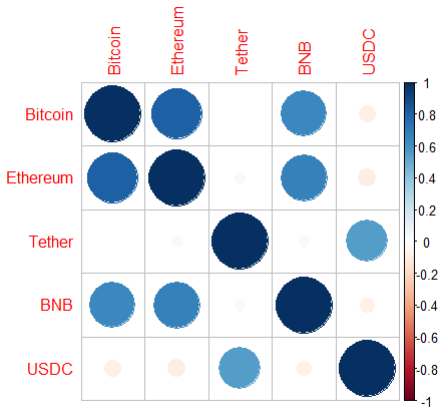


Figure 25: Correlation between daily returns for last 1231 trading days

A clear pattern can be seen that, bitcoin, ethereum, binance coin is highly correlated among each other, on the other hand tether and usd coin are correlated among themselves. If we pick members one from each they seem to be uncorrelated.

9.1.2 Boxplot of daily returns

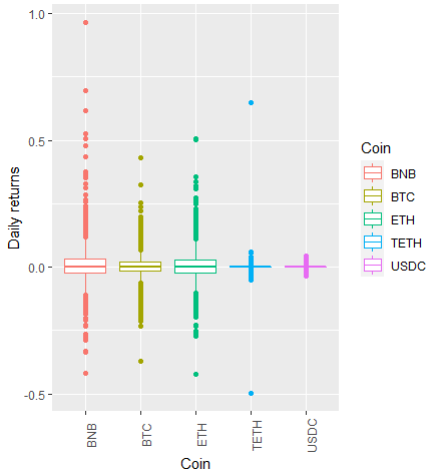


Figure 26: Boxplot of daily returns

It can be seen that the daily returns cenetred around 0, and have a lot of variability in them. Heavier tails can be seen in the first 3 cryptocurrencies. Thus, we can have an idea that normal distribution is probably not appropriate to model them.

9.2 Test of Normality

The returns data of cryptocurrencies are checked for normality. The Jaraque-Bera (JB) test statistics for normality test was employed for this purpose. The null hypothesis of this test is sample distribution is normally distributed.

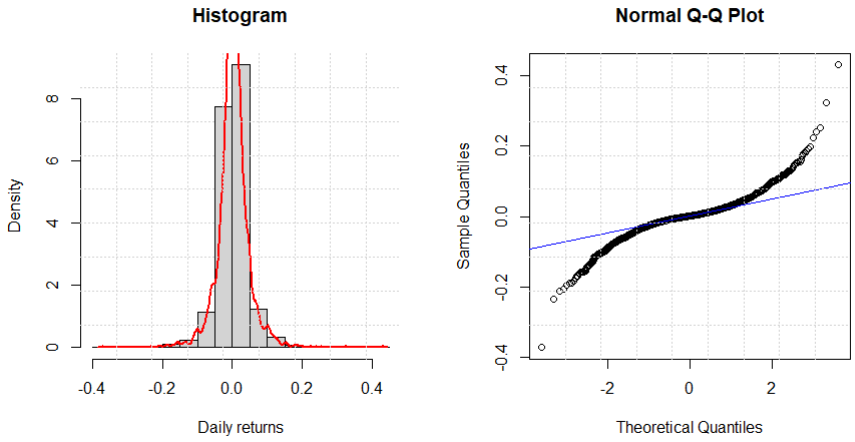


Figure 27: Histogram and normal q-q plot for return series of BTC

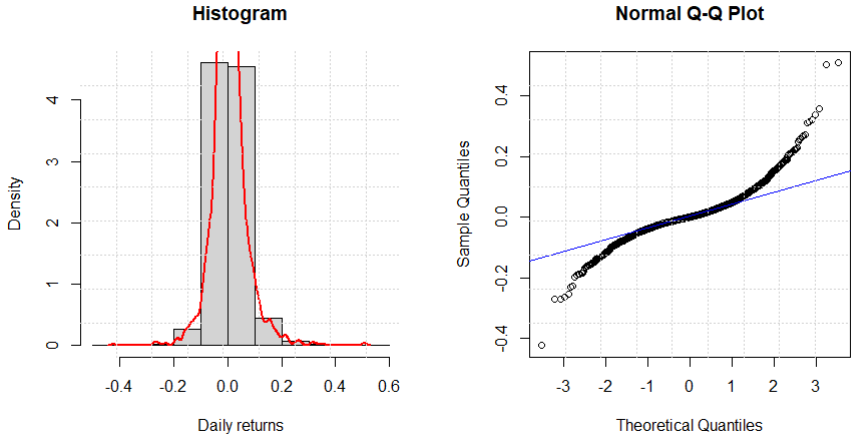


Figure 28: Histogram and normal q-q plot for return series of ETH

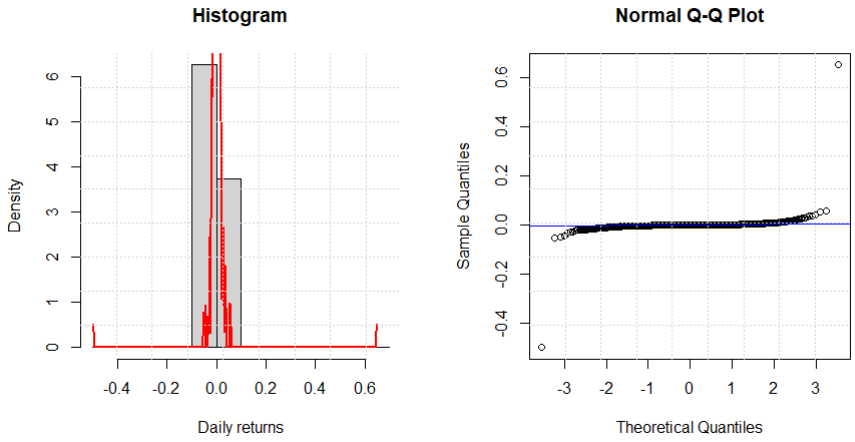


Figure 29: Histogram and normal q-q plot for return series of TETH

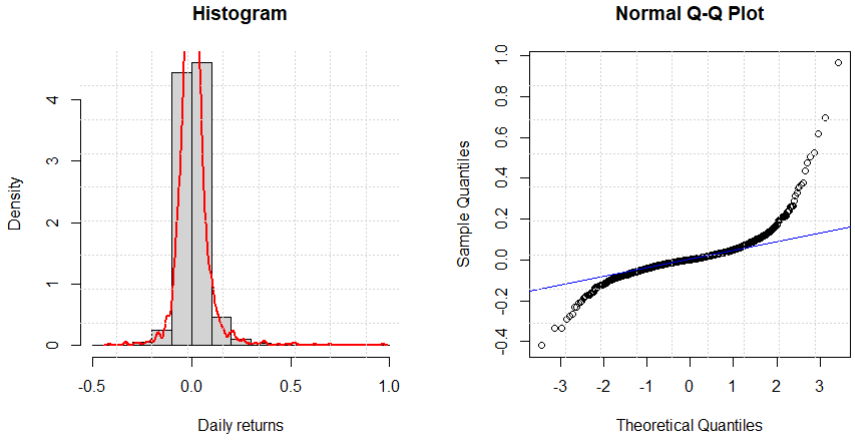


Figure 30: Histogram and normal q-q plot for return series of BNB

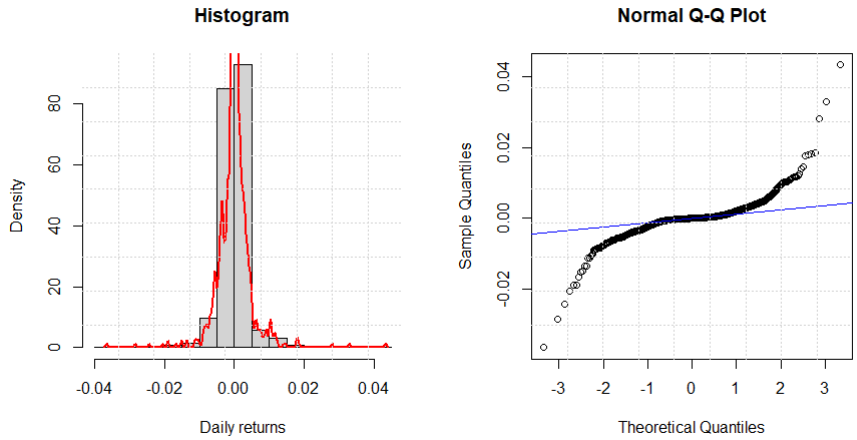


Figure 31: Histogram and normal q-q plot for return series of USDC

From all the above plots it can be seen that presence of heavy tails and positive skewness is common is all of the daily returns.Hence can not be normally distributed. To confirm the above we use Jaraque-Bera test.

Table 13: Jaraque Bera test

	p-value
Bitcoin	$< 10^{-16}$
Ethereum	$< 10^{-16}$
Tether	$< 10^{-16}$
Binance Coin	$< 10^{-16}$
USD coin	$< 10^{-16}$

9.3 Testing for auto regressive conditional heteroscedasticity (ARCH) Effects

To apply GARCH models to the Bitcoin returns series, the presence of stationarity and ARCH effects in return series are tested. The Ljung-Box and Lagrange multiplier (LM) test are used to test for the presence of ARCH effects in the data.
The ARCH test is a vital tool for examining the time dynamics of the second moments (i.e. conditional variance). The presence of a significant excess kurtosis is not indicative of time-varying volatility, but the reverse is true: a

significant ARCH effect identifies time-varying conditional volatility,and, as a result, the presence of a fat-tailed distribution (i.e. excess kurtosis > 0). Null hypothesis of the test is absence of ARCH effects.

Table 14: p-values of ARCH LM test and ADF test

	LM test	ADF test
Bitcoin	< 0.01	< 0.01
Ethereum	< 0.01	< 0.01
Tether	0.97	< 0.01
Binance Coin	< 0.01	< 0.01
USD coin	< 0.01	< 0.01

As we can see ARCH effects are present in all 4 cryptocurrencies except for tether. Stationarity of all 4 series is confirmed by ADF test.

9.4 Conditional variance equation

9.4.1 Standard Generalized AutoRegressive Conditional Heteroskedasticity (sGARCH) with ARMA(p, q)

sGarch(1,1) model is given by

$$x_t = \mu + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{j=1}^{q} \theta_j e_{t-j} + \sigma_t e_t$$

$$\sigma_t^2 = \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2$$

where, $\omega > 0, \alpha, \beta > 0, \alpha + \beta < 1$
e_t is a sequence of independent and identically distributed random variable.

9.4.2 Integrated Generalized AutoRegressive Conditional Heteroskedasticity (iGARCH) with ARMA(p, q)

igarch(1, 1) model is given by

$$x_t = \mu + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{j=1}^{q} \theta_j e_{t-j} + \sigma_t e_t$$

$$\sigma_t^2 = \omega + \beta_1 e_{t-1}^2 + \beta_2 \sigma_{t-1}^2$$

where, $\omega > 0, \alpha, \beta > 0, \beta_1 + \beta_2 = 1$
e_t is a sequence of independent and identically distributed random variable.

9.5 Distribution assumption of error e_t

From the test of normality in table 13 and figure 14, it is clear that the distribution of the returns are not normally distributed. There is the presence of excess kurtosis and heavy-tails in the distribution of the returns. To account for the excess kurtosis and fat tails that are present in the residuals of the returns series, the error term in the GARCH is modelled with Student-t distribution, Generalized Error Distribution (GED), and Normal Inverse Gaussian (NIG) types of distributions. These distributions are appropriate to capture the excess kurtosis and the skewness in the residuals return series.

9.5.1 Students's t-distribution

The Student's t-distribution or the t-distribution is a sub-class of the continuous probability distributions. It is used when the sample size is small and the population standard deviation is unknown. The probability density function (pdf) of the t-distribution is defined as

$$f_t(x;v) = \frac{\Gamma(\frac{v+1}{2})}{\sqrt{v\pi}\Gamma(v/2)}(1+\frac{x^2}{v})^{-\frac{v+1}{2}}, x \in (\infty, \infty)$$

where, v is the number of degrees of freedom and $\Gamma(.)$ is the gamma function.

9.5.2 Generalized error distribution (GED)

The standardized GED or the error distribution is a symmetrical unimodal sub-class of the exponential family with shape parameter v The pdf of the GED is defined as

$$f_{GED}(x;v) = k(v)exp(\frac{1}{2}|\frac{x}{\lambda(v)}|^v), x \in (-\infty, \infty)$$

where v determines the tail weight with larger value of v giving lesser tail weight, $k(v)$, and $\lambda(v)$ are constant and are defined as

$$k(v) = \frac{v}{\lambda(v)2^{1+\frac{1}{v}}\Gamma(1/v)}$$

$$\lambda(v) = (\frac{2^{-\frac{2}{v}}\Gamma(1/v)}{\Gamma(3/v)})^{1/2}$$

9.5.3 Normal Inverse Gaussian (NIG) distribution

The NIG distribution is a sub-class of the generalized hyperbolic distribution. The pdf of an NIG distribution is given by

$$f_{NIG}(x;\alpha,\kappa,\mu,\delta) = \frac{\alpha\delta}{\pi} exp(\delta\sqrt{\alpha^2-\beta^2} + \beta(x-\mu))K_1[aq(x)]$$

where, $q(x) = ((x-\mu)^2+\delta^2)^{1/2}, \alpha > 0$ determines the shape, $0 \leq |\kappa| \leq \alpha$ determines skewness, $\delta > 0$ is the scaling and $\mu \in R$ is the location parameter, for $x \in R$, $K_1(x)$ is the modified Bessel function of second kind with index 1.
The NIG distribution has a heavier tail than the normal distribution and can take different kinds of shapes.The shape of an NIG density is described using a four dimensional parameters $(\alpha, \kappa, \mu, \delta)$.The NIG distribution is appropriate for capturing data sets with extremal observations, skewness, and fat tails or semi-heavy tails.

9.6 Results and discussions

9.6.1 Bitcoin

Below table shows the results of the maximum likelihood estimate (MLE) of sGARCH(1, 1), iGARCH(1, 1) models for Bitcoin daily returns using Normal Inverse Gaussian distribution. From the table, the log-likelihood value (6483.3) is maximum for sGARCH(1, 1) model. The values of the two information criterions (AIC = -4.02, BIC = -3.99) of sGARCH(1, 1) are minimum as compared to iGARCH(1, 1). The visual QQ plot is consistent with the AIC, BIC, and log likelihood values of the sGARCH(1, 1)-NIG. P-value of goodness of fit test is 0.33, indicating a good fit. These results indicate that sGARCH(1, 1)-NIG model is the optimal model to describe the volatility of the return series of Bitcoin.

Table 15: ML estimates of sGARCH(1,1) and iGARCH(1,1) on NIG distributed errors

Model	μ	ω	α	β	$\frac{1}{\nu}$	κ	AIC	BIC	LL
sGarch(1,1)	0.0011	0.0000	0.1229	0.8761	0.4008	-0.0448	-4.0201	-3.9993	6483.2870
	0.0007	0.0000	0.0154	0.0159	0.0399	0.0350			
iGarch(1,1)	0.0011	0.0000	0.1227	0.8773	0.4052	-0.0481	-4.0180	-3.9991	6478.9080
	0.0005	0.0000	0.0149		0.0354	0.0354			

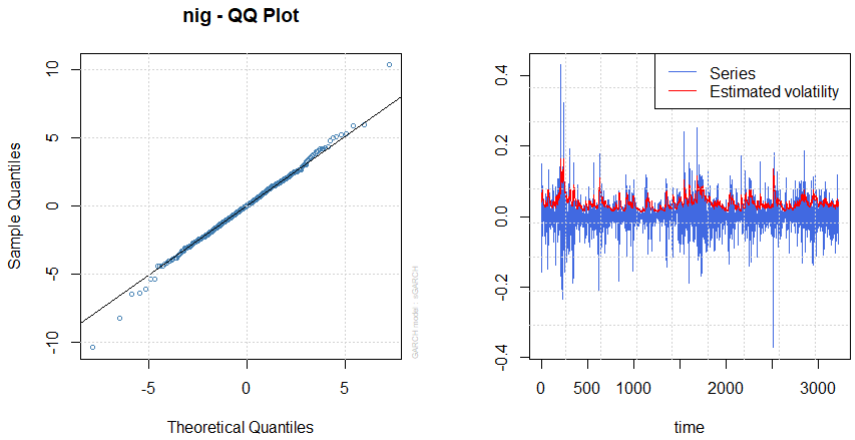


Figure 32: Q-Q plot of residuals of sGARCH(1,1)-NIG and fitted volatlity for BTC returns series

9.6.2 Ethereum

Below table shows the results of the maximum likelihood estimate (MLE) of sGARCH(1, 1), iGARCH(1, 1) models for Ethereum daily returns using Normal Inverse Gaussian distribution. From the table, the log-likelihood value (3772.65) is maximum for iGARCH(1, 1) model. The values of the two information criterions (AIC = -3.14, BIC = -3.10) of iGARCH(1, 1) are almost same as compared to sGARCH(1, 1). The visual QQ plot is consistent with the AIC, BIC, and log likelihood values of the iGARCH(1, 1)-NIG. P-value of goodness of fit test is 0.60, indicating a good fit. These results indicate that sGARCH(1, 1)-NIG model is the optimal model to describe the volatility of the return series of Ethereum.

Table 16: ML estimates of sGARCH(1,1) and iGARCH(1,1) on NIG distributed errors

Model	μ	ω	α	β	$\frac{1}{\nu}$	κ	AIC	BIC	LL
sGarch(1,1)	0.0032	0.0002	0.1659	0.8032	0.5261	0.1556	-3.1444	-3.1057	3771.9620
	0.0012	0.0000	0.0286	0.0295	0.0675	0.0418			
iGarch(1,1)	0.0031	0.0002	0.2102	0.7898	0.4793	0.1486	-3.1458	-3.1095	3772.6490
	0.0001	0.0000	0.0309		0.0542	0.0323			

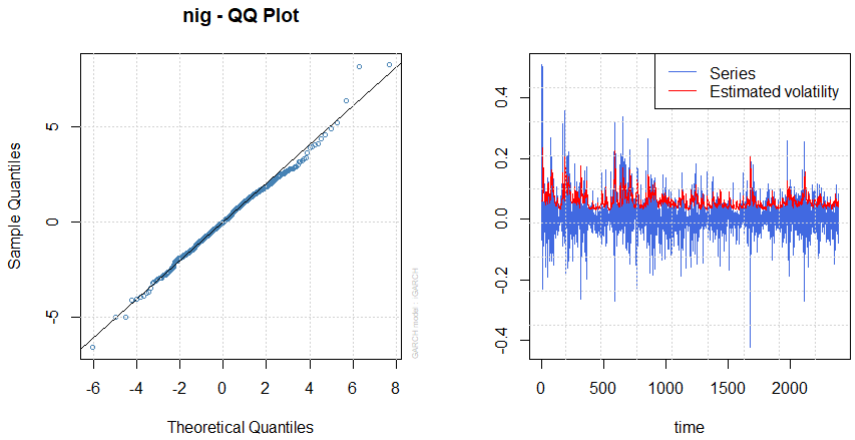


Figure 33: Q-Q plot of residuals of iGARCH(1,1)-NIG and fitted volatlity for ETH returns series

9.6.3 Binance Coin

Below table shows the results of the maximum likelihood estimate (MLE) of sGARCH(1, 1), iGARCH(1, 1) models for BNB daily returns using Normal Inverse Gaussian distribution. From the table, the log-likelihood value (-5114.06) is maximum for sGARCH(1, 1) model. The values of the two information criterions (AIC = 6.13, BIC = 6.17) of iGARCH(1, 1) are almost same as compared to sGARCH(1, 1). The visual QQ plot is consistent with the AIC, BIC, and log likelihood values of the iGARCH(1, 1)-NIG. P-value of goodness of fit test is 0.23, indicating a good fit. These results indicate that iGARCH(1, 1)-NIG model is the optimal model to describe the volatility of the return series of Binance coin.

Table 17: ML estimates of sGARCH(1,1) and iGARCH(1,1) on NIG distributed errors

Model	μ	ω	α	β	$\frac{1}{\nu}$	κ	AIC	BIC	LL
sGarch(1,1)	0.1425	1.5286	0.1652	0.8102	0.6646	0.0058	6.1365	6.1787	-5114.0660
	0.1150	0.5097	0.0375	0.0377	0.1049	0.0504			
iGarch(1,1)	0.1299	1.2919	0.1879	0.8121	0.6117	-0.0095	6.1364	6.1753	-5114.9640
	0.1126	0.4449	0.0366		0.0889	0.0519			

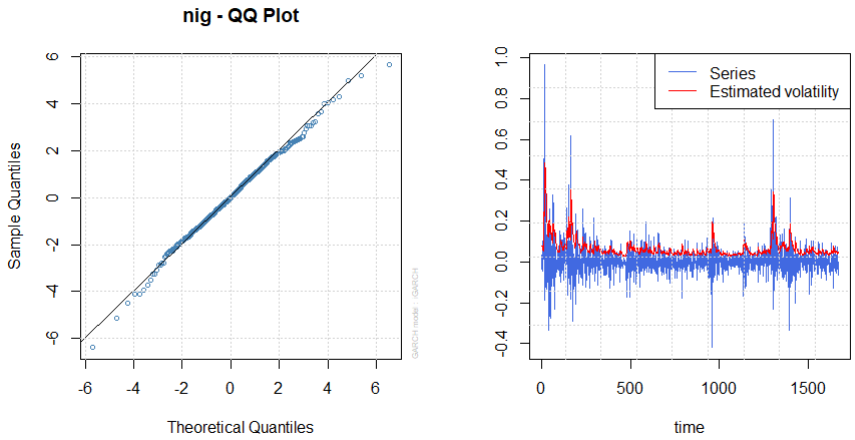


Figure 34: Q-Q plot of residuals of iGARCH(1,1)-NIG and fitted volatlity for BNB returns series

9.6.4 USD Coin

Below table shows the results of the maximum likelihood estimate (MLE) of sGARCH(1, 1), iGARCH(1, 1) models for USDC daily returns using studenst t distribution. From the table, the log-likelihood value (656.50) is maximum for iGARCH(1, 1) model. The values of the two information criterions (AIC = -1.05, BIC = -1.03) of iGARCH(1, 1) are almost same as compared to sGARCH(1, 1). The visual QQ plot is consistent with the AIC, BIC, and log likelihood values of the iGARCH(1, 1)-t. P-value of goodness of fit test is 0.40, indicating a good fit. These results indicate that iGARCH(1, 1)-t model is the optimal model to describe the volatility of the return series of USD coin.

Table 18: ML estimates of sGARCH(1,1) and iGARCH(1,1) on t distributed errors

Model	μ	ω	α	β	$\frac{1}{\nu}$	AIC	BIC	LL
sGarch(1,1)	-0.0002	0.0000	0.1727	0.8263	4.3093	-1.0548	-1.0258	656.2560
	0.0003	0.0000	0.0273	0.0248	0.3711			
iGarch(1,1)	-0.0002	0.0000	0.1737	0.8263	4.2936	-1.0569	-1.0319	656.5033

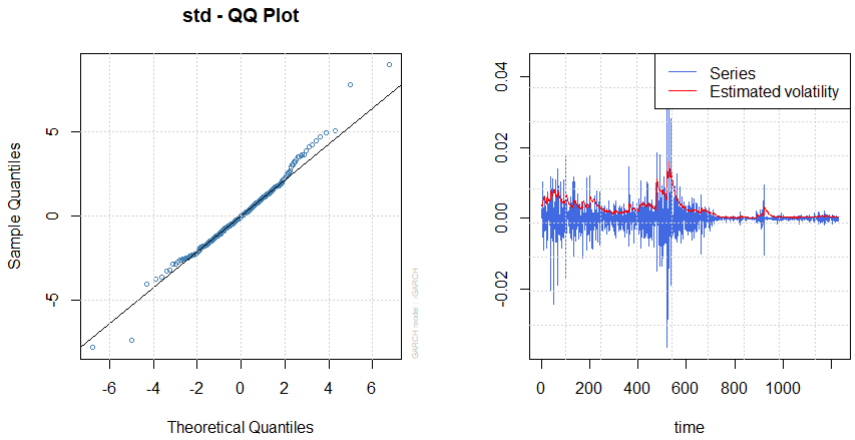


Figure 35: Q-Q plot of residuals of iGARCH(1,1)-NIG and fitted volatlity for BNB returns series

10 Conclusion

In this study 5 cryptocurrencies were analysed used different approaches. In the first half of the study, daily closing prices were modelled using auto regressive moving average models and exponential smoothing models. It was observed ARIMA models outperformed the ES models in all the cases. Training and testing errors using both the methods were reasonably low. In case of testing data predictions were off by about 4% at max.
In the second half of the study, volatility of daily returns data of 4 cryptocurrencies were analysed using different versions on GARCH models(sGarch, iGacrh). To account for the skewness and fat tails in the returns time series the Student-t, Generalized Error Distribution (GED), and Normal Inverse Gaussian (NIG) distribution were used to capture the tail distribution in the GARCH models.The NIG distribution performed better (3 out of 4 currencies) in capturing the fat tail and skewness in the return series distribution.

11 Acknowledgement

I would like to wholeheartedly thank my supervisors as well as all my professors at the Department of Statistics, University of Calcutta, for their constant support.
I would like to express my sincere gratitude and gratefulness to *Dr. Gaurangadeb Chattopadhyay* and *Dr. Bhaswati Ganguly*, my supervising professors, for their valuable guidance with all necessary materials for my project. Their suggestions and instructions have served as a major contributor towards completion of this project.

12 References

- The Analysis of Time Series an introduction with R : Chris Chatfield, Haipeng Xing.
- Time Series Analysis and its applications : Robert H. Shumway, David S. Stoffer.

13 Appendix-R codes

```
library(ggplot2)
library(dplyr)
library(plotly)
library(crypto2)
library(forecast)
library(tseries)
library(rugarch)
btc = read.csv("Bitcoin data.csv")
eth = read.csv("Ethereum data.csv")
teth = read.csv("Tether data.csv")
bnb = read.csv("BNB data.csv")
usdc = read.csv("USD Coin data.csv")
eth_df_test = read.csv("Ethereum data test.csv")
eth_acc = eth_df_test$close
d = read.csv("Bitcoin data_test.csv")
btc_acc = d$close
test_tether = read.csv("Tether data_test.csv")
teth_acc = test_tether$close
d = read.csv("BNB data_test.csv")
```

```
bnb_acc = d$close
d = read.csv("USDC data_test.csv")
usdc_acc = d$close
btc2 = btc$close[2491:3221]
btc2date = btc$timestamp[2491:3221]
btc2yrs = data.frame(btc2date, btc2)
head(btc2date)
tail(btc2date)
btc2 %>% ggtsdisplay()
btc2 %>% diff() %>% ggtsdisplay()
adf.test(btc2)
btc2 %>% diff() %>% adf.test()
eth2 = eth$close[1660:2390]
eth2date = eth$timestamp[1660:2390]
eth2yrs = data.frame(eth2date, eth2)
eth2 %>% ggtsdisplay()
eth2 %>% diff() %>% ggtsdisplay()
adf.test(eth2)
eth2 %>% diff() %>% adf.test()
teth2 = teth$close[1818:2548]
teth2date = teth$timestamp[1818:2548]
teth2yrs = data.frame(teth2date, teth2)
teth2 %>% ggtsdisplay()
eth2 %>% diff() %>% ggtsdisplay()
adf.test(eth2)
eth2 %>% diff() %>% adf.test()
bnb2 = bnb$close[942:1672]
bnb2date = bnb$timestamp[942:1672]
bnb2yrs = data.frame(bnb2date, bnb2)
bnb2 %>% ggtsdisplay()
bnb2 %>% diff() %>% ggtsdisplay()
adf.test(bnb2)
bnb2 %>% diff() %>% adf.test()
usdc2 %>% ggtsdisplay()
usdc2 %>% diff() %>% ggtsdisplay()
adf.test(usdc2)
usdc2 %>% diff() %>% adf.test()
usdc2 = usdc$close[502:1232]
usdc2date = usdc$timestamp[502:1232]
usdc2yrs = data.frame(usdc2date, usdc2)
```

```
crypto_arima = function(coin, max.pq){
  G = matrix(NA, nrow = max.pq, ncol = max.pq)
  for(i in 1:max.pq){
    for(j in 1:max.pq){
      eth_arima = arima(coin, order = c(i,1,j))
      G[i,j] = AIC(eth_arima)
    }
  }
  return(list(pos = which(G == min(G), arr.ind = TRUE),
              min.AIC = min(G)))
}

crypto_arima2 = function(coin, max.pq){
  G = matrix(NA, nrow = max.pq, ncol = max.pq)
  for(i in 1:max.pq){
    for(j in 1:max.pq){
      eth_arima = arima(coin, order = c(i,0,j))
      G[i,j] = eth_arima$aic
    }
  }
  return(list(pos = which(G == min(G), arr.ind = TRUE),
              min.AIC = min(G)))
}

crypto_arima(btc2,5)
btc1 = arima(btc2, order = c(3,1,2))
checkresiduals(btc1)
crypto_arima(eth2,5)
eth1 = arima(btc2, order = c(5,1,5))
checkresiduals(eth1)
crypto_arima(teth2, 5)
teth1 = arima(teth2, order = c(4,1,5))
checkresiduals(teth1)
crypto_arima(bnb2,5)
bnb1 = arima(bnb2, order = c(4,1,3))
checkresiduals(bnb1)
crypto_arima(usdc2,5)
usdc1 = arima(bnb2, order = c(5,1,5))
checkresiduals(usdc1)
rollin_predictions = function(coin, coin_acc){
```

```
  y_hat = c()
  for(i in 1:length(coin_acc)){
    model = auto.arima(coin)
    f = data.frame(forecast(model, 1))
    y_hat[i] = f$Point.Forecast
    coin[length(coin)+1] = coin_acc[i]
  }
  return(y_hat)
}
btc_yhat = rollin_predictions(btc2, btc_acc)
eth_yhat = rollin_predictions(eth2, eth_acc)
teth_yhat = rollin_predictions(teth2, teth_acc)
bnb_yhat = rollin_predictions(bnb2, bnb_acc)
usdc_yhat = rollin_predictions(usdc2, usdc_acc)

plot(fitted(btc1), col = "red", xlab = "Days", ylab = "US Dollars", lwd = 2)
lines(btc2, col = "blue")
legend("topright", legend = c("Actual", "Fitted"),
 col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")
plot(btc_acc, type = "l", col = "blue", xlab = "Days", ylab = "US Dollars",
     lwd = 1)
lines(btc_yhat, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
 col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")

plot(fitted(eth1), col = "red", xlab = "Days", ylab = "US Dollars", lwd = 2)
lines(eth2, col = "blue")
legend("topright", legend = c("Actual", "Fitted"),
col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")
plot(eth_acc, type = "l", col = "blue", xlab = "Days",
 ylab = "US Dollars",
     lwd = 1)
lines(eth_yhat, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
 col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")
```

```
plot(fitted(teth1), col = "red", xlab = "Days", ylab = "US Dollars", lwd = 2)
lines(teth2, col = "blue")
legend("topright", legend = c("Actual", "Fitted"),
col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")
plot(teth_acc, type = "l", col = "blue", xlab = "Days", ylab = "US Dollars",
     lwd = 1)
lines(teth_yhat, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
 col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")

plot(fitted(bnb1), col = "red", xlab = "Days", ylab = "US Dollars", lwd = 2)
lines(bnb2, col = "blue")
legend("topright", legend = c("Actual", "Fitted"),
 col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")
plot(bnb_acc, type = "l", col = "blue", xlab = "Days", ylab = "US Dollars",
     lwd = 1)
lines(bnb_yhat, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
 col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")

plot(fitted(usdc1), col = "red", xlab = "Days", ylab = "US Dollars", lwd = 2)
lines(usdc2, col = "blue")
legend("topright", legend = c("Actual", "Fitted"),
 col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")
plot(usdc_acc, type = "l", col = "blue", xlab = "Days", ylab = "US Dollars",
     lwd = 1)
lines(usdc_yhat, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
 col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")

model_acc = function(y_hat, acc){
  N = length(y_hat)
  RMSE = sqrt(sum((y_hat-acc)^2)/N)
  MAE = (sum(abs(y_hat - acc)))/N
```

```
  MAPE = ((sum(abs((acc-y_hat)/acc)))/N)*100
  return(list(
    RMSE = RMSE,
    MAE = MAE,
    MAPE = MAPE
  ))
}

model_acc(btc_yhat, btc_acc)
accuracy(btc1)
model_acc(eth_yhat, eth_acc)
accuracy(eth1)
model_acc(teth_yhat, teth_acc)
accuracy(teth1)
model_acc(bnb_yhat, bnb_acc)
accuracy(bnb1)
model_acc(usdc_yhat, usdc_acc)
accuracy(usdc1)
rollin_predictions_holt = function(coin, coin_acc){
  y_hat = c()
  for(i in 1:length(coin_acc)){
    model = holt(coin)
    f = data.frame(forecast(model, 1))
    y_hat[i] = f$Point.Forecast
    coin[length(coin)+1] = coin_acc[i]
  }
  return(y_hat)
}


rollin_predictions_ses = function(coin, coin_acc){
  y_hat = c()
  for(i in 1:length(coin_acc)){
    model = ses(coin)
    f = data.frame(forecast(model, 1))
    y_hat[i] = f$Point.Forecast
    coin[length(coin)+1] = coin_acc[i]
  }
  return(y_hat)
}
```

```
btc_holt = holt(btc2)
summary(btc_holt)
eth_holt = holt(eth2)
summary(eth_holt)
teth_holt = holt(teth2)
summary(teth_holt)
bnb_holt = holt(bnb2)
summary(bnb_holt)
usdc_ses = ses(usdc2)
summary(usdc_holt)
btc_pred_holt = rollin_predictions_holt(btc2, btc_acc)
eth_pred_holt = rollin_predictions_holt(eth2, eth_acc)
teth_pred_holt = rollin_predictions_holt(teth2, teth_acc)
bnb_pred_holt = rollin_predictions_holt(bnb2, bnb_acc)
usdc_pred_ses = rollin_predictions_ses(usdc2, usdc_acc)

plot(btc_acc, type = "l", col = "blue", xlab = "Days", ylab = "US Dollars")
lines(btc_pred_holt, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")

plot(eth_acc, type = "l", col = "blue",  xlab = "Days", ylab = "US Dollars")
lines(eth_pred_holt, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")

plot(teth_acc, type = "l", col = "blue",  xlab = "Days", ylab = "US Dollars")
lines(teth_pred_holt, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")

plot(bnb_acc, type = "l", col = "blue",  xlab = "Days", ylab = "US Dollars")
lines(bnb_pred_holt, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")
```

```
plot(usdc_acc, type = "l", col = "blue" , xlab = "Days", ylab = "US Dollars")
lines(usdc_pred_ses, col = "red")
legend("bottomright", legend = c("Actual", "Predicted"),
col = c("blue", "red"), lty = 1, lwd = 2)
grid(nx = 9, ny = 9, col = "lightgray", lty = "dotted")

daily_returns = function(coin_){
  coin_today_cp = coin_$close[-c(1)]
  coin_yesterday_cp = coin_$close[-c(length(coin_$close))]
  coin_daily_returns = (coin_today_cp - coin_yesterday_cp)/(coin_yesterday_cp)
  return(coin_daily_returns)
}

q = 1231
Bitcoin = tail(btc_daily_returns,q)
tethereum = tail(teth_daily_returns,q)
Ttether = tail(tteth_daily_returns,q)
BNB = tail(bnb_daily_returns,q)
USDC = tail(usdc_daily_returns,q)

dr_df = data.frame(Dr = c(btc_daily_returns,
teth_daily_returns, tteth_daily_returns,
                          bnb_daily_returns, usdc_daily_returns),
                   Coin = rep(c("BTC", "teth", "Tteth", "BNB", "USDC"), c(
                     length(btc_daily_returns),
                     length(teth_daily_returns),
                     length(tteth_daily_returns),
                     length(bnb_daily_returns),
                     length(usdc_daily_returns)
                   )))
ggplot(dr_df, aes(x=Coin, y=Dr, color=Coin)) +
  geom_boxplot() +
theme(axis.text.x = element_text(angle = 90))+ylab("Daily returns")

m = data.frame(Bitcoin, tethereum, Ttether, BNB, USDC)
mm = cor(m)
corrplot::corrplot(mm)
adf.test(btc_daily_returns)
archtest(btc_daily_returns)
```

```
jarque.bera.test(btc_daily_returns)

adf.test(eth_daily_returns)
archtest(eth_daily_returns)
jarque.bera.test(teth_daily_returns)

adf.test(teth_daily_returns)
archtest(teth_daily_returns)
jarque.bera.test(tteth_daily_returns)

adf.test(bnb_daily_returns)
archtest(bnb_daily_returns)
jarque.bera.test(bnb_daily_returns)

adf.test(usdc_daily_returns)
archtest(usdc_daily_returns)
jarque.bera.test(bnb_daily_returns)

hist(btc_daily_returns, freq = F,
main = "Histogram", xlab = "Daily returns")
lines(density(btc_daily_returns), col = "red", lwd = 2)
grid(9,9)
qqnorm(btc_daily_returns)
qqline(btc_daily_returns, col = "blue")
grid(9,9)

hist(eth_daily_returns, freq = F,
main = "Histogram", xlab = "Daily returns")
lines(density(eth_daily_returns), col = "red", lwd = 2)
grid(9,9)
qqnorm(eth_daily_returns)
qqline(eth_daily_returns, col = "blue")
grid(9,9)

hist(teth_daily_returns, freq = F,
 main = "Histogram", xlab = "Daily returns")
lines(density(teth_daily_returns), col = "red", lwd = 2)
grid(9,9)
qqnorm(teth_daily_returns)
qqline(teth_daily_returns, col = "blue")
```

```
grid(9,9)

hist(bnb_daily_returns, freq = F,
main = "Histogram", xlab = "Daily returns")
lines(density(bnb_daily_returns), col = "red", lwd = 2)
grid(9,9)
qqnorm(bnb_daily_returns)
qqline(bnb_daily_returns, col = "blue")
grid(9,9)

hist(usdc_daily_returns, freq = F,
main = "Histogram", xlab = "Daily returns")
lines(density(usdc_daily_returns), col = "red", lwd = 2)
grid(9,9)
qqnorm(usdc_daily_returns)
qqline(usdc_daily_returns, col = "blue")
grid(9,9)



btc_daily_returns = daily_returns(btc)
btc_dr_df = data.frame(time = btc$timestamp[-1],
dr = btc_daily_returns)

s_t = ugarchspec(mean.model = list(armaOrder = c(2,3)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'std')

s_n = ugarchspec(mean.model = list(armaOrder = c(2,3)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'nig')

s_g = ugarchspec(mean.model = list(armaOrder = c(2,3)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'ged')

i_t = ugarchspec(mean.model = list(armaOrder = c(2,3)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'std')
```

```
i_n = ugarchspec(mean.model = list(armaOrder = c(2,3)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'nig')

i_g = ugarchspec(mean.model = list(armaOrder = c(2,3)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'ged')



m1 = ugarchfit(data = btc_dr_df$dr, spec = s_t)
m2 = ugarchfit(data = btc_dr_df$dr, spec = s_n)

vol_m2 = m2@fit$sigma
plot(btc_daily_returns, type = "l",
col = "royalblue", xlab = "time", ylab = "")
lines(vol_m2, col = "red")
grid(9,9)
legend("topright", legend = c("Series", "Estimated volatility"),
col = c("royalblue", "red"), lty = 1)
plot(m2)
m3 = ugarchfit(data = btc_dr_df$dr, spec = s_g)

m4 = ugarchfit(data = btc_dr_df$dr, spec = i_t)
m5 = ugarchfit(data = btc_dr_df$dr, spec = i_n)
m6 = ugarchfit(data = btc_dr_df$dr, spec = i_g)

eth = read.csv("Ethereum data.csv")
d = read.csv("Ethereum data test.csv")
eth_daily_returns = daily_returns(eth)
test_eth_dr = daily_returns(d)

s_t = ugarchspec(mean.model = list(armaOrder = c(5,5)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'std')

s_n = ugarchspec(mean.model = list(armaOrder = c(5,5)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'nig')
```

```
s_g = ugarchspec(mean.model = list(armaOrder = c(5,5)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'ged')

i_t = ugarchspec(mean.model = list(armaOrder = c(5,5)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'std')

i_n = ugarchspec(mean.model = list(armaOrder = c(5,5)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'nig')

i_g = ugarchspec(mean.model = list(armaOrder = c(5,5)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'ged')



m7 = ugarchfit(data = eth_daily_returns, spec = s_t)
m8 = ugarchfit(data = eth_daily_returns, spec = s_n)
m9 = ugarchfit(data = eth_daily_returns, spec = s_g)

m10 = ugarchfit(data = eth_daily_returns, spec = i_t)
m11= ugarchfit(data = eth_daily_returns, spec = i_n)

vol_m11 = m11@fit$sigma
plot(eth_daily_returns, type = "l", col = "royalblue", xlab = "time", ylab = "")
lines(vol_m11, col = "red")
grid(9,9)
legend("topright", legend = c("Series", "Estimated volatility"),
col = c("royalblue", "red"), lty = 1)
plot(m11)
m12 = ugarchfit(data = eth_daily_returns, spec = i_g)

bnb = read.csv("BNB data.csv")
bnb_daily_returns_train = daily_returns(bnb)

bnb_daily_returns = daily_returns(bnb)
```

```
s_t = ugarchspec(mean.model = list(armaOrder = c(3,4)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'std')

s_n = ugarchspec(mean.model = list(armaOrder = c(3,4)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'nig')

s_g = ugarchspec(mean.model = list(armaOrder = c(3,4)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'ged')

i_t = ugarchspec(mean.model = list(armaOrder = c(3,4)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'std')

i_n = ugarchspec(mean.model = list(armaOrder = c(3,4)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'nig')

i_g = ugarchspec(mean.model = list(armaOrder = c(3,4)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'ged')



m13 = ugarchfit(data = bnb_daily_returns, spec = s_t)
m14 = ugarchfit(data = bnb_daily_returns, spec = s_n)
m15 = ugarchfit(data = bnb_daily_returns, spec = s_g)

m16 = ugarchfit(data = bnb_daily_returns, spec = i_t)
m17 = ugarchfit(data = bnb_daily_returns, spec = i_n)
vol_m17 = m17@fit$sigma
plot(bnb_daily_returns, type = "l", col = "royalblue", xlab = "time", ylab = "")
lines(vol_m17, col = "red")
grid(9,9)
legend("topright", legend = c("Series", "Estimated volatility"),
col = c("royalblue", "red"), lty = 1)
plot(m17)
m18 = ugarchfit(data = bnb_daily_returns, spec = i_g)
```

```
d = read.csv("USD Coin data.csv")
usdc_daily_returns = daily_returns(d)

d = read.csv("BNB data_test.csv")
bnb_daily_returns_test = daily_returns(d)
bnb_daily_returns = daily_returns(bnb)


s_t = ugarchspec(mean.model = list(armaOrder = c(1,1)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'std')

s_n = ugarchspec(mean.model = list(armaOrder = c(1,1)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'nig')

s_g = ugarchspec(mean.model = list(armaOrder = c(1,1)),
                 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                 distribution.model = 'ged')

i_t = ugarchspec(mean.model = list(armaOrder = c(1,1)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'std')

i_n = ugarchspec(mean.model = list(armaOrder = c(1,1)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'nig')

i_g = ugarchspec(mean.model = list(armaOrder = c(1,1)),
                 variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                 distribution.model = 'ged')

m19 = ugarchfit(data = usdc_daily_returns, spec = s_t)
m20 = ugarchfit(data = usdc_daily_returns, spec = s_n)
m21 = ugarchfit(data = usdc_daily_returns, spec = s_g)

m22 = ugarchfit(data = usdc_daily_returns, spec = i_t)
vol_m22 = m22@fit$sigma
plot(usdc_daily_returns, type = "l", col = "royalblue", xlab = "time", ylab = "")
```

```
lines(vol_m22, col = "red")
grid(9,9)
legend("topright", legend = c("Series", "Estimated volatility"),
col = c("royalblue", "red"), lty = 1)
plot(m22)
m23 = ugarchfit(data = usdc_daily_returns, spec = i_n)
m24 = ugarchfit(data = usdc_daily_returns, spec = i_g)
```