RMIT UNIVERSITY

Assignment 2:

Time Series Analysis

Student :	Phuong Pham. ID: 4078692
Tutorial :	Friday 05.30 PM. Subject Coordinator: Dr. Haydar Demirhan
Campus:	Melbourne City Campus
Due Date :	Wednesday, 30th April 2025, 11:59 PM

I. INTRODUCTION

Since its creation in 2009, Bitcoin has evolved from an experimental digital currency to such a globally recognized asset, flashing its strong technical signal throughout the years thanks to the increasing volatility and speculative investment patterns. Interpreting the Bitcoin market index from August 2011 to the beginning of the year 2025, the dataset reflects the monthly historical performance of this cryptocurrency, offering an opportunity to identify its dynamic behaviour for future predictions that will not only be beneficial to investors or traders but also various corporations as well as authorities, by leveraging the prediction model to maximizing profits, monitor stability and enforce regulations to optimize efficiency and security. This report presents a comprehensive analysis of the Bitcoin time series object that appropriately builds a robust forecasting model, emphasizing the ARIMA model. Following the descriptive exploration in the first phase, the time series is transformed and differenced to satisfy the stationarity requirement of fitting the ARIMA model. Then, model specification tools are systematically employed to propose potential sets of candidates to achieve the most accurate and precise predictability. After evaluating fitting methods and several goodness-of-fit criteria, the final model selections are expected to balance forecasting accuracy with model simplicity.

II. TIME SERIES ANALYSIS

1. Descriptive Analysis

i. Time Series Conversion

First and foremost, the Bitcoin dataset read into R is converted to a time series object using the function ts() of the package TSA. By looking at the original dataset, the indexes are collected monthly, starting from August 2011 to January 2025, therefore, the 'start', 'end', and 'frequency' parameters are input accordingly for the accurate conversion. In this dataset, the frequency is specified to be 12, which represents the monthly basis of the data points.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2011								10000.0	6025.0	4125.0	3987.5	5725.0
2012	6625.0	6237.5	6125.0	6250.0	6462.5	8262.5	11600.0	12700.0	15275.0	13750.0	15537.5	16550.0
2013	25575.0	41912.5	120187.5	174850.0	159887.5	111912.5	122850.0	160000.0	157812.5	254425.0	1399400.0	915000.0
2014	1003750.0	689862.5	568537.5	561062.5	784750.0	801387.5	727550.0	598800.0	488862.5	422487.5	471262.5	401250.0
2015	271125.0	316837.5	305300.0	295250.0	286137.5	328125.0	355412.5	287325.0	295250.0	387725.0	470887.5	538612.5
2016	460612.5	544037.5	518325.0	560462.5	664800.0	834937.5	779787.5	713437.5	760287.5	871262.5	927575.0	1207875.0
2017	1204987.5	1489012.5	1337887.5	1687762.5	2872512.5	3081862.5	3569762.5	5917825.0	5407612.5	8042762.5	12434587.5	17350000.0
2018	12686250.0	12893625.0	8660775.0	11552712.5	9365350.0	7982137.5	9656787.5	8771687.5	8247262.5	7879087.5	4963825.0	4616625.0
2019	4266387.5	4739200.0	5120100.0	6586250.0	10684000.0	13450125.0	12606675.0	11992687.5	10373062.5	11437587.5	9438337.5	8960450.0
2020	11659812.5	10659675.0	8026425.0	10784150.0	11807737.5	11416200.0	14195925.0	14568750.0	13472775.0	17270100.0	24625237.5	36240987.5
2021	41426725.0	56551200.0	73478225.0	72219087.5	46676762.5	43796537.5	51862662.5	58945112.5	54793037.5	76699300.0	71217650.0	57767962.5
2022	48114900.0	53973725.0	56896587.5	47049750.0	39703450.0	24906112.5	29153162.5	25073750.0	24281250.0	25622500.0	21462500.0	20660000.0
2023	28908750.0	28920000.0	35595000.0	36543750.0	34023750.0	38086250.0	36538750.0	32415000.0	33707500.0	43326250.0	47148750.0	52822500.0
2024	53197500.0	76451250.0	89106250.0	75790000.0	84346250.0	78345000.0	80765000.0	73708750.0	79127500.0	87788750.0	120551250.0	116726250.0
2025	128015000.0											

Figure 1: Time Series Object of Bitcoin Index

With the construction of the time variables, the object is plotted to provide an intuitive visualization to identify the crucial characteristics of the dataset itself. The time series plot helps analysts observe the trend and pattern, as well as any potential outliers that may not be visible in the summary statistics. The five key attributes that must be focused on are “trend”, “seasonality”, “changing variance”, “behavior”, and “change point”.

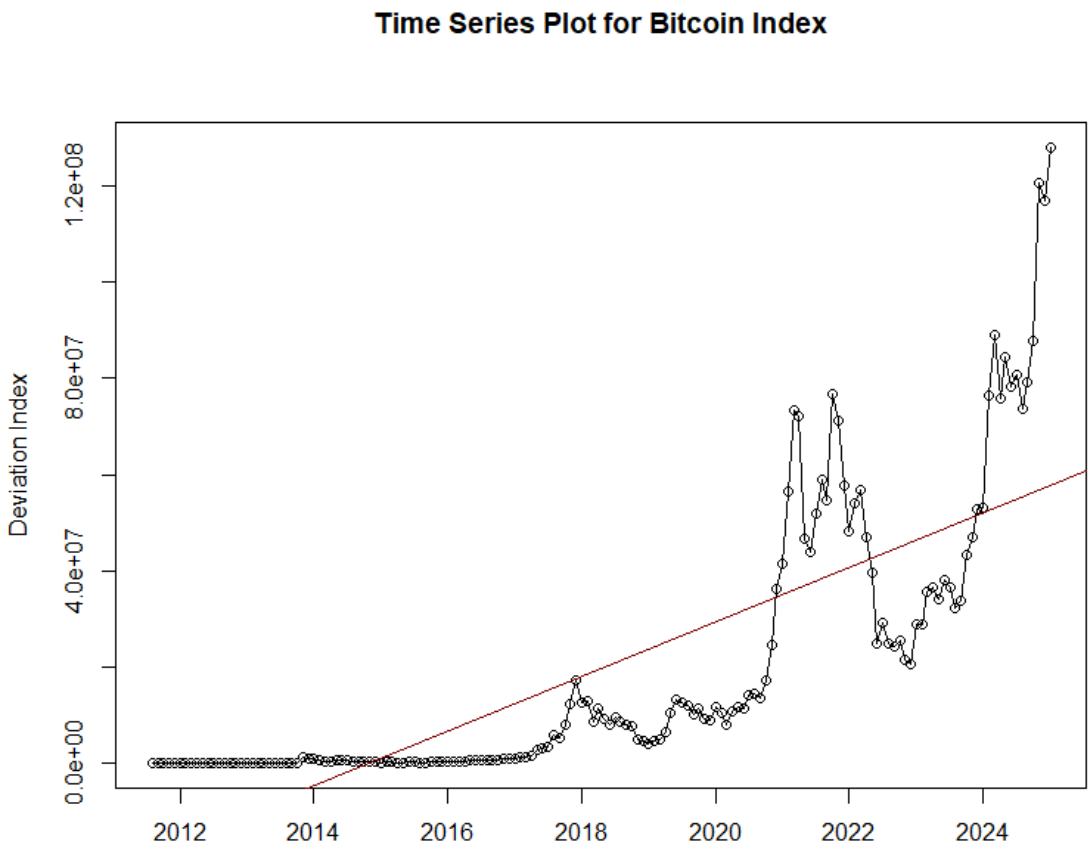

Figure 2: Time Series Plot of Bitcoin Index

Looking into the time series plot for Bitcoin index from 2011 to 2025, it is apparent that the data is following an uptrend pattern, specifically resulting in a relatively upward linear regression slope. The indexes were only seen to dramatically rise since the 2017-2018 period, with various fluctuations observed throughout the 7-year span, even though the flat movement of data points in previous years. Furthermore, there was neither seasonality nor consistent repeating patterns over time shown in the plot, besides visible sharp spikes and flat periods. In terms of heteroscedasticity, the plot displays a changing variance over time, with large spikes and fluctuations indicating the increasing volatility of Bitcoin values. As a result, a critical change point was evident after the rise in 2017, where the time series visualization started to exposively skyrocket, corresponding to a real-world major event called the Bitcoin boom driven by media factors leading to broader adoption and speculative investment.

ii. Summary Statistics

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
3988	470981	7930612	19781743	31599541	128015000

Figure 3: Summary statistics

The summary statistics in Figure 3 appear to display the overall traits of the dataset, giving a sense of how the Bitcoin market has evolved over time. It presents a skewed distribution due to a significantly higher mean compared to the median, at 3,988 and 7,930,612 respectively, suggesting a few substantially large prices that drive the average higher, while 50% of Bitcoin values are observed to be clustered between 407,981 and 31,599,541 of the interquartile range. Besides, the wide range from the minimum to the maximum values indicates significant growth in the market volatility.

iii. First Lag of Time Series

In relation, the succeeding observation implies the existence of strong autoregressive (AR) behaviour instead of moving average (MA). On top of that, the first lag of the Bitcoin time series would be calculated to compute the correlation between the current and previous values.

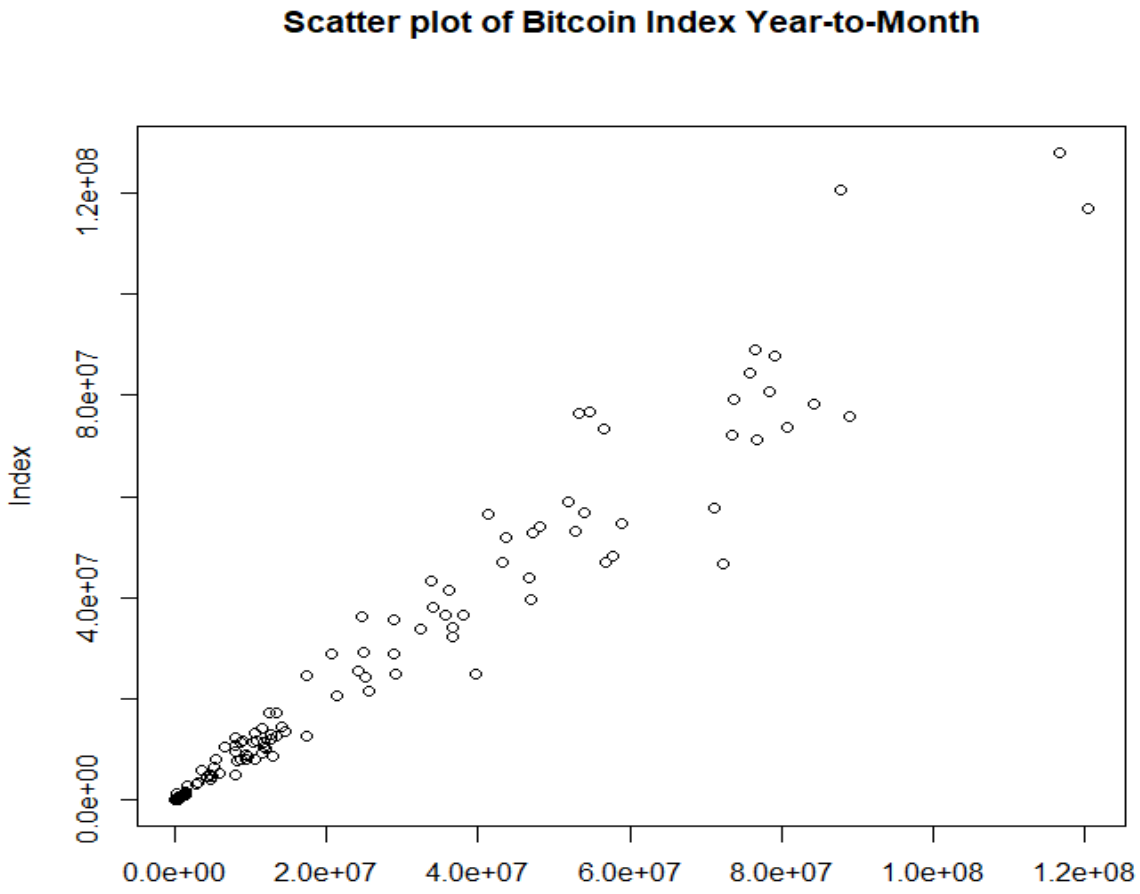

Figure 4: Scatter Plot of Correlation between Original and First Lag

The scatter plot of the correlation between the original data and its first lag confirms a highly positive correlation that indicates a strong persistence of the Bitcoin values over time. To be specific, the past prices are seen to be indicative of future indexes, which is typically expected in financial series where historical records have predictive power.

iv. Normality Check

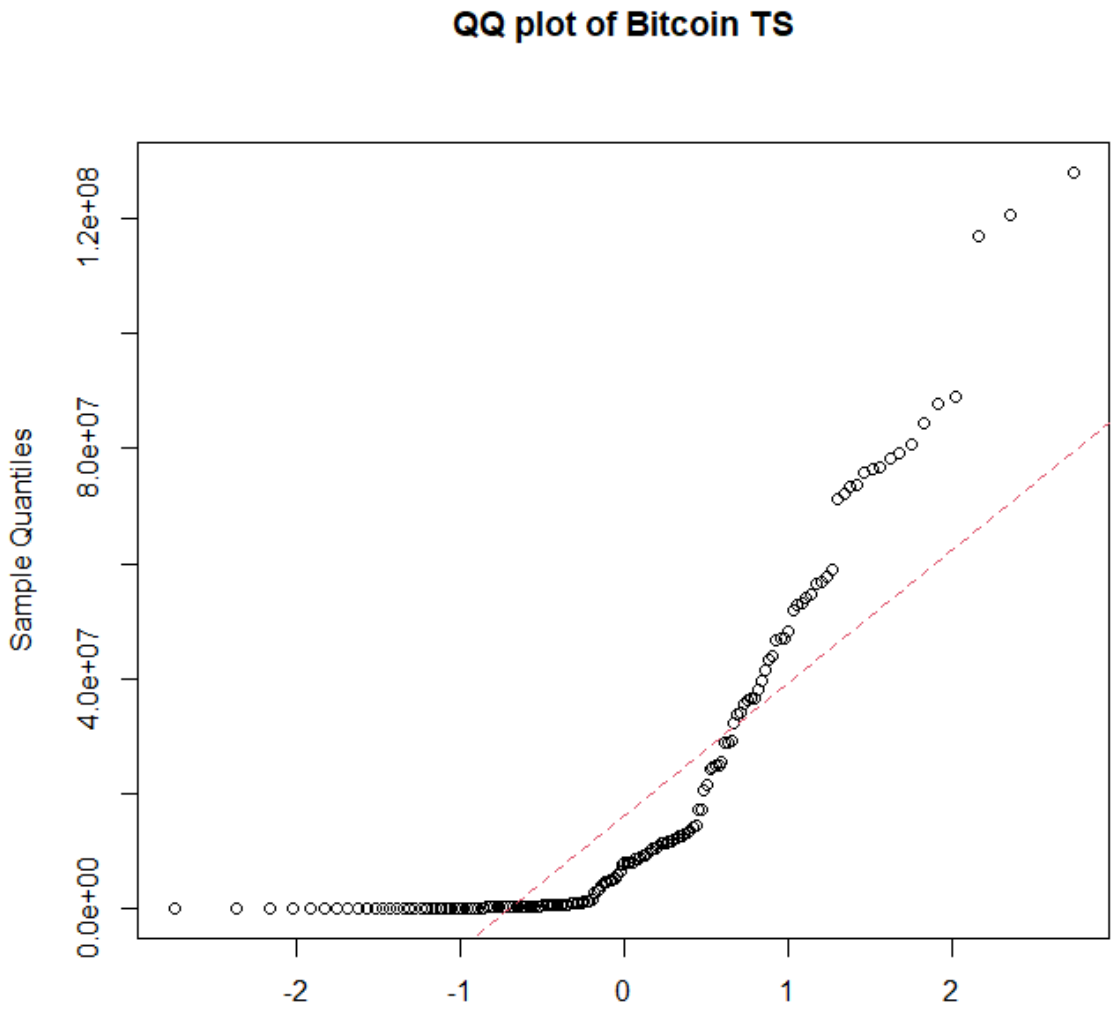

Figure 5: Q-Q Plot of Bitcoin Time Series

```
        Shapiro-Wilk normality test

data:  bitcoin_ts
W = 0.73793, p-value = 1.102e-15
```

Figure 6: Shapiro-Wilk Test of Bitcoin Time Series

In terms of the normal distribution analysis, Figure 5 presents the Q-Q plot of the time series object for the Bitcoin index, verifying the normality of the dataset. It is obvious that the assumption is not satisfied

since the majority of data points do not align close to the diagonal line, where their heads and tails seem to vary distantly, with multiple deviations as well as outliers captured within this Q-Q plot. In the meantime, the Shapiro-Wilk test logged in Figure 6 also supports the non-normality assumption of the dataset. In particular, since the p-value of 1.102e-15 is exceptionally smaller than the significance level α = 0.05, thus, the null hypothesis of normality is strongly rejected.

2. Model Specification

i. Stationarity Check for The Time Series

In the AutoRegressive Integrated Moving Average (ARIMA) modeling process, stationarity is one of the most important expectations in order to fit the model precisely. The stationarity of the object ensures the constancy of statistical properties such as the mean, variance, and autocorrelation. It is because the ARIMA model is built based on the stationarity assumption, capturing the combination of the remaining consistent behaviours from the historical pattern to conduct reliable forecasts. Autocorrelation Function (ACF), Partial Autocorrelation Function (PACF), and the Augmented Dickey-Fuller (ADF) Test are used to formally inspect the validation of stationarity.

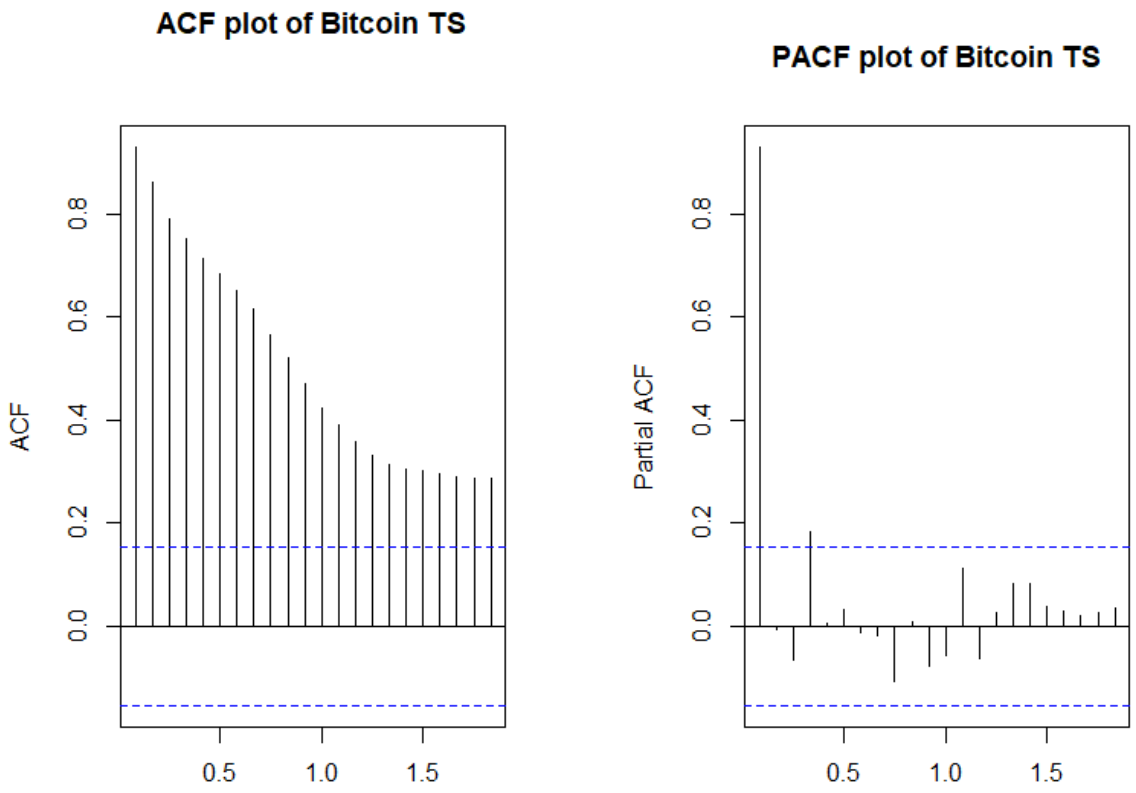

Figure 7: ACF & PACF Plot of Bitcoin Time Series

The bars of the ACF plot in Figure 7, measuring the correlation of the series with its several lags, show a strong autocorrelation over many lags due to the slow decay over time. In order to achieve stationarity, the ACF is expected to drop off quickly after the first few bars rather than slowly decreasing. Moreover, the adjacent PACF plot removes the influence of shorter lags, looking forward to observing the partial autocorrelation to remain close to zero after one or two significant lags. Although the PACF drops off after the first spike with no further significant autocorrelation, it is still consistent with the presence of non-stationarity due to its observed trend. For further clarification, the ADF stationarity test is conducted.

```
        Augmented Dickey-Fuller Test

data:  bitcoin_ts
Dickey-Fuller = -0.27056, Lag order = 5, p-value = 0.99
alternative hypothesis: stationary
```

Figure 8: Augmented Dickey-Fuller Test

In this hypothesis test, the null hypothesis H_0 states that the series is non-stationary, whereas the alternative hypothesis of confirming stationarity. The test statistic at around -0.27, along with the associated p-value, determines whether the null hypothesis is valid to be rejected, in other words, to confirm if the series has a potential unit root leading to non-stationarity. Given the large p-value at 0.99 that exceeds the significance level of 0.05, there is not enough evidence to reject the null hypothesis, indicating that the Bitcoin series is likely to exhibit some forms of non-stationarity. In consequence, no further stationarity test should be done while performing transformation and differencing in the next step is essential to achieve a stationary series for ARIMA model fitting.

ii. Box-Cox and Log Transformation

First of all, transformation is applied not only to enhance the stationarity but also to handle skewness to accomplish a normal distribution. For Box-Cox transformation, the function BOXCOX.ar() is used to execute the transformation specifically for autoregressive (ar), which is part of the package 'forecast'.

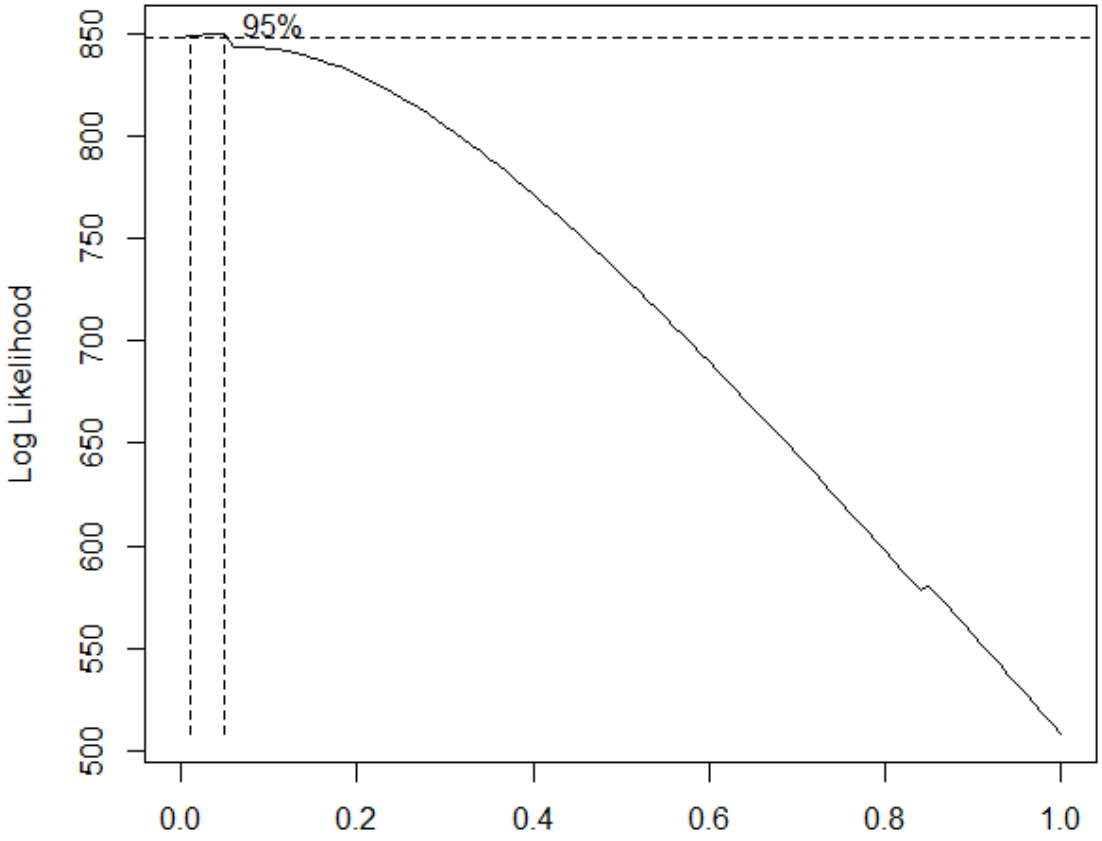

Figure 9: Box-Cox Transformation of Bitcoin Time Series

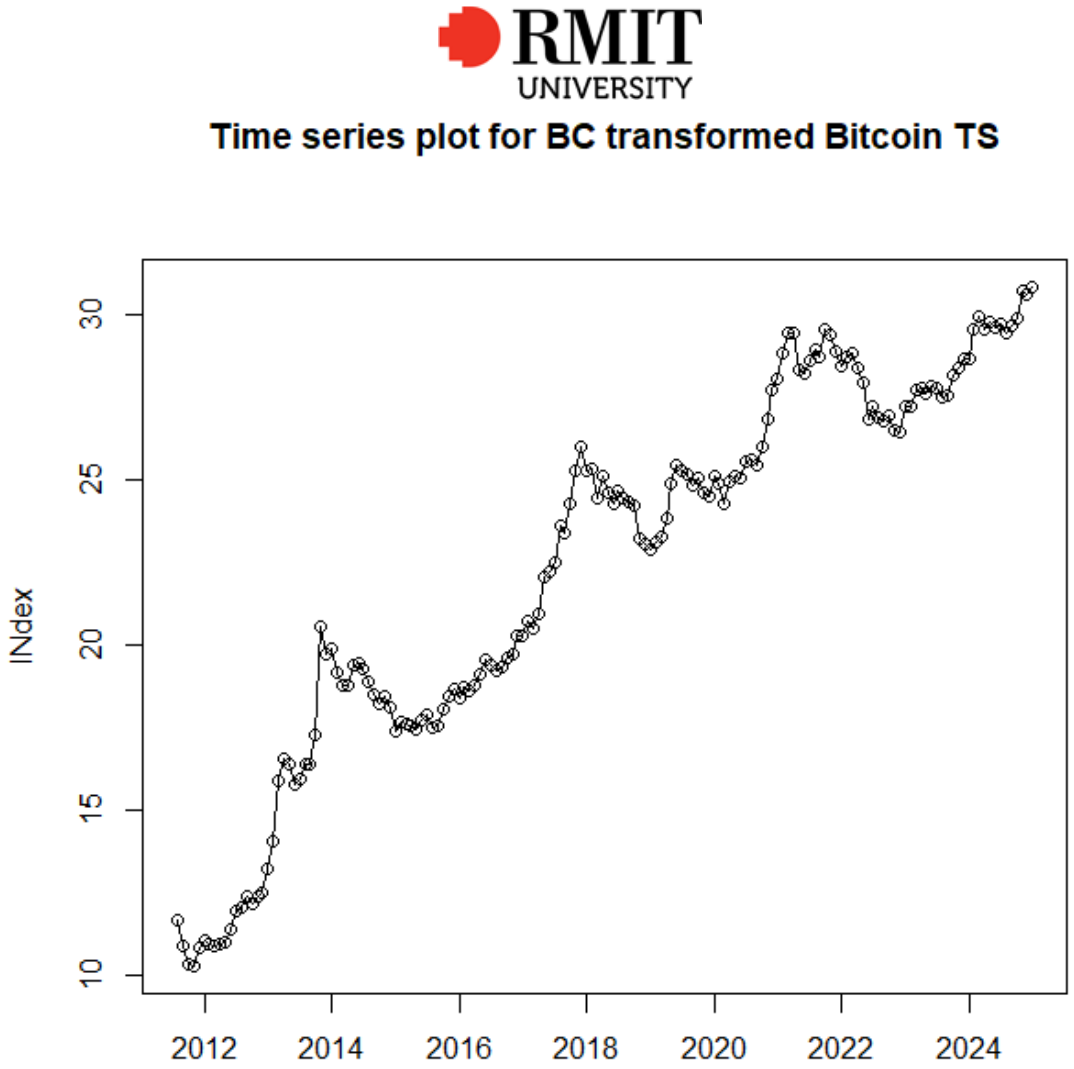

Figure 10: Time Series Plot for Box-Cox Transformed Bitcoin Series

During the process, the RStudio encounters a systematic error in solving computational singularity if the lambda is not specified. Thus, the sequence of candidate power transformation values from 0 to 1 with a step size of 0.01 is generated to define the lambda parameter. The limited range was first set as -2 and 2, but then restricted to [0,1] to avoid extreme distortion. The following step is to extract the optimal Box-Cox lambda, which is related to the highest log-likelihood values. This approach ensures the return of the transformation parameter that maximizes the likelihood. With the optimal lambda of 0.05, the time series object is manually transformed using the formula $BoxCox(y, \lambda) = \frac{y^{\lambda}-1}{\lambda}$ shown in Figure 9, with y is the time series, and lambda equals 0.05 computed from the range above. Figure 10 illustrates the time series plot after a smooth transformation.

Secondly, the log transformation is applied to verify whether the identified optimal Box-Cox lambda value is associated with the log-likelihood values. Using the function log() to perform the transformation. Figure 11 below shows the final version of log log-transformed time series object.

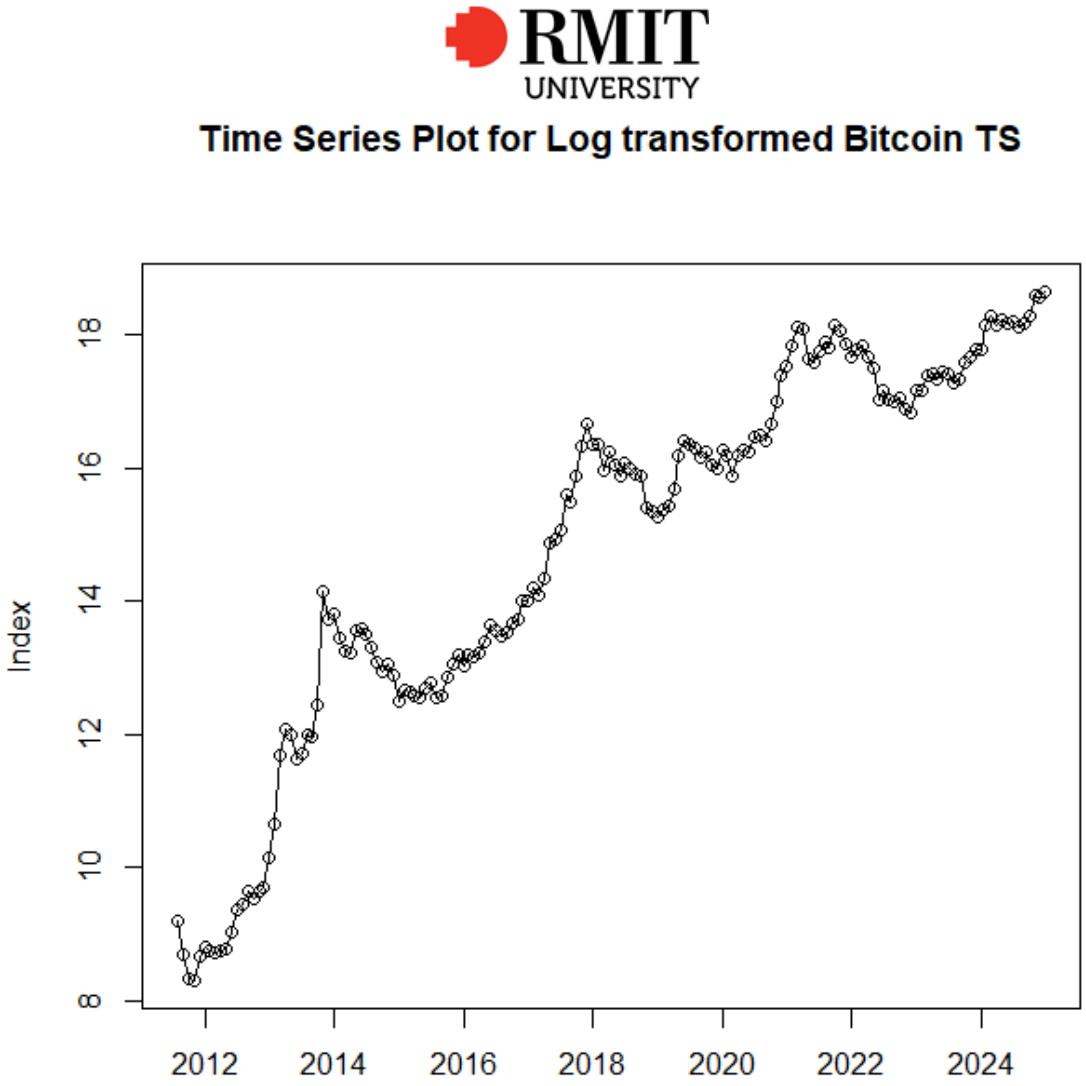

Figure 11: Time Series Plot for Log-transformed Bitcoin Series

In comparison to Figure 10 of the Box-Cox-transformed time series, the log-transformed one in Figure 11 encounters an almost complete similarity in terms of trend, pattern, and movement of data points. It is explicit that the lambda computed previously is accepted to be the ideal outcomes that is able to maximize the likelihood of fulfilling the ARIMA model fitting predictions. For that reason, the Box-Cox transformed version is selected for further analysis. Moreover, the Box-Cox transformation is a powerful tool for stabilizing variance and normalizing data, especially those with heteroscedasticity or skewness.

Before considering the differencing process, the object is checked again for the normality assumption using the Shapiro-Wilk test and Q-Q Plot.

```
        Shapiro-Wilk normality test

data:  bitcoin_bc
W = 0.93252, p-value = 6.461e-07
```

Figure 12: Shapiro-Wilk Test for Box-Cox-transformed Bitcoin Series

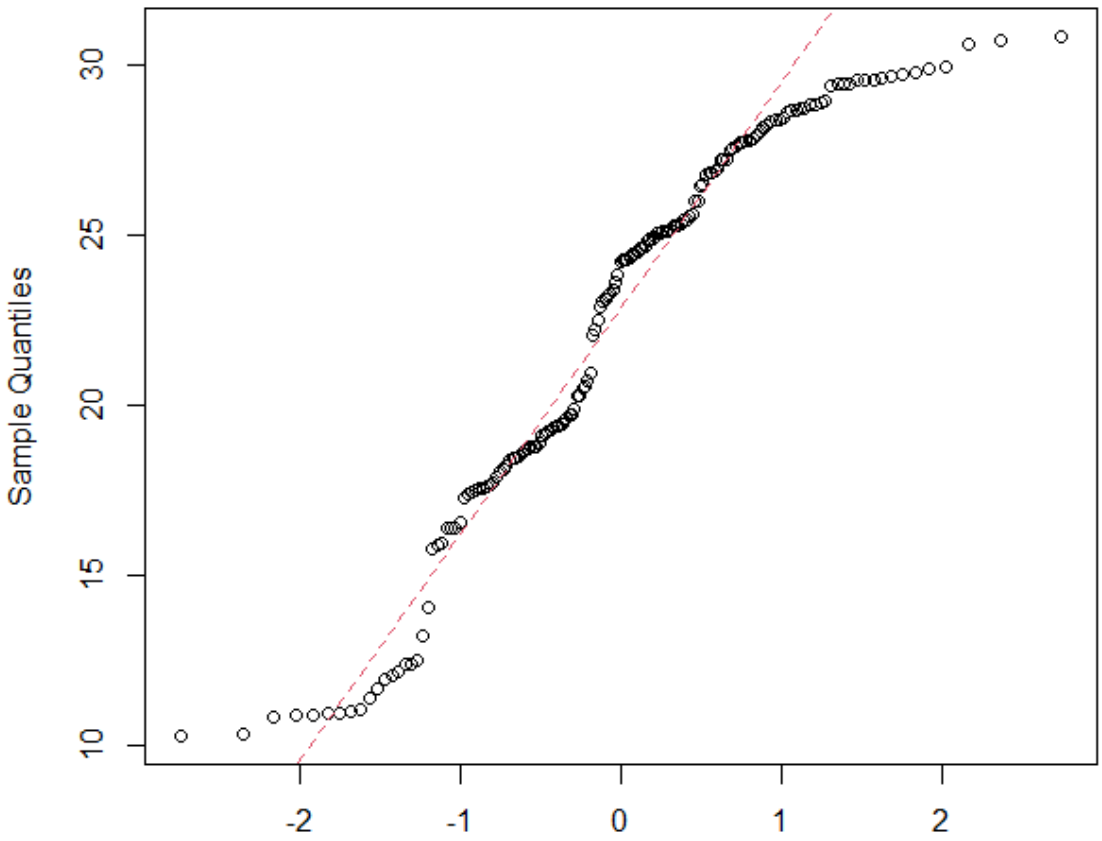

Figure 13: Q-Q Plot for Box-Cox-transformed Bitcoin Series

The Shapiro-Wilk test for normality in Figure 12 suggests that the transformed Bitcoin time series still do not follow a normal distribution due to a close-to-zero p-value of 6.6461e-07. The p-value smaller than the typical significance level $\alpha = 0.05$ results in a rejected null hypothesis and leads to a conclusion of non-normally distributed nature of the dataset. Although the test shows a negative outcome, the Q-Q plot, on the other hand, presents a visualization in which the majority of the dataset lies around the diagonal line. Despite the heavy tails, the transformed series is found to be closer to normal, as strict normality is hard to achieve, especially for financial datasets.

iii. Differencing

After applying the Box-Cox transformation to the time series of the Bitcoin index, it is necessary to continue differencing to ensure the trends or seasonality have been removed to satisfy the stationarity

assumption. The first differencing is performed using the diff() function to subtract the current value from the previous one.

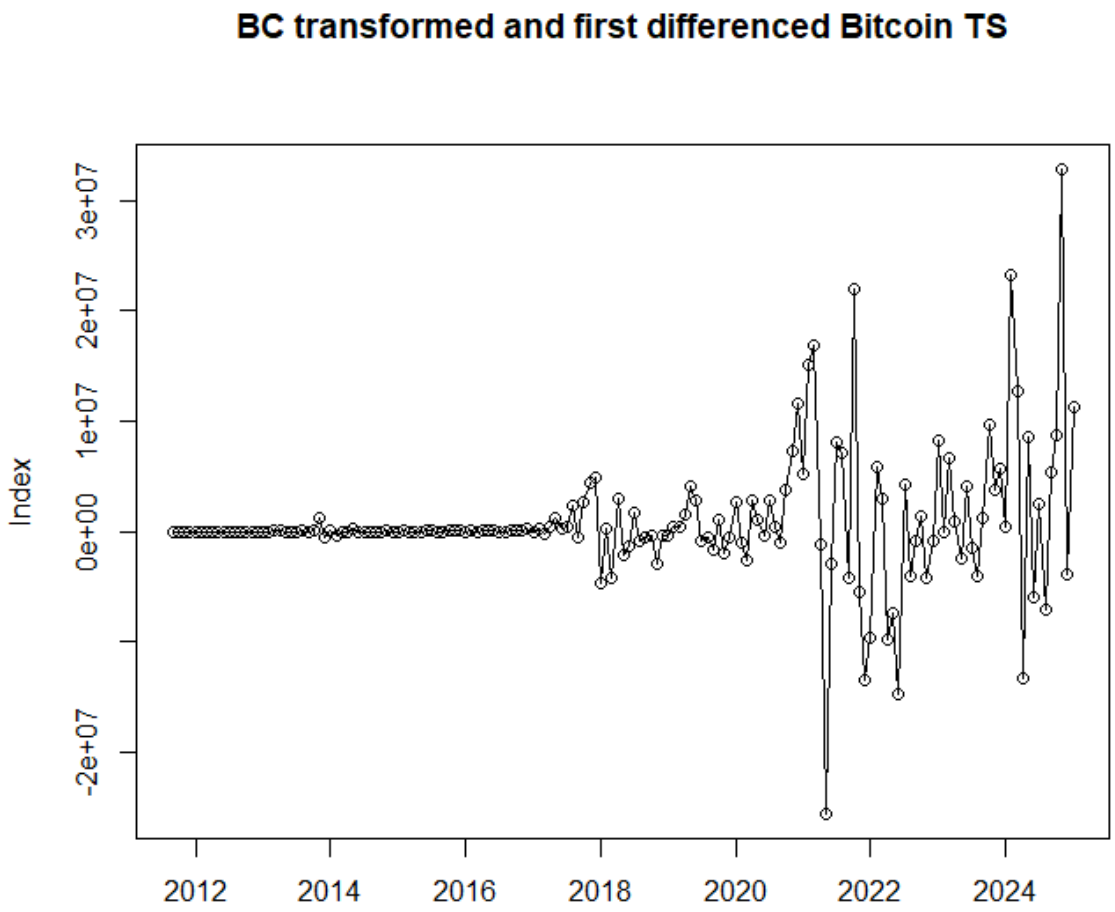

Figure 14: First Differencing of Box-Cox-transformed Bitcoin Series

According to the plot, the data oscillates around the value of 0 with no persistent trend. However, the volatility is still visible as there are significant spikes and fluctuations from 2017-2018 onwards. Thereafter, the first differencing suggests the lack of a long-term trend along with the zero-centered oscillation that satisfies the nature of stationarity. To strengthen the determination, the stationary check is run again with ACF and PACF (Figure 15), and ADF (Figure 16).

```
        Augmented Dickey-Fuller Test

data:  bitcoin_dif
Dickey-Fuller = -5.0788, Lag
order = 5, p-value = 0.01
alternative hypothesis: stationary
```

Figure 15: Augmented Dickey-Fuller Test for First Differenced Bitcoin Series

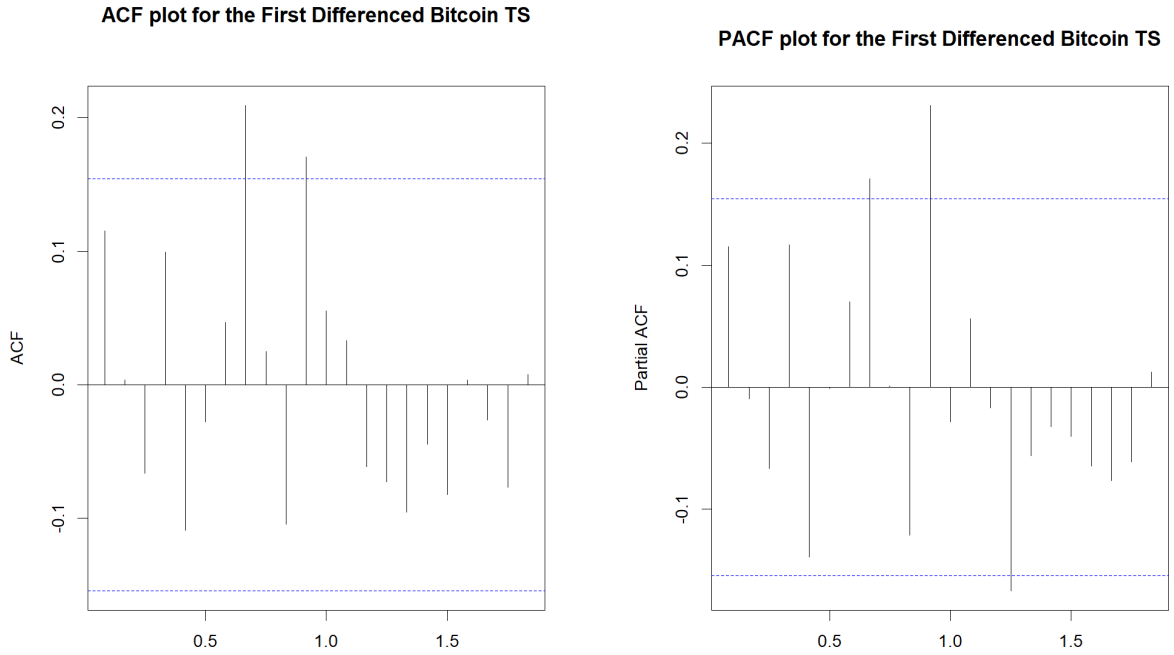

Figure 16: ACF & PACF Plots for First Differenced Bitcoin Series

The ACF and PACF plots show a good sign of stationarity, where the autocorrelation is not lingering for many lags, despite a short-term dependence at the early lags. Additionally, the ADF test's p-value is computed to be 0.01, which is smaller than the significance level of 0.05. This result indicates that the first differencing meets the requirement for stationarity, which also aligns with the interpretation of ACF and PACF plots. There are two new tests, namely the Phillips-Perron (PP) and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests, conducted to support the first three tests' findings.

```
        Phillips-Perron Unit Root Test

data:  bitcoin_dif
Dickey-Fuller Z(alpha) = -140.59, Truncation lag parameter = 4, p-value
= 0.01
alternative hypothesis: stationary
```

Figure 17: Phillips-Perron Unit Root Test for First Differenced Bitcoin Series

```
        KPSS Test for Level Stationarity

data:  bitcoin_dif
KPSS Level = 0.38253, Truncation lag parameter = 4, p-value = 0.08468
```

Figure 18: KPSS Test for First Differenced Bitcoin Series

With the PP Test's result in Figure 17, since the p-value of 0.01 is less than 0.05, it is valid to reject the null hypothesis then declare that the Bitcoin series after the first differencing is stationary following the alternative hypothesis. In contrast to the PP Test, the KPSS Test's null hypothesis implies stationarity of the

time series, while the alternative states the non-stationarity. Due to a greater than 0.05 p-value, at approximately 0.085, the null hypothesis is inevident to be rejected, thus, the series is likely stationary around the constant level. Both of the two advanced tests confirm the results of the previous analyses to conclude the legitimate stationarity. Considering that the first differencing of the series has achieved stationarity, the parameter estimation process will be executed for fitting the model without the second differencing to avoid over-differencing.

iv. Parameter Estimation

The ARIMA(p,d,q) model requires three parameters, including p for the number of autoregressive terms, d for the number of differencing terms, and q is the number of moving average terms, to forecast the time series data. Since the d is set to 1 thanks to the first differencing's stationarity, there are three approaches to identify the best set of parameters to fit the model, which is to evaluate the ACF and PACF plot, the Extended AutoCorrelation Function also known as EACF, and finally the Bayesian Information Criterion (BIC) table.

Looking at Figure 16 for the ACF and PACF plots, the number of significant lags in the ACF is assigned to the q parameter, while those for the PACF are the p parameter. The eighth and eleventh bars of the ACF are significant; therefore, q equals 2 in this evaluation. Using the same approach, the p parameter is 3 because there are three significant lags recorded in the PACF, during the eighth, eleventh, and fifteenth lags, resulting in the first possible ARIMA(3,1,2) model.

The EACF diagnostic method is manufactured using the function eacf() in R, providing an extension of the traditional ACF and PACF by showing the matrix of combined effects between the AR and MA processes.

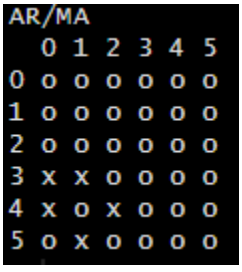

Figure 19: EACF matrix for First Differenced Bitcoin Series

In interpreting the EACF outcome, the initial step is to identify the zero in the top-left corner of the table, then take its neighbors as the significant variables to form the sets of p and q parameters. With

the top-left zero at (0,0), there are three neighbors, listed as (0,1), (1,0), and (1,1), that can shape another three ARIMA models: ARIMA(0,1,1), ARIMA(1,1,0), ARIMA(1,1,1).

Last but not least, the BIC table is a tool for time series model selection, using the context of ARIMA model to find ones that minimize the BIC values. Hence, the model chosen is expected to offer the best fit with the fewest parameters. In this process, the armasubsets() function is utilized to set the limits on the maximum possible values for AR/MA terms, in other words, the p and q parameters. With the number of AR and MA set to 14, the table is allowed to compute the maximum of 14 orders to test the combinations. The limits are chosen because of the positions of significant lags of ACF and PACF plots. Although the risk of overfitting is aware, the smaller limits might not capture the full complexity of the time series.

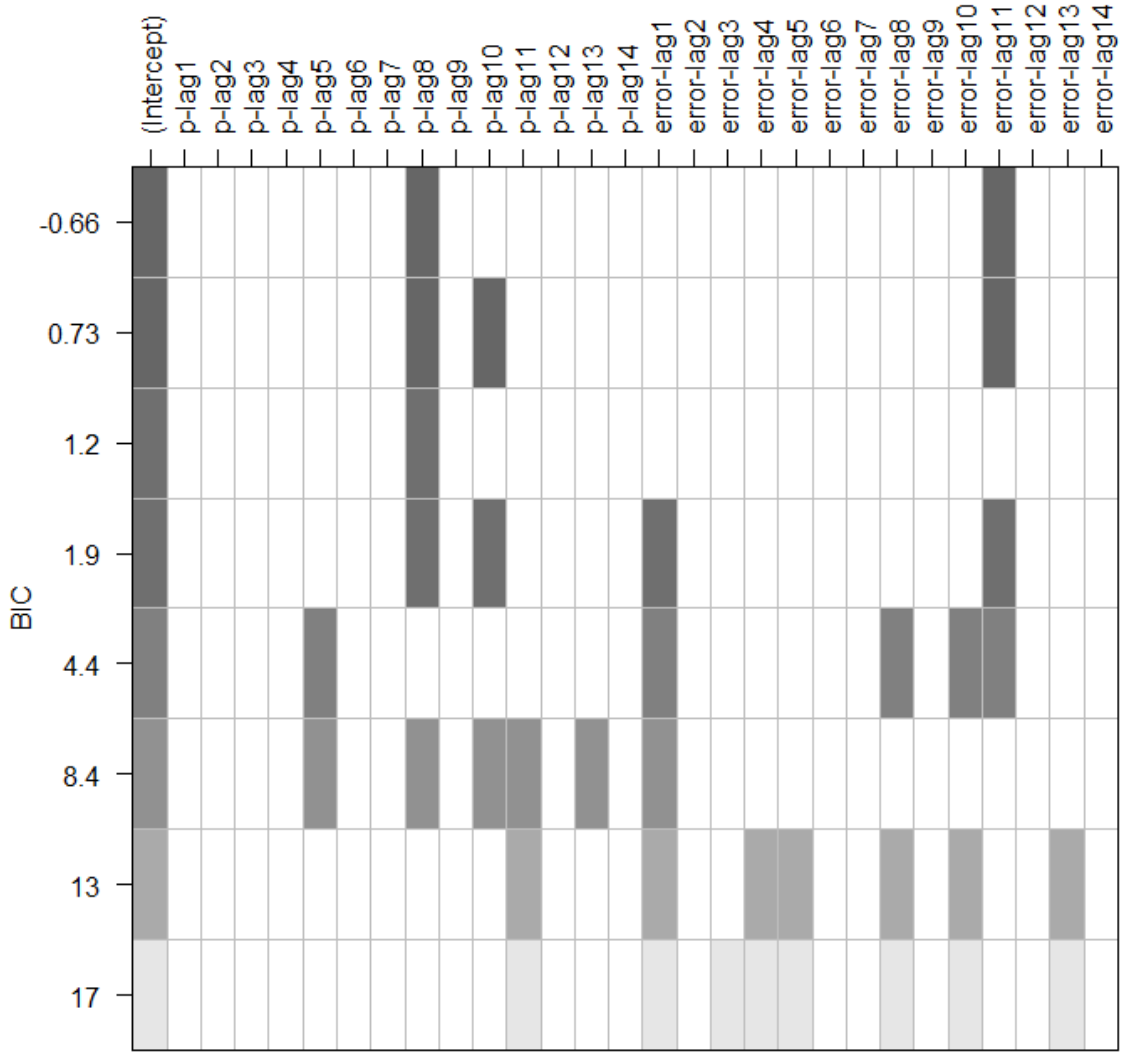

Figure 20: BIC Table for First Differenced Bitcoin Series

With ‘p-lag’ representing p and ‘error-lag’ representing q, the first 3 rows are considered to identify the best ARIMA model based on the combination of AR and MA orders. Figure 20 displays that ‘p-lag8’ is significant for all three orders. As a result, 8 is selected as the p parameter for one of the ARIMA models. Moving on to q, the ‘error-lag11’ appears to be significant for the first two rows, therefore, the q for the ARIMA model is 11. The ‘p-lag10’ was initially considered, yet it is not significant in the first and third rows to be listed as a potential p parameter. ARIMA(8,1,11) is the only model that is identified.

The set of possible ARIMA models across three approaches for the parameters specification process is: {ARIMA(3,1,2), ARIMA(0,1,1), ARIMA(1,1,0), ARIMA(1,1,1), ARIMA(8,1,11)}

3. Model fitting

The ARIMA(3,1,1) is fitted to the Bitcoin time series to find the related significant tests. Three estimation methods are used to find the model’s coefficients for AR and MA terms, which control how the likelihood of the prediction is optimized. The most recommended methods are “Maximum Likelihood” (ML), “Conditional Sum of Squares” (CSS), and a combination of both (CSS-ML) when the other two show ambiguous outcomes. Afterward, the AIC, BIC, and Error measures are evaluated to find the best model to achieve the most precise prediction for the Bitcoin index in upcoming periods.

i. ARIMA((3,1,2)

```
z test of coefficients:

     Estimate Std. Error  z value Pr(>|z|)
ar1  0.753783   0.081509   9.2479  < 2e-16 ***
ar2 -0.973263   0.064155 -15.1704  < 2e-16 ***
ar3  0.180991   0.080673   2.2435  0.02486 *
ma1 -0.599517   0.024508 -24.4618  < 2e-16 ***
ma2  0.999998   0.027958  35.7678  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 21: ML method for ARIMA(3,1,2)

```
z test of coefficients:

      Estimate Std. Error  z value  Pr(>|z|)
ar1  0.6487960  0.0235279  27.5756 < 2.2e-16 ***
ar2 -0.9755851  0.0121714 -80.1540 < 2.2e-16 ***
ar3  0.2064720  0.0238042   8.6738 < 2.2e-16 ***
ma1 -0.5232884  0.0248406 -21.0659 < 2.2e-16 ***
ma2  1.0618293  0.0056187 188.9799 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 22: CSS method for ARIMA(3,1,2)

```
z test of coefficients:

     Estimate Std. Error  z value Pr(>|z|)
ar1  0.753767   0.081508   9.2478  < 2e-16 ***
ar2 -0.973225   0.064152 -15.1705  < 2e-16 ***
ar3  0.180960   0.080672   2.2432  0.02489 *
ma1 -0.599520   0.024504 -24.4666  < 2e-16 ***
ma2  0.999999   0.027962  35.7625  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 23: CSS-ML method for ARIMA(3,1,2)

Figure 21 shows the ARIMA model's coefficient values with the full exact likelihood, which helps produce statistically efficient estimates. The statistics record all terms to be significant with p-values that are much lower than 0.05, suggesting the validity of this model, but further analysis is required to confirm the usability of this model. When looking at Figure 22 for the CSS methods, the third AR's p-value reduces to reach an extremely significant value. The CSS-ML method in Figure 23 displays a similar p-value for AR3 at 0.025, which satisfies the significance at the 5% level. This method is often proposed to get both decent starting values using CSS, then switching to ML to refine the final estimates. With all AR and MA considered significant, this model may be a proper one to fit the time series object.

ii. ARIMA(1,1,0)

```
z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ar1 0.188468   0.077681  2.4262  0.01526 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 24: ML method for ARIMA(1,1,0)

```
z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ar1 0.187169   0.076937  2.4328  0.01498 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 24: CSS method for ARIMA(1,1,0)

Both the MA1 coefficients' p-values in Figures 24 and 25 are statistically significant at a significance level of α=0.05. With similar results for both ML and CSS methods, the ARIMA(0,1,1) model may require further assessment to assume a good model, though the moving average term MA is related to the general behavior, and the CSS-ML method is unnecessary in this situation, as ML and CSS show similar outputs.

iii. ARIMA(0,1,1)

```
z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ma1 0.167265   0.072664  2.3019  0.02134 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 24: ML method for ARIMA(0,1,1)

```
z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ma1 0.166782   0.072675  2.2949  0.02174 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 25: CSS method for ARIMA(0,1,1)

Comparable to ARIMA(0,1,1), the ARIMA(1,1,0) also shows significant autoregressive terms AR in both ML and CSS methods when fitting the model. Although the p-values are roughly significant, there is not enough evidence to indicate the usability of this model without evaluating any other advanced metrics.

iv. ARIMA(1,1,1)

```
z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ar1  0.46509    0.37010  1.2567   0.2089
ma1 -0.28739    0.40144 -0.7159   0.4741
```

Figure 26: ML method for ARIMA(1,1,1)

```
z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ar1  0.45349    0.25895  1.7513   0.0799 .
ma1 -0.28098    0.27890 -1.0075   0.3137
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 27: CSS method for ARIMA(1,1,1)

```
z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ar1  0.46433    0.36964  1.2562   0.2090
ma1 -0.28668    0.40080 -0.7153   0.4744
```

Figure 28: CSS-ML method for ARIMA(1,1,1)

Dissimilar to ARIMA(0,1,1) and ARIMA(1,1,0) where further assessment is considered, the ARIMA(1,1,1) model is decided not to be an appropriate model to be fitted into the series. It is because both the moving average and autoregressive terms are statistically insignificant, regardless of the 10% or 5% significance level. The AR in CSS method becomes marginally significant is not evident enough.

v. ARIMA(8,1,11)

```
z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1   0.120146   0.715418  0.1679  0.86663
ar2  -0.075102   0.672956 -0.1116  0.91114
ar3   0.034895   0.521539  0.0669  0.94666
ar4   0.187803   0.185128  1.0144  0.31037
ar5  -0.243277   0.169953 -1.4314  0.15231
ar6  -0.204268   0.265179 -0.7703  0.44112
ar7   0.374476   0.279445  1.3401  0.18022
ar8  -0.312673   0.383720 -0.8148  0.41516
ma1   0.066386   0.718072  0.0925  0.92634
ma2   0.136291   0.745680  0.1828  0.85498
ma3   0.026416   0.649458  0.0407  0.96756
ma4  -0.210054   0.294500 -0.7133  0.47569
ma5   0.152857   0.215653  0.7088  0.47844
ma6   0.364290   0.208785  1.7448  0.08102 .
ma7  -0.270160   0.362567 -0.7451  0.45619
ma8   0.627258   0.383690  1.6348  0.10209
ma9   0.063051   0.335334  0.1880  0.85086
ma10 -0.018334   0.356687 -0.0514  0.95901
ma11 -0.042250   0.291825 -0.1448  0.88489
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 29: ML method for ARIMA(8,1,11)

When fitting the specifically large mode with AR(8)I using the Maximum Likelihood Method (Figure 29), the statistical test of coefficients shows that the entire number of autoregressive and moving average terms are insignificant. The p-values for those terms are all above the 0.05 significance level, while only one moving average named MA6, is marginally significant with a 0.08 p-value at the 10% level of significance. Due to this issue, the model is fitted again following the CSS method to see if the minimized sum of squares of residuals may vary the output.

```
z test of coefficients:

       Estimate Std. Error   z value  Pr(>|z|)
ar1  -0.0421682  0.0060728   -6.9438 3.817e-12 ***
ar2   0.0756106  0.0033706   22.4325 < 2.2e-16 ***
ar3   0.3321102  0.0034766   95.5284 < 2.2e-16 ***
ar4  -0.1171685  0.0052833  -22.1771 < 2.2e-16 ***
ar5  -0.4391863  0.0012365 -355.1909 < 2.2e-16 ***
ar6   0.1228742  0.0029529   41.6107 < 2.2e-16 ***
ar7   0.1960335  0.0023723   82.6349 < 2.2e-16 ***
ar8  -0.1231282  0.0031943  -38.5465 < 2.2e-16 ***
ma1   0.2413427  0.0700857    3.4435 0.0005742 ***
ma2   0.1019135        NaN       NaN       NaN
ma3  -0.3880855  0.0426130   -9.1072 < 2.2e-16 ***
ma4   0.0450182  0.0713786    0.6307 0.5282394
ma5   0.4110145  0.0915328    4.4904 7.111e-06 ***
ma6   0.0376484  0.0872155    0.4317 0.6659809
ma7  -0.0577674  0.0789784   -0.7314 0.4645145
ma8   0.4774437  0.0860558    5.5481 2.888e-08 ***
ma9   0.0186473  0.0894134    0.2086 0.8347979
ma10 -0.0211949  0.0743694   -0.2850 0.7756476
ma11 -0.1049262  0.0789447   -1.3291 0.1838117
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 30: CSS method for ARIMA(8,1,11)

By taking Figure 30 of the CSS method into consideration, there are more AR and MA terms are witnessed to be significant compared to the previous method, indicating that this is probably a better approach to quickly estimate a larger ARIMA model. While all autoregressive terms turn out to be significant, numerous figures for the moving average are yet to be the same. It is suggested that further inference be made to conclude whether the model is credible to be fitted.

vi. AIC & BIC Values

The Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are model selection criteria that ease the process of defining the best ARIMA model among candidates by balancing the likelihood level of fitting and the simplicity of the chosen one. The lower the values for both criteria, the better the model is.

	df	AIC
model.110	2	260.3679
model.312	6	260.4386
model.011	2	261.0479
model.111	3	261.8676
model.8111	20	272.9771

Figure 31: AIC Values for all models

	df	BIC
model.110	2	266.5307
model.011	2	267.2107
model.111	3	271.1118
model.312	6	278.9271
model.8111	20	334.6052

Figure 32: BIC Values for all models

The AIC table showing all the computed values for all specified models identifies two finalists, ARIMA(3,1,2) and ARIMA(1,1,0), respectively. About the large model with AR(8) in practice, the 20 parameters make the model (8,1,11) complex and unstable for forecasting, which may trigger warnings about non-stationarity and overfitting risk. Furthermore, both the AIC and BIC tables declare the high complexity at the largest value. In Figures 31 and 32, model (1,1,0) seems to annotate the smallest value among others. Besides, despite a high BIC for the model ARIMA(3,1,2), the difference is not considerable enough; this model still records the second lowest AIC. In accordance with AIC, the (1,1,0) may be the most practical option thanks to its stability, reliability, interpretability, and simplicity, while the (3,1,2) is the second best one with a good compromise between AIC and BIC. The two models are followed by the third-best one: the ARIMA(0,1,1) model.

vii. Error Measures

The final professional step is to compare models using forecast accuracy metrics, also known as error measures, such as Mean Error (ME), Root Mean Square Error (RMSE), etc.. Generally, the lower the error, the more likely the model will perform accurate predictions.

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
ARIMA(3,1,2)	0.08862280	0.5153958	0.3819917	0.4260851	1.822494	0.1731648	-0.02917559
ARIMA(0,1,1)	0.10170432	0.5358849	0.3868470	0.4849851	1.827695	0.1753658	-0.02451594
ARIMA(1,1,0)	0.09643414	0.5347416	0.3877852	0.4611352	1.831496	0.1757911	-0.04510516
ARIMA(1,1,1)	0.08948625	0.5339000	0.3872616	0.4299748	1.832874	0.1755538	-0.03026349
ARIMA(8,1,11)	0.06976801	0.4857513	0.3569416	0.3460627	1.709861	0.1618091	-0.02321891

Figure 33: Table for Error Measures of all models

Surprisingly, no matter observing a high complexity, the ARIMA(8,1,11) model indicates the best raw forecast accuracy in terms of lowest ME, RMSE, MAE, and MASE. It also has the lowest ACF1, which means the model's residuals are the closest to white noise. Besides, the ARIMA(3,1,2) is the following model with the second lowest ME, RMSE, MAE, and MPE. The ARIMA(1,1,0) seems to drop to the third best model in the list to be fitted in this Bitcoin series according to these metrics, thanks to its not much difference in forecast performance, though the metrics are unremarkable compared to ARIMA(1,1,1).. It is due to the fact that ARIMA(1,1,1) is witnessed to be insignificant in terms of the coefficient test.

III. CONCLUSION

From 2011 to 2015, the Bitcoin index recorded a dramatic increase since the year 2017, rocketing up to the present after a flat period for 6 years straight. The analysis started with descriptive analytics with combinations of summary statistics and visual exploration, revealing the overall uptrend of the bull market, which resulted from the increase in volatility. Because of non-stationary and non-cyclical patterns, the Box-Cox transformation was applied to perform the first differencing that initiates the dataset's stationarity to fit the ARIMA model. ACF-PACF plots, EACF matrices, and BIC tables were conducted to endorse a set of proposed (p,d,q) models including {ARIMA(3,1,2), ARIMA(0,1,1), ARIMA(1,1,0), ARIMA(1,1,1), ARIMA(8,1,11)}. Each candidate was then fitted using multiple estimation methods to assess the statistical significance of estimated coefficients, together with accuracy metrics, assisting in determining the best-suited performance. On the one hand, the model ARIMA(8,1,11) seemed to achieve the lowest forecast errors despite capturing the poor AIC as well as BIC values. On the other hand, the risk of overfitting due to a high complexity model was ultimately deemed impractical. As alternatives, ARIMA(1,1,0) and ARIMA(3,1,2) were the second and best model that offers a balance between predictive accuracy, simplicity, and stability to be fitted. They are believed to provide a reliable tool for forecasting the index of the Bitcoin market while maintaining model interpretability as well as robustness.

IV. APPENDICES

```
rm(list = ls())

#============================ Install Packages ============================

library(TSA)

library(lmtest)

library(tseries)

library(forecast)

library(readr)

library(stats)


#============================ Set working directory & Read file ============================

# Set working directory

setwd("D:/Phạm Võ Vĩnh Phương/RMIT University/Assessments/Sem 3/Time Series Analysis/Asm 2")


# Read dataset

bitcoin <- read_csv("assignment2Data2025.csv")


#============================ Converting to time series & plot ============================

# Convert to time series object

bitcoin_ts <- ts(bitcoin$Bitcoin, start = c(2011, 8), end = c(2025, 1), frequency = 12)

bitcoin_ts


#Fit linear regression model

t <- time(bitcoin_ts)

t

linear_bitcoin <- lm(bitcoin_ts ~ t)
```

```
# Plot the time series
plot(bitcoin_ts,
    ylab = "Deviation Index",
    xlab = "Time",
    type = 'o',
    main = "Time Series Plot for Bitcoin Index")
abline(linear_bitcoin, col = "#7e0000")

# Create and Plot The First Lag of the Bitcoin Series
y = bitcoin_ts
x = zlag(bitcoin_ts)
index = 2:length(x)
cor(y[index],x[index])

plot(y=bitcoin_ts,x=zlag(bitcoin_ts),
    ylab='Index',
    xlab='Time',
    main= "Scatter plot of Bitcoin Index Year-to-Month")

# Descriptive
summary(bitcoin_ts)

#============================ Stationarity & normality check for time series object
============================
# ACF & PACF
par(mfrow=c(1,2))
acf(bitcoin_ts, main = "ACF plot of Bitcoin TS")
pacf(bitcoin_ts, main = "PACF plot of Bitcoin TS")
par(mfrow=c(1,1))
```

```
# Augmented Dickey-Fuller Test
adf.test(bitcoin_ts)

# Q-Q PLot
qqnorm(y=bitcoin_ts, main = "QQ plot of Bitcoin TS")
qqline(y=bitcoin_ts, col = 2, lwd = 1, lty = 2)

# Shapiro-Wilk Test
shapiro.test(bitcoin_ts)

#============================ Transformation ============================
# Box-cox Transformation
BC <- BoxCox.ar(bitcoin_ts, lambda = seq(0, 1, 0.01))
BC$ci
lambda <- BC$lambda[which(max(BC$loglike) == BC$loglike)]
lambda
bitcoin_bc = ((bitcoin_ts^lambda)-1)/lambda

par(mfrow=c(1,1))
plot(bitcoin_bc,
    type='o',
    ylab ="INdex",
    main="Time series plot for BC transformed Bitcoin TS")

# Log Transformation
bitcoin_log <- log(bitcoin_ts)
plot(bitcoin_log,
    ylab = "Index",
```

```
    xlab = "Time",
    main = "Time Series Plot for Log transformed Bitcoin TS",
    type = "o")



 # Q-Q PLot
qqnorm(y=bitcoin_bc, main = "QQ plot for BC transformed Bitcoin TS")
qqline(y=bitcoin_bc, col = 2, lwd = 1, lty = 2)


 # Shapiro-Wilk Test
shapiro.test(bitcoin_bc)


#============================ Differencing ============================
 # Plot
bitcoin_dif <- diff(bitcoin_ts, differences = 1)
plot(bitcoin_dif,
    ylab = "Index",
    xlab = "Time",
    main = "BC transformed and first differenced Bitcoin TS",
    type = "o")


# ACF & PACF
par(mfrow=c(1,2))
acf(bitcoin_dif, main = "ACF plot for the First Differenced Bitcoin TS")
pacf(bitcoin_dif, main = "PACF plot for the First Differenced Bitcoin TS")
par(mfrow=c(1,1))


# Augmented Dickey-Fuller Test
adf.test(bitcoin_dif)
```

```
# Phillips-Perron Unit Root Test
pp.test(bitcoin_dif)

# KPSS Test
kpss.test(bitcoin_dif)

#============================ Propose possible models using specification tools
============================
# ACF & PACF
par(mfrow=c(1,2))
acf(bitcoin_dif, main = "ACF plot for the First Differenced Bitcoin TS")
pacf(bitcoin_dif, main = "PACF plot for the First Differenced Bitcoin TS")
par(mfrow=c(1,1))
#-- {ARIMA(3,1,2)}

# EACF
eacf(bitcoin_dif, ar.max = 5, ma.max = 5)
#-- {ARIMA(0,1,1), ARIMA(1,1,0), ARIMA(1,1,1)}

# BIC Table
resBit = armasubsets(y = bitcoin_dif,
                     nar = 14, nma = 14,
                     y.name = 'p',
                     ar.method = 'ols')
plot(resBit)
#-- {ARIMA(8,1,11), ARIMA(8,1,15)}

#-- Possible models: {ARIMA(3,1,2), ARIMA(0,1,1), ARIMA(1,1,0), ARIMA(1,1,1), ARIMA(8,1,11)}
```

```
#============================ Fit the models ============================
# ARIMA(3,1,2)
model.312 = arima(bitcoin_bc, order = c(3,1,2), method = 'ML')
coeftest(model.312)

model.312CSS = arima(bitcoin_bc, order = c(3,1,2), method = 'CSS')
coeftest(model.312CSS)

model.312CSSML = arima(bitcoin_bc, order = c(3,1,2), method = 'CSS-ML')
coeftest(model.312CSSML)



# ARIMA(0,1,1)
model.011 = arima(bitcoin_bc, order = c(0,1,1), method = 'ML')
coeftest(model.011)

model.011CSS = arima(bitcoin_bc, order = c(0,1,1), method = 'CSS')
coeftest(model.011CSS)

model.011CSSML = arima(bitcoin_bc, order = c(0,1,1), method = 'CSS-ML')
coeftest(model.011CSSML)



# ARIMA(1,1,0)
model.110 = arima(bitcoin_bc, order = c(1,1,0), method = 'ML')
coeftest(model.110)

model.110CSS = arima(bitcoin_bc, order = c(1,1,0), method = 'CSS')
```

```
coeftest(model.110CSS)


model.110CSSML = arima(bitcoin_bc, order = c(1,1,0), method = 'CSS-ML')
coeftest(model.110CSSML)



# ARIMA(1,1,1)
model.111 = arima(bitcoin_bc, order = c(1,1,1), method = 'ML')
coeftest(model.111)

model.111CSS = arima(bitcoin_bc, order = c(1,1,1), method = 'CSS')
coeftest(model.111CSS)

model.111CSSML = arima(bitcoin_bc, order = c(1,1,1), method = 'CSS-ML')
coeftest(model.111CSSML)



# ARIMA(8,1,11)
model.8111 = arima(bitcoin_bc, order = c(8,1,11), method = 'ML')
coeftest(model.8111)

model.8111CSS = arima(bitcoin_bc, order = c(8,1,11), method = 'CSS')
coeftest(model.8111CSS)

model.8111CSSML = arima(bitcoin_bc, order = c(8,1,11), method = 'CSS-ML')
coeftest(model.8111CSSML)



# Fit all models
```

```
aic_table <- AIC(model.312, model.011, model.110, model.111, model.8111)
aic_table[order(aic_table$AIC), ]

bic_table <- BIC(model.312, model.011, model.110, model.111, model.8111)
bic_table[order(bic_table$BIC), ]

# Use forecast package to get error measures
model.312A = Arima(bitcoin_bc,order=c(3,1,2),method='ML')
model.011A = Arima(bitcoin_bc,order=c(0,1,1),method='ML')
model.110A = Arima(bitcoin_bc,order=c(1,1,0),method='ML')
model.111A = Arima(bitcoin_bc,order=c(1,1,1),method='ML')
model.8111A = Arima(bitcoin_bc,order=c(8,1,11),method='ML')

Smodel.312A <- accuracy(model.312A)[1:7]
Smodel.011A <- accuracy(model.011A)[1:7]
Smodel.110A <- accuracy(model.110A)[1:7]
Smodel.111A <- accuracy(model.111A)[1:7]
Smodel.8111A <- accuracy(model.8111A)[1:7]

df.Smodels <- data.frame(
  rbind(Smodel.312A,Smodel.011A,Smodel.110A,
        Smodel.111A,Smodel.8111A)
)
colnames(df.Smodels) <- c("ME", "RMSE", "MAE", "MPE", "MAPE",
                          "MASE", "ACF1")
rownames(df.Smodels) <- c("ARIMA(3,1,2)", "ARIMA(0,1,1)", "ARIMA(1,1,0)",
                          "ARIMA(1,1,1)", "ARIMA(8,1,11)")

df.Smodels
```