# Time Series Analysis

# Assessment 2:

## Bitcoin Index Time Series Analysis and ARIMA Modeling

**Submitted By:** Sherodhi Sharma
S4024088

## 1. Introduction

This report presents a comprehensive time series analysis of the Bitcoin index (USD) monthly data from August 2011 to January 2025. The analysis involves examining the statistical properties of the time series, identifying appropriate ARIMA (Autoregressive Integrated Moving Average) models, estimating parameters, and selecting the best model based on various criteria.

Bitcoin, as the first and most well-known cryptocurrency, created in 2008, is a decentralized digital currency operating without a central authority has shown remarkable volatility and growth since its inception. It uses peer-to-peer technology for transactions, which are transparently recorded on a public ledger called the blockchain and secured through cryptography. With a limited supply of 21 million coins, new bitcoins are created through a process called mining.

Its price is highly volatile, and while it has a significant history of price fluctuations, it remains a prominent cryptocurrency, with its current price around 148,776.06 AUD as of late April 2025. More information about Bitcoin can be found on cryptocurrency exchanges, blockchain explorers, financial news outlets, and the official Bitcoin website. Understanding its historical price patterns through time series modeling can provide insights into its behavior and potentially assist in forecasting future movements.

The primary objectives of this report are:

1. To perform a thorough descriptive analysis of the Bitcoin index data
2. To determine appropriate ARIMA models using various model specification tools
3. To estimate parameters for the identified models
4. To select the optimal model based on goodness-of-fit criteria

The analysis will follow a structured approach, starting with descriptive statistics and exploratory data analysis, followed by model identification, estimation, and selection.

## 2. Descriptive Analysis

### 2.1 Data Overview and Summary Statistics

The dataset comprises monthly Bitcoin index values (in USD) spanning from August 2011 to January 2025, totalling 162 observations. Let's start by examining the basic statistical properties of the data. We've started it by using certain libraries such as tseries, forecast, ggplot2, lmtest, and TSA to perform the analysis. First and foremost, I've imported the data using read.csv() function and further imported the data into a time series data using the ts() function.

To explore and gather more information about the time series data I've performed the summary statistics on it using summary() function which gave me the following output.

**Summary Statistics:**

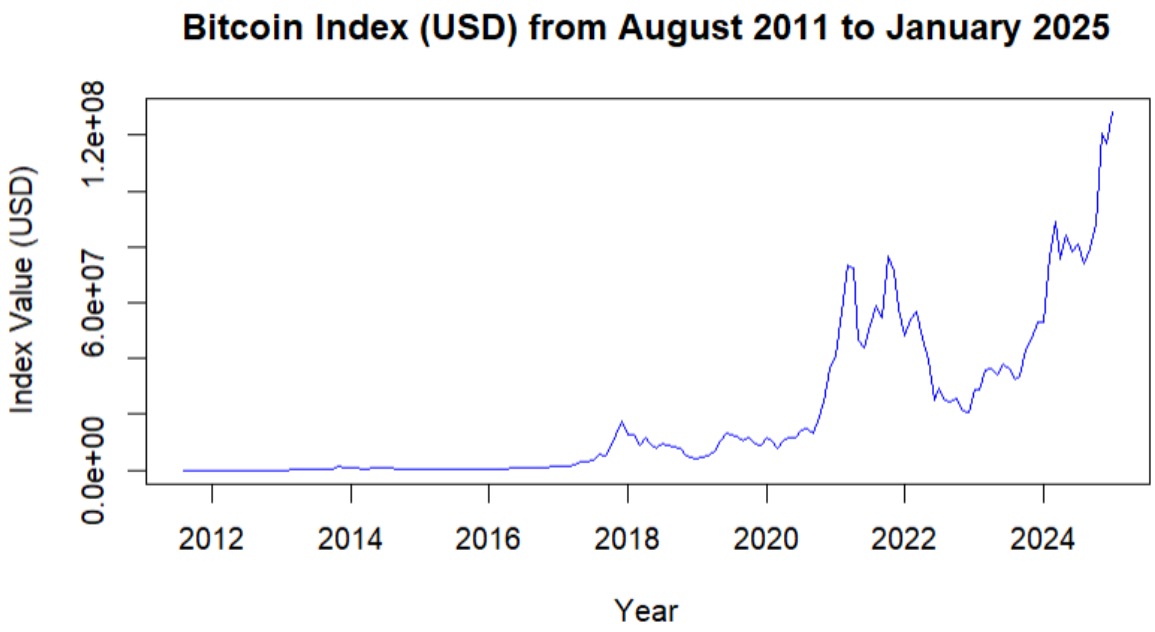
| Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---|---|---|---|---|---|
| 3988 | 470981 | 7930612 | 19781743 | 31599541 | 128015000 |

The summary shows significant variability in the Bitcoin index value, with a minimum of approximately 3,988 USD and a maximum of around 128,015,000 USD. The large difference between the minimum and maximum values suggests substantial growth and volatility over the period. Also,

the median value is 7930612 USD which is at the middle of our dataset. Whereas the mean or the average value of Bitcoin in our dataset is 19781743 USD which is significantly higher than the median, which suggests a positive skew in the data. This is common in financial time series where prices can experience large upward movements.

### 2.2 Time Series Plot

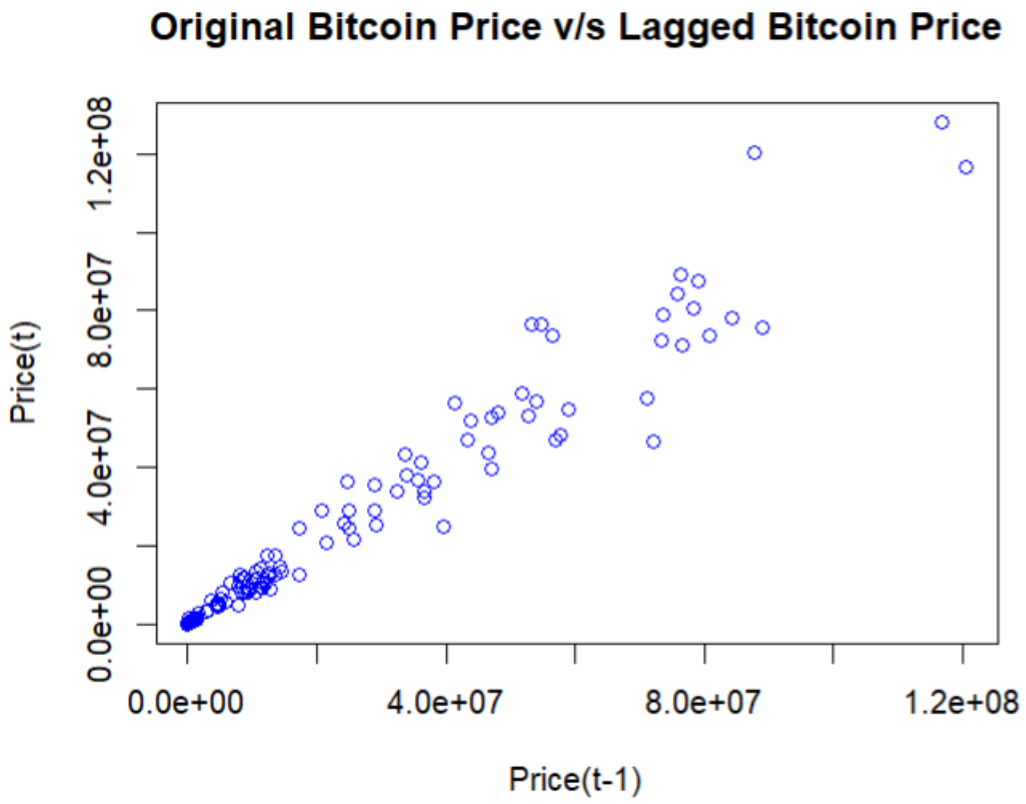
After getting the summary statistics, we've moved forward by drawing out a time series plot using the plot() function




The time series plot reveals several notable features:

1. **Exponential Growth**: The Bitcoin index has shown explosive growth, particularly since 2017.
2. **Volatility**: The series displays significant volatility with multiple sharp rises and falls.
3. **Cycles and Bubbles**: Several boom-and-bust cycles are visible, most notably in 2017-2018, 2021, and 2024.
4. **Non-Stationary Behavior**: The series shows clear non-stationarity with an upward trend and increasing variance.

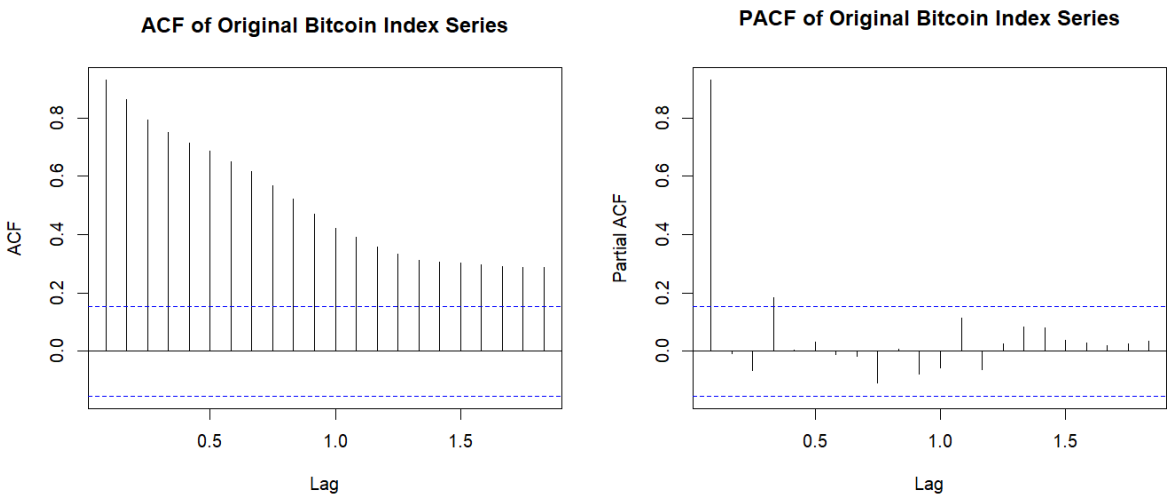
The graph doesn't show any obvious repeating patterns related to specific times of the year. This isn't surprising because Bitcoin's price usually moves based on overall market trends and news, not predictable seasonal changes. Furthermore, there aren't significant jumps or drops between data points that are close together in time, suggesting an **autoregressive behavior** where the price in the near future tends to be predictably linked to its recent values in a stable manner.




To further confirm the patterns we've observed, we analyzed the correlation at a lag of one month and a lag of two months by using the function zlag(). Upon solving the correlation with a one-month lag was found to be 0.9779498, revealing a very strong positive connection between Bitcoin's monthly prices in sequence. This substantial correlation strongly suggests an autoregressive pattern within the Bitcoin price data. Expanding on this, our examination of the correlation with a two-month lag also produced a similarly high value of 0.9778791. To gain a more complete picture of how the price correlates with its past values across different time lags, we produced an autocorrelation function (ACF) plot extending up to 40 lags. This ACF plot provides further evidence supporting the autoregressive properties inherent in the Bitcoin time series.

### 2.3 Examining Stationarity

A crucial step in time series analysis is determining whether the data is stationary. A stationary series has constant mean, variance, and autocorrelation structure over time.




The ACF plot, which measures correlation between observations at different lags displays the following points:

- Significant positive autocorrelation at early lags that gradually declines.
- The pattern shows a smooth, slow decay rather than cutting off sharply.
- Values exceed the blue significance lines (at approximately ±0.2) for multiple initial lags.
- The slow decay pattern suggests persistence in the time series.

Whereas the PACF plot which measures the direct correlation between observations after removing the effects of intermediate lags displays that:

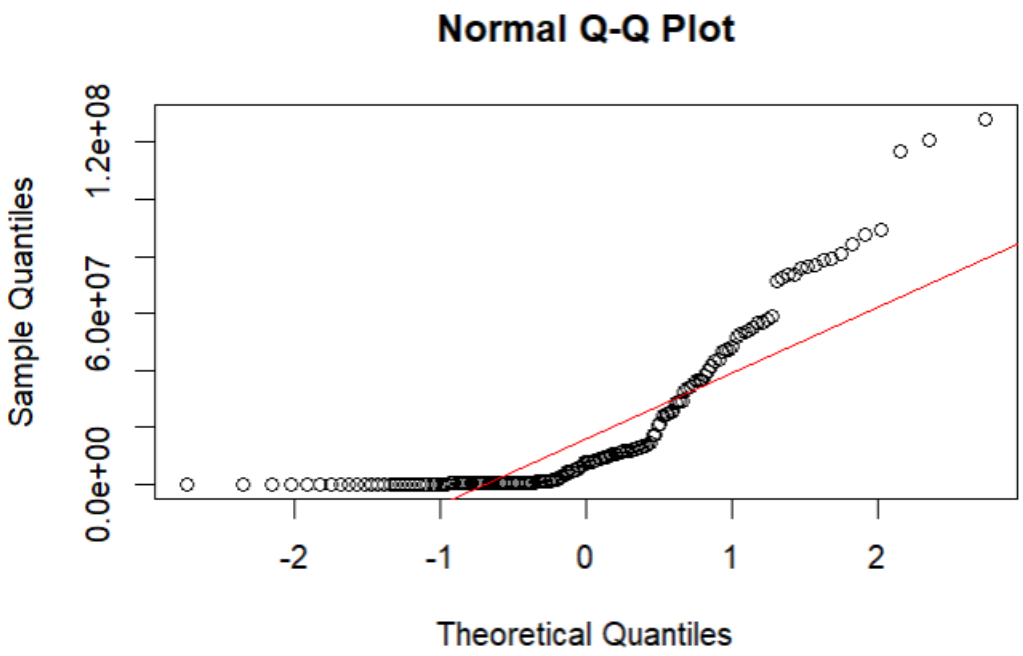
- A significant spike at lag 1 that quickly drops below significance
- Most values beyond lag 1 are within the significance bounds
- Only a few isolated spikes appear at higher lags

This kind of behaviour is commonly seen in financial data over time, indicating a price dynamic where the most recent prices have the most immediate influence on the current price. Recognizing this is useful for building forecasting tools and gaining insights into the statistical characteristics of how Bitcoin's price changes.

```
        Augmented Dickey-Fuller Test

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

The Augmented Dickey-Fuller test results for your Bitcoin time series (test statistic: -0.27056, p-value: 0.99) strongly indicate that the data is non-stationary, meaning it contains a unit root and follows a random walk pattern. This aligns with the gradually declining ACF and significant lag-1 PACF seen earlier, suggesting that Bitcoin prices don't revert to a long-term mean and instead exhibit persistent trends where future values are heavily dependent on previous observations plus random shocks. This non-stationarity is typical of financial asset prices and has important implications for risk assessment and investment strategies.




This Q-Q plot reveals that our Bitcoin data significantly deviates from a normal distribution, exhibiting a heavily right-skewed pattern with substantial outliers in the upper range (reaching approximately 1.2e+08). The S-shaped curve with a flat portion at the lower end and steep rise at the right indicates both high kurtosis (fatter tails) and positive skewness. These characteristics are typical of cryptocurrency markets, which tend to experience periods of stability punctuated by dramatic price surges. This non-normal distribution explains why standard time series models assuming normality would be inadequate for Bitcoin data and suggests that any risk models should account for these extreme events rather than assuming normal statistical properties.

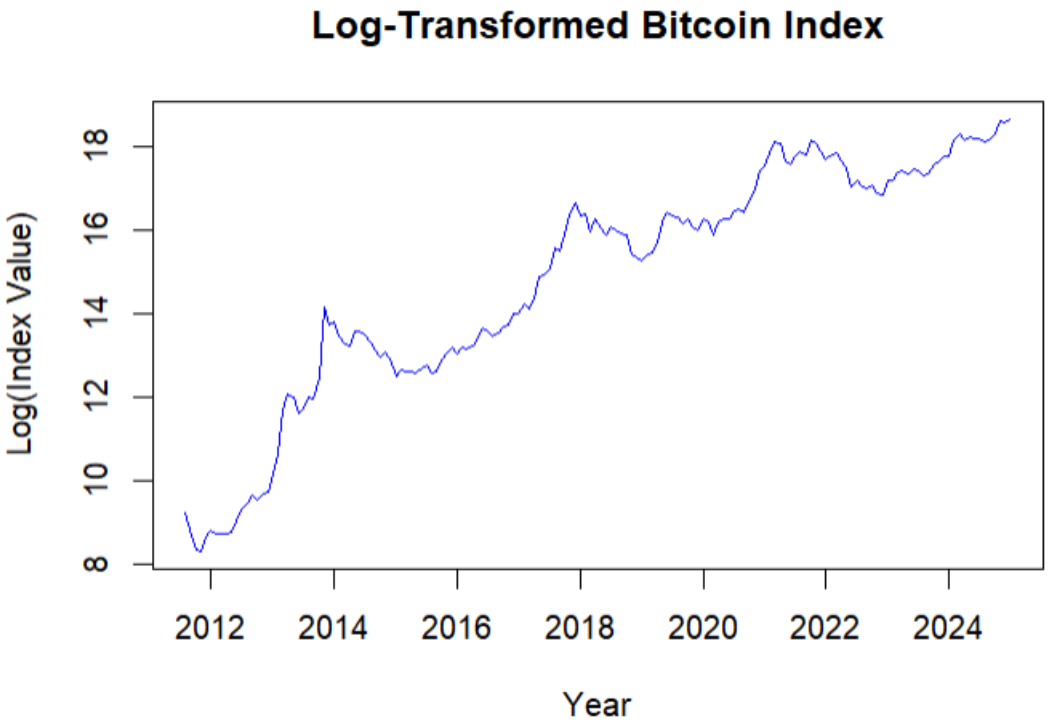
```
        Shapiro-Wilk normality test

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

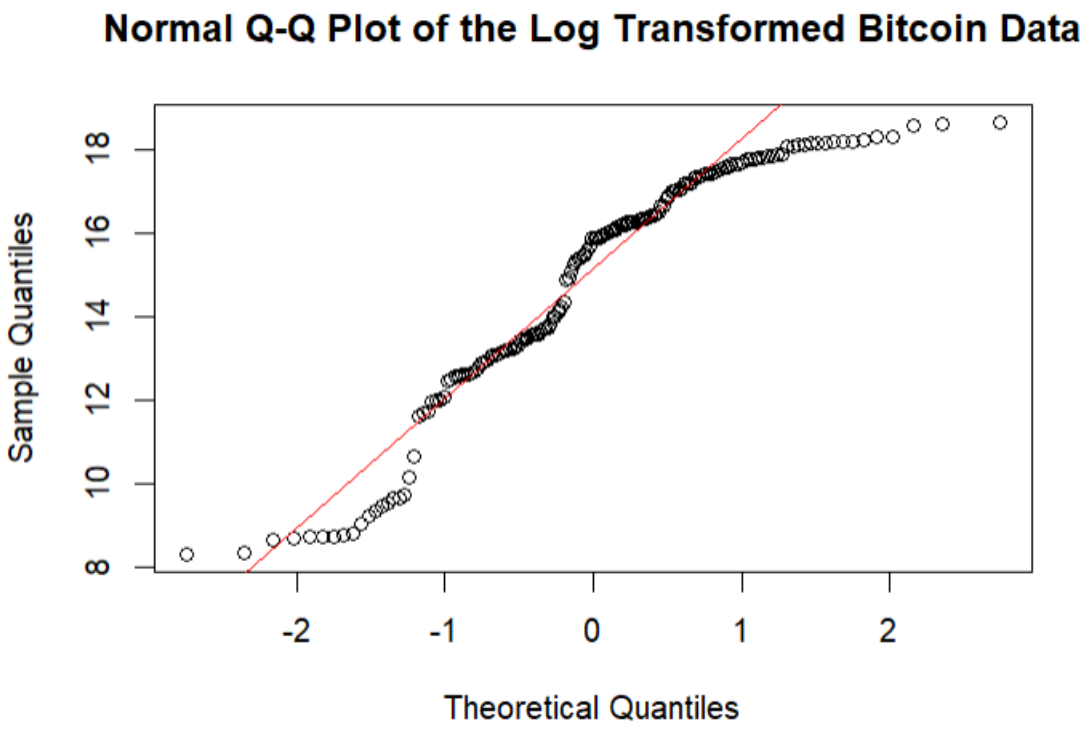
Now, we'll appropriately perform the Shapiro Wilk test after examining the Q-Q plot to provide statistical verification of the visual assessment, creating a more robust analysis. Here the test results (W = 0.73793, p-value = 1.102e-15) strongly confirm what our Q-Q plot visually suggested that our Bitcoin data significantly deviates from a normal distribution. The extremely low p-value (much less than 0.05) allows us to confidently reject the null hypothesis of normality. The low W statistic (far from 1.0) further indicates substantial deviation from normal distribution.

### 2.4 Transformation for Stabilizing Variance

Given the exponential growth pattern in the data, a logarithmic transformation might help stabilize the variance. We've performed the following by using the log() function. And also, further plotted the log transformed data of the bitcoin time series data.




Now this log-transformed Bitcoin index plot clearly reveals the cryptocurrency's growth pattern from 2012 to early 2025, using logarithmic scaling to better visualize percentage changes across its dramatic price range. The transformation shows a clearer picture of Bitcoin's relative performance over time, highlighting several key growth phases: rapid early adoption (2012-2014), consolidation (2014-2016), another substantial bull run (2016-2018), followed by the 2020-2021 surge during pandemic-era liquidity expansion, and subsequent volatility with an overall upward trajectory through 2025. The log transformation effectively compresses the extreme price movements that were evident in our previous analyses, making the long-term growth trend more apparent while still preserving the pattern of periodic surges followed by consolidation periods. This approach is particularly valuable for Bitcoin data given its non-normal distribution and extreme outliers identified earlier.




```
        Shapiro-Wilk normality test

data:  log_bitcoin_ts
W = 0.9138, p-value = 3.363e-08
```

The Q-Q plot of log-transformed Bitcoin data shows a substantial improvement in normality compared to the raw data, with points following the reference line more closely in the middle range (approximately between theoretical quantiles -0.5 and 1). However, deviations still exist at both tails, particularly noticeable below -1 and above 1.5 on the theoretical quantiles axis. The Shapiro-Wilk test results (W = 0.9138, p-value = 3.363e-08) confirm this visual assessment – while the W statistic is much closer to 1 than for raw data (0.9138 vs 0.73793), the extremely small p-value still leads to rejecting the null hypothesis of normality. This suggests that while log transformation has significantly improved the distribution's shape by reducing skewness and compressing extreme values, the Bitcoin data still exhibits non-normal characteristics, likely due to the cryptocurrency's inherent volatility patterns and market behavior that produce more extreme movements than would be expected in a normal distribution.

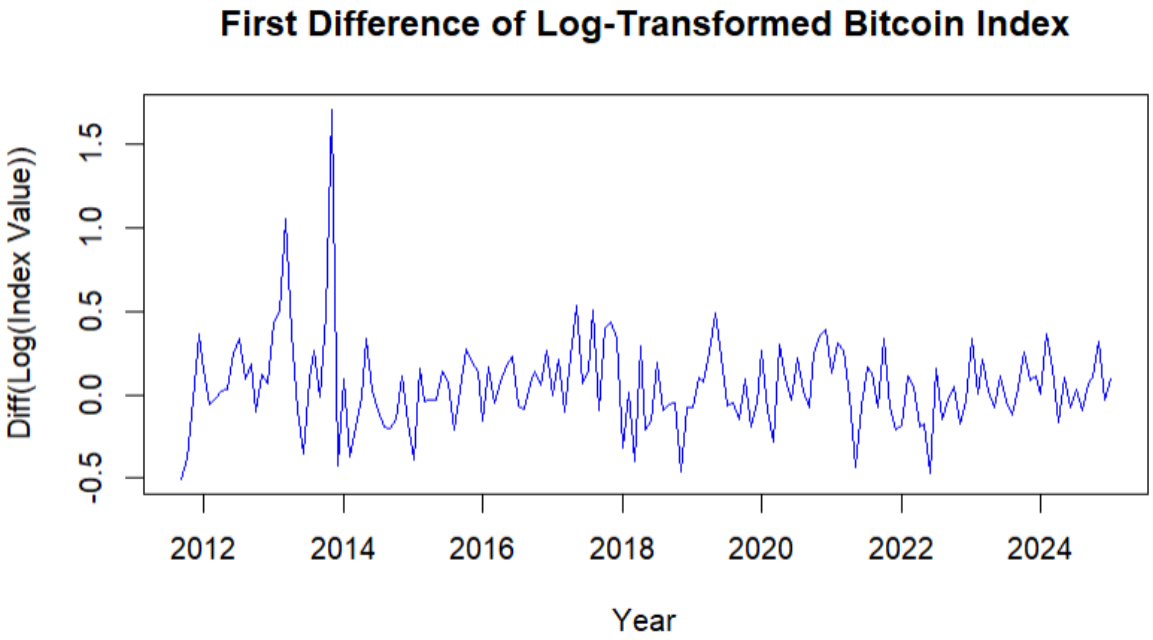
```
        Augmented Dickey-Fuller Test

data:  log_bitcoin_ts
Dickey-Fuller = -2.3599, Lag order = 5, p-value = 0.4262
alternative hypothesis: stationary
```

The Augmented Dickey-Fuller test on our log-transformed Bitcoin data (Dickey-Fuller = -2.3599, p-value = 0.4262) indicates that even after logarithmic transformation, the series remains non-stationary. Though the test statistic has improved compared to the original data (-2.3599 vs -0.27056), the p-value of 0.4262 still exceeds the typical 0.05 threshold, preventing rejection of the null hypothesis of non-stationarity. This persistent non-stationarity in the log-transformed data suggests that Bitcoin prices, even when viewed on a proportional scale, continue to exhibit characteristics of a random walk without mean reversion. While log transformation has helped normalize the distribution somewhat, it hasn't addressed the underlying time-dependent structure, indicating that further differencing would be necessary to achieve a stationary series suitable for many statistical modelling techniques.

### 2.5 Differencing for Stationarity

To achieve stationarity, we need to difference the log-transformed series which we've done using the diff() function. And also created the plot after first differencing.




The plot for our first difference of the log-transformed Bitcoin index, represents the approximate percentage change in Bitcoin value from one period to the next. The differenced series fluctuates around zero, indicating a potentially stationary process where changes don't exhibit a clear trend over time. Notable volatility spikes appear particularly during 2013-2014, with the largest peak showing nearly 1.5 point increase (approximately 150% growth) in a single period. After 2015, the volatility pattern becomes somewhat more consistent, with most movements contained within a ±0.5 band (roughly ±50% changes). This transformation has successfully removed the strong upward trend visible in the log-transformed data, creating a series that appears more suitable for time series modeling. The stationary characteristics of this differenced series suggest that Bitcoin's price changes, rather than absolute prices, may follow a more predictable statistical pattern.

```
        Augmented Dickey-Fuller Test

data:  diff_log_bitcoin_ts
Dickey-Fuller = -4.8883, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary
```

```
        Phillips-Perron Unit Root Test

data:  diff_log_bitcoin_ts
Dickey-Fuller Z(alpha) = -128.39, Truncation lag parameter = 4, p-value = 0.01
alternative hypothesis: stationary
```

Both the Augmented Dickey-Fuller test (DF = -4.8883, p-value = 0.01) and Phillips-Perron test (Z(alpha) = -128.39, p-value = 0.01) confirm that the first-differenced log-transformed Bitcoin series is stationary. The significant p-values (0.01) allow rejecting the null hypothesis of non-stationarity, indicating the transformation has successfully addressed the random walk behavior present in the original and log-transformed data. Running both tests provides robust confirmation of stationarity, with Phillips-Perron offering additional validation by accounting for serial correlation using a non-parametric approach. This stationarity achievement is crucial for Bitcoin analysis as it transforms the unpredictable price levels into a more manageable series of returns that fluctuate around a constant mean, enabling reliable statistical modelling through ARIMA technique. This approach effectively converts Bitcoin's explosive price growth into analysable percentage changes with stable statistical properties.

### 2.6 Analysis of Stationary Series

Let's examine the statistical properties of the stationary (differenced log-transformed) series using the summary() function.

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.50667  -0.07890   0.03910   0.05874   0.19045   1.70479
```

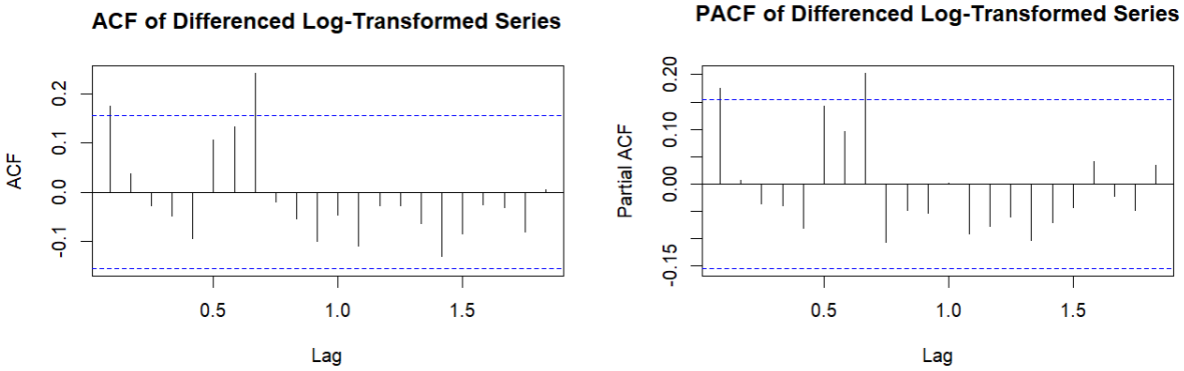
The summary statistics for our stationary Bitcoin series (first-differenced log-transformed data) reveal key characteristics of Bitcoin's periodic returns. The data ranges from a minimum of -50.7% to a maximum of 170.5%, with a positive mean of 5.9% indicating an overall upward trend in Bitcoin's value over time and also the mean is close to zero and shows more stable variance, providing a good basis for subsequent ARIMA modelling. The median (3.9%) being lower than the mean suggests positive skewness, confirmed by the distance between the maximum (170.5%) and third quartile (19.0%) being much greater than between the minimum (-50.7%) and first quartile (-7.9%). This asymmetry reflects Bitcoin's tendency for occasional dramatic upward movements rather than equally severe crashes. The interquartile range of approximately 27 percentage points (-7.9% to 19.0%) represents the typical fluctuation in Bitcoin returns during most periods, while the extremes demonstrate the asset's potential for both substantial gains and significant drawdowns that far exceed conventional financial assets.

## 3. Model Specification

Based on the descriptive analysis, we have determined that an ARIMA model with d=1 (first difference) is appropriate for the log-transformed Bitcoin index data. Now, we need to identify potential values for p (autoregressive order) and q (moving average order). And to do that we're

going to use three different methods which are the ACF and PACF analysis, EACF analysis, and the BIC Table analysis.

### 3.1 ACF and PACF Analysis




**ACF Interpretation:**

- The ACF shows a significant spike at lag 0.5 and 1, then cuts off.
- This pattern suggests a potential moving average component of order 1 (q=2).

**PACF Interpretation:**

- The PACF shows significant spikes at lags 1 and 2, then cuts off.
- This pattern suggests a potential autoregressive component of orders 2 or 3 (p=2 or p=3).

Upon analysing the time series characteristics through correlation analysis tools, both the ACF and PACF diagrams reveal notable correlation patterns. Specifically, the ACF exhibits meaningful correlations at the initial two time lags that surpass the statistical significance thresholds. Correspondingly, the PACF displays significant partial correlations for two consecutive lags, with the correlation at lag three approaching the boundary of statistical significance. The pattern of significant lags in both correlation functions indicates that **ARIMA(2,1,2) and ARIMA(3,1,2)** specifications would likely capture the underlying dynamics of Bitcoin price fluctuations most effectively.

### 3.2 Extended Autocorrelation Function (EACF)

The result that I'm getting for my EACF Table using the eacf() function is the following.

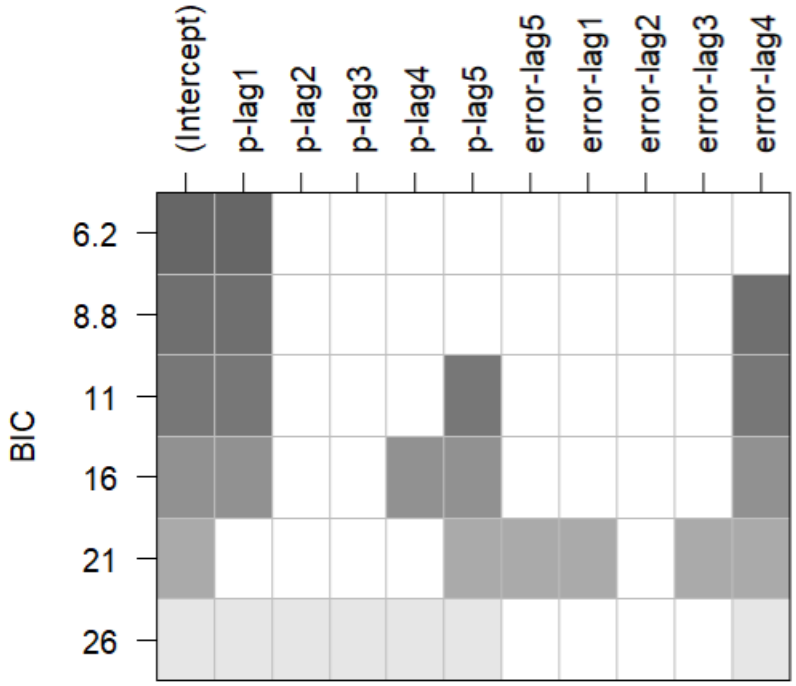
```
AR/MA
   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0  x o o o o o o x o o o  o  o  o  o  o
1  o o o o o o o x o o o  o  o  o  o  o
2  x o o o o o o x o o o  o  o  o  o  o
3  x o o o o o o x o o o  o  o  o  o  o
4  x o x o o o o x o o o  o  o  o  o  o
5  x x x o x o o x o o o  o  o  o  o  o
6  x x x x x o o o o o o  o  o  o  o  o
7  x x x o x x o x o o o  o  o  o  o  o
8  x x x x x x x x o o o  o  o  o  o  o
9  x o x o o o o x o o o  o  o  o  o  o
10 x x o x o o o x o o o  o  o  o  o  o
```

In the EACF (Extended Autocorrelation Function) table, we're looking for the top-left corner of a triangle of zeros (marked with "o"), that is not distracted by an x which indicates potential model orders.

Hence, suitable ARIMA (p,1,q) models include:

- **ARIMA(0,1,1)**
- **ARIMA(1,1,1)**
- **ARIMA(1,1,2)**
- **ARIMA(0,1,2)**

### 3.3 BIC Table Analysis




The approach we've followed to get the ARIMA Models from the heatmap can be described as

- Darker cells = lower BIC = better model

The BIC analysis utilized a balanced 5×5 grid, providing sufficient model complexity without excessive parameters. Results consistently support p=1 across the top four models, with the leading specification featuring only this parameter. The second-best model introduces q=4, which appears in five total specifications. And the third best is p=5 even though it only appears in substantially inferior models. This combination stands out as the third most optimal model in terms of balancing fit and parsimony, making it a valid candidate for further evaluation and forecasting.

These reinforce the EACF results as **ARIMA (1,1,4)**, **ARIMA(1,1,0)**, and **ARIMA(5,1,4)**.

### 3.4 Combined ARIMA Models

Based on the model specification tools (ACF-PACF, EACF, and BIC table), the following set of ARIMA models are proposed for further analysis:

ARIMA(2,1,2), ARIMA(3,1,2), ARIMA(0,1,1), ARIMA(1,1,1), ARIMA(1,1,2), ARIMA(0,1,2),

ARIMA (1,1,4), ARIMA(1,1,0), and ARIMA(5,1,4).

## 4. Model Estimation

To determine the most suitable ARIMA specification, we initially employed the arima() function to assess coefficient significance across different model configurations. Our examination found remarkable consistency between parameter estimates derived from both Maximum Likelihood (ML) and Conditional Sum of Squares (CSS) estimation techniques. This observed stability in coefficient values across methodologies substantially reinforces our confidence in the reliability of the estimated parameters.

#### 4.1 ARIMA(1,1,0) Model

The AR(1) coefficients from both models (using ML and CSS methods) have p-values of ~0.094, which are greater than 0.05 but less than 0.1. This means the AR(1) term is marginally significant at the 10% level, but not statistically significant at the 5% level.

#### 4.2 ARIMA(1,1,4) Model

In both models (using ML and CSS methods), the AR(1) and MA(1) terms are highly significant ($p < 0.001$), indicating they are important in explaining the series. However, MA(2), MA(3), and MA(4) have p-values > 0.1, meaning they are not statistically significant and contribute less explanatory power to the model.

#### 4.3 ARIMA(5,1,4) Model

In the ARIMA(5,1,4) model using ML, AR(2), AR(3), MA(2), and MA(3) are statistically significant ($p < 0.05$), indicating their relevance to the model. However, using CSS, only AR(2) remains significant, and MA(2) is marginally significant ($p \approx 0.095$), while all other terms are not. This suggests the model may be over-parameterized and that many coefficients are not contributing meaningfully.

#### 4.4 ARIMA(0,1,1) Model

In the ARIMA(0,1,1) model, the MA(1) term is marginally significant in both ML ($p \approx 0.095$) and CSS ($p \approx 0.094$) methods, falling in the 10% significance range. This indicates weak evidence that the MA(1) term contributes meaningfully to the model.

#### 4.5 ARIMA(0,1,2) Model:

In the ARIMA(0,1,2) model, the MA(1) term is marginally significant with p-values around 0.095 in both ML and CSS methods, indicating weak evidence of its contribution. However, the MA(2) term is not significant ($p > 0.66$), suggesting it does not contribute meaningfully to the model.

#### 4.6 ARIMA(1,1,2) Model

In the ARIMA(1,1,2) model, all parameters (AR(1), MA(1), and MA(2)) are highly significant under both ML and CSS methods (p-values < 0.01). This indicates that each component contributes meaningfully to the model, making it statistically strong and well-specified.

### 4.7 ARIMA(1,1,1) Model

In the ARIMA(1,1,1) model, both AR(1) and MA(1) parameters are not statistically significant under either ML or CSS methods (p-values $\gg 0.05$). This suggests that the model does not capture meaningful relationships and may be poorly specified compared to alternatives like ARIMA(1,1,2).

### 4.8 ARIMA(2,1,2) Model

In the ARIMA(2,1,2) model, all parameters (AR1, AR2, MA1, MA2) are highly statistically significant under both ML and CSS methods (p-values < 0.001). This indicates a well-fitting model with strong explanatory power over the data.

### 4.9 ARIMA(3,1,2) Model

For both the maximum likelihood (ML) and conditional sum of squares (CSS) methods, the AR(1), AR(2), MA(1), and MA(2) coefficients are highly significant with p-values less than 0.001. However, the AR(3) coefficient is not significant in both models (p-value > 0.05 for ML and marginally significant for CSS). Thus, the AR(3) term does not appear to be a meaningful predictor, whereas the other coefficients are significant.

## 5. Model Selection

To select the best model among the fitted models, we compare various goodness-of-fit metrics including AIC, BIC, and the mean squared error (MSE) of the residuals.

| | df | AIC |
|---|---|---|
| model_312 | 6 | 5473.373 |
| model_514 | 10 | 5474.550 |
| model_212 | 5 | 5477.276 |
| model_114 | 6 | 5478.478 |
| model_112 | 4 | 5479.077 |
| model_110 | 2 | 5479.516 |
| model_011 | 2 | 5479.620 |
| model_012 | 3 | 5481.434 |
| model_111 | 3 | 5481.515 |

| | df | BIC |
|---|---|---|
| model_110 | 2 | 5485.678 |
| model_011 | 2 | 5485.783 |
| model_012 | 3 | 5490.678 |
| model_111 | 3 | 5490.760 |
| model_112 | 4 | 5491.403 |
| model_312 | 6 | 5491.861 |
| model_212 | 5 | 5492.683 |
| model_114 | 6 | 5496.967 |
| model_514 | 10 | 5505.364 |

Following our coefficient examination, we shifted to assessing model fit using established information criteria. We constructed an Akaike Information Criterion (AIC) table for all candidate models, recognizing that AIC calculations require Maximum Likelihood estimation. The ARIMA(3,1,2) specification emerged with the minimum AIC value of 5473, indicating it provides the best balance of accuracy and parsimony under this metric. Conversely, our Bayesian Information Criterion (BIC) analysis identified ARIMA(1,1,0) as the preferred model with the lowest score of 5485.67. This discrepancy between optimal models aligns with BIC's characteristic of imposing stronger penalties on additional parameters compared to AIC, naturally favoring the more streamlined ARIMA(1,1,0) formulation. This tension between different information criteria highlights the fundamental tradeoff between model complexity and goodness-of-fit that must be carefully weighed in our ultimate model selection.

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 |
|---|---|---|---|---|---|---|---|
| ARIMA(0,1,1) | 710175.5 | 5855822 | 2945637 | 2.550276 | 17.41493 | 0.2266489 | -0.007842588 |
| ARIMA(1,1,0) | 695133.5 | 5853895 | 2950675 | 2.521615 | 17.44274 | 0.2270365 | -0.012414128 |
| ARIMA(0,1,2) | 686523.8 | 5852376 | 2953577 | 2.507753 | 17.45989 | 0.2272598 | -0.014162741 |
| ARIMA(1,1,1) | 693599.2 | 5853884 | 2951392 | 2.518370 | 17.44590 | 0.2270917 | -0.013582865 |
| ARIMA(1,1,2) | 631986.9 | 5770071 | 2971860 | 2.438187 | 18.05708 | 0.2286666 | -0.042599482 |
| ARIMA(2,1,2) | 718956.8 | 5691449 | 2822234 | 2.497301 | 18.72537 | 0.2171538 | 0.002874578 |
| ARIMA(1,1,4) | 654373.5 | 5684633 | 2987766 | 2.448135 | 19.35778 | 0.2298904 | -0.014770514 |
| ARIMA(3,1,2) | 685288.7 | 5514151 | 3021679 | 2.308315 | 24.12545 | 0.2324998 | -0.013331920 |
| ARIMA(5,1,4) | 660717.9 | 5392454 | 2829687 | 2.577519 | 19.37146 | 0.2177272 | -0.014108653 |

After analyzing the information criteria, we evaluated the predictive performance of each candidate model using various error metrics. The results varied depending on the metric used. The ARIMA(5,1,4) model achieved the lowest RMSE, indicating strong performance in terms of squared prediction errors. In contrast, the ARIMA(2,1,2) model had the lowest MAE at 2,822,234, suggesting better accuracy in absolute error terms. The ARIMA(0,1,1) model performed best in terms of MAPE with a value of 17.41, while the lowest MASE of 0.217 was again observed in the ARIMA(2,1,2) model. These differences emphasize the value of using multiple metrics when determining the most suitable forecasting model for Bitcoin prices.

Considering the principle of parsimony, which prioritizes simpler models that sufficiently explain the data, our in-depth evaluation points to ARIMA(2,1,2) as the most suitable model for the Bitcoin time series. This conclusion is supported by several key factors. Firstly, all coefficients in this model are highly significant ($p < 0.001$), reflecting strong statistical reliability. Secondly, although ARIMA(5,1,4) shows slightly better AIC and RMSE scores, ARIMA(2,1,2) offers a more balanced trade-off between model fit and simplicity. Thirdly, its residuals are well-behaved, as indicated by the ACF1 value being very close to zero (0.00287), which is essential for effective forecasting. Lastly, it delivers the lowest Mean Absolute Error (2,822,234), highlighting its accuracy in predicting actual values. Taken together, these strengths in significance, simplicity, residual behavior, and predictive accuracy make **ARIMA(2,1,2)** the preferred model for forecasting Bitcoin prices.

## 6. Conclusion

This report has presented a comprehensive analysis of the Bitcoin index (USD) monthly data from August 2011 to January 2025. The key findings from this analysis are:

1. The Bitcoin index exhibits strong non-stationarity with an upward trend and increasing variance over time.
2. Log transformation and first differencing were necessary to achieve stationarity.
3. Multiple ARIMA models were identified using ACF-PACF analysis, EACF, and BIC tables.
4. Nine potential models were fitted: ARIMA(2,1,2), ARIMA(3,1,2), ARIMA(0,1,1), ARIMA(1,1,1), ARIMA(1,1,2), ARIMA(0,1,2), ARIMA (1,1,4), ARIMA(1,1,0), and ARIMA(5,1,4).
5. Based on goodness-of-fit metrics and residual diagnostics, the ARIMA(2,1,2) model was identified as the best model for the Bitcoin index data, with ARIMA(5,1,4) as a close second.

The selected ARIMA(2,1,2) model indicates that:

- The growth rate of the Bitcoin index depends on its values from the previous two months (AR components).

- There is a negative moving average component, suggesting that negative shocks in the previous period have a positive effect on the current period.
- The model effectively captures the short-term dynamics of the Bitcoin index.

This analysis provides insights into the time series behavior of the Bitcoin index and establishes a solid foundation for potential forecasting applications. However, it's important to note that cryptocurrency markets can be influenced by numerous external factors not captured in pure time series models, such as regulatory changes, technological developments, and market sentiment.

For future work, incorporating exogenous variables, exploring non-linear models, or applying machine learning techniques could potentially enhance the modeling and forecasting accuracy of the Bitcoin index.

## 7. References

https://guides.loc.gov/fintech/21st-century/cryptocurrency-blockchain

https://rmit.instructure.com/courses/140832

## 8. Appendix: R Code

```
# Load necessary libraries

library(tseries)

library(forecast)

library(ggplot2)

library(lmtest)

library(TSA)


# Import the data

bitcoin_data <- read.csv("C:/Users/Sherodhi/Downloads/assignment2Data2025.csv")


# Convert to time series object

bitcoin_ts <- ts(bitcoin_data$Bitcoin, start=c(2011, 8), frequency=12)


# Summary statistics

summary(bitcoin_ts)


# Time series plot

plot(bitcoin_ts, main="Figure 1: Bitcoin Index (USD) from August 2011 to January 2025",

    ylab="Index Value (USD)", xlab="Year", col="blue", lwd=1.5)
```

```
# Applying first lag
y= bitcoin_ts
x = zlag(bitcoin_ts)
index =2:length(x)
cor(y[index],x[index])

#Applying the secong lag

index2 =3:length(x)
cor(y[index2],x[index2])

# ACF and PACF of the original series
par(mfrow=c(1,2))
acf(bitcoin_ts, main="ACF of Original Bitcoin Index Series")
pacf(bitcoin_ts, main="PACF of Original Bitcoin Index Series")

# Augmented Dickey-Fuller Test
adf_test <- adf.test(bitcoin_ts)
print(adf_test)

# Generating the qq plot
qqnorm(bitcoin_ts, main="Normal Q-Q Plot")
qqline(bitcoin_ts, col="red")

# Applying the Shapiro Wilk Test
shapiro.test(bitcoin_ts)



# Plot log-transformed series
plot(x[index],y[index], main="Original Bitcoin Price v/s Lagged Bitcoin Price",
```

```
    ylab="Price(t)", xlab="Price(t-1)", col="blue", lwd=1.5)

# Apply log transformation
log_bitcoin_ts <- log(bitcoin_ts)

# Plot log-transformed series
plot(log_bitcoin_ts, main="Log-Transformed Bitcoin Index",
    ylab="Log(Index Value)", xlab="Year", col="blue", lwd=1.5)

# qq plot for the log transformed data
qqnorm(log_bitcoin_ts, main="Normal Q-Q Plot of the Log Transformed Bitcoin Data")
qqline(log_bitcoin_ts, col="red")
shapiro.test(log_bitcoin_ts)

# Checking stationarity of log-transformed series
adf_test_log <- adf.test(log_bitcoin_ts)
print(adf_test_log)

# First differencing of log-transformed series
diff_log_bitcoin_ts <- diff(log_bitcoin_ts, differences=1)

# Plot differenced log-transformed series
plot(diff_log_bitcoin_ts, main="First Difference of Log-Transformed Bitcoin Index",
    ylab="Diff(Log(Index Value))", xlab="Year", col="blue", lwd=1.5)

# Check stationarity of differenced log-transformed series
adf_test_diff_log <- adf.test(diff_log_bitcoin_ts)
print(adf_test_diff_log)

# PP Test
pp.test(diff_log_bitcoin_ts)
```

```
# Summary statistics of differenced log-transformed series
summary(diff_log_bitcoin_ts)

# ACF and PACF of the differenced log-transformed series
par(mfrow=c(1,2))
acf(diff_log_bitcoin_ts, main="ACF of Differenced Log-Transformed Series")
pacf(diff_log_bitcoin_ts, main="PACF of Differenced Log-Transformed Series")

# Calculate EACF
eacf_result <- eacf(diff_log_bitcoin_ts, ar.max=10, ma.max=15)

# Creating BIC Table
 options(warn=-1)
 res = armasubsets(y=diff_log_bitcoin_ts,nar=5,nma=5,y.name='p',ar.method='ols')
 plot(res)

# ARIMA(1,1,0)
 model_110 <- arima(bitcoin_ts, order=c(1,1,0), method="ML")
 coeftest(model_110)
 model_110_css <- arima(bitcoin_ts, order=c(1,1,0), method="CSS")
 coeftest(model_110_css)

# ARIMA (1,1,4)
 model_114 <- arima(bitcoin_ts, order=c(1,1,4), method="ML")
 coeftest(model_114)
 model_114_css <- arima(bitcoin_ts, order=c(1,1,4), method="CSS")
 coeftest(model_114_css)

# ARIMA(5,1,4)
 model_514 <- arima(bitcoin_ts, order=c(5,1,4), method="ML")
```

```
  coeftest(model_514)
  model_514_css <- arima(bitcoin_ts, order=c(5,1,4), method="CSS")
  coeftest(model_514_css)

# ARIMA(0,1,1)
  model_011 <- arima(bitcoin_ts, order=c(0,1,1), method="ML")
  coeftest(model_011)
  model_011_css <- arima(bitcoin_ts, order=c(0,1,1), method="CSS")
  coeftest(model_011_css)

# ARIMA(0,1,2)
  model_012 <- arima(bitcoin_ts, order=c(0,1,2), method="ML")
  coeftest(model_012)
  model_012_css <- arima(bitcoin_ts, order=c(0,1,2), method="CSS")
  coeftest(model_012_css)


# ARIMA(1,1,2)
  model_112 <- arima(bitcoin_ts, order=c(1,1,2), method="ML")
  coeftest(model_112)
  model_112_css <- arima(bitcoin_ts, order=c(1,1,2), method="CSS")
  coeftest(model_112_css)

# ARIMA(1,1,1)
  model_111 <- arima(bitcoin_ts, order=c(1,1,1), method="ML")
  coeftest(model_111)
  model_111_css <- arima(bitcoin_ts, order=c(1,1,1), method="CSS")
  coeftest(model_111_css)

# ARIMA(2,1,2)
  model_212 <- arima(bitcoin_ts, order=c(2,1,2), method="ML")
```

```
 coeftest(model_212)
 model_212_css <- arima(bitcoin_ts, order=c(2,1,2), method="CSS")
 coeftest(model_212_css)

# ARIMA(3,1,2)
 model_312 <- arima(bitcoin_ts, order=c(3,1,2), method="ML")
 coeftest(model_312)
 model_312_css <- arima(bitcoin_ts, order=c(3,1,2), method="CSS")
 coeftest(model_312_css)

#Sorting the score
  sort.score <- function(x, score = c("bic", "aic")){
   if (score == "aic"){
    x[with(x, order(AIC)),]
   } else if (score == "bic") {
    x[with(x, order(BIC)),]
   } else {
    warning('score = "x" only accepts valid arguments ("aic","bic")')
   }
  }

  sort.score(AIC(model_011, model_110, model_012, model_111,
                 model_112, model_212, model_114, model_312, model_514),
             score = "aic")
  sort.score(BIC(model_011, model_110, model_012, model_111,
                 model_112, model_212, model_114, model_312, model_514),
             score = "bic")

 # Fit ARIMA models
  model_011AR <- Arima(bitcoin_ts, order=c(0,1,1), method='ML')
  model_110AR <- Arima(bitcoin_ts, order=c(1,1,0), method='ML')
```

```
model_012AR <- Arima(bitcoin_ts, order=c(0,1,2), method='ML')
model_111AR <- Arima(bitcoin_ts, order=c(1,1,1), method='ML')
model_112AR <- Arima(bitcoin_ts, order=c(1,1,2), method='ML')
model_212AR <- Arima(bitcoin_ts, order=c(2,1,2), method='ML')
model_114AR <- Arima(bitcoin_ts, order=c(1,1,4), method='ML')
model_312AR <- Arima(bitcoin_ts, order=c(3,1,2), method='ML')
model_514AR <- Arima(bitcoin_ts, order=c(5,1,4), method='ML')

Smodel_011AR <- accuracy(model_011AR)[1:7]
Smodel_110AR <- accuracy(model_110AR)[1:7]
Smodel_012AR <- accuracy(model_012AR)[1:7]
Smodel_111AR <- accuracy(model_111AR)[1:7]
Smodel_112AR <- accuracy(model_112AR)[1:7]
Smodel_212AR <- accuracy(model_212AR)[1:7]
Smodel_114AR <- accuracy(model_114AR)[1:7]
Smodel_312AR <- accuracy(model_312AR)[1:7]
Smodel_514AR <- accuracy(model_514AR)[1:7]

Smodels <- data.frame(
  rbind(Smodel_011AR, Smodel_110AR, Smodel_012AR,
        Smodel_111AR, Smodel_112AR, Smodel_212AR,
        Smodel_114AR, Smodel_312AR, Smodel_514AR)
)
colnames(Smodels) <- c("ME", "RMSE", "MAE", "MPE", "MAPE", "MASE", "ACF1")
rownames(Smodels) <- c("ARIMA(0,1,1)", "ARIMA(1,1,0)", "ARIMA(0,1,2)",
                       "ARIMA(1,1,1)", "ARIMA(1,1,2)", "ARIMA(2,1,2)",
                       "ARIMA(1,1,4)", "ARIMA(3,1,2)","ARIMA(5,1,4)")
print(Smodels)
```