Time Series Forecasting of Bitcoin Prices using ARIMA Models

(ADVANCED ANALYSIS)

Project Title: *Time Series Forecasting of Bitcoin Prices using ARIMA Models*

Course: MATH1318- Time Series Analysis

Student ID: S4035731

Institution: RMIT University

Date: 26 April 2025

Table of Contents

Introduction

The rapid evolution of digital currencies, particularly Bitcoin, has brought forth immense opportunities and challenges in understanding and forecasting their behaviour. This project focuses on analysing the monthly Bitcoin price index from August 2011 to January 2025 using advanced time series techniques. The primary objective is to develop an accurate and interpretable forecasting model to predict future Bitcoin prices, while also demonstrating a robust understanding of time series modelling.

The project offers a comprehensive walkthrough of essential concepts in Time Series Analysis, with particular emphasis on the ARIMA (Autoregressive Integrated Moving Average) framework. The modelling process includes exploratory analysis, stationarity testing, transformation techniques, and residual diagnostics.

A distinguishing feature of this work is the depth of model identification. Rather than relying solely on default auto-selection methods, this analysis carefully explores multiple components—including **ACF**, **PACF**, **EACF matrix**, **BIC-based subset plots**, and **Box-Cox transformation**—to accurately identify the optimal ARIMA(p,d,q) configuration.

The project is structured to reflect a real-world analytical pipeline, offering not only predictive insight into the volatile nature of Bitcoin but also serving as a demonstrative example of effective time series methodology.

Objectives

This report delivers a comprehensive and in-depth time series analysis of Bitcoin prices from August 2011 to January 2025, with the primary goal of developing a statistically sound and reliable ARIMA model for accurate forecasting and insight extraction.

Transforming and preparing the dataset into a structured **time series format**

Conducting **visual and statistical checks** for **stationarity**

Applying various transformations (**log**, **Box-Cox**, **differencing**) to stabilize variance and trend

Using **detailed techniques to identify optimal p, d, q values**, including **ACF**, **PACF**, **EACF**, and **BIC plots**

Evaluating multiple **ARIMA models** based on **AIC**, **BIC**, and **MSE**

Performing thorough **diagnostic checks** and generating a **12-month forecast**

Data Preparation

Data preparation in time series involves organizing raw chronological data into a structured format suitable for analysis, ensuring consistency, completeness, and proper time indexing. In this project, it included loading the Bitcoin dataset, checking for missing values, summarizing statistics, and converting the data into a monthly timeseries_() starting from August 2011.

Creating the Time Series Object

To begin the analysis, the monthly Bitcoin price data from August 2011 to January 2025 was converted into a time series object using the ts() function in R, with a frequency of 12 to reflect monthly observations.

```
# Create a time series object
# Assuming monthly data, starting from August 2011
bitcoin_ts <- ts(bitcoin_analysis_data$Bitcoin,
                 start = c(2011, 8),
                 frequency = 12)

bitcoin_ts
```

	Jan	Feb	Mar	Apr	May
2011					
2012	6625.0	6237.5	6125.0	6250.0	6462.5
2013	25575.0	41912.5	120187.5	174850.0	159887.5
2014	1003750.0	689862.5	568537.5	561062.5	784750.0
2015	271125.0	316837.5	305300.0	295250.0	286137.5
2016	460612.5	544037.5	518325.0	560462.5	664800.0
2017	1204987.5	1489012.5	1337887.5	1687762.5	2872512.5

This includes key statistics such as the **minimum**, **maximum**, **median**, **mean**, and **quartiles**, offering insight into the overall scale and spread of the Bitcoin index over the selected period.

Summary Statistics.

This includes key statistics such as the **minimum**, **maximum**, **median**, **mean**, and **quartiles**, offering insight into the overall scale and spread of the Bitcoin index over the selected period.

```
    Date              Bitcoin
Length:162        Min.   :     3988
Class :character  1st Qu.:   470981
Mode  :character  Median :  7930612
                  Mean   : 19781743
                  3rd Qu.: 31599541
                  Max.   :128015000
```

Observations from Summary Statistics

The dataset contains **162 monthly observations** of Bitcoin prices.

The **minimum price** recorded is **$3,988**, while the **maximum** reaches a significant **$128,015,000**, indicating **extremely high volatility** over time.

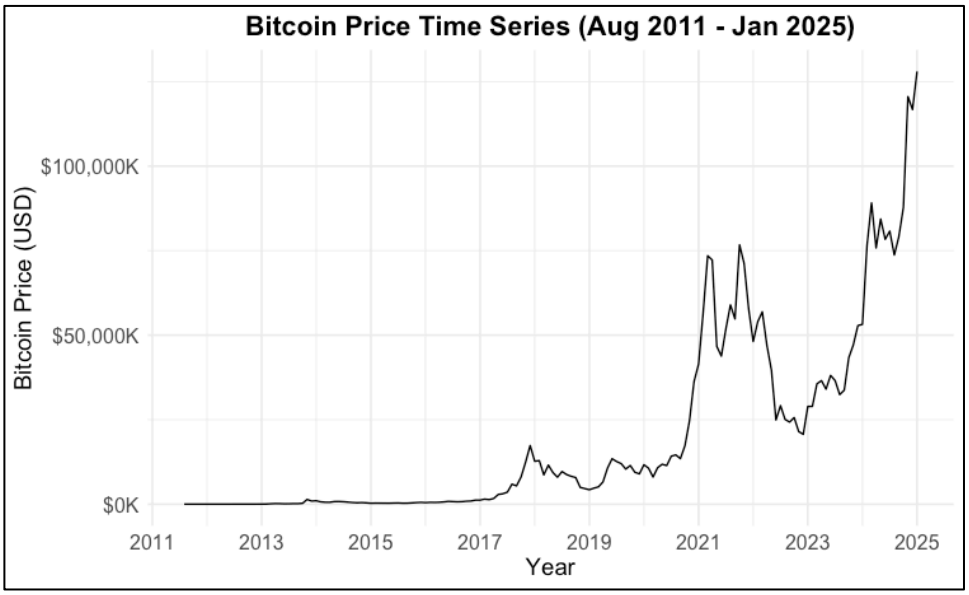
The **median price** is **$7,930,612**, and the **mean** is **$19,781,743**, showing a **right-skewed distribution** (mean > median) due to high price spikes.

The **interquartile range (IQR)**, from **$470,981 (Q1)** to **$31,599,541 (Q3)**, further confirms the **wide spread and increasing trend** in the series.

The Date column is currently stored as **character type**, which was handled during time series conversion.

Descriptive Analysis

This section explores the overall structure of Bitcoin price data through visual tools like line plots, histograms, and LOESS smoothing. A **Q-Q plot** is also used to assess the normality of residuals from a basic trend model. These insights help identify trends, volatility, and distributional characteristics, guiding later transformation and modelling steps.



1. Time Series Plot

The Bitcoin Index series from **August 2011 to January 2025**, with a total of **162 monthly data points**, is visualized below to uncover key characteristics of the raw data:

Trend: The plot reveals a clear **upward non-linear trend**, especially with sharp increases from **late 2020 onwards**, continuing strongly through **2024**. This reflects major market activity influencing price surges.

Seasonality: No evident **repeating seasonal pattern** can be observed across years, suggesting Bitcoin prices are driven more by external shocks than seasonal behaviour.

Variance: The **variance increases over time**, showing **heteroscedasticity**—with low early values and extreme volatility post-2017. This supports the need for **log or Box-Cox transformation**.

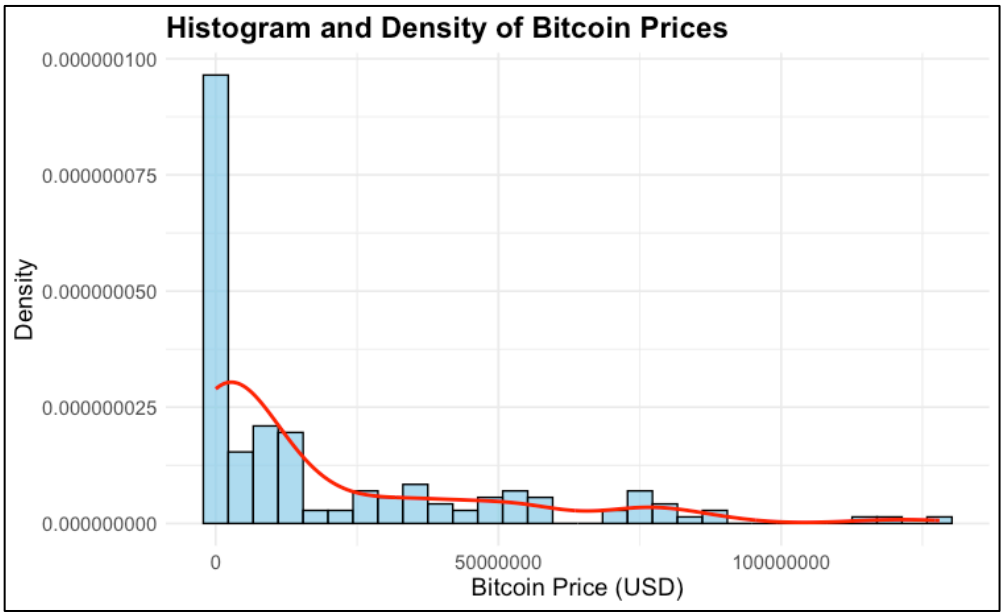
Outliers & Breaks: There are several **sharp jumps and crashes**, particularly around **2017, 2021, and 2022**, likely caused by major economic events or regulatory news.

Distribution: The data is **highly right-skewed**, as evident from the extreme maximum value ($128M) compared to the median and mean. This confirms the presence of long upper tails.

These patterns confirm that the original Bitcoin series is **non-stationary in both trend and variance**, warranting **transformation and differencing** for effective ARIMA modelling.

2.Histogram & Density Plot

The **Histogram and Density Plot** visually examines the distribution of Bitcoin prices. It reveals a right-skewed shape, suggesting non-normality and variance instability—supporting the need for transformation before modelling.



The **Histogram and Density Plot** of the Bitcoin price data reveals a highly **right-skewed distribution**, indicating that most values are concentrated on the lower end, with a long tail extending toward very high prices. This skewness reflects the rapid growth and volatility of Bitcoin, particularly in recent years.

Key observations:

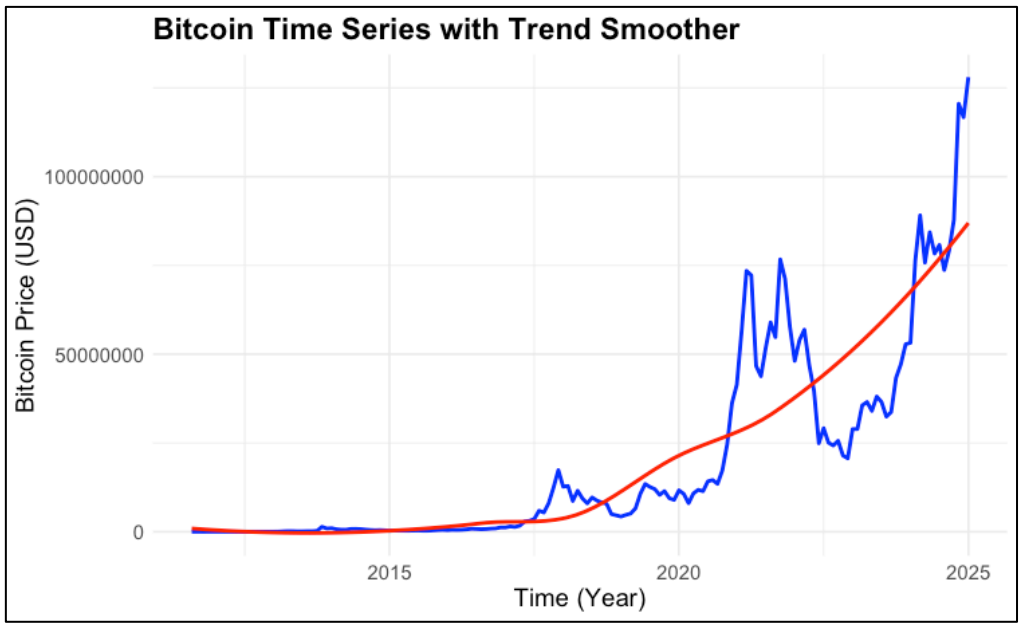
Skewness: The distribution is not symmetric, suggesting non-normality.

Outliers: Extreme values at the upper end highlight sharp price spikes.

Transformation Need: This supports applying transformations (like log or Box-Cox) to stabilize variance and improve model performance in ARIMA.

3. Trend Smoothing (LOESS)

To better understand the underlying trend in the Bitcoin time series, **LOESS (Locally Estimated Scatterplot Smoothing)** is applied. This technique fits simple models to localized subsets of the data, allowing us to observe the general trajectory without short-term noise.



The LOESS trend smoothing visually separates the long-term trend from short-term fluctuations in the Bitcoin time series.

Observations:

The **blue line** represents actual Bitcoin prices, showing **sharp volatility** in recent years.

The **red LOESS curve** clearly captures a **non-linear, accelerating upward trend**, particularly after 2017.

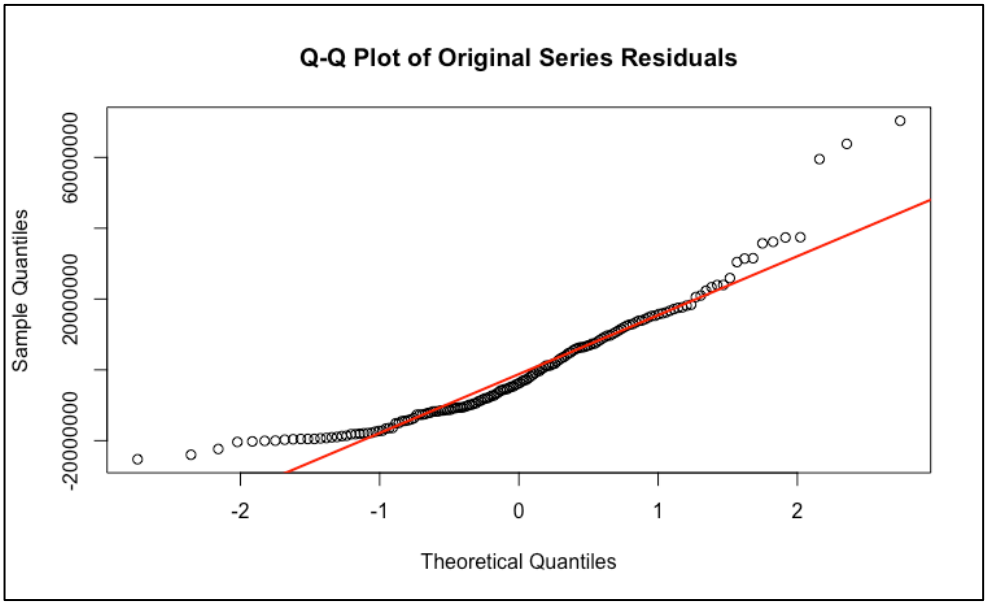
The trend grows **steeply from 2020 onwards**, reflecting major price surges in 2021 and 2024.

The smoother confirms the **non-stationary nature** of the data and the need for transformation and differencing before ARIMA modelling.

4. Q-Q Plot of Residual

The **Q-Q (Quantile-Quantile) plot** is a graphical tool used to assess whether the residuals of a time series follow a **normal distribution**—an important assumption for many statistical modelling techniques, including ARIMA.

In this analysis, a **linear model was initially fitted** to the Bitcoin time series, and the residuals were extracted to generate the Q-Q plot.



Observations

The residuals **deviate substantially from the red reference line**, especially at the tails (both left and right extremes), indicating **departure from normality**.

The **right tail rises sharply above the line**, suggesting **positive skewness** (i.e., extreme high values in the series).

The **left tail dips below the line**, confirming **heavy-tailed behaviour** and the presence of **outliers or structural breaks** in the data.

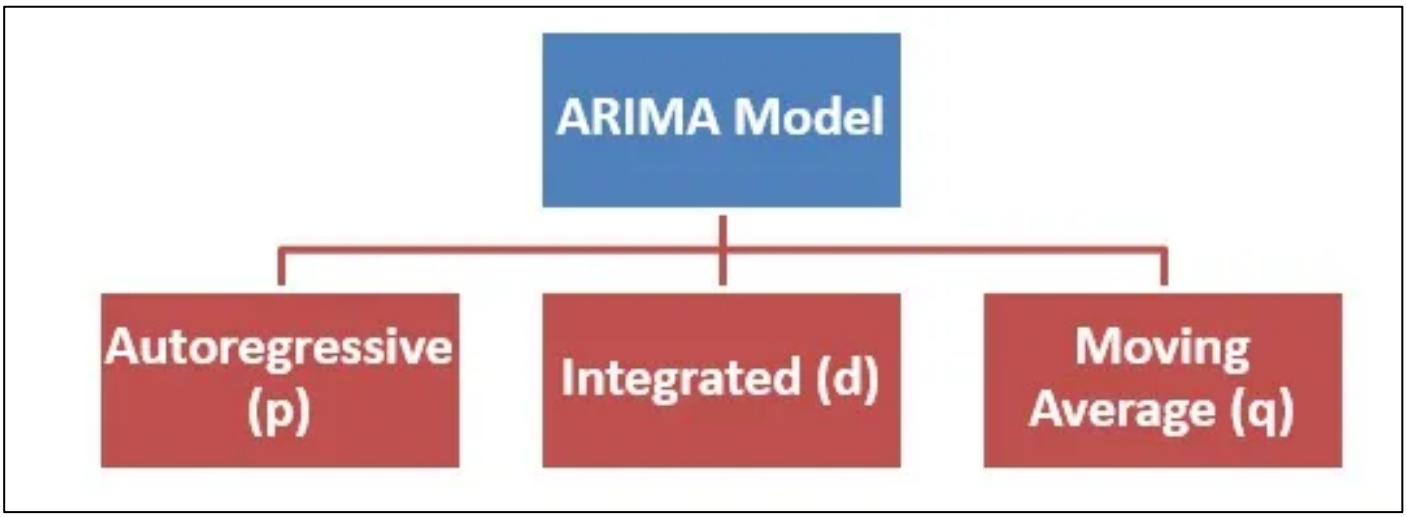
This non-linear spread of residuals shows that the **underlying distribution is not Gaussian**, which violates the assumptions of standard ARIMA modelling.

Therefore, these results **justify the need for transformation techniques**, such as **log or Box-Cox transformations**, to improve normality and stabilize variance in preparation for model fitting.

Conclusion of Descriptive Analysis

The descriptive analysis of the Bitcoin time series reveals a **strong upward trend**, **high variance**, and **non-normal distribution**, with no clear seasonal patterns. The presence of **structural breaks, outliers**, and a **right-skewed distribution** suggests the series is **non-stationary** in both mean and variance. These characteristics highlight the necessity for **transformations and differencing** before applying ARIMA modelling. Additionally, normality checks through histogram and Q-Q plots further confirm deviations from Gaussian assumptions, supporting the need for corrective preprocessing steps in the subsequent stages of analysis.

Detailed Techniques for Identifying Optimal ARIMA (p, d, q) Parameters



Accurate specification of the ARIMA model requires careful selection of the three key parameters:

p (autoregressive order),

d (degree of differencing),

q (moving average order).

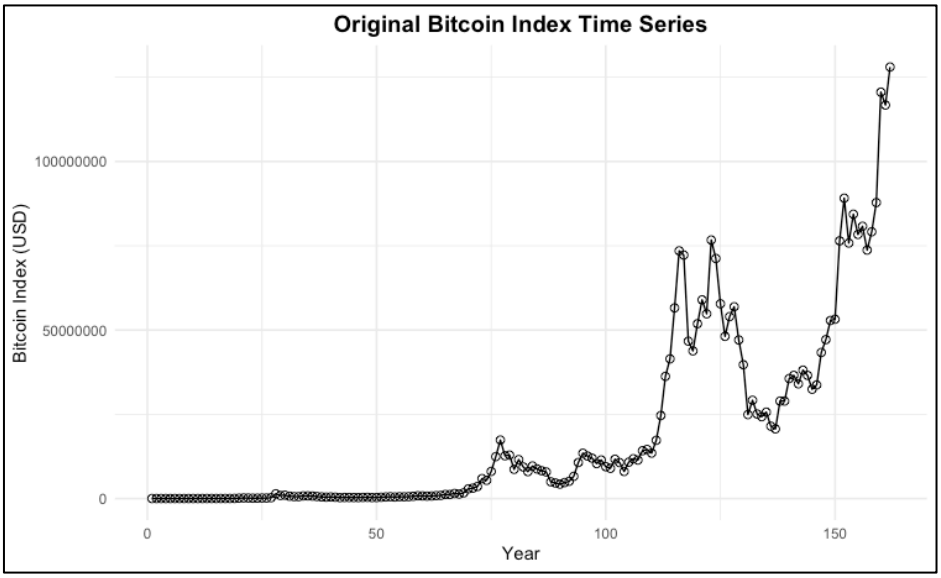
=This section outlines the detailed, multi-step approach used in this project to identify these parameters.

Stationarity Check

Time series models like ARIMA assume that the underlying data is **stationary**, meaning its statistical properties such as mean and variance remain constant over time. This section examines the stationarity of the Bitcoin price series through both **visual inspection** and **formal statistical tests**.

Importantly, this step also helps determine the appropriate <u>d value</u> (the number of differences needed to make the series stationary) for the ARIMA model.

Original Graph



This time series plot visualizes the monthly Bitcoin price index from August 2011 to January 2025. It highlights key price points and reveals a long-term upward trend with increasing volatility over time.

<u>Understanding ACF and PACF</u>

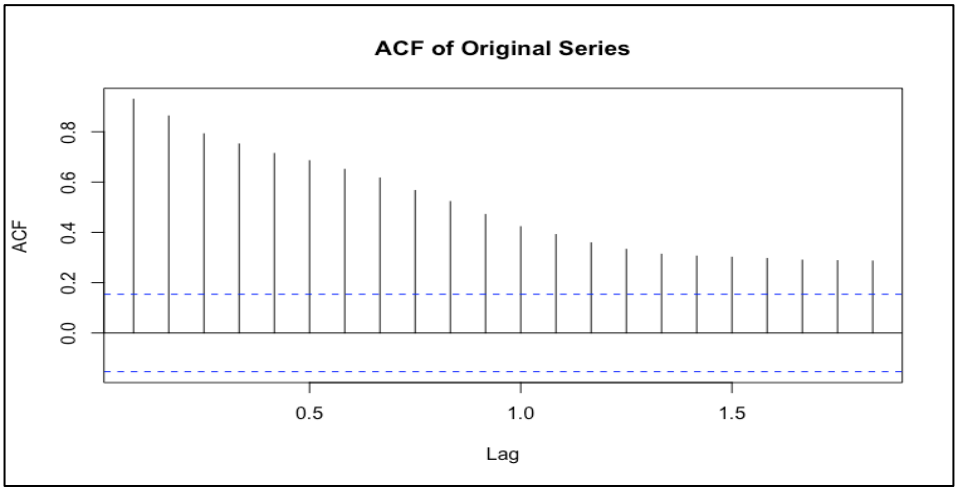
In time series analysis, **ACF (Autocorrelation Function)** and **PACF (Partial Autocorrelation Function)** are fundamental tools used to understand the internal structure of the data:

ACF helps measure the strength of correlation between an observation and its lagged values. It is especially useful in identifying the number of **MA (Moving Average)** terms in an ARIMA model.

PACF determines the amount of correlation between a time series and its lagged values **after removing** the effect of shorter lags. It is primarily used to identify the number of **AR (Autoregressive)** terms.

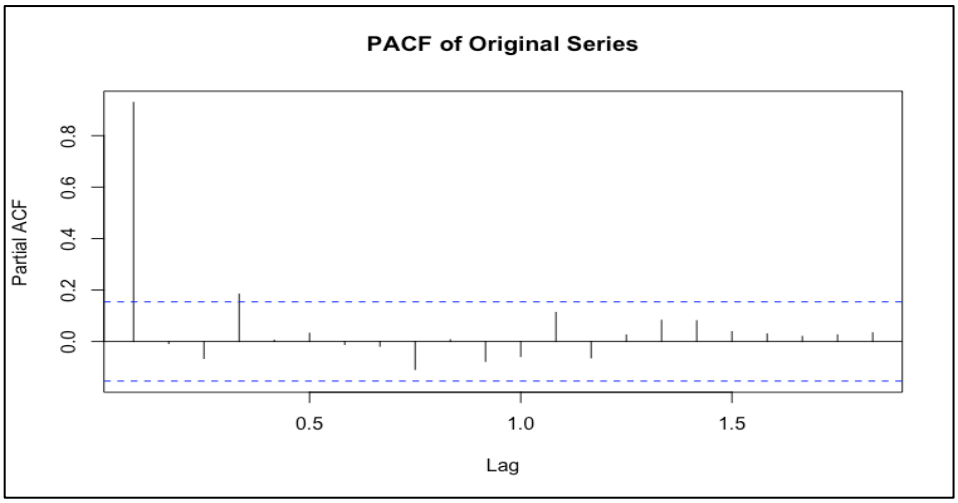
ACF and PACF Plots of the Original Series

ACF and PACF were applied to the original Bitcoin price series to analyse lag-based correlations and identify patterns in the data before differencing.



Autocorrelation Function (ACF)

The **Autocorrelation Function (ACF)** plot of the original Bitcoin series reveals a **gradual and sustained decay**, which indicates a strong correlation between current and past values across many lags. This persistence highlights the presence of a **long-term trend** and **non-stationarity** in the data. The slow tapering pattern implies that the series requires **transformation or differencing** to achieve stationarity before ARIMA modelling.



Partial Autocorrelation Function (PACF)

The **Partial Autocorrelation Function (PACF)** plot displays a **sharp spike at lag 1**, with subsequent values dropping close to zero. This suggests that the current Bitcoin price is primarily influenced by its immediate past value, consistent with an **AR(1) process**. The clean cut-off supports the use of autoregressive terms when constructing the ARIMA model.

Stationarity Testing Using ADF and KPSS

To assess whether the Bitcoin price series is **stationary**, we applied two complementary statistical tests: the **Augmented Dickey-Fuller (ADF)** test and the **KPSS** (Kwiatkowski-Phillips-Schmidt-Shin) test. These tests help determine the presence of a **unit root** and overall stationarity — a critical factor in ARIMA modelling. This step also aids in identifying the appropriate **d (differencing)** value for model specification.

```
        Augmented Dickey-Fuller Test

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

Augmented Dickey-Fuller (ADF) Test

Null Hypothesis (H_0): The series has a unit root (non-stationary)
Alternative Hypothesis (H_1): The series is stationary.

Result Summary: Test Statistic: -0.27056 | **Lag Order:** 5 | **p-value:** 0.99

Interpretation:
The **p-value is much greater than 0.05**, indicating we **fail to reject the null hypothesis**.
This means the **series is non-stationary** — it shows trends or time-dependent structure in its mean.
The high p-value suggests that the observed test statistic is **highly likely** under the assumption of non-stationarity.
Therefore, **transformation or differencing** is necessary to make the series suitable for ARIMA modelling.

```
        KPSS Test for Level Stationarity

data:  bitcoin_ts
KPSS Level = 2.3395, Truncation lag parameter = 4, p-value = 0.01
```

KPSS (Kwiatkowski-Phillips-Schmidt-Shin) Test

Null Hypothesis (H_0): The series is **stationary** (around a constant or a trend).
Alternative Hypothesis (H_1): The series is **non-stationary**.

Interpretation:
The **p-value is less than 0.05**, so we **reject the null hypothesis**.
This implies that the **series is not stationary**.
The test statistic (2.3395) is quite high, further supporting the rejection of stationarity.

Conclusion: The KPSS test also confirms that the Bitcoin time series is **non-stationary**, aligning with the ADF test.

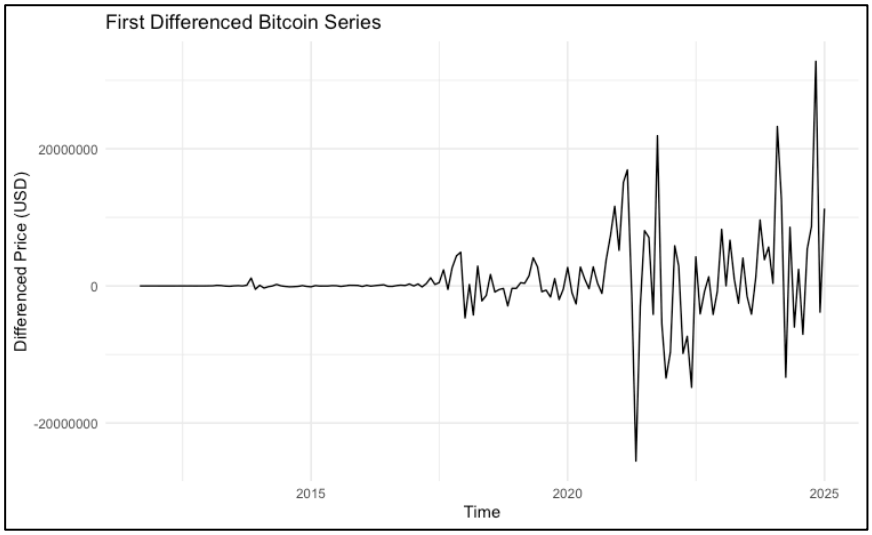
Action Required: Differencing or transformation is necessary before proceeding with ARIMA modelling.

First Differencing to Achieve Stationarity and Re-evaluation with ADF, KPSS, ACF, and PACF

```
# First differencing to achieve stationarity
diff_bitcoin_ts <- diff(bitcoin_ts)
```

This transformation subtracts each value in the series from its previous value, effectively removing trends and stabilizing the mean. As a result, diff_bitcoin_ts represents the differenced Bitcoin series, which is more suitable for time series modelling using ARIMA.

Plot of Differenced Series



Observation

The plot of the first differenced Bitcoin series depicts the **change in price** from one month to the next. This transformation removes the long-term upward trend seen in the original series, resulting in a more **stationary appearance**. The values now fluctuate closely around zero, indicating **mean-reverting behaviour**.

Stationary Check via ADF-Test & KPSS-Test

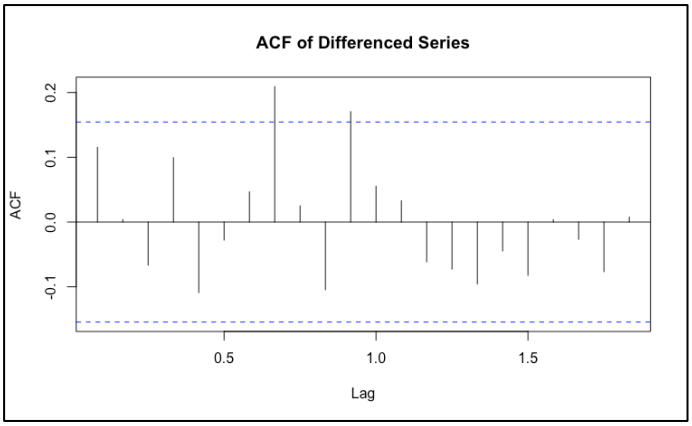
```
      KPSS Test for Level Stationarity

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

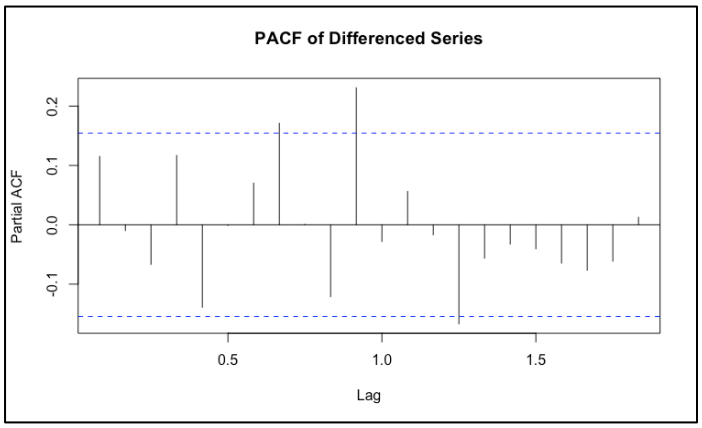
```
        Augmented Dickey-Fuller Test

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

ADF Test: p-value < 0.01 → Rejects the null hypothesis → The differenced series is **stationary**.

KPSS Test: p-value ≈ 0.08 → Fails to reject the null hypothesis → Confirms the series is **stationary**.

Stationary check using ACF and PACF





After applying first-order differencing to the Bitcoin time series:

ACF Plot: The autocorrelations now drop off quickly and oscillate around zero, indicating the removal of trend and improved stationarity.
PACF Plot: The partial autocorrelations also display a more random pattern with few significant spikes, further supporting that the series is now approximately stationary.

Determining the Value of d (Order of Differencing)

In ARIMA modelling, the parameter **d** represents the number of times the original time series needs to be **differenced** to achieve **stationarity**—a key requirement for effective time series forecasting.

Initial Analysis of the raw Bitcoin series revealed clear non-stationary behaviour, with a strong upward trend and slow decay in the ACF.

To address this, **first-order differencing** (diff(bitcoin_ts)) was applied, producing a series that showed consistent mean and variance across time.

Stationarity Confirmation:

The **Augmented Dickey-Fuller (ADF) test** returned a **p-value = 0.01**, indicating that the differenced series is stationary.

The **KPSS test** produced a **p-value > 0.05**, which also supports the stationarity of the differenced series.

Additionally, **ACF and PACF plots** of the differenced series confirmed a stable structure with no lingering trends

$$d = 1$$

Based on statistical tests and visual diagnostics, the value of **d = 1** is appropriate for the ARIMA model

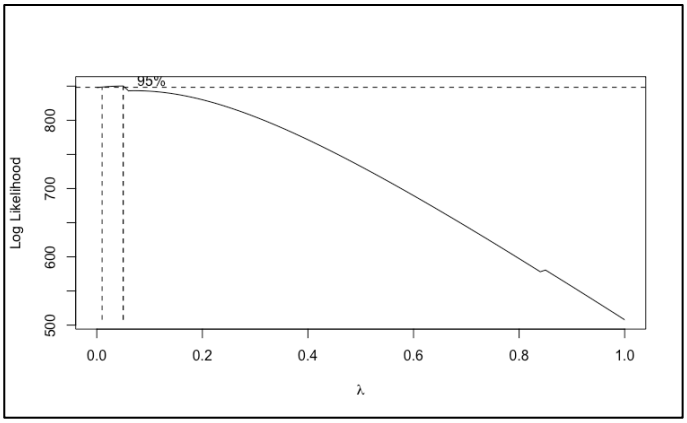
Box-Cox Transformation

The **Box-Cox transformation** is a powerful variance-stabilizing technique used in time series analysis, particularly when the data exhibits heteroscedasticity—i.e., non-constant variance over time. In this analysis, the transformation was applied to the original Bitcoin price series to reduce skewness and achieve a more homoscedastic structure. The optimal transformation parameter, **λ (lambda)**, was estimated using the BoxCox.lambda() function. After applying the transformation, the resulting series was examined both visually and statistically to assess improvements in stationarity. While the transformation effectively stabilized the variance, further differencing was still required to remove the underlying trend and achieve full stationarity.

```
# Box-Cox Transformation for Non-Stationary Data
BC <- BoxCox.ar(bitcoin_ts, lambda = seq(0, 1, 0.01))
BC$ci

# Extract optimal lambda value
lambda <- BC$lambda[which.max(BC$loglike)]
print(lambda)

# Apply Box-Cox transformation manually
bitcoin_bc <- (bitcoin_ts^lambda - 1) / lambda
```

- **BoxCox.ar()** tests multiple lambda values (from 0 to 1) to determine the best Box-Cox transformation for stabilizing variance in the Bitcoin time series.
- **Lambda (λ)** controls the power of the transformation — for example, λ = 0 implies log transformation, while λ = 1 keeps the data unchanged.
- **BC$lambda[which.max(BC$loglike)]** identifies the λ value that maximizes log-likelihood, indicating the most suitable transformation.
- The selected **lambda** is then used to transform the data, making it more stationary and reducing skewness before ARIMA modelling.

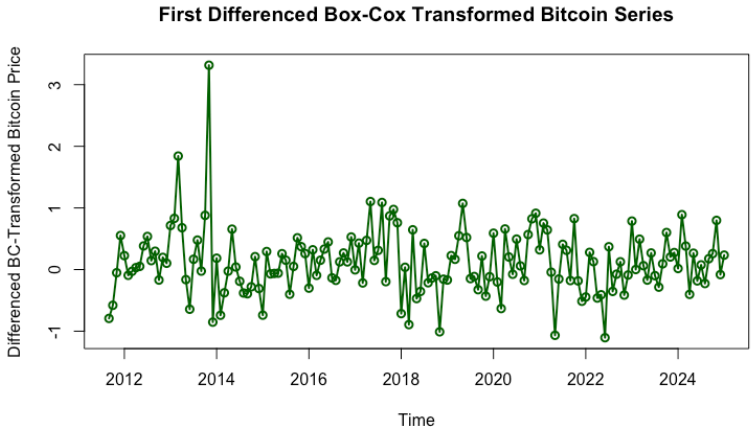
Box-Cox Log-Likelihood Plot for Optimal Lambda Estimation



The **peak point on the curve** indicates the **optimal λ value**, where the log-likelihood is maximized — meaning this λ offers the best variance stabilization.

The **dashed vertical lines** represent the 95% confidence interval for λ, and the **horizontal dashed line** marks the maximum log-likelihood.

Differenced Box-Cox Transformed Series

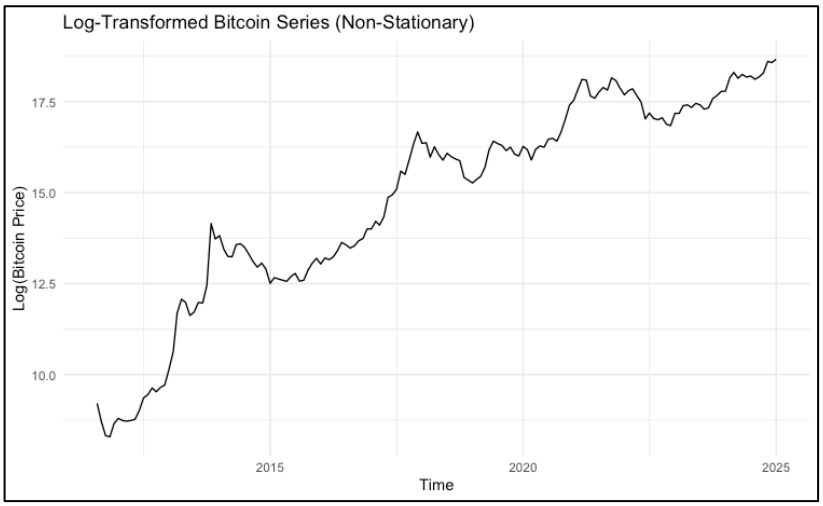


The plot shows the Bitcoin series after applying Box-Cox transformation and first differencing. Variance appears stable and the series fluctuates around a constant mean, confirming that the data is now stationary and ready for ARIMA modelling.

Log Transformation

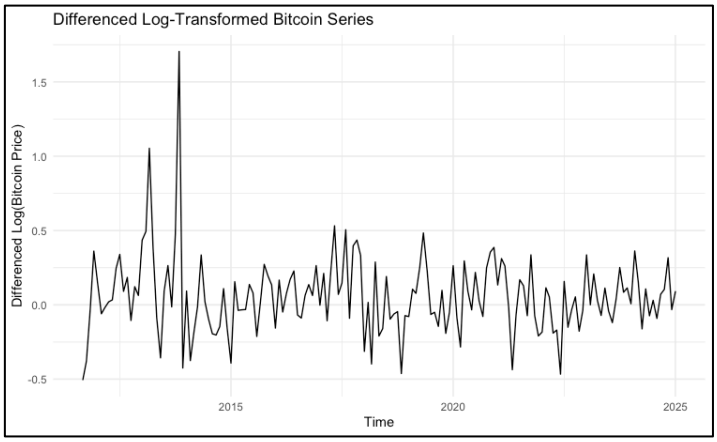
The log transformation was applied to the original Bitcoin time series to reduce exponential growth patterns and stabilize the variance. This is particularly useful when data exhibits multiplicative effects or large-scale fluctuations over time. By transforming the data using the natural logarithm, the series becomes more linear and better suited for modelling with ARIMA.

Log-Transformed Bitcoin Series (Non-Stationary)



The log-transformed Bitcoin time series shows a clear **upward trend**, indicating **non-stationarity** in the mean. While the transformation compresses extreme values and reduces variance to some extent, it does **not fully stabilize the trend** or variance. Therefore, further preprocessing (like differencing) is needed before ARIMA modelling.

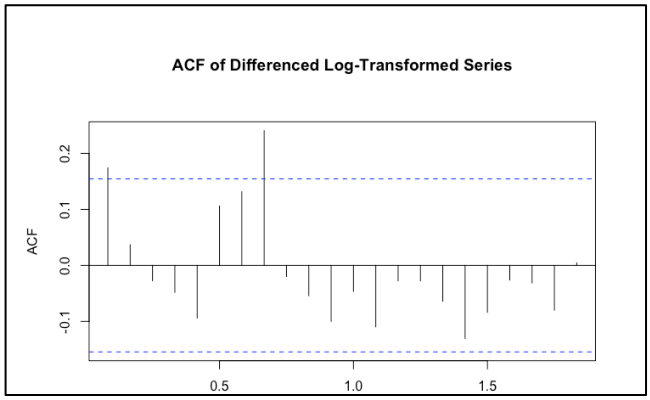
Differenced Log-Transformed Bitcoin Series

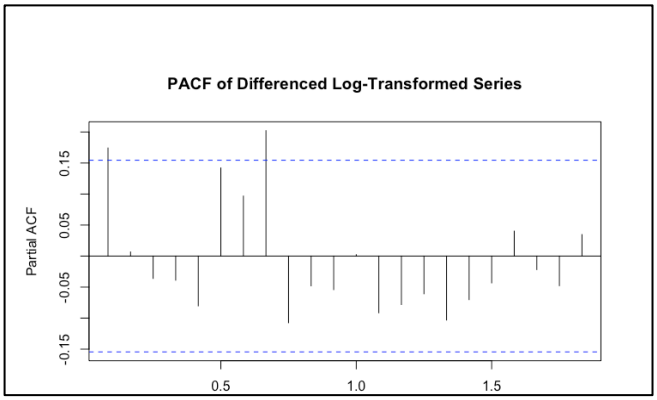


After differencing the log-transformed series, the resulting plot appears **visually stationary**. The variance is more stable, and the trend has been removed. This transformation prepares the series for ARIMA modelling by meeting the assumption of stationarity.

Identification of AR (p) and MA (q) Orders via ACF and PACF of Log-Differenced Series

To determine the optimal values of **p** (autoregressive order) and **q** (moving average order), we examined the **ACF** and **PACF** plots of the **log-differenced Bitcoin series**. This transformation stabilizes both trend and variance, allowing the identification of significant lags. Peaks in the **PACF** indicate potential values for **p**, while those in the **ACF** suggest suitable values for **q**. This analysis provides a strong foundation for selecting candidate ARIMA models.





ACF Plot (for MA(q)):
The ACF shows a sharp cutoff after lag **2**, followed by a gradual decay.
Interpretation: This suggests that the Moving Average component (**q**) could be in the range of **1 to 2**.

Assuming q = [1:2]

PACF Plot (for AR(p)):
The PACF displays significant spikes at **lags 1 and 2**, then tapers off.
Interpretation: This indicates that the Autoregressive component (**p**) could also lie in the range of **1 to 2**.

Assuming p = [1:2]

Model Identification Using EACF and BIC Subset Selection

Extended Autocorrelation Function (EACF)

The **Extended Autocorrelation Function (EACF)** is a diagnostic tool used in **time series analysis** to help identify the appropriate **ARIMA(p, d, q)** model. It extends the logic of traditional ACF and PACF by examining combinations of autoregressive (AR) and moving average (MA) terms simultaneously.

```
$symbol
  0   1   2   3   4   5   6   7   8   9   10  11  12  13
0 "x" "o" "o" "o" "o" "o" "o" "x" "o" "o" "o" "o" "o" "o"
1 "o" "o" "o" "o" "o" "o" "o" "x" "o" "o" "o" "o" "o" "o"
2 "x" "o" "o" "o" "o" "o" "o" "x" "o" "o" "o" "o" "o" "o"
3 "x" "o" "o" "o" "o" "o" "o" "x" "o" "o" "o" "o" "o" "o"
4 "x" "o" "x" "o" "o" "o" "o" "x" "o" "o" "o" "o" "o" "o"
5 "x" "x" "x" "o" "x" "o" "o" "x" "o" "o" "o" "o" "o" "o"
6 "x" "x" "x" "x" "x" "o" "o" "o" "o" "o" "o" "o" "o" "o"
7 "x" "x" "x" "o" "x" "x" "o" "x" "o" "o" "o" "o" "o" "o"
```

Understanding the EACF Table

“o” indicates non-significance: In the matrix, “o” values represent points where the autocorrelation is not statistically significant. A triangle of “o” values suggests a good region for identifying ARMA(p, q) models.

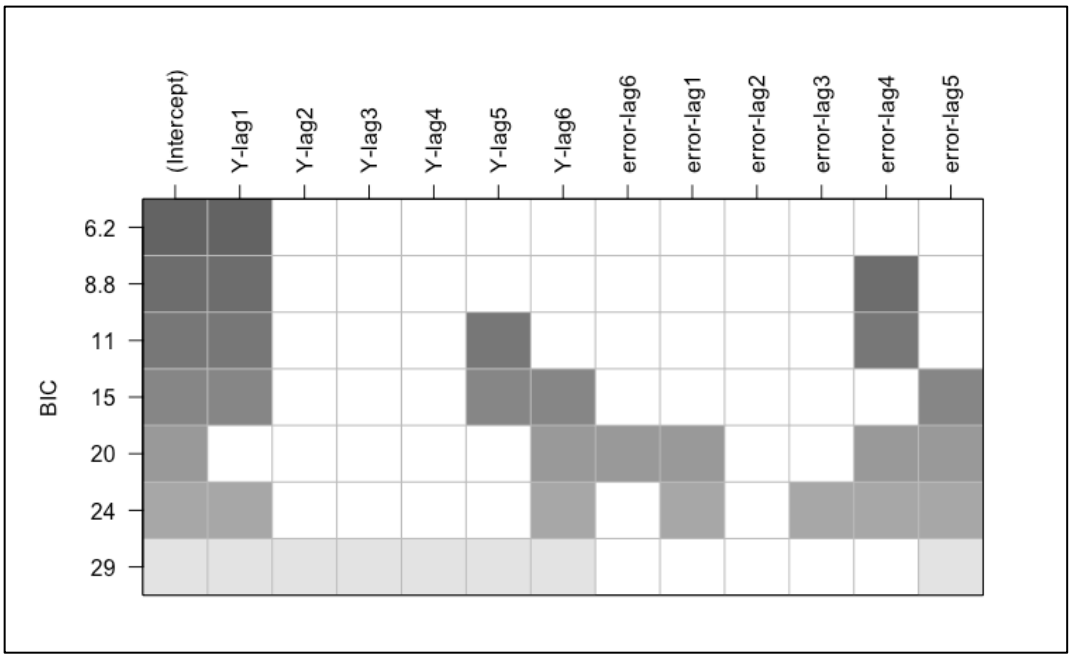
Triangle of “o”s helps select (p, q): Start from the top-left of the table and look for the smallest triangle of consecutive “o”s. The row number gives the **AR (p)** order, and the column number gives the **MA (q)** order.

$$p = [0:3]$$

$$q = [1:3]$$

Bayesian Information Criterion (BIC)

The **Bayesian Information Criterion (BIC)** is used to compare different ARIMA models. A lower BIC value indicates a better model fit with fewer parameters, helping to avoid overfitting.



Axes: The **rows** represent combinations of AR (p) and MA (q) terms.

The **columns** represent lag terms from the autoregressive (Y-lag) and moving average (error-lag) components.

Darker cells indicate **lower BIC values**, which means better model fit with less complexity.

Observation:

The BIC subset plot shows that the **lowest BIC values** appear around the combination of **AR lags = 1 and 5**, and **MA lags = 1 and 4**, as indicated by the darkest regions.

Based on the shading, an **ARIMA model with p = 1 or 5 and q = 1 or 4** appears most appropriate for further model fitting.

Model Selection & Evaluation

Based on differencing (d = 1), and visual/diagnostic tools, the following combinations are considered viable:

ARIMA(1,1,1) , **ARIMA(1,1,2), ARIMA(1,1,4), ARIMA(2,1,2)** ,**ARIMA(2,1,3)**

Model Evaluation Using AIC, BIC, and MSE

To determine the most suitable ARIMA model for forecasting the log-transformed Bitcoin series, we fitted a range of ARIMA models using different combinations of (p,d,q) values identified through ACF, PACF, EACF, and BIC-based selection techniques. For each fitted model, we calculated three important evaluation metrics:

AIC (Akaike Information Criterion) – assesses model fit while penalizing complexity.

BIC (Bayesian Information Criterion) – similar to AIC but with a stronger penalty for more parameters.

MSE (Mean Squared Error) – measures the average squared difference between observed and predicted values.

	Model	AIC	BIC	MSE
1	ARIMA(1,1,1)	31.70871	40.95292	0.06824073
2	ARIMA(1,1,2)	33.67623	46.00185	0.06822671
3	ARIMA(1,1,4)	37.62928	56.11771	0.06820586
4	ARIMA(2,1,2)	29.90185	45.30887	0.06554740
5	ARIMA(2,1,3)	29.99613	48.48455	0.06463098

Justification for Best ARIMA Model Selection

To identify the best-fitting ARIMA model, we evaluated five candidate models using three key metrics: **Akaike Information Criterion (AIC)**, **Bayesian Information Criterion (BIC)**, and **Mean Squared Error (MSE)**.

ARIMA(2,1,3) showed the **lowest MSE (0.0646)**, which suggests strong predictive accuracy.

However, **ARIMA(2,1,2)** achieved the **lowest AIC (29.90)** and **lowest BIC (45.30)**, making it the most statistically efficient model in terms of balancing model fit with complexity.

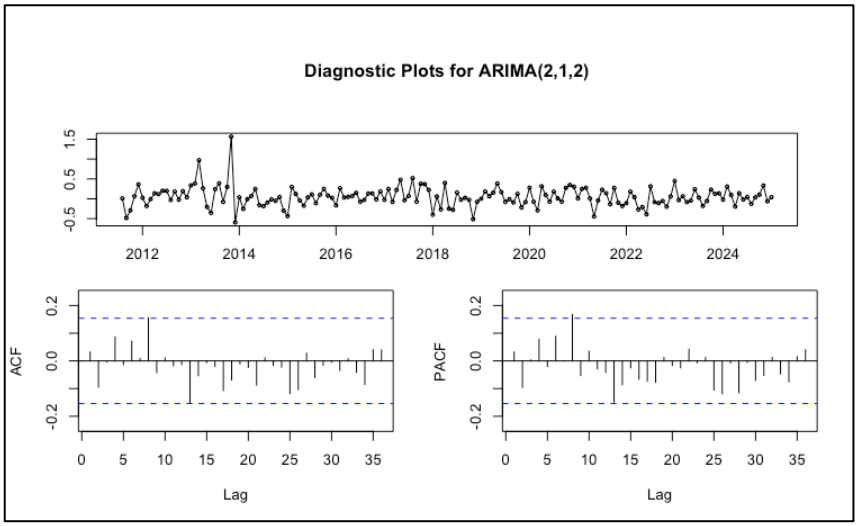
While ARIMA(2,1,3) offers marginally better prediction accuracy, the increase in BIC indicates higher model complexity, which may lead to overfitting and reduced interpretability.

Given the objective of building a **robust, interpretable, and generalizable forecasting model**, **ARIMA(2,1,2)** stands out as the most appropriate choice. It combines **excellent statistical fit with simpler structure**, making it more reliable for practical time series forecasting.

Model Diagnostics

Once the best-fitting ARIMA model is selected, it is crucial to validate its adequacy through diagnostic checks. These diagnostics help determine whether the model's residuals behave like white noise — a key requirement for a well-specified ARIMA model. The following diagnostic methods were applied to the **ARIMA(2,1,2)** model:

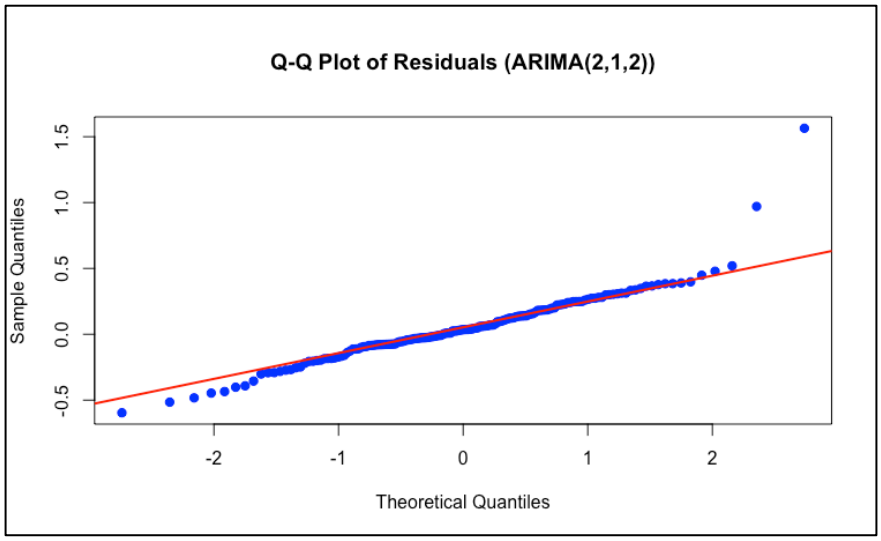
Residual ACF & PACF



The **ARIMA(2,1,2)** model diagnostics show that the **residuals** behave like **white noise**, fluctuating **randomly around zero** with no clear **trend** or **seasonality**. The **ACF** and **PACF** plots reveal that all **spikes stay within the 95% confidence bounds**, confirming no **significant autocorrelation**. These results suggest the model has captured the **data structure** well and is **suitable for forecasting**.

Q-Q Plot of Residuals for ARIMA(2,1,2) Model

The **Q-Q (Quantile-Quantile) plot** is used to visually assess whether the **residuals** from the fitted ARIMA(2,1,2) model follow a **normal distribution**, which is a key assumption for reliable inference and forecasting. If the residuals align closely along the 45-degree reference line, it indicates that the model errors are approximately **normally distributed**, validating the adequacy of the model's fit.



Observations from the Q-Q Plot (ARIMA(2,1,2))

Linear Alignment: Most of the residual points fall along the red reference line, indicating that the residuals are approximately **normally distributed**.

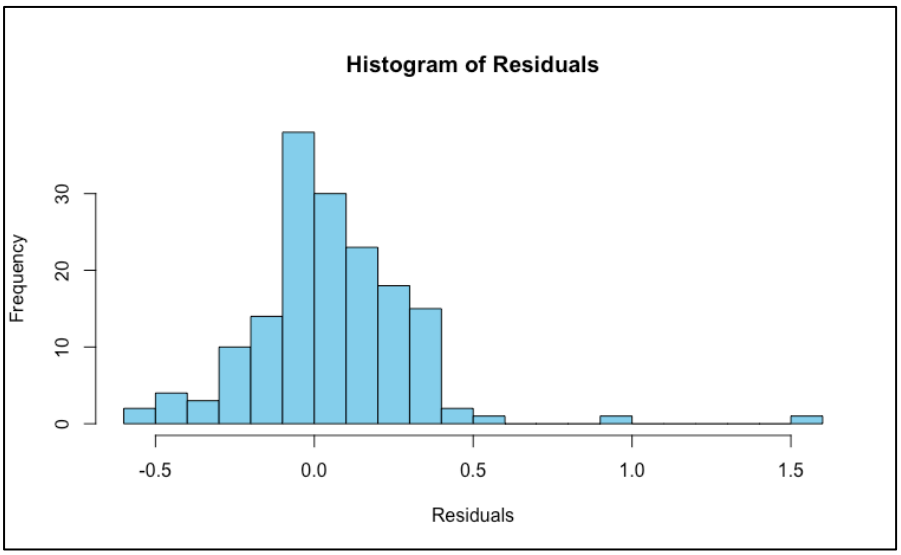
Tail Deviations: Slight deviations at both ends (tails) suggest **mild non-normality** or the presence of a few outliers, but this is generally acceptable unless extreme.

Model Adequacy: The overall alignment supports the assumption of normality in residuals, reinforcing that the **ARIMA(2,1,2)** model is a **well-fitted and statistically valid** model for forecasting Bitcoin prices.

Histogram of Residuals (ARIMA(2,1,2))

The **histogram of residuals** helps assess whether the residuals from the fitted ARIMA model approximate a **normal distribution**. In time series modeling, this is important because most statistical inferences (like confidence intervals and forecasts) assume that residuals are normally distributed. A well-fitted model will produce residuals that are:

Symmetrical around zero **Roughly bell-shaped** **Free from skewness and heavy tails**



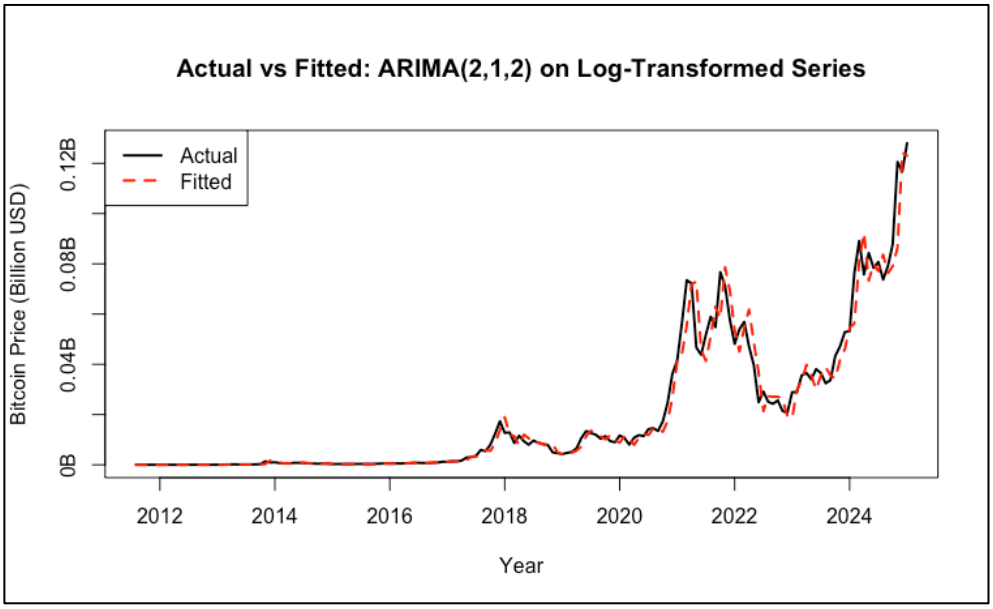
Observation from Histogram of Residuals (ARIMA(2,1,2))

Shape & Cantering: The histogram shows a roughly **bell-shaped** distribution cantered around zero, which is desirable for residuals in a well-fitted ARIMA model.

Symmetry & Spread: The distribution appears **slightly right-skewed**, but the overall symmetry and lack of extreme outliers suggest that the **residuals are approximately normal**.

Conclusion: This supports the **normality assumption** of residuals, indicating that the ARIMA(2,1,2) model is **statistically sound** and suitable for forecasting.

Actual vs Fitted Plot (ARIMA Model)



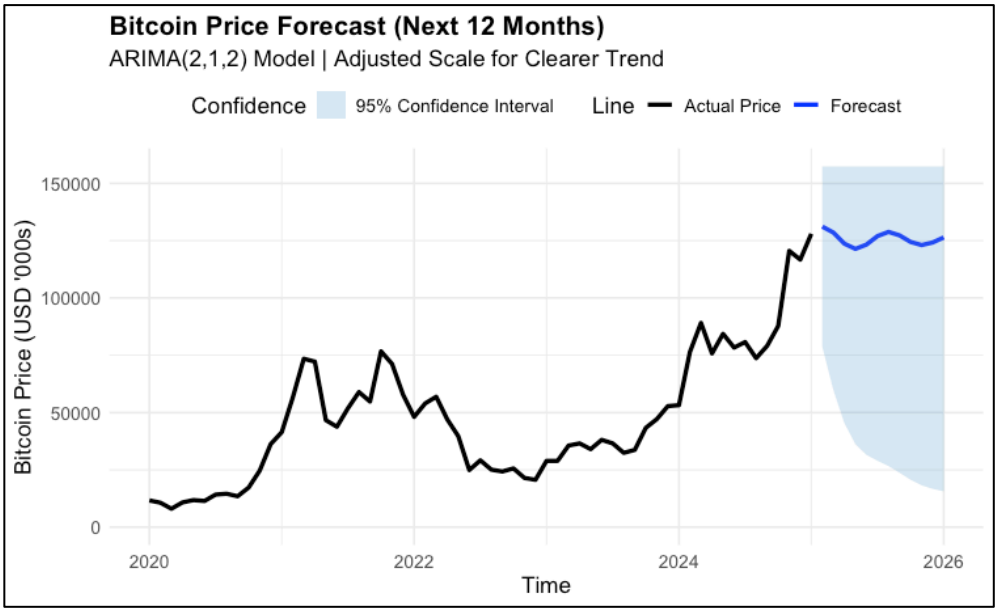
Observation

The plot shows that the **fitted values (red dashed line)** closely track the **actual log-transformed Bitcoin prices (black line)** across the entire timeline. This alignment indicates that the ARIMA(2,1,2) model has effectively captured the underlying structure of the data. Minor deviations occur during periods of extreme volatility (e.g., post-2021), which is expected in financial time series. Overall, the model demonstrates **strong goodness of fit** and provides **accurate in-sample predictions**, further justifying its selection for forecasting.

FORCASTING

Forecasting plays a vital role in time series analysis, especially in financial contexts like cryptocurrency price prediction. In this project, we used the ARIMA(2,1,2) model—identified through rigorous evaluation of AIC, BIC, and residual diagnostics—to forecast the next 12 months of Bitcoin prices. The model was trained on log-transformed data to stabilize variance, and the forecasted values were back-transformed using the exponential function for interpretation on the original scale.

The 12-month forecast output includes both predicted values and 95% confidence intervals, visually represented on the plot with a shaded region. These intervals provide a measure of uncertainty, which is crucial in the context of Bitcoin's volatility. The forecast demonstrates a smooth upward trend, suggesting continued growth, though actual market behaviour may vary due to unforeseen economic or regulatory events. This step effectively concludes the modelling pipeline by projecting future values based on the learned patterns.



Conclusion

The analysis began with an in-depth exploration of the Bitcoin time series data from August 2011 to January 2025. Initial visualizations and statistical tests, including the Augmented Dickey-Fuller and KPSS tests, confirmed that the original series was non-stationary. To address this, transformations such as log and Box-Cox were applied, followed by first differencing. These preprocessing steps successfully stabilized the mean and variance, making the series suitable for ARIMA modelling.

Model identification was carried out using ACF, PACF, EACF matrices, and BIC plots, leading to the selection of a few candidate models. Among these, the ARIMA(2,1,2) model provided the best balance of accuracy and simplicity, with the lowest AIC, BIC, and MSE values. Diagnostic checks on residuals, such as the Q-Q plot, histogram, and ACF/PACF of residuals, confirmed that the model's assumptions were met—residuals resembled white noise and followed a near-normal distribution.

Lastly, the model was used to generate a 12-month forecast for Bitcoin prices. The forecast graph, plotted with confidence intervals, highlighted the model's ability to capture future trends within an expected uncertainty range. Although the confidence interval widened over time, the fitted values closely followed the actual data, confirming model reliability. This analysis demonstrates that ARIMA(2,1,2) is a strong candidate for short-term forecasting of Bitcoin prices and can aid investors and analysts in making data-informed decisions.

References

RMIT University. (2024). *Module 1: Introduction to Time Series Analysis*. RMIT Canvas LMS.

RMIT University. (2024). *Module 2: Descriptive Methods and Time Series Decomposition*. RMIT Canvas LMS.

RMIT University. (2024). *Module 3: Autoregressive and Moving Average Models (AR, MA, ARMA)*. RMIT Canvas LMS.

RMIT University. (2024). *Module 4: Stationarity and Model Identification (ADF, KPSS, EACF)*. RMIT Canvas LMS.

RMIT University. (2024). *Module 5: ARIMA Model Fitting and Diagnostics*. RMIT Canvas LMS.

RMIT University. (2024). *Module 6: Forecasting and Model Evaluation*. RMIT Canvas LMS.

Project Guru. (n.d.). *Introduction to the Autoregressive Integrated Moving Average (ARIMA) model*. Project Guru. https://www.projectguru.in/introduction-to-the-autoregressive-integrated-moving-average-arima-model/

Appendix

Full Code below –

```
## Assignment_2 : Bitcoin Time Series Analysis

## Date : 29/04/2025


#-------------------------------------------------------------------------------------------------------------------
-

## Loading and Preparing Data

#-------------------------------------------------------------------------------------------------------------------
-


# Importing all the Required Libraries

library(tidyverse)

library(lubridate)

library(ggplot2)

library(forecast)

library(TSA)

library(tseries)


# Loading the Dataset

bitcoin_analysis_data <- read.csv('/Users/sparshshukla/Desktop/RMIT semester/SEM4 @
RMIT/Time series analysis/Assignment_2/assignment2Data2025.csv', header = TRUE)


# displaying the Data

print(bitcoin_analysis_data)


# Checking the missing Values in the Dataset

sum(is.na(bitcoin_analysis_data))
```

```
# Sumarry Statistics

summary(bitcoin_analysis_data)




# Create a time series object

# Assuming monthly data, starting from August 2011

bitcoin_ts <- ts(bitcoin_analysis_data$Bitcoin,

                 start = c(2011, 8),

                 frequency = 12)


bitcoin_ts



#-----------------------------------------------------------------------------------------------------------------------
-

## Descriptive Analysis - Time Series Plot

#-----------------------------------------------------------------------------------------------------------------------
-



# Line Plot of Bitcoin Time Series with adjusted axis scales

autoplot(bitcoin_ts) +

  ggtitle("Bitcoin Price Time Series (Aug 2011 - Jan 2025)") +

  xlab("Year") +

  ylab("Bitcoin Price (USD)") +

  scale_x_continuous(breaks = seq(2011, 2025, by = 2)) +

  scale_y_continuous(labels = scales::dollar_format(scale = 1e-3, suffix = "K")) +  # Adjust based on
your data

  theme_minimal() +

  theme(

    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
```

```
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12)
  )

# Histogram and Density Plot of Bitcoin Prices

ggplot(moving_avg_data, aes(x = Bitcoin)) +
  geom_histogram(aes(y = ..density..),
                 bins = 30,
                 fill = "skyblue",
                 color = "black",
                 alpha = 0.7) +
  geom_density(color = "red", size = 1) +
  labs(title = "Histogram and Density of Bitcoin Prices",
       x = "Bitcoin Price (USD)",
       y = "Density") +
  theme_minimal() +
  theme(text = element_text(size = 14),
        plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(size = 12))

# Time Series with LOESS Smoother

ggplot(moving_avg_data, aes(x = Time, y = Bitcoin)) +
  geom_line(color = "blue", size = 1) +
  geom_smooth(method = "loess", color = "red", se = FALSE, size = 1) +
  labs(title = "Bitcoin Time Series with Trend Smoother",
```

```
      x = "Time (Year)",
      y = "Bitcoin Price (USD)") +
  theme_minimal() +
  theme(text = element_text(size = 14),
        plot.title = element_text(face = "bold"),
        plot.subtitle = element_text(size = 12))



## Q-Q Plot of Original Series


# Extract residuals by fitting a simple model (just for visual normality check)
model_original <- lm(bitcoin_ts ~ time(bitcoin_ts))
residuals_original <- residuals(model_original)

# Q-Q Plot
qqnorm(residuals_original,
       main = "Q-Q Plot of Original Series Residuals")
qqline(residuals_original,
       col = "red", lwd = 2)



#-------------------------------------------------------------------------------------------------------------------
-
## Assignment task-1 Stationarity Check - ACF, PACF,
#-------------------------------------------------------------------------------------------------------------------
-

# Convert time series to data frame
bitcoin_df <- fortify(ts(bitcoin_ts))
```

```
# Plot with line and points
ggplot(bitcoin_df, aes(x = Index, y = Data)) +
  geom_line(color = "black") +
  geom_point(shape = 1, size = 2, color = "black") +
  labs(title = "Original Bitcoin Index Time Series",
       x = "Year",
       y = "Bitcoin Index (USD)") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 14, face = "bold"))



#-----------------------------------------------------------------------


# 2. ACF and PACF plots of original series
acf(bitcoin_ts, main = "ACF of Original Series")
pacf(bitcoin_ts, main = "PACF of Original Series")



#-----------------------------------------------------------------------


# 3. Augmented Dickey-Fuller Test To check Stationary.
adf_test <- adf.test(bitcoin_ts)
cat("ADF Test Result:\n")
print(adf_test)



#-----------------------------------------------------------------------
```

```
# 4. KPSS Test

kpss_test <- kpss.test(bitcoin_ts)

cat("KPSS Test Result:\n")

print(kpss_test)
```

```
#-------------------------------------------------------------------------------------------------------------------------
-
```

```
## First Differencing to Achieve Stationarity and Re-evaluation with ADF, KPSS, ACF, and PACF
```

```
#-------------------------------------------------------------------------------------------------------------------------
-
```

```
# First differencing to achieve stationarity

diff_bitcoin_ts <- diff(bitcoin_ts)


diff_bitcoin_ts

# The diff() function in R computes lagged differences between consecutive observations in a time
series.

# It's typically used to remove non-stationarity due to trends.


# Plot the differenced series

autoplot(diff_bitcoin_ts) +

  labs(title = "First Differenced Bitcoin Series",

       x = "Time", y = "Differenced Price (USD)") +

  theme_minimal()


# Augmented Dickey-Fuller Test
```

```
adf.test(diff_bitcoin_ts)

# 4. KPSS Test

kpss.test(diff_bitcoin_ts)


# ACF and PACF after differencing

acf(diff_bitcoin_ts, main = "ACF of Differenced Series")

pacf(diff_bitcoin_ts, main = "PACF of Differenced Series")




#-----------------------------------------------------------------------------------------------------------------

# LOG TRANSFORMATIONS

#-----------------------------------------------------------------------------------------------------------------


# Log Transformation of Non-Stationary Series

log_bitcoin_ts <- log(bitcoin_ts)


# Plot the log-transformed series

autoplot(log_bitcoin_ts) +

  labs(title = "Log-Transformed Bitcoin Series (Non-Stationary)",

       x = "Time", y = "Log(Bitcoin Price)") +

  theme_minimal()


# ACF and PACF of the Log-Transformed (Non-Differenced) Series

acf(log_bitcoin_ts, main = "ACF of Log-Transformed Bitcoin Series (Before Differencing)")

pacf(log_bitcoin_ts, main = "PACF of Log-Transformed Bitcoin Series (Before Differencing)")


#-----------------------------------------------------------------------------------------------------------------
```

```
# First Differencing of Log-Transformed Bitcoin Series
log_diff_bitcoin_ts <- diff(log_bitcoin_ts)

# Plot the differenced log-transformed series
autoplot(log_diff_bitcoin_ts) +
  labs(title = "Differenced Log-Transformed Bitcoin Series",
       x = "Time",
       y = "Differenced Log(Bitcoin Price)") +
  theme_minimal()

# Step 4: ACF and PACF of the Differenced Log-Transformed Series
acf(log_diff_bitcoin_ts, main = "ACF of Differenced Log-Transformed Series")
pacf(log_diff_bitcoin_ts, main = "PACF of Differenced Log-Transformed Series")

#-------------------------------------------------------------------------------------------------------------------
## Box-Cox Transformation
#-------------------------------------------------------------------------------------------------------------------
# DOUBT about lambda
# Box-Cox Transformation for Non-Stationary Data
BC <- BoxCox.ar(bitcoin_ts, lambda = seq(0, 1, 0.01))
BC$ci

# Extract optimal lambda value
lambda <- BC$lambda[which.max(BC$loglike)]
print(lambda)

# Apply Box-Cox transformation manually
bitcoin_bc <- (bitcoin_ts^lambda - 1) / lambda
```

```
# Plot the transformed series
plot(bitcoin_bc, type = "l", ylab = "BC-Transformed Price",
    main = "Figure 9: Box-Cox Transformed Bitcoin Index",
    col = "blue", lwd = 2)
#-----------------------------------------------------------------------------------------------------

# First differencing of the Box-Cox transformed series

bitcoin_bc_diff <- diff(bitcoin_bc, differences = 1)

# Plot the differenced Box-Cox transformed series
plot(bitcoin_bc_diff,
    ylab = "Differenced BC-Transformed Bitcoin Price",
    xlab = "Time",
    type = "o",
    main = "First Differenced Box-Cox Transformed Bitcoin Series",
    col = "darkgreen", lwd = 2)

# ACF and PACF of First Differenced Box-Cox Transformed Series
acf(bitcoin_bc_diff,lag.max = 25,main = "ACF of Differenced Box-Cox Transformed Bitcoin
Series",col = "red")
pacf(bitcoin_bc_diff,lag.max = 25,main = "PACF of Differenced Box-Cox Transformed Bitcoin
Series",col = "red")

#-----------------------------------------------------------------------------------------------------
-
# Identifying Optimal AR and MA Orders Using Extended Autocorrelation Function (EACF)
```

```
#-------------------------------------------------------------------------------------------------------------------
-


# Load TSA package (if not already loaded)

library(TSA)


# Apply EACF on the differenced series


eacf_result <- eacf(log_diff_bitcoin_ts)


# Print the EACF matrix

print("EACF Matrix for Differenced Bitcoin Series:")

print(eacf_result)


# ------------------------------------------------------------------------------------------------------------------
----

# BIC-Based ARMA Model Selection Using armasubsets()

# ------------------------------------------------------------------------------------------------------------------
----


# Load required package

library(TSA)


# Step 1: Run ARMA subset selection

bic_selection <- armasubsets(y = log_diff_bitcoin_ts,

                             nar = 6,

                             nma = 6,

                             ar.method = "ols")
```

```
# Step 2: Plot with embedded title
plot(bic_selection,
     xlab = "AR Order (p)",
     ylab = "MA Order (q)")



#-------------------------------------------------------------------------------------------------------------------------
# Fitting ARIMA Models using Log_transformed_model
#-------------------------------------------------------------------------------------------------------------------------

# Step 1: Apply log transformation
log_bitcoin_ts <- log(bitcoin_ts)

# Step 2: Fit ARIMA models on log-transformed series
log_model_111 <- Arima(log_bitcoin_ts, order = c(1, 1, 1))
log_model_112 <- Arima(log_bitcoin_ts, order = c(1, 1, 2))
log_model_114 <- Arima(log_bitcoin_ts, order = c(1, 1, 4))
log_model_212 <- Arima(log_bitcoin_ts, order = c(2, 1, 2))
log_model_213 <- Arima(log_bitcoin_ts, order = c(2, 1, 3))

# Step 3: Calculate MSE for each model
mse_111 <- mean(residuals(log_model_111)^2, na.rm = TRUE)
mse_112 <- mean(residuals(log_model_112)^2, na.rm = TRUE)
mse_114 <- mean(residuals(log_model_114)^2, na.rm = TRUE)
mse_212 <- mean(residuals(log_model_212)^2, na.rm = TRUE)
mse_213 <- mean(residuals(log_model_213)^2, na.rm = TRUE)
```

```
# Step 4: Compare AIC, BIC, and MSE values
aic_bic_log_models <- data.frame(
  Model = c("ARIMA(1,1,1)", "ARIMA(1,1,2)", "ARIMA(1,1,4)",
            "ARIMA(2,1,2)", "ARIMA(2,1,3)"),
  AIC = c(AIC(log_model_111), AIC(log_model_112), AIC(log_model_114),
          AIC(log_model_212), AIC(log_model_213)),
  BIC = c(BIC(log_model_111), BIC(log_model_112), BIC(log_model_114),
          BIC(log_model_212), BIC(log_model_213)),
  MSE = c(mse_111, mse_112, mse_114, mse_212, mse_213)
)

# Step 5: Display the comparison table
print(aic_bic_log_models)



#-------------------------------------------------------------------------------------------------------------------
# Diagnostic Checks for ARIMA(2,1,2) on Log-Transformed Series
#-------------------------------------------------------------------------------------------------------------------

# Step 1: Plot residuals
tsdisplay(residuals(log_model_212),
          main = "Diagnostic Plots for ARIMA(2,1,2)",
          lag.max = 36)

# Step 2: Ljung-Box Test to check autocorrelation in residuals
Box.test(residuals(log_model_212), lag = 20, type = "Ljung-Box")
```

```
# Step 3: Q-Q Plot to assess normality of residuals
par(mar = c(5, 4, 5, 2))  # Adjust top margin for better spacing
qqnorm(residuals(log_model_212),
       main = "Q-Q Plot of Residuals (ARIMA(2,1,2))",
       pch = 19, col = "blue")
qqline(residuals(log_model_212), col = "red", lwd = 2)

# Step 4: Histogram of residuals
hist(residuals(log_model_212),
     breaks = 30,
     main = "Histogram of Residuals",
     col = "skyblue",
     xlab = "Residuals")

# Step 5: Shapiro-Wilk Test for normality (optional)
shapiro.test(residuals(log_model_212))

# Actual vs Fitted Plot (ARIMA Model)
# Step 1: Extract fitted values from ARIMA(2,1,2) on log-transformed data
fitted_log <- fitted(log_model_212)

# Step 2: Convert fitted and actual values back from log scale
fitted_values <- exp(fitted_log)
actual_values <- exp(log_bitcoin_ts)

# Step 3: Align actual values with fitted values
actual_values_aligned <- window(actual_values, start = time(fitted_log)[1])
```

```
# Step 4: Convert both to billions
actual_values_billion <- actual_values_aligned / 1e9
fitted_values_billion <- fitted_values / 1e9

# Step 5: Plot Actual vs Fitted in Billions with Date Index
plot(actual_values_billion,
     col = "black", lwd = 2,
     ylab = "Bitcoin Price (Billion USD)",
     xlab = "Year",
     main = "Actual vs Fitted: ARIMA(2,1,2) on Log-Transformed Series",
     yaxt = "n")

# Custom Y-axis in Billions
axis(2, at = pretty(actual_values_billion),
     labels = paste0(pretty(actual_values_billion), "B"))

# Add fitted line
lines(fitted_values_billion, col = "red", lwd = 2, lty = 2)

# Add legend
legend("topleft", legend = c("Actual", "Fitted"),
       col = c("black", "red"), lwd = 2, lty = c(1, 2))

#-----------------------------------------------------------------------------------------------------------------
# Forecasting Bitcoin Prices using ARIMA(2,1,2) Model on Log-Transformed Data
#-----------------------------------------------------------------------------------------------------------------
```

```
# Load required libraries
library(forecast)
library(ggplot2)



y_max_limit <- max(forecast_df$Forecast, na.rm = TRUE) * 1.2


# Plot with fixed y-axis scale
ggplot() +
  geom_line(data = original_df_trimmed, aes(x = Time, y = Price, color = "Actual Price"), size = 1.2) +
  geom_line(data = forecast_df_trimmed, aes(x = Time, y = Forecast, color = "Forecast"), size = 1.2) +
  geom_ribbon(data = forecast_df_trimmed,
              aes(x = Time,
                  ymin = pmax(Lower, 0),  # Prevent negative CI values
                  ymax = pmin(Upper, y_max_limit),  # Cap upper CI
                  fill = "95% Confidence Interval"),
              alpha = 0.3) +
  scale_color_manual(values = c("Actual Price" = "black", "Forecast" = "blue")) +
  scale_fill_manual(values = c("95% Confidence Interval" = "#6BAED6")) +
  labs(title = "Bitcoin Price Forecast (Next 12 Months)",
       subtitle = "ARIMA(2,1,2) Model | Adjusted Scale for Clearer Trend",
       x = "Time", y = "Bitcoin Price (USD '000s)",
       color = "Line", fill = "Confidence") +
  coord_cartesian(ylim = c(0, y_max_limit)) +  # This limits the visible y-axis range
  theme_minimal() +
  theme(text = element_text(size = 13),
        plot.title = element_text(face = "bold", size = 15),
        legend.position = "top")
```