Time Series Predictive Analysis of Bitcoin

Riya Minesh Amin - s3807007
MATH1318

Contents

Introduction

Bitcoin is a cryptocurrency that was initially described by Satoshi Nakamoto in his 2008 article "Bitcoin: A Peer-to-Peer Electronic Cash System". A cryptocurrency is defined as "a virtual coinage system that operates similarly to a traditional currency, allowing users to submit virtual payment for products and services without the need for a central trustworthy authority. Bitcoin is the first cryptocurrency and one of the most common in the world, having been created in 2009. The value of Bitcoin is determined by what people are willing to pay in, so it is highly unpredictable and fluctuates on a daily basis.

Objective

The goal of this report is to examine the daily closing price of bitcoin from the *30th April 2016* to *30th April 2021* using time series techniques, then select the best model from a collection of viable models for this dataset and provide Bitcoin projections for the following 10 days.

Methodology

Data Description

The data used here is the Bitcoin price data from the year 30th April 2016 to 30th April 2021, which corresponds to a total of 1827 observations. The data has been collected from coinmarketcap.com. The data is derived from Bitstamp, the largest Bitcoin exchange, and includes a daily database valued in US dollars, which is the most commonly traded currency against Bitcoin.

Approach

The Model Building Strategy is used to create the most significant time series Model, which is then used for prediction. Our technique is described in detail in the following steps:

- The data is fitted with several Trend Models to see whether any of them match the series, then the ADF test is performed to determine the series' stationarity.
- By parameter estimation and diagnostic verification, we investigated three models: ARIMA, GARCH, and ARMA+GARCH to determine the best model to match Bitcoin data. In addition, we computed MASE values for our best model overfitted values and projections.
- Model parameters are calculated, and significance tests are run to determine whether or not the models are significant.
- AIC values to aid in choosing the best fit among the Probable models.
- Model Diagnostics is used to examine the fitness and normality of the models.
- Forecast the bitcoin closing price for the next 10 years using the best model available.

Data Analysis

Several requirements must be met to develop an efficient time series model. To begin, we plot a time series plot and scatter plot to assess the data's Trend, Changing Variance, Seasonality, Interventions, and Behaviour.

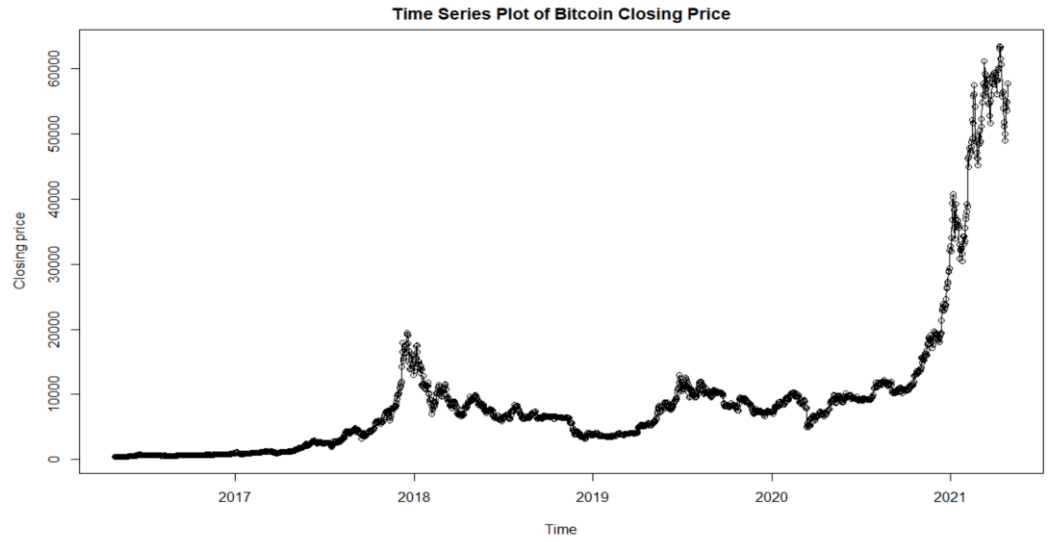

Figure 1: Time series plot of Bitcoin Closing Price

Based on the plot above, the following four primary features may be observed:

- Yes, there is a trend. There is an evident tendency, an upward trend, and it says that the trend is nonstationary.
- Changing variance: visible changing variance.
- Repeat pattern: no repeating pattern, no seasonality components (for this series, no seasonality or harmonic model is required).
- Following observations suggest that there is a presence of autoregressive behavior (it may result from the trend in the time series).

To determine whether or whether consecutive years are connected in any manner, we may use each day's bitcoin closing price value to estimate the next day. The scatter plot below analyses the relationship between successive numbers of bitcoin closing values.

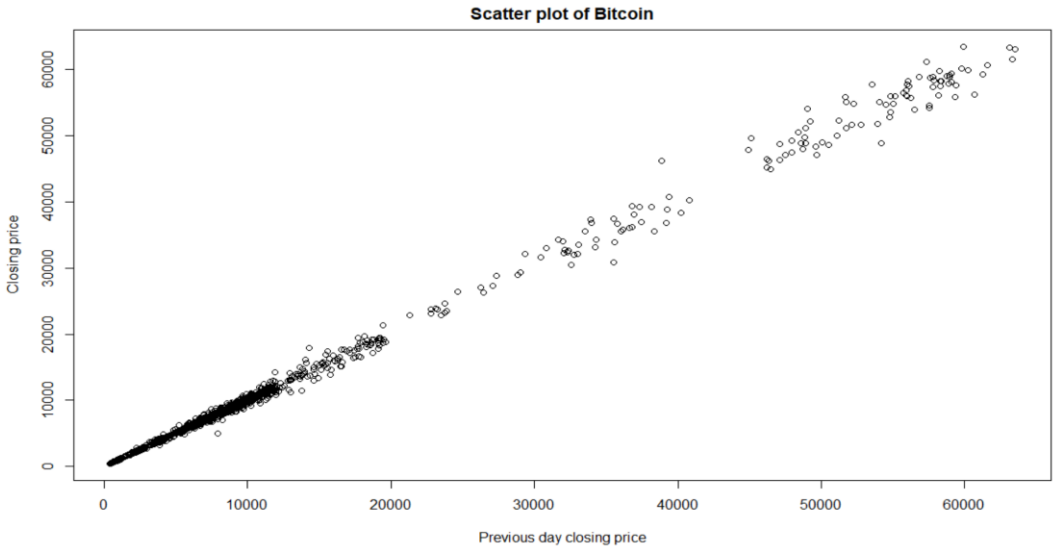

Figure 2: Scatter plot of Bitcoin Closing Price

Looking at the scatterplot suggests that there is a very high positive correlation among the successive points.

Checking Stationarity

Normal Q-Q plot

Before modelling time index data, we must first verify for stationarity, as many statistical and economic approaches rely on it. According to the Q-Q plot below, it did not fulfill the criterion of normalcy. And the Shapiro test verified it with a p-value of less than 0.01. As a result, the null hypothesis that it is random was rejected.

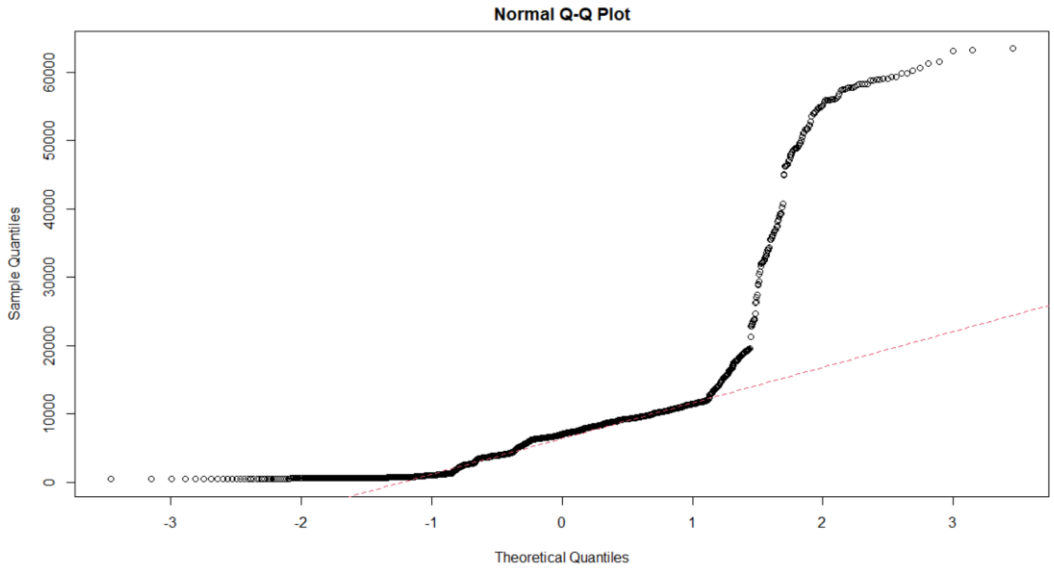

Figure 3: Q-Q Normal Plot

ACF and PACF plot

Below ACF and PACF plots support the evidence of non-stationarity. ACF has a slowly decaying pattern, and PACF has a very strong first correlation, indicating the presence of trend and non-stationarity. With the ADF test, a p-value of 0.99, (which is greater than 0.05), that indicates cannot reject the null hypothesis stating that the series is non-station.

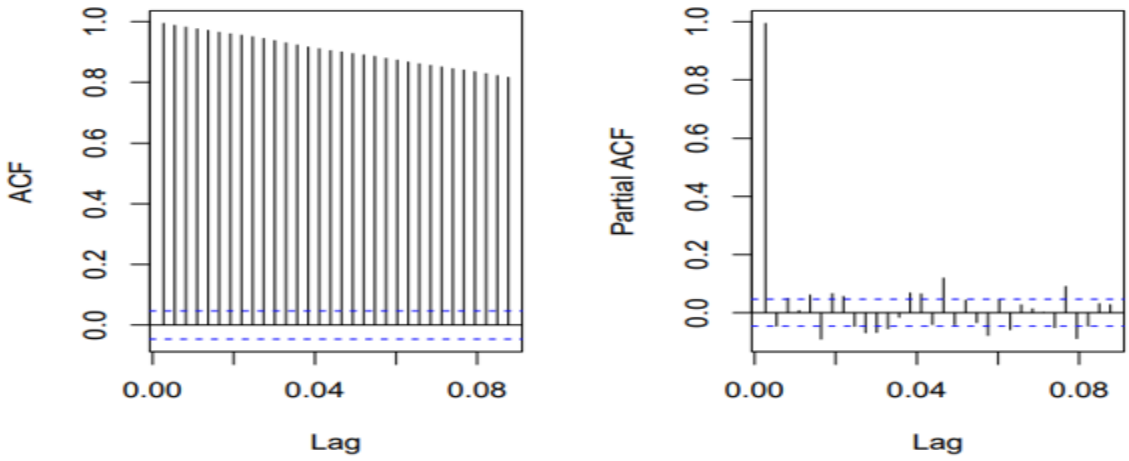

Figure 4: ACF and PACF plots

```
        Augmented Dickey-Fuller Test

data:  BTC_ts
Dickey-Fuller = 0.88071, Lag order = 12, p-value = 0.99
alternative hypothesis: stationary
```

BOXCOX Transformation

To overcome non-stationarity, BOXCOX transformation was applied.

Because the log probability plot seemed to catch 0 in this case, we chose to apply the log transformation to the bitcoin series.

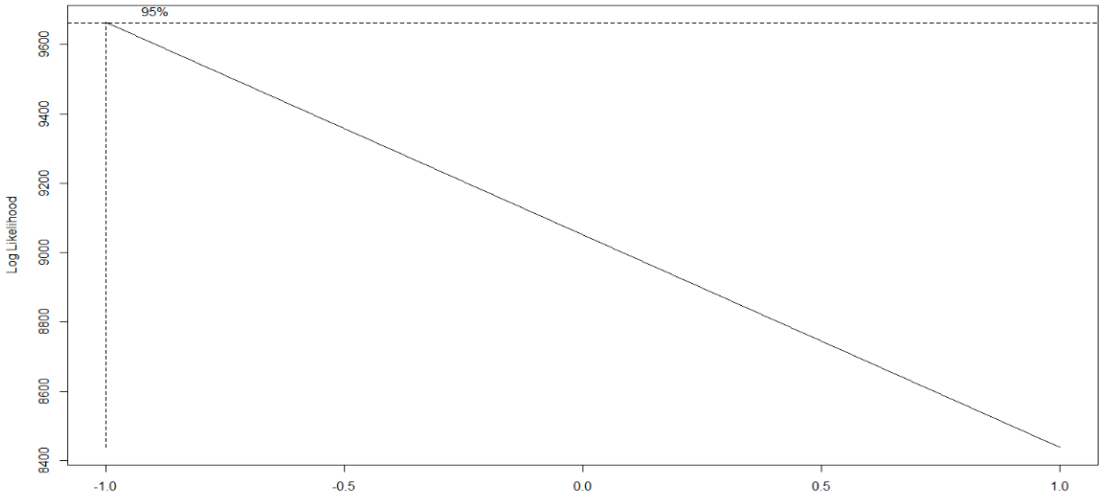

Figure 5: Log-likelihood for BOXCOX transformation

The existence of a trend and nonstationarity in log-transformed coin data is implied by the presence of a slowly declining pattern in ACF and a very high initial correlation in PACF.

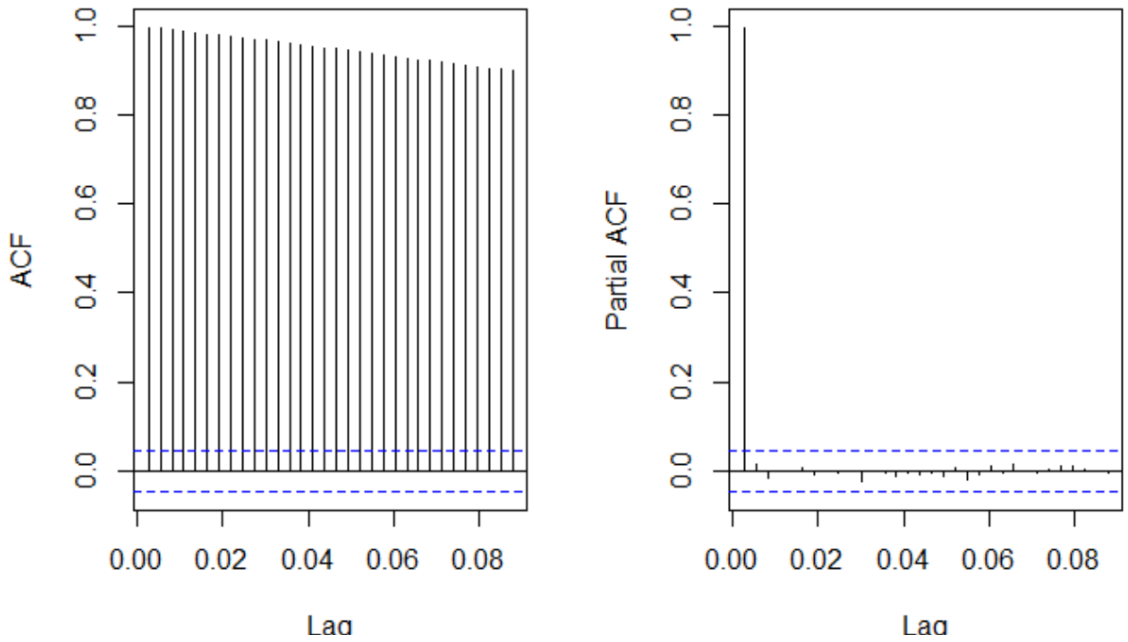

Figure 6: ACF and PACF plots of transformed log series

The ADF test yielded a p-value of 0.7319, indicating that the null hypothesis claiming that the series is stationary cannot be rejected.

```
        Augmented Dickey-Fuller Test

data:  log(BTC_ts)
Dickey-Fuller = -1.6372, Lag order = 12, p-value = 0.7319
alternative hypothesis: stationary
```

Therefore, further, we need differencing on the transformed data.

Differencing

We applied First differencing on the log-transformed data and we see that the trend has successfully detrended.

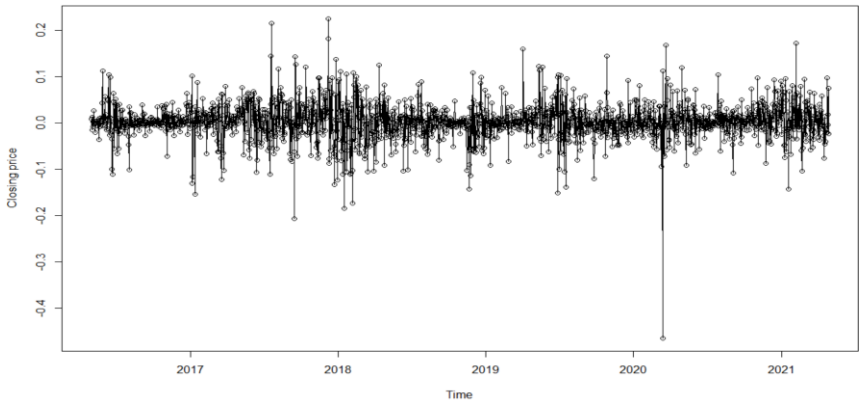

Figure 7: Time series plot of Fir differencing of log-transformed series.

With the ADF test, a p-value of 0.01 demonstrates that we may reject the null hypothesis that the series is non-stationary, and the series is now stationary.

```
Title:
 Augmented Dickey-Fuller Test

Test Results:
  PARAMETER:
    Lag Order: 19
  STATISTIC:
    Dickey-Fuller: -7.7619
  P VALUE:
    0.01

Description:
 Fri Jun 04 14:31:51 2021 by user: user
```

For all delays, the McLeod-Li test is significant at the 5% level of significance. This is a strong indication of the existence of volatility clustering. It can also explain the plot's clear shifting variance.

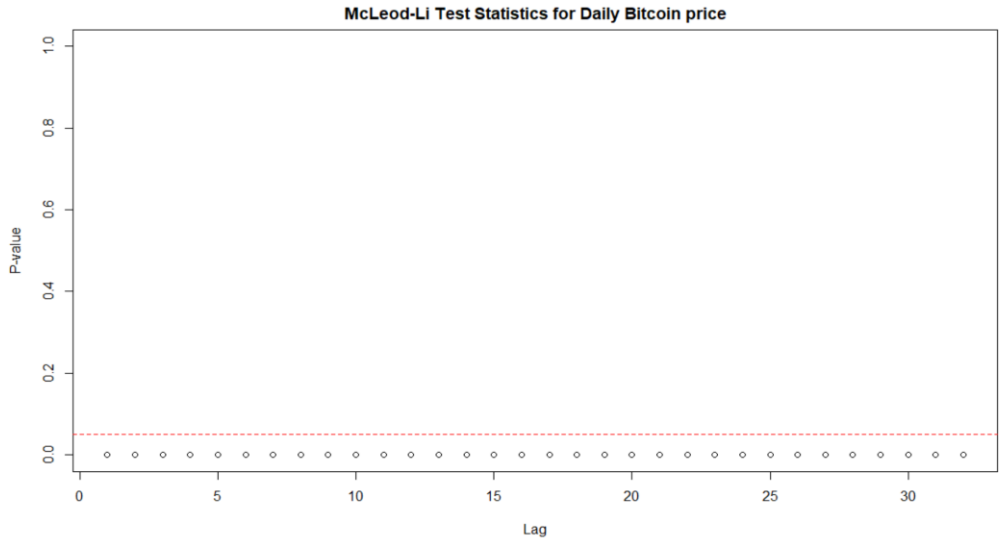

Figure 8: McLeod-Li of log transformation and first differencing of bitcoin series data

Modelling the Bitcoin series

ARIMA Model

Model Specifications

To establish the orders of the ARIMA model, we will examine the EACF output, the BIC table, and the ACF and PACF displays.

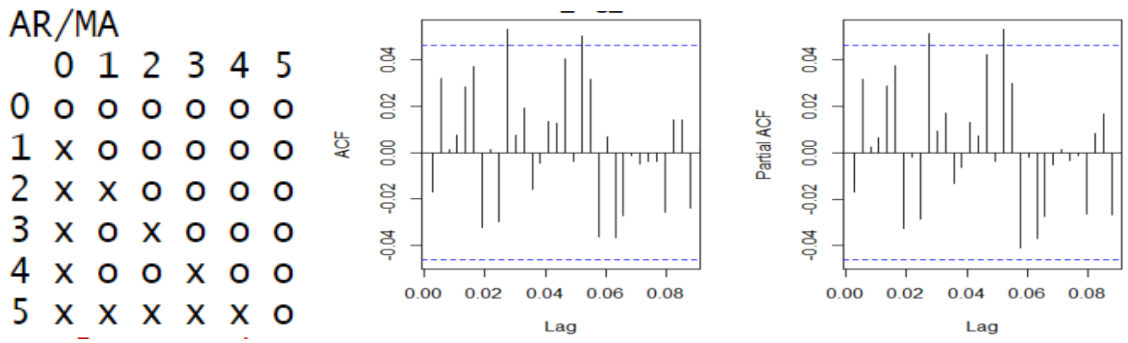

Figure 9: EACF output and ACF and PACF plots

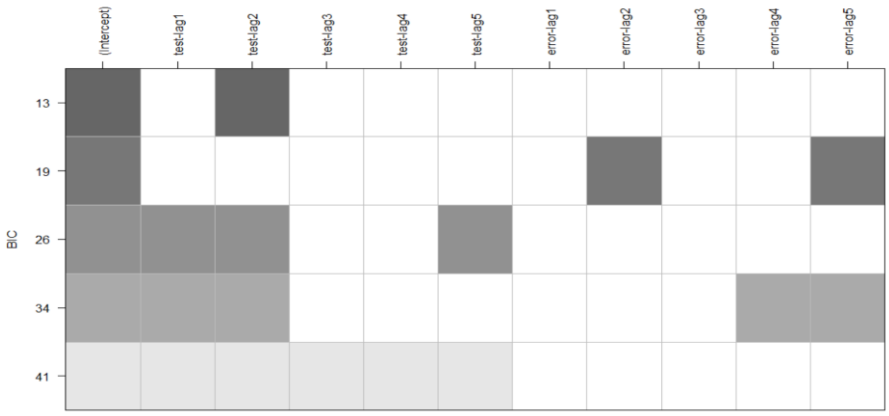

Figure 10: BIC Table

From the output of EACF, we can include ARIMA(0,1,1) ARIMA(1,1,1), ARIMA(1,1,2), ARIMA(2,1,2). From BIC table the models will be ARIMA(2,1,0), ARIMA(0,1,2), ARIMA(2,1,5) and ARIMA(5,1,0). Overall, the model could be included are ARIMA(0,1,1), ARIMA(0,1,2), ARIMA(1,1,1), ARIMA(1,1,2), ARIMA(2,1,0), ARIMA(2,1,2), ARIMA(2,1,5) and ARIMA(5,1,0).

ARIMA Parameter Estimation

To estimate parameters, two methods would be considered.

- Least Squares estimation of the AR coefficient with significance tests, method='CSS'
- Maximum likelihood estimation of the AR coefficient with significance tests, method="ML"

Significance Test for coefficients

ARIMA (0,1,1)

We see that both methods 'CSS' & 'ML' are insignificant.

Least Square Estimation	**Maximum likelihood estimation**
z test of coefficients: Estimate Std. Error z value Pr(>|z|) ma1 -0.011872 0.022639 -0.5244 0.6	z test of coefficients: Estimate Std. Error z value Pr(>|z|) ma1 -0.011866 0.022634 -0.5243 0.6001

ARIMA (0,1,2)

We see that both methods 'CSS' & 'ML' are insignificant.

Least Square Estimation	**Maximum likelihood estimation**
z test of coefficients: Estimate Std. Error z value Pr(>|z|) ma1 -0.012593 0.023443 -0.5372 0.5911 ma2 0.035562 0.023237 1.5304 0.1259	z test of coefficients: Estimate Std. Error z value Pr(>|z|) ma1 -0.012588 0.023436 -0.5371 0.5912 ma2 0.035527 0.023225 1.5297 0.1261

ARIMA (1,1,1)

We see that both methods 'CSS' & 'ML' are significant.

Least Square Estimation	**Maximum likelihood estimation**
z test of coefficients: Estimate Std. Error z value Pr(>|z|) ar1 -0.891663 0.083806 -10.6396 < 2.2e-16 *** ma1 0.873466 0.089037 9.8101 < 2.2e-16 *** --- Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1	z test of coefficients: Estimate Std. Error z value Pr(>|z|) ar1 -0.847567 0.098015 -8.6473 < 2.2e-16 *** ma1 0.829507 0.100456 8.2574 < 2.2e-16 *** --- Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

ARIMA (1,1,2)

We see that both methods 'CSS' & 'ML' are insignificant.

Least Square Estimation	**Maximum likelihood estimation**

```
z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1 -0.0060778  0.8989328 -0.0068   0.9946
ma1 -0.0064112  0.8986673 -0.0071   0.9943
ma2  0.0354384  0.0260460  1.3606   0.1736
```

```
z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1 -0.0059695  0.8725484 -0.0068   0.9945
ma1 -0.0065345  0.8722845 -0.0075   0.9940
ma2  0.0354459  0.0258619  1.3706   0.1705
```

ARIMA (2,1,0)

We see that both methods 'CSS' & 'ML' are insignificant.

Least Square Estimation	**Maximum likelihood estimation**

`z test of coefficients:`
 	
`Estimate Std. Error z value Pr(>|z|)`
`ar1 -0.012252 0.023433 -0.5229 0.6011`
`ar2 0.035980 0.023435 1.5353 0.1247`
	`z test of coefficients:`
	
	`Estimate Std. Error z value Pr(>|z|)`
	`ar1 -0.012286 0.023428 -0.5244 0.6000`
	`ar2 0.035944 0.023424 1.5345 0.1249`

ARIMA (2,1,2)

We see that both methods 'CSS' & 'ML' are significant.

Least Square Estimation	**Maximum likelihood estimation**

`z test of coefficients:`
 	
`Estimate Std. Error z value Pr(>|z|)`
`ar1 0.098878 0.076716 1.2889 0.1974`
`ar2 0.882093 0.077390 11.3980 <2e-16 ***`
`ma1 -0.109394 0.082266 -1.3298 0.1836`
`ma2 -0.854792 0.082932 -10.3071 <2e-16 ***`
`---`
`Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1`
	`z test of coefficients:`
	
	`Estimate Std. Error z value Pr(>|z|)`
	`ar1 0.082792 0.071572 1.1568 0.2474`
	`ar2 0.898670 0.070984 12.6601 <2e-16 ***`
	`ma1 -0.092955 0.078771 -1.1801 0.2380`
	`ma2 -0.873318 0.077823 -11.2218 <2e-16 ***`
	`---`
	`Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1`

ARIMA (2,1,5)

We see that both methods 'CSS' & 'ML' P-values are significant at AR(1) & MA(1).

Least Square Estimation	**Maximum likelihood estimation**

`z test of coefficients:`
 	
`Estimate Std. Error z value Pr(>|z|)`
`ar1 0.651689 0.178472 3.6515 0.0002607 ***`
`ar2 -0.721424 0.158835 -4.5420 5.573e-06 ***`
`ma1 -0.666325 0.179076 -3.7209 0.0001985 ***`
`ma2 0.768840 0.160872 4.7792 1.760e-06 ***`
`ma3 -0.028592 0.033766 -0.8468 0.3971196`
`ma4 0.028334 0.029970 0.9454 0.3444527`
`ma5 0.043111 0.025405 1.6970 0.0897034 .`
`---`
`Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1`
	`z test of coefficients:`
	
	`Estimate Std. Error z value Pr(>|z|)`
	`ar1 0.633832 0.170996 3.7067 0.0002100 ***`
	`ar2 -0.741017 0.160875 -4.6062 4.101e-06 ***`
	`ma1 -0.648428 0.171746 -3.7755 0.0001597 ***`
	`ma2 0.787946 0.162628 4.8451 1.266e-06 ***`
	`ma3 -0.028057 0.033669 -0.8333 0.4046629`
	`ma4 0.026898 0.029666 0.9067 0.3645745`
	`ma5 0.043219 0.025162 1.7176 0.0858626 .`
	`---`
	`Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1`

ARIMA (5,1,0)

We see that both methods 'CSS' & 'ML' are insignificant.

Least Square Estimation	**Maximum likelihood estimation**

```
z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1 -0.0127054  0.0234366 -0.5421   0.5877
ar2  0.0353762  0.0234397  1.5092   0.1312
ar3  0.0053138  0.0234538  0.2266   0.8208
ar4  0.0109154  0.0234396  0.4657   0.6414
ar5  0.0324381  0.0234757  1.3818   0.1670
```

```
z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1 -0.0128383  0.0234302 -0.5479   0.5837
ar2  0.0354194  0.0234271  1.5119   0.1306
ar3  0.0052785  0.0234369  0.2252   0.8218
ar4  0.0108943  0.0234173  0.4652   0.6418
ar5  0.0323541  0.0234466  1.3799   0.1676
```

After determining the significance of the coefficients, we concentrate on AIC values to find the best fitting ARIMA model.

	df <dbl>	AIC <dbl>
model_212_ml	5	-6490.760
model_111_ml	3	-6487.670
model_210_ml	3	-6487.633
model_012_ml	3	-6487.602
model_011_ml	2	-6487.260
model_112_ml	4	-6485.599
model_215_ml	8	-6485.237
model_510_ml	6	-6483.809

From the above AIC table, we can see ARIMA(2,1,2) has the lowest AIC value, as well as the coefficients, were significant. Thus, we chose it to be the best fitting ARIMA model.

Model Diagnostics

Residual Analysis aids in determining the model's fit. If the model is right, it should display White Noise qualities. The Residual Plots are shown below aid in determining the Trend. The histogram, QQ plot, and Shapiro test all aid in determining the data's normality. The ACF and PACF plots aid in determining the independence of the mistakes.

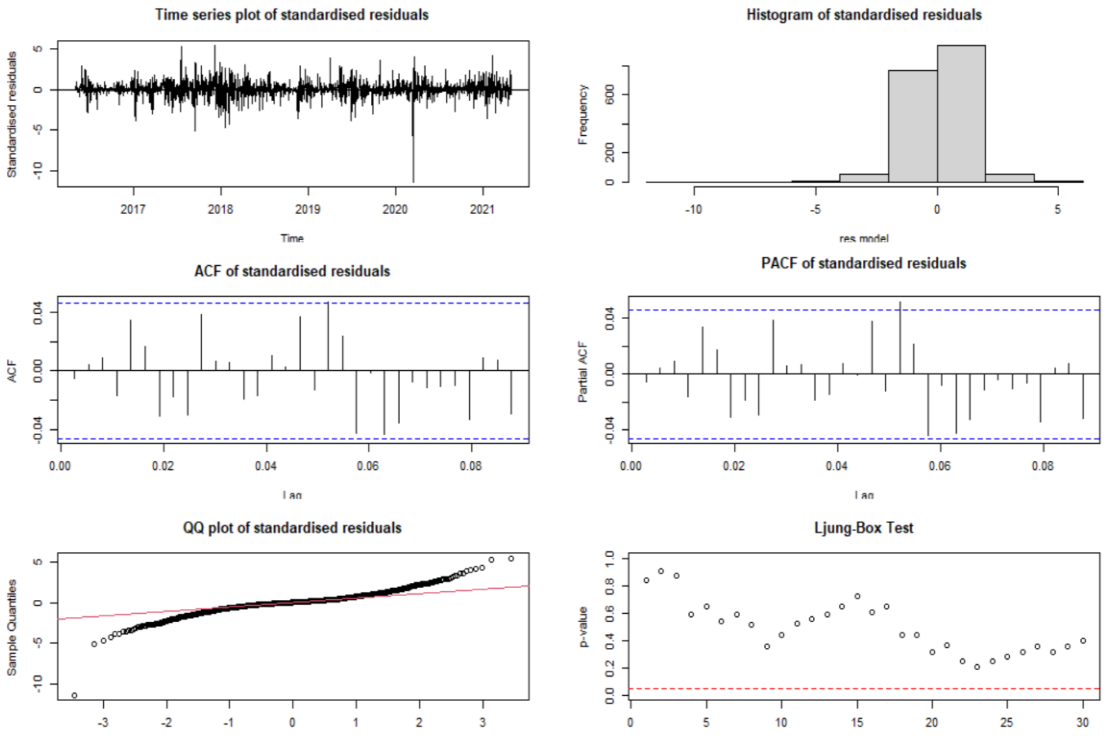

Figure 11: Residual analysis of ARIMA(2,1,2) model

- Residuals plots: Although the plot depicts the model detrend trends in series, the shifting variance in residuals is still highly visible.
- Histogram: unsymmetric.
- Q-Q plot: Normality assumption does not hold for this series. The thick tail implies the ARCH effect in series.
- Shapiro-Wail test: With P-value is less than 0.01, so we rejected the null hypothesis of normal distribution.
- ACF and PACF plot: There are no significant lags seen in the ACF plot, but we can see there is one significant lag in the PACF plot that confirms there are some autocorrelations left in the residuals.
- The Ljung-Box Test: All points are above the red dashed line; we have no evidence to reject the null hypothesis that the error terms are uncorrelated.

Overall, normality did not hold for ARIMA based on the residual analysis shown above (2,1,2). The model does not capture the dependent structure of the Bitcoin time-series very well.

Forecasting the ARIMA Model

We validated that the best fit model is ARIMA(2,1,2) using Model Diagnostics. We forecast the Bitcoin closing price for the following 10 days using this model.

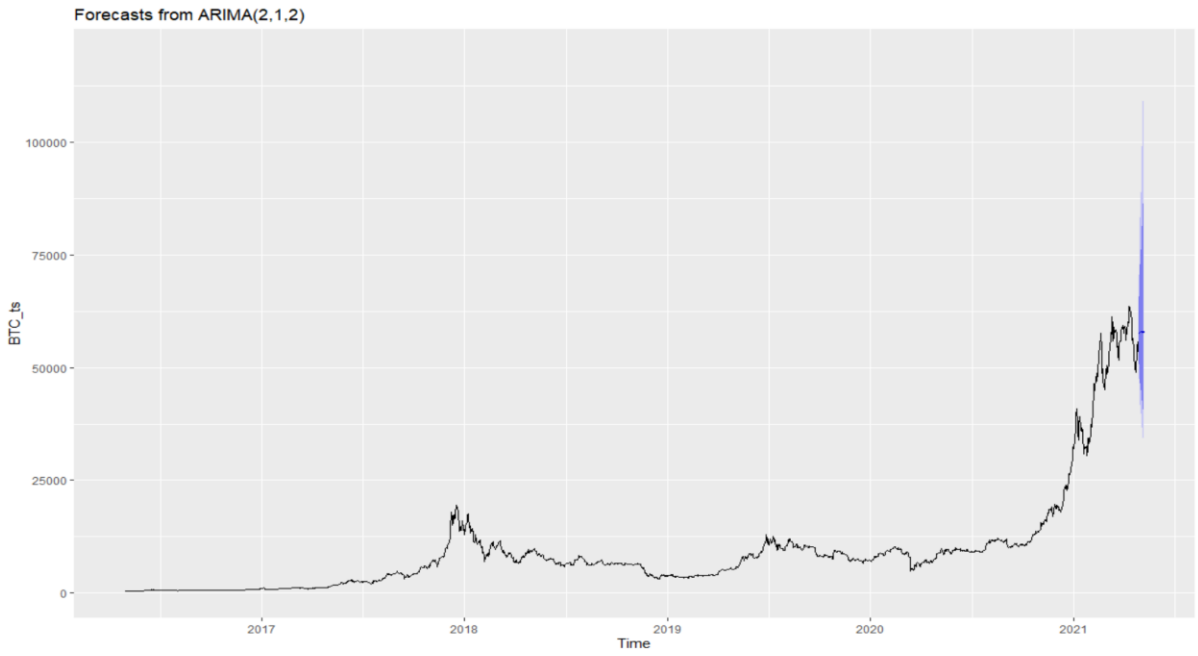

Figure 12: Forecasting plot of Bitcoin closing price for next 10 days

	Point Forecast <dbl>	Lo 80 <dbl>	Hi 80 <dbl>	Lo 95 <dbl>	Hi 95 <dbl>
2021.3233	57639.50	50858.88	65696.78	47700.47	70583.63
2021.3260	57874.53	48566.13	69756.11	44444.95	77399.78
2021.3288	57744.24	46548.77	72893.25	41788.53	83129.98
2021.3315	57913.14	45179.89	75996.57	39940.81	88734.16
2021.3342	57791.38	43779.48	78585.08	38181.18	93804.59
2021.3370	57920.83	42756.13	81325.87	36851.62	99062.48
2021.3397	57815.37	41668.65	83667.95	35529.73	103915.88
2021.3425	57918.10	40838.97	86207.64	34485.09	109058.63
2021.3452	57829.51	39947.93	88416.56	33429.45	113865.69
2021.3479	57912.45	39244.67	90829.49	32567.56	119002.70

The results of the Forecast at 80% and 95% confidence intervals indicate that Bitcoin's closing price in the next 10 days seems to be increasing gradually.

According to the McLeod-Li test, this series is significant at the 5% level of significance for all lags, indicating volatility clustering. Furthermore, even after detrending using the ARIMA model, the shifting variance is visible. As a result, we decide to deal with conditional variance in the next section by employing GARCH models.

ARMA+GARCH Model

GARCH(p,q) order selection process

To determine the GARCH impact, we use log-transformed data and first differencing.

ACF and PACF plots

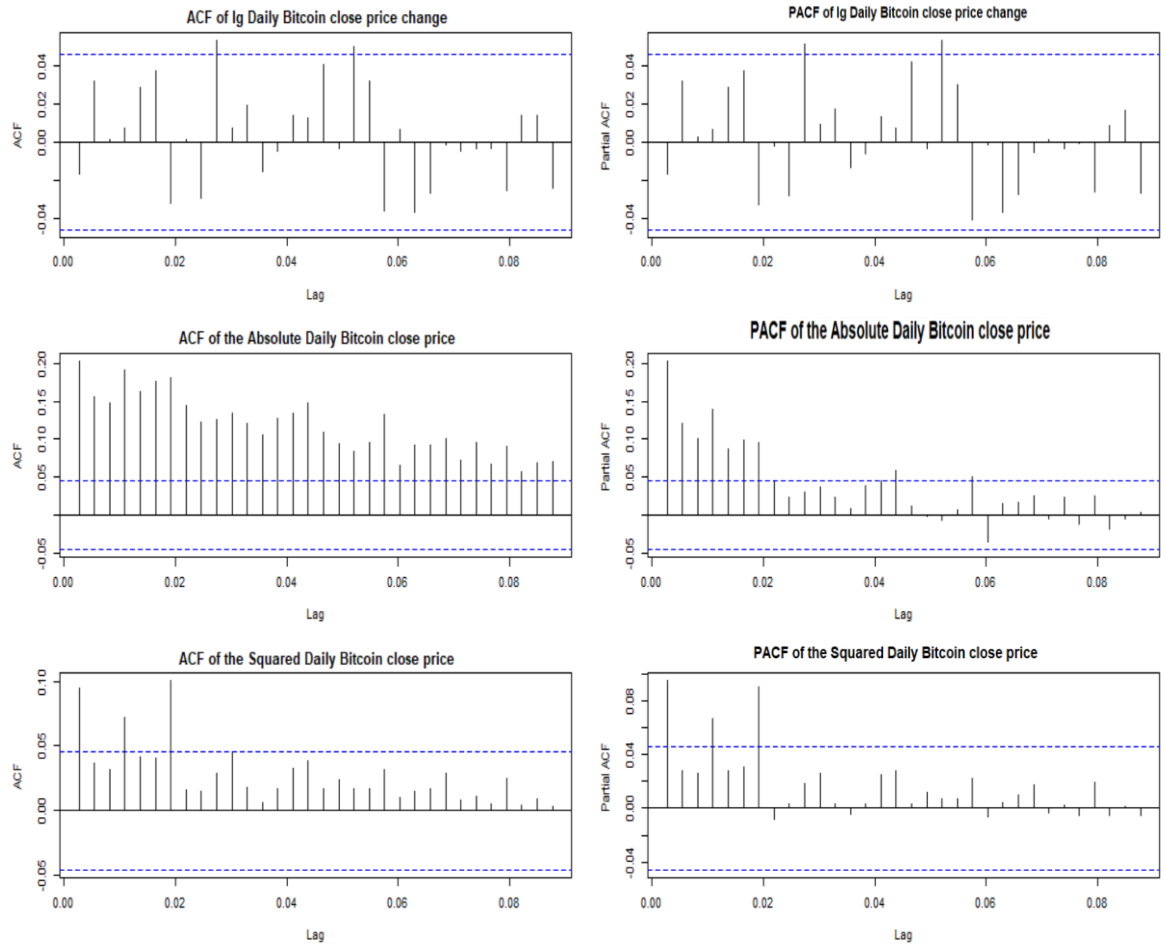

Figure 13: ACF and PACF plot of log-transformed data with the first differencing, absolute value, and squared Daily Bitcoin closing price.

There are several strong correlations in both ACF and PACF with the influence of extremely volatile series for both squared and absolute value series, indicating that the daily closing price of Bitcoin is not distributed independently and identically.

Normality Test for Absolute and Squared Bitcoin closing price.

Absolute Bitcoin Closing Price	Squared Bitcoin Closing Price
Shapiro-Wilk normality test data: abs(r.BTC_ts) W = 0.73116, p-value < 2.2e-16	Shapiro-Wilk normality test data: r.BTC_ts^2 W = 0.20586, p-value < 2.2e-16

From both the Absolute and Squared Bitcoin Closing price, Shapiro-Wilk reveals that normality did not hold.

GARCH Model specifications

To establish the orders of the GARCH model, we will examine the EACF outputs.

Absolute Transformation	Squared Residuals
AR/MA 0 1 2 3 4 5 0 x x x x x x 1 x o o o o o 2 x x o o o o 3 x x x o o o 4 x x x x o o 5 x x x x x o	AR/MA 0 1 2 3 4 5 0 x o o x o o 1 x o o o o o 2 x x o o o o 3 x x x o o o 4 x x x x o o 5 x x x x o o

Form the EACF plot of applying absolute residuals we include GARCH(1,1) and GARCH(1,2) models. And from the EACF plot of applying square transformations, we include GARCH(2,2), GARCH(2,3) models.

Model Diagnostic checking.

GARCH(1,1)

```
Call:
garch(x = r.BTC_ts, order = c(1, 1), trace = FALSE)

Model:
GARCH(1,1)

Residuals:
      Min        1Q    Median        3Q       Max
-12.11775  -0.33316   0.07245   0.49398   6.93134

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 8.591e-05   7.612e-06    11.29   <2e-16 ***
a1 1.672e-01   1.261e-02    13.26   <2e-16 ***
b1 8.062e-01   1.403e-02    57.46   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Diagnostic Tests:
        Jarque Bera Test

data:  Residuals
X-squared = 19406, df = 2, p-value < 2.2e-16


        Box-Ljung test

data:  Squared.Residuals
X-squared = 0.039815, df = 1, p-value = 0.8418
```

```
Call:
 garchFit(formula = ~garch(1, 1), data = r.BTC_ts)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x000001b730f1a400>
 [data = r.BTC_ts]

Conditional Distribution:
 norm

Coefficient(s):
        mu       omega      alpha1       beta1
2.7557e-03  8.7346e-05  1.6418e-01  8.0649e-01

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     2.756e-03   7.730e-04    3.565 0.000364 ***
omega  8.735e-05   1.404e-05    6.222 4.91e-10 ***
alpha1 1.642e-01   2.178e-02    7.536 4.84e-14 ***
beta1  8.065e-01   1.997e-02   40.380  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 3406.395    normalized:  1.869591

Description:
 Mon Jun 07 13:05:50 2021 by user: user


Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  16871.55  0
 Shapiro-Wilk Test  R    W      0.89944   0
 Ljung-Box Test     R    Q(10)  21.09534  0.0204385
 Ljung-Box Test     R    Q(15)  24.59509  0.05565256
 Ljung-Box Test     R    Q(20)  30.70573  0.05919714
 Ljung-Box Test     R^2  Q(10)  3.71517   0.9592828
 Ljung-Box Test     R^2  Q(15)  4.641849  0.994749
 Ljung-Box Test     R^2  Q(20)  5.430693  0.9994766
 LM Arch Test       R    TR^2   3.917826  0.9848739

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-3.734791 -3.722700 -3.734801 -3.730330
```

Residual Analysis of GARCH(1,1)

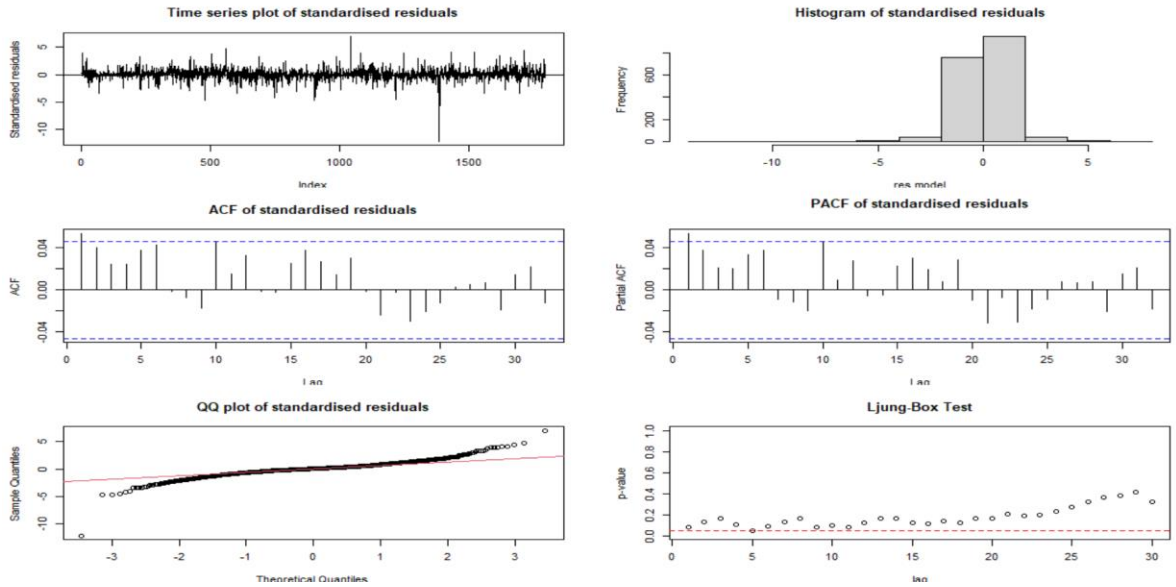

Figure 14: Residual Analysis of GARCH(1,1) Model

We were able to determine that all coefficients are significant at the 5% level of significance by combining two estimate methods. Normality does not hold here. There is considerable autocorrelation left in the residuals.

After performing the same procedure on the remaining GARCH models, we discovered that all of the models included had a similar residual analysis. However, based on the n parameter estimates, it appears that GARCH(1,1) is the best model comparatively, despite the fact that AIC suggested GARCH(2,2). We will continue with the GARCH(1,1) model to anticipate estimated conditional variance.

Description: df [4 x 2]

	df <dbl>	AIC <dbl>
m.22	5	-6797.118
m.11	3	-6789.632
m.12	4	-6782.543
m.23	6	-6751.885

4 rows

ARMA(m,n) order selection process

To determine the optimum order(m, n), we experiment with many combinations and pick the one that produces the best results. The below table demonstrates how ARMA(1,1) has the lowest AIC value.

ARMA(m, n)	AIC
(0,1)	-6493.31
(0,2)	-6492.35
(1,1)	-6494.02
(1,2)	-6490.37
(2,2)	-6490.63

ARMA(1,1)

```
Call:
arma(x = r.BTC_ts, order = c(1, 1))

Model:
ARMA(1,1)

Residuals:
       Min         1Q     Median         3Q        Max
-4.669e-01 -1.519e-02 -1.181e-05  1.628e-02  2.255e-01

Coefficient(s):
           Estimate  Std. Error  t value Pr(>|t|)
ar1       -0.895529    0.080418  -11.136  < 2e-16 ***
ma1        0.877046    0.086164   10.179  < 2e-16 ***
intercept  0.005047    0.001800    2.803  0.00506 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Fit:
sigma^2 estimated as 0.001653,  Conditional Sum-of-Squares = 3.01,  AIC = -6494.02
```

ARMA+GARCH Model

After obtaining the best GARCH and ARMA order, let's check the full model parameters of ARMA(1,1)+GARCH(1,1)

```
*----------------------------------*
*          GARCH Model Fit         *
*----------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(1,1)
Mean Model       : ARFIMA(1,0,1)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
ar1    -0.863121    0.123870  -6.9679        0
ma1     0.834075    0.137711   6.0567        0
omega   0.000092    0.000014   6.4619        0
alpha1  0.208615    0.027187   7.6733        0
beta1   0.773171    0.021472  36.0081        0

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
ar1    -0.863121    0.128779  -6.7023 0.000000
ma1     0.834075    0.152002   5.4872 0.000000
omega   0.000092    0.000032   2.8760 0.004027
alpha1  0.208615    0.059488   3.5068 0.000453
beta1   0.773171    0.027825  27.7873 0.000000

LogLikelihood : 3054.8

Information Criteria
------------------------------------

Akaike       -3.7605
Bayes        -3.7439
Shibata      -3.7606
Hannan-Quinn -3.7544
```

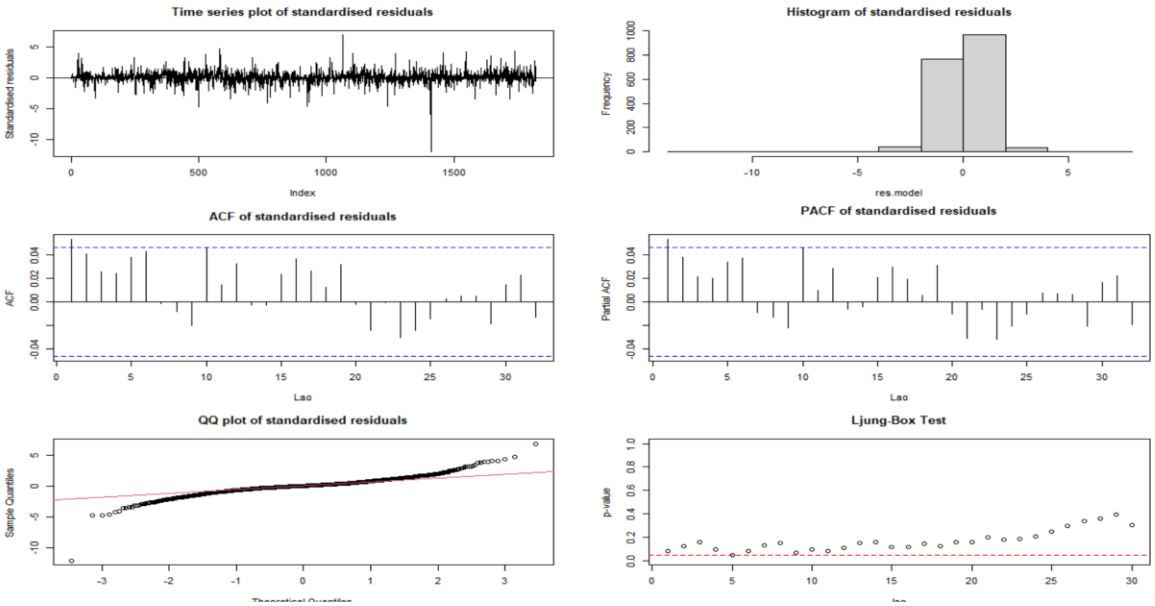

Figure 15: Residual Analysis of the ARMA(1,1)+GARCH(1,1) Model

Forecasting ARMA(1,1)+GARCH(1,1)

We can anticipate future Bitcoin prices (from 2016-04-30 to 2021-01-30) using the methods described above and the resulting ARMA(1,2)-GARCH(1,2) model.

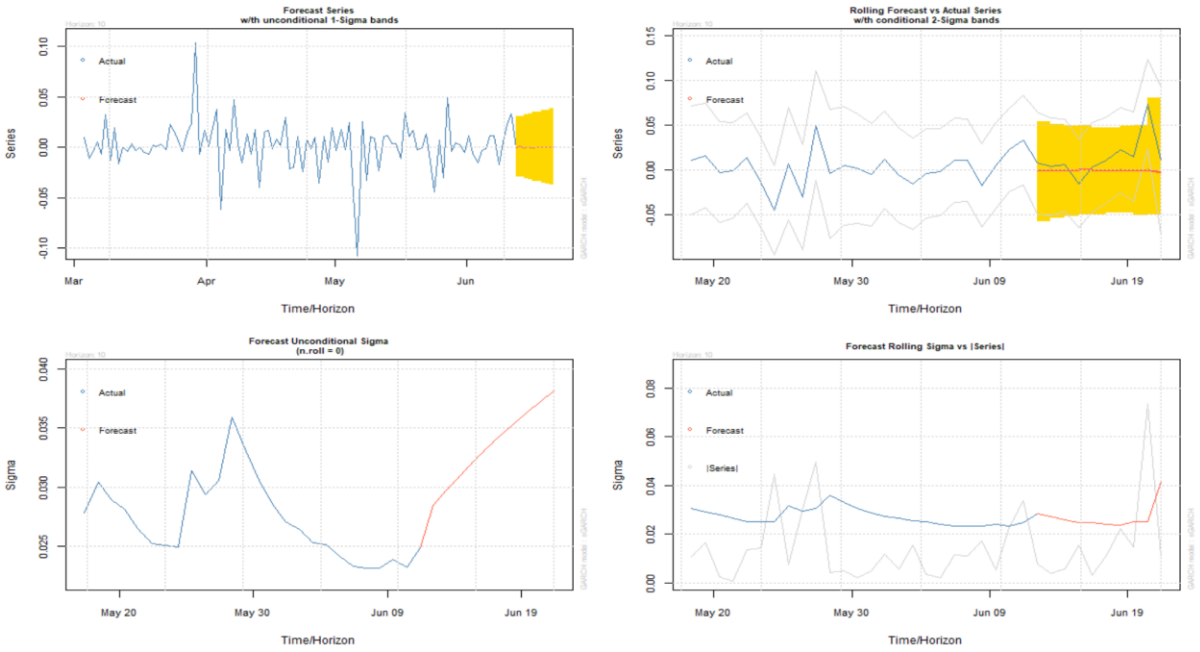

Figure 15: Forecasting of transformed bitcoin series using the ARMA(1,1)+GARCH(1,1) model.

The first of four plots are the standard forecast plot, which suggests a general slightly declining trend in the following 10 days. In general, rolling forecasts are employed for long-term forecasting. We discovered that conditional variance would rise in the future. And it shows that the Bitcoin closing price will continue to fluctuate, with the price gradually increasing in the future.

Our model can roughly forecast the ups and downs of Bitcoin's closing price movement when compared to real values. However, it fails to reflect the considerable volatility of the daily return/price, resulting in a forecast that is relatively stable in comparison to the actual return/price.

Conclusion

In this report, we forecasted future Bitcoin values, one of the most extensively used and traded cryptocurrencies, and investigate the predictive capability of the ARIMA and ARMA-GARCH model on the Bitcoin closing price series. Although the residual analysis is not ideal, ARMA(1,1)+GARCH(1,1) appears to match the log-transformed bitcoin series with the first differencing by coefficient significance tests. Furthermore, the research anticipated bitcoin closing price forecasts for the following 10 days, implying a minor falling trend, but it will climb in the long term and conditional variance will rise as significantly.

References:

- Bitcoin. (2019). CoinMarketCap. https://coinmarketcap.com/currencies/bitcoin/

Appendix

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
#lOADING THE LIBRARIES
library(TSA)
library(tseries)
library(readr)
library(FSAdata)
library(fUnitRoots)
library(forecast)
library(knitr)
library(CombMSC)
library(lmtest)
library(fGarch)
library(FitAR)
library(tswge)
library(rugarch)
#Firstly, read Bitcoin data into R and check the data format.
BTC <- read_csv("D:/Time Series Notes/BTC-USD.csv")
class(BTC)
sum(is.na(BTC))
df<- na.omit(BTC)
sum(is.na(df))
Time <- seq(as.Date("2016-04-30"), as.Date("2021-04-30"), by = "day")
BTC_ts <- ts(df$Close,start = c(2016, as.numeric(format(Time[1], "%j"))),
frequency = 365)
#Time series plot
plot(BTC_ts,type='o',ylab='Closing price',
main='Time Series Plot of Bitcoin Closing Price')
#Scatter Plot
plot(y=BTC_ts,x=zlag(BTC_ts),
```

```
ylab='Closing price',
xlab='Previous day closing price', main = "Scatter plot of Bitcoin")
# Normality Check
qqnorm(BTC_ts)
qqline(BTC_ts, col = 2, lwd = 1, lty = 2)
shapiro.test(BTC_ts)
#ARIMA MODELLING
par(mfrow=c(1,2))
a <- acf(BTC_ts)
b<- pacf(BTC_ts)
plot(a)
plot(b)
par(mfrow=c(1,1))
adf.test(BTC_ts) #indicating non-stationarity
BTC_ts_tr = BoxCox.ar(BTC_ts+abs(min(BTC_ts))+0.1, lambda = c(-1,1))
BTC_ts_tr$ci
par(mfrow=c(1,2))
acf(log(BTC_ts))
pacf(log(BTC_ts))
par(mfrow=c(1,1))
adf.test(log(BTC_ts))
#NON STATIONARY
qqnorm(log(BTC_ts), main = 'QQ plot for the natural log')
qqline(log(BTC_ts))
#First Differencing
diff_log_BTC = diff(log(BTC_ts), difference = 1)
plot(diff_log_BTC, type = 'o', ylab = 'Closing price')
order = ar(diff(diff_log_BTC))$order
adfTest(diff_log_BTC, lags = order, title = NULL, description = NULL)
McLeod.Li.test(y=diff_log_BTC, main="McLeod-Li Test Statistics for Daily Bitcoin price")
#ARIMA MODEL SPECIFICATIONS
par(mfrow=c(1,2))
```

```
acf(diff_log_BTC)
pacf(diff_log_BTC)
eacf(diff_log_BTC,ar.max = 5, ma.max = 5)
par(mfrow = c(1,1))
res = armasubsets(y=diff_log_BTC, nar = 5, nma = 5, y.name = 'test', ar.method = 'ols')
plot(res)
#Fit Models and find estimations
#ARIMA MODEL (0,1,1)
model_011_css = arima(log(BTC_ts),order=c(0,1,1),method='CSS')
coeftest(model_011_css)
model_011_ml = arima(log(BTC_ts),order=c(0,1,1),method='ML')
coeftest(model_011_ml)
# ARIMA MODEL (1,1,1)
model_111_css = arima(log(BTC_ts),order=c(1,1,1),method='CSS')
coeftest(model_111_css)
model_111_ml = arima(log(BTC_ts),order=c(1,1,1),method='ML')
coeftest(model_111_ml)
# ARIMA MODEL (1,1,2)
model_112_css = arima(log(BTC_ts),order=c(1,1,2),method='CSS')
coeftest(model_112_css)
model_112_ml = arima(log(BTC_ts),order=c(1,1,2),method='ML')
coeftest(model_112_ml)
#ARIMA MODEL (2,1,2)
model_212_css = arima(log(BTC_ts),order=c(2,1,2),method='CSS')
coeftest(model_212_css)
model_212_ml = arima(log(BTC_ts),order=c(2,1,2),method='ML')
coeftest(model_212_ml)
# ARIMA MODEL (2,1,0)
model_210_css = arima(log(BTC_ts),order=c(2,1,0),method='CSS')
coeftest(model_210_css)
model_210_ml = arima(log(BTC_ts),order=c(2,1,0),method='ML')
coeftest(model_210_ml)
```

```
#ARIMA(0,1,2)
model_012_css = arima(log(BTC_ts),order=c(0,1,2),method='CSS')
coeftest(model_012_css)
model_012_ml = arima(log(BTC_ts),order=c(0,1,2),method='ML')
coeftest(model_012_ml)

# ARIMA MODEL (2,1,5)
model_215_css = arima(log(BTC_ts),order=c(2,1,5),method='CSS')
coeftest(model_215_css)
model_215_ml = arima(log(BTC_ts),order=c(2,1,5),method='ML')
coeftest(model_215_ml)
#ARIMA(5,1,0).
model_510_css = arima(log(BTC_ts),order=c(5,1,0),method='CSS')
coeftest(model_510_css)
model_510_ml = arima(log(BTC_ts),order=c(5,1,0),method='ML')
coeftest(model_510_ml)
#AIC and BIC vaules
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(stats::AIC(model_111_ml, model_112_ml, model_210_ml, model_012_ml,
model_212_ml,
model_215_ml, model_011_ml,  model_510_ml), c( "aic"))

#RESIDUAL ANALYSIS
doPar <- function(
```

```
  mfrow = c(1,1),
  mai = c(1,0.5,0.5,0.5)
  ) {
  par(
    bg = 'white',
    col = "black",
    col.axis =       'black',
    col.lab = "black",
    col.main = 'black',
    col.sub = 'black',
    fg = 'black',
    mai = mai,
    mfrow = mfrow
  )
}
residual.analysis <- function(
  model,
  std = TRUE,start = 2,
  class = c("ARIMA","GARCH","ARMA-GARCH")[1],
  title = ''
  ){
  # If you have an output from arima() function use class = "ARIMA"
  # If you have an output from garch() function use class = "GARCH"
  # If you have an output from ugarchfit() function use class = "ARMA-GARCH"
  library(TSA)
  library(FitAR)
  if (class == "ARIMA"){
    if (std == TRUE){
      res.model = rstandard(model)
    }else{
      res.model = residuals(model)
    }
```

```
 }else if (class == "GARCH"){
  res.model = model$residuals[start:model$n.used]
 }else if (class == "ARMA-GARCH"){
  res.model = model@fit$residuals
 }else {
  stop("The argument 'class' must be either 'ARIMA' or 'GARCH' ")
 }
 doPar(mfrow = c(3,2), mai = c(0.5,0.5,0.5,0.5))
 plot(res.model,type='l',ylab='Standardised residuals', main="Time series plot of standardised
residuals")
 abline(h=0)
 hist(res.model,main="Histogram of standardised residuals")
 acf(res.model,main="ACF of standardised residuals")
 pacf(res.model,main="PACF of standardised residuals")
 qqnorm(res.model,main="QQ plot of standardised residuals")
 qqline(res.model, col = 2)
 title(title, line = -1, outer = TRUE)
 print(shapiro.test(res.model))
 k=0
 LBQPlot(res.model, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
}



residual.analysis(model = model_212_css, class = "ARIMA" )


r.BTC_ts = diff(log(BTC_ts))


fit2 <- Arima(BTC_ts, order=c(2,1,2), lambda = "auto")
summary(fit2)


autoplot(forecast(fit2, h=10))
forecast(fit2, h=10)
```

```
# Modelling GARCH model
par(mfrow=c(3,2))
acf(r.BTC_ts,main=" ACF of lg Daily Bitcoin close price change")
pacf(r.BTC_ts,main=" PACF of lg Daily Bitcoin close price change")
acf(abs(r.BTC_ts),main=" ACF of the Absolute Daily Bitcoin close price")
pacf(abs(r.BTC_ts),main=" PACF of the Absolute Daily Bitcoin close price")
acf(r.BTC_ts^2,main=" PACF of the Squared Daily Bitcoin close price")
pacf(r.BTC_ts^2,main=" PACF of the Squared Daily Bitcoin close price")

#Test for Absolute value

McLeod.Li.test(y=abs(r.BTC_ts), main = "Mcleod-Li Test for Absolute Daily Bitcoin Closing
Price")

qqnorm(abs(r.BTC_ts), main="Q-Q Normal Plot of the Absolute Daily Bitcoin closing price")
qqline(abs(r.BTC_ts))
shapiro.test(abs(r.BTC_ts))

#Test for Squared Value

McLeod.Li.test(y=r.BTC_ts^2,main="McLeod-Li Test Statistics for the Squared Daily Bitcoin
price")
qqnorm(r.BTC_ts^2, main="Q-Q Normal Plot of the Squared Daily Bitcoin price")
qqline(r.BTC_ts^2)
shapiro.test(r.BTC_ts^2)

eacf(r.BTC_ts,ar.max = 5, ma.max = 5)

eacf(abs(r.BTC_ts))

eacf(r.BTC_ts^2)
```

```
#GARCH(1,1)

m.11 = garch(r.BTC_ts,order=c(1,1),trace = FALSE)
summary(m.11) # All coefficients are significant at 5% level of significance.
residual.analysis(m.11, class = "GARCH", start = 25)
m.11_1 = garchFit(formula = ~garch(1,1), data =r.BTC_ts )
summary(m.11_1)

#GARCH(1,2)

m.12 = garch(r.BTC_ts,order=c(1,2),trace = FALSE)
summary(m.12) # All coefficients are significant at 5% level of significance.
m.12_2 = garchFit(formula = ~garch(1,2), data =r.BTC_ts )
summary(m.12_2)
residual.analysis(m.12,class="GARCH",start=3)
#GARCH (2,2)
m.22 = garch(r.BTC_ts,order=c(2,2),trace = FALSE)
summary(m.22) # Not all coefficients are significant at 5% level of significance.
m.22_2 = garchFit(formula = ~garch(2,2), data =r.BTC_ts )
summary(m.22_2)
residual.analysis(m.22,class="GARCH",start=11)
#GARCH (2,3)
m.23 = garch(r.BTC_ts,order=c(2,3),trace = FALSE)
summary(m.23) # Not all coefficients are significant at 5% level of significance.
m.23_2 = garchFit(formula = ~garch(2,3), data =r.BTC_ts )
summary(m.23_2)
residual.analysis(m.23,class="GARCH",start=4)
sort.score(AIC(m.11,m.12,m.22,m.23), score = "aic")
# Conditional Variance Prediction
plot((fitted(m.11)[,1])^2,type='l',
ylab='Conditional Variance',
```

```
xlab='time',main=" Conditional Variances of the Daily Bitcoin Closing Price")
fGarch::predict(m.11_1,n.ahead=10,trace=FALSE,plot=TRUE)
# ARMA model
ar.model.01 = arma(r.BTC_ts, order= c(0,1))
summary(ar.model.01)
ar.model.02 = arma(r.BTC_ts, order= c(0,2))
summary(ar.model.02)
ar.model.11 = arma(r.BTC_ts, order= c(1,1))
summary(ar.model.11)
ar.model.12 = arma(r.BTC_ts, order= c(1,2))
summary(ar.model.12)
ar.model.22 = arma(r.BTC_ts, order= c(2,2))
summary(ar.model.22)
# ARMA(1,1) + GARCH(1,1)
model1<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
            mean.model = list(armaOrder = c(1, 1), include.mean = FALSE),
            distribution.model = "norm")
m.11_11<-ugarchfit(spec=model1,data=r.BTC_ts, out.sample = 200)
m.11_11
par(mfrow=c(2,2))
plot(m.11_11, which=8)
plot(m.11_11, which=9)
plot(m.11_11, which=10)
par(mfrow=c(1,1))
# Forecasting
forc.11_11 = ugarchforecast(m.11_11, data = r.BTC_ts, n.ahead = 10,n.roll = 10)
plot(forc.11_11, which = "all")
forc.11_11@forecast$seriesFor
rev.100 <- as.vector(forc.11_11@forecast$seriesFor/100) # divided by 100 to revert the
rev.100
```