Time Series Analysis of Daily Bitcoin Closing Price Data from 27th of April 2013 to the 3rd of March 2018.

MATH1318 - Time Series Analysis

Abhishekh Shankar, Wesley Nderi
RMIT

May 26, 2018

1 Introduction

Bitcoin is a peer-to-peer cryptocurrency that was created by an unknown person or group of persons known as Satoshi Nakamoto that relies on encryption techniques to regulate the generation of coins. A dataset representing daily closing prices of bitcoin (in USD) from the 27th of April 2013 to the 3rd of March 2018 was provided and the R programming language was utilised as a means for time series analysis. Initially, a GARCH model was specified, followed by an ARMA and GARCH combination, and their quality of fit analysed. Finally, predictions were made on the closing prices of Bitcoin for the following 10 days from the 4th of March to the 13th of March 2018 in an effort to forecast the behaviour of Bitcoin prices.

2 Bitcoin Data Inspection

After reading in the dataset and converting to a time series object, a time series plot of the data is shown in Figure 1 below alongside the appropriate R code which is displayed in Listing 1[1].

```
# Import the data set.
bitcoin <- read.csv("~/RMIT/2018 Semester One/Time Series Analysis/Final
    Project/Bitcoin/Bitcoin_Historical_Price.csv")

# Create a daily date object for time series conversion.
date <- seq(as.Date("2013-04-27"), as.Date("2018-03-03"), by="day")

# Convert to a ts object.
bitcoin.ts <- ts(bitcoin$Close, start = c(2013, as.numeric(format(date[1], "%j"))),
    frequency = 365)

# Plot the time series data.
plot(bitcoin.ts, ylab='Closing Price ($)', main="Time Series Plot of Bitcoin Closing
    Price Data")
```

Listing 1: Code snippet required for reading in the dataset, converting to a time series object and plotting a time series plot.

[1] All R code is found in Appendix One and all used R functions shown in Appendix Two.

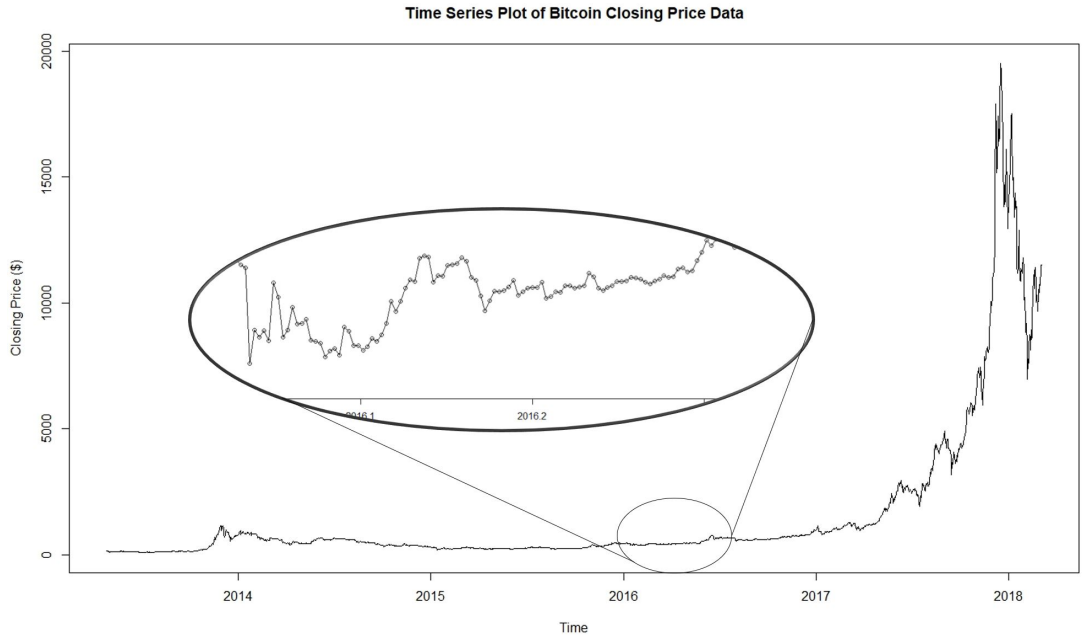


Figure 1: Time series plot of Bitcoin closing prices between 2013 and 2018. A close-up inspection of Bitcoin time series plot is shown.

The time series plot shown in Figure 1 above shows multiple trends and periods where volatitility is both high and low which implies the existence of an ARCH effect. The zoomed in portion of the plot allows for a clearer viewing of the behaviour of the dataset and it can be seen that it exhibits both autoregressive and moving average behaviours. Furthermore, there is no sign of seasonality in the plot. From this, the data can be converted to a return series and a GARCH model applied.

The return series is accomplished by taking the first difference of the log-transformed dataset. Listing 2 shows the code snippet required for the necessary computations and the resulting plot of the return series is shown below in Figure 2.

```
# Convert the series to a return series.
r.bitcoin <- diff(log(bitcoin.ts))

# Plot the return series.
plot(r.bitcoin, main = "Daily Return Series for Bitcoin Price Data", ylab = 'Closing
    Price ($)')
```

Listing 2: Code snippet for the conversion and plotting of the bitcoin dataset to a return series.

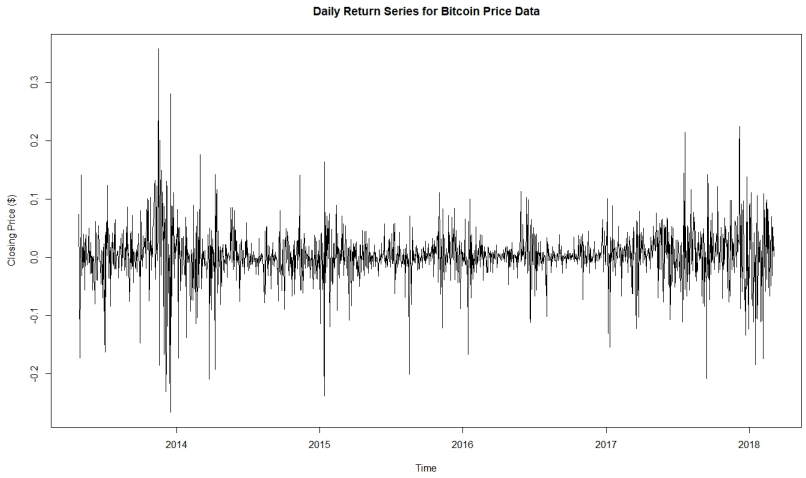


Figure 2: Plot of the daily return series of the bitcoin closing price data.

The plot of the return series shows obvious volatility and there is no sign of a trend. Figures 3, 4, and 5 show the QQ plot, ACF, PACF, and EACF plots of the return series respectively, and the code snippets can be found in Listings 3, 4, and 5.

```
qqnorm(r.bitcoin, main = "QQ Plot of Daily Return Bitcoin Price Series.")
qqline(r.bitcoin, col = 2)
```

Listing 3: Code snippet for the QQ plot.

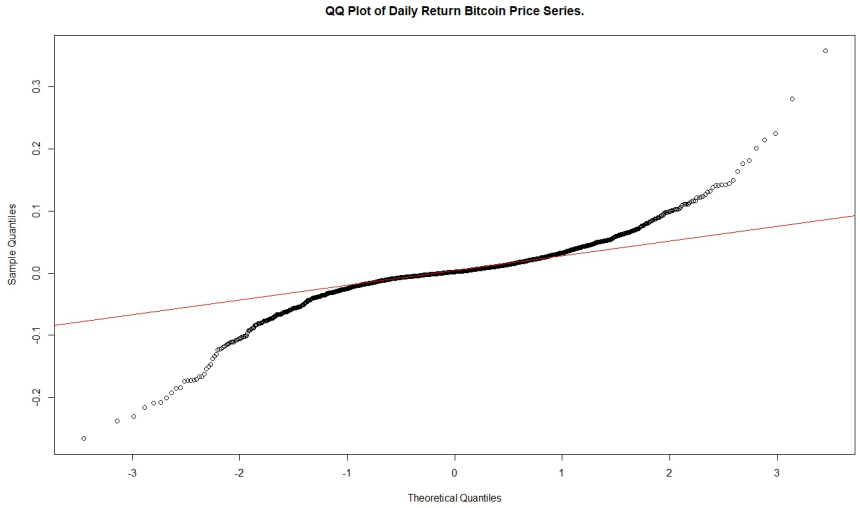


Figure 3: QQ plot of the daily return bitcoin price series.

```
par(mfrow=c(1,2))
acf(r.bitcoin, main = "The Sample ACF Plot for the Daily Return Bitcoin Price Series.")
pacf(r.bitcoin, main = "The Sample PACF Plot for the Daily Return Bitcoin Price Series.")
par(mfrow=c(1,1))
```

Listing 4: Code snippet for the ACF and PACF plots of the bitcoin return series.

Figure 4: ACF and PACF plots of the daily return bitcoin price series.

```
eacf(r.bitcoin)
```

Listing 5: Code snippet for the EACF plot of the bitcoin return series.

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

Figure 5: EACF plot of the daily return bitcoin price series.

The QQ plot displays tails which deviate from the line of normality and while both the ACF and PACF plots show multiple significant correlations, the EACF plot seems to suggest an ARMA(0, 0) model.

The McLeod-Li test was run using the code snipet shown in Listing 6, with a plot of the results shown in Figure 6. The test proved to be highly significant which corresponds to the normality assumption being violated. Furthermore, the results of the McLeod-Li test agree with that of the QQ plot.

```
McLeod.Li.test(y=r.bitcoin, main = "McLeod-Li Test Statistics for Daily Return Series.")
```

Listing 6: Code snippet for the EACF plot of the bitcoin return series.

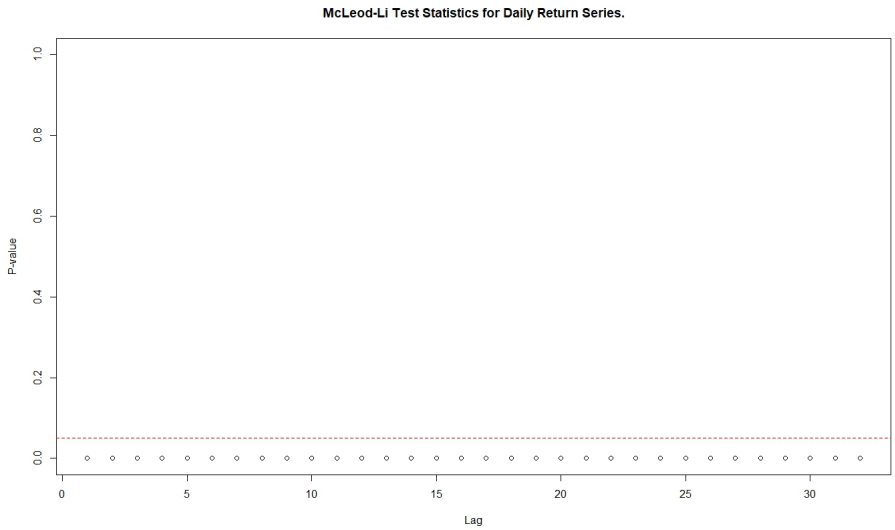


Figure 6: EACF plot of the daily return bitcoin price series.

3 GARCH Model Specification and Selection

In order to determine the parameters of the GARCH model, both an absolute value and a squared return series was computed. The code for the two transformations are shown below in Listing 7.

```
# Compute the absolute value and squared series of the return.
abs.r.bitcoin = abs(r.bitcoin)
sq.r.bitcoin = r.bitcoin^2
```

Listing 7: Code snippet for the absolute value and squared series of the return.

The ACF and PACF plots for the squared series are shown below in Figure 7 and the plot of the EACF are shown in Figure 8. There are multiple significant correlations in both the ACF and PACF of the squared return series and since the vertex of the EACF can be taken at the position (2, 2), the suggested model is no longer an ARMA(0, 0) model, but potentially from a list including:

- ARMA(2, 2)
- ARMA(2, 3)
- ARMA(3, 3)

```
par(mfrow=c(1,2))
acf(sq.r.bitcoin, ci.type="ma", main = "The Sample ACF Plot for the Squared Bitcoin
    Price Series.")
pacf(sq.r.bitcoin, main = "The Sample PACF Plot for the Squared Bitcoin Price Series.")
par(mfrow=c(1,1))
```

Listing 8: Code snippet for the ACF and PACF plots of the squared bitcoin return series.

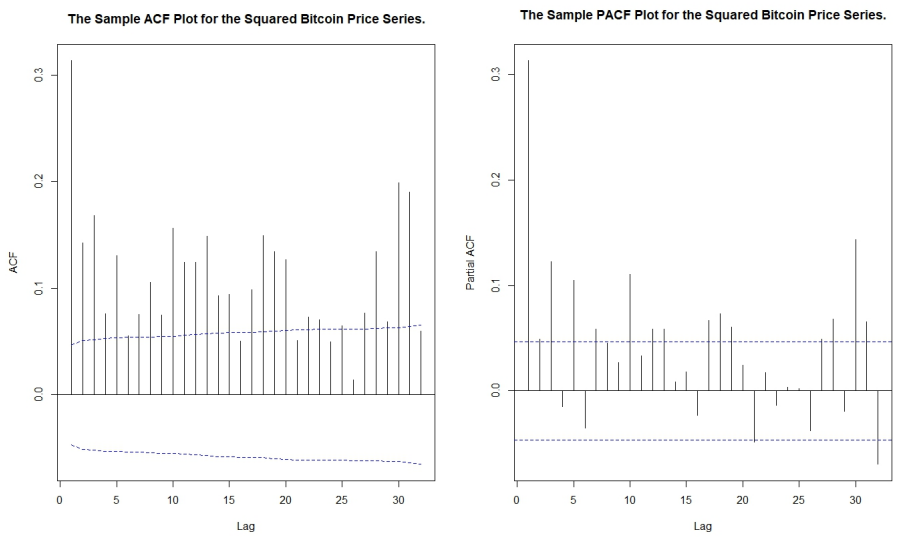


Figure 7: ACF and PACF plots of the squared daily return bitcoin price series.

```
eacf(sq.r.bitcoin)
```

Listing 9: Code snippet for the EACF plot of the squared bitcoin return series.

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

Figure 8: EACF plot of the squared daily return bitcoin price series.

```
par(mfrow=c(1,2))
acf(abs.r.bitcoin, ci.type="ma", main = "The Sample ACF Plot for the Bitcoin Price
    Series.")
pacf(abs.r.bitcoin, main = "The Sample PACF Plot for the Bitcoin Price Series.")
par(mfrow=c(1,1))
```

Listing 10: Code snippet for the ACF and PACF plots of the absolute value bitcoin return series.

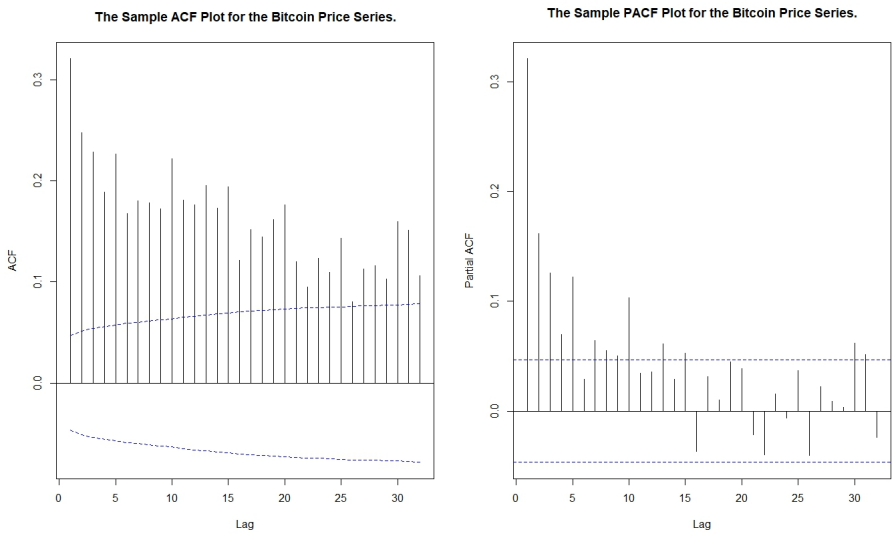


Figure 9: ACF and PACF plots of the absolute value daily return bitcoin price series.

```
eacf(abs.r.bitcoin)
```

Listing 11: Code snippet for the EACF plot of the absolute value bitcoin return series.

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

Figure 10: EACF plot of the squared daily return bitcoin price series.

Similarly, the ACF and PACF plots for the absolute value return series is shown in Figure 9 and the resulting EACF is shown in Figure 10.

Like that of the squared return series, the ACF and PACF of the absolute value return series shows many signifiant correlations. The vertex of the EACF can be taken at the position (1, 1) which corresponds the following potential models:

- ARMA(1, 1)
- ARMA(1, 2)
- ARMA(2, 2)

From this, the potential models presented from both the squared and absolute value return series correspond to parameter settings of:

- (max(1, 1), 1)
- (max(1, 2), 2)
- (max(2, 2), 2)
- (max(3, 3), 3)

Therefore, the set of possible GARCH models are:

- GARCH(1, 1)
- GARCH(2, 2)
- GARCH(3, 3)

The code snippet required to fit the GARCH(1, 1) is presented in Listing 12, while the snippets for fitting both the GARCH(2, 2) and GARCH(3, 3) models can be found in Appendix One.

Similarly, the model summary for the GARCH(1, 1) model is presented in Figure 11 and the model summaries for all GARCH models fitted are found in Appendix Three.

```
# Fit GARCH(1, 1) model.
model.11 = garch(r.bitcoin, order= c(1,1), trace = FALSE)
# Produce model summary.
summary(model.11)

# Fit GARCH(1, 1) model using different method.
model.11_2 = garchFit(formula ~garch(1,1), data = bitcoin.ts)
# Produce model summary.
summary(model.11_2)
```

Listing 12: Code snippet for fitting the GARCH(1, 1) model

```
##
## Call:
## garch(x = r.bitcoin, order = c(1, 1), trace = FALSE)
##
## Model:
## GARCH(1,1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.55811 -0.33740  0.06798  0.53194  4.76716
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 3.632e-05   3.816e-06    9.518   <2e-16 ***
## a1 1.485e-01   9.555e-03   15.546   <2e-16 ***
## b1 8.522e-01   7.025e-03  121.310   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 5626.9, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 1.7522, df = 1, p-value = 0.1856
```

Figure 11: Model summary for the fitted GARCH(1, 1) model.

As seen in Figure 11, the GARCH(1, 1) model resulted in all highly significant parameters. Conversely, the GARCH(2, 2) model had an insignificant $a2$ parameter, while the GARCH(3, 3) resulted in insignificant $a2$, $b1$, and $b3$ parameters.

Utilising the AIC information criteria for each of the models allows another method of determining the ideal model to fit. The code snippet required is presented in Listing 13 below, with the results of the score ranking shown in Figure 12.

```
# Rank all GARCH models using their AIC information criteria.
sort.score(AIC(model.11, model.22, model.33), score = "aic")
```

Listing 13: Code snippet required to rank the GARCH models in order of their AIC information criteria.

```
##           df       AIC
## model.11   3 -6633.729
## model.22   5 -6604.584
## model.33   7 -6597.135
```

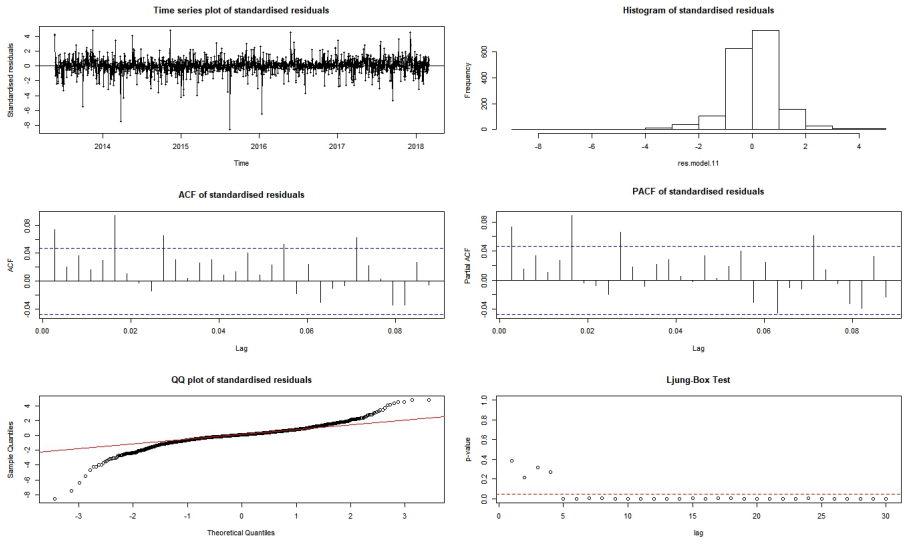
Figure 12: Model summary for the fitted GARCH(1, 1) model.

The results of the ranking indicate that the best model to fit is the GARCH(1, 1) model and this aligns with the results of the model summary which determined that it was the only model of the three that had all significant parameters.

A final comparison of models is a look into their residuals. Appendix Four shows all the residuals for each of the GARCH models alongside their code snippets and it can be observed that for every model, the residuals are similar.

Figure 13 shows the residuals of the GARCH(1, 1) model. There does not appear to be a trend in the time series plot of the residuals, and the Shapiro-Wilk test is highly significant for all models indicating non-normality of residuals at a 5% level of significance.

Given that the residuals of each of the potential GARCH models do not differ significantly, the results of the ranking of the AIC information criteria and parameter significance is sufficient enough to conclude that the GARCH(1, 1) model is the ideal model to fit to the data in order to account for the conditional variance in the dataset.




```
##
##  Shapiro-Wilk normality test
##
## data:  res.model.11
## W = 0.9021, p-value < 2.2e-16
```

Figure 13: GARCH(1, 1) model residuals.

4 ARMA + GARCH Model Speicification and Selection

A number of ARMA models is fit alongside the GARCH(1, 1) model and the results of the model fitting is found in Appendix Five alongside their respective code snippets. Table 1 serves as a summary of each of the fitted models listing each of the significant and insignificant components.

Table 1: ARMA + GARCH model summary table.

Fitted Model	**Significant**	**Insignificant**
ARMA(1, 1) + GARCH(1, 1)	AR(1)	MA(1)
ARMA(1, 2) + GARCH(1, 1)	AR(1)	MA(1), MA(2)
ARMA(2, 1) + GARCH(1, 1)	AR(1), AR(2), MA(1)	Nil
ARMA(3, 1) + GARCH(1, 1)	AR(1), AR(2), AR(3)	MA(1)
ARMA(3, 2) + GARCH(1, 1)	AR(1), AR(2), AR(3), MA(1), MA(2)	Nil
ARMA(4, 2) + GARCH(1, 1)	AR(1), AR(2), AR(3), AR(4), MA(1), MA(2)	Nil
ARMA(5, 2) + GARCH(1, 1)	AR(1), AR(2), AR(3), MA(1), MA(2)	AR(4), AR(5)
ARMA(4, 3) + GARCH(1, 1)	AR(1), AR(2), AR(3), AR(4)	MA(3)

After steadily increasing the AR and MA components and establishing their significance, the list of candidate models was determined to be:

- ARMA(2, 1) + GARCH(1, 1)
- ARMA(3, 2) + GARCH(1, 1)
- ARMA(4, 2) + GARCH(1, 1)

The ARMA(4, 2) + GARCH(1, 1) model was successfully overfit in both the MA and AR components with the ARMA(5, 2) resulting in insignificance in the AR(4) and AR(5) parameters, and the ARMA(4, 3) resulting in an insignificant MA(3) parameter.

Table 2: ARMA + GARCH candidate model information criteria ranking.

Fitted Model	**Akaike**	**Bayes**	**Shibata**	**Hannan-Quinn**
ARMA(2, 1) + GARCH(1, 1)	9.2457	9.2674	9.2457	9.2537
ARMA(3, 2) + GARCH(1, 1)	**9.2353**	**9.2632**	**9.2353**	**9.2456**
ARMA(4, 2) + GARCH(1, 1)	9.2377	9.2686	9.2376	9.2491

Finally, a look into the information criteria for each of the candidate models is shown in Table 2 and from this, it can be observed that the ideal model is the ARMA(3, 2) + GARCH(1, 1) model.

5 Forecasting

While the ARMA(3, 2) + GARCH(1, 1) model has been determined to be the best of the candidate models through the results of the information criteria, all three of the candidate models was utilised for forecasting in order to determine the ideal model through the results.

Figure 14 shows the results of the forecast on the ARMA(2, 1) + GARCH(1, 1) model with the respective code snippet shown in Listing 14.

```
# Compute the forecast.
forc.11.21 = ugarchforecast(m.11.21, data = bitcoin.ts, n.ahead = 10)
# View the forecast values.
forc.11.21
```

Listing 14: Code snippet for forecasting with the ARMA(2, 1) + GARCH(1, 1) model.

```
##
## *-----------------------------------*
## *       GARCH Model Forecast        *
## *-----------------------------------*
## Model: sGARCH
## Horizon: 10
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=1974-11-08 11:00:00]:
##      Series Sigma
## T+1   11498 394.9
## T+2   11487 394.7
## T+3   11478 394.5
## T+4   11472 394.3
## T+5   11467 394.1
## T+6   11464 393.9
## T+7   11463 393.7
## T+8   11463 393.5
## T+9   11464 393.4
## T+10  11465 393.2
```

Figure 14: Forecast summary for the ARMA(2, 1) + GARCH(1, 1) model.

The results of the forecast indicate that the closing price of Bitcoin decreases from \$11,498 on the first day to \$11,465 on the tenth day. This behaviour is seen in the plot of the forecast shown in Figure 15 below.

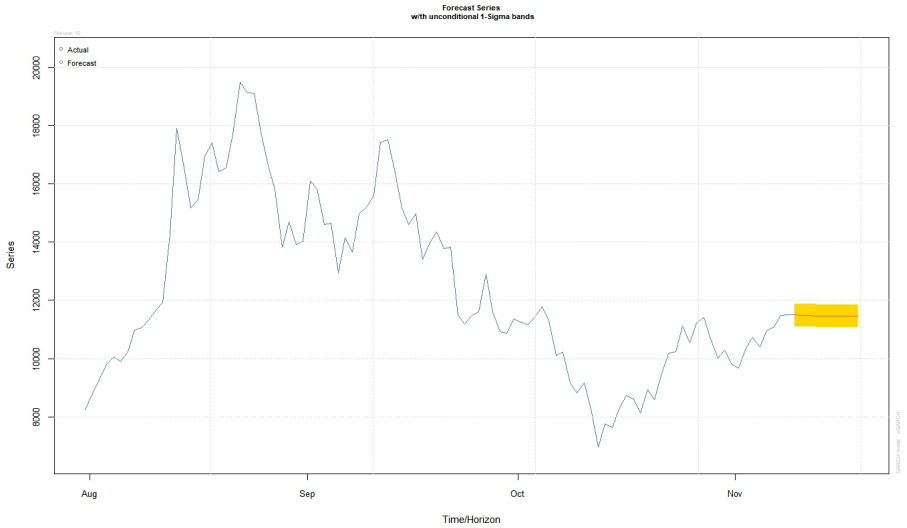


Figure 15: Forecast plot for the ARMA(2, 1) + GARCH(1, 1) model.

Figure 16 shows the forecast summary on the ARMA(3, 2) + GARCH(1, 1) model with its respective code listing shown in Listing 15 and the plot of the forecast shown in Figure 17.

```
# Compute the forecast.
forc.11.32 = ugarchforecast(m.11.32, data = bitcoin.ts, n.ahead = 10)
# View the forecast values.
forc.11.32
```

Listing 15: Code snippet for forecasting with the ARMA(3, 2) + GARCH(1, 1) model.

```
##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
## Model: sGARCH
## Horizon: 10
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=1974-11-08 11:00:00]:
##      Series Sigma
## T+1   11627 426.0
## T+2   11584 425.8
## T+3   11453 425.6
## T+4   11445 425.4
## T+5   11557 425.2
## T+6   11585 425.0
## T+7   11477 424.8
## T+8   11408 424.6
## T+9   11481 424.4
## T+10  11556 424.2
```

Figure 16: Forecast summary for the ARMA(3, 2) + GARCH(1, 1) model.

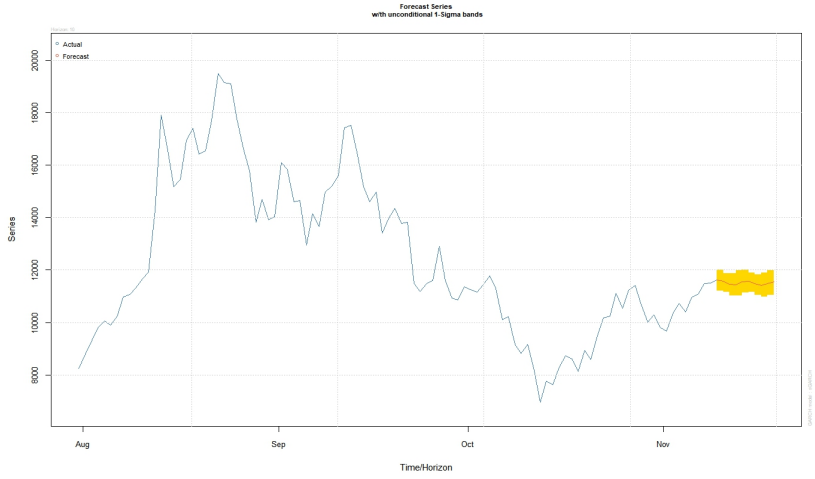


Figure 17: Forecast plot for the ARMA(3, 2) + GARCH(1, 1) model.

The ARMA(3, 2) + GARCH(1, 1) model predicts the closing price of Bitcoin to fluctuate between $11,627 on the first day and the lowest price of $11,408 on the eighth day before slowly rising again until the tenth day to $11,556.

Finally, Figure 18 shows the forecast summary on the ARMA(4, 2) + GARCH(1, 1) model with its respective code listing shown in Listing 16 and the plot of the forecast shown in Figure 19.

```
# Compute the forecast.
forc.11.42 = ugarchforecast(m.11.42, data = bitcoin.ts,n.ahead = 10)
# View the forecast values.
forc.11.42
```

Listing 16: Code snippet for forecasting with the ARMA(4, 2) + GARCH(1, 1) model.

```
##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
## Model: sGARCH
## Horizon: 10
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=1974-11-08 11:00:00]:
##      Series Sigma
## T+1   11633 428.3
## T+2   11596 428.1
## T+3   11462 427.9
## T+4   11451 427.7
## T+5   11570 427.5
## T+6   11608 427.2
## T+7   11496 427.0
## T+8   11419 426.8
## T+9   11498 426.6
## T+10  11587 426.4
```

Figure 18: Forecast summary for the ARMA(4, 2) + GARCH(1, 1) model.

The ARMA(4, 2) + GARCH(1, 1) model predicts the closing price of Bitcoin to fluctuate between $11,633 on the first day and the lowest price of $11,419 on the eighth day before slowly rising again until the tenth day to $11,587.

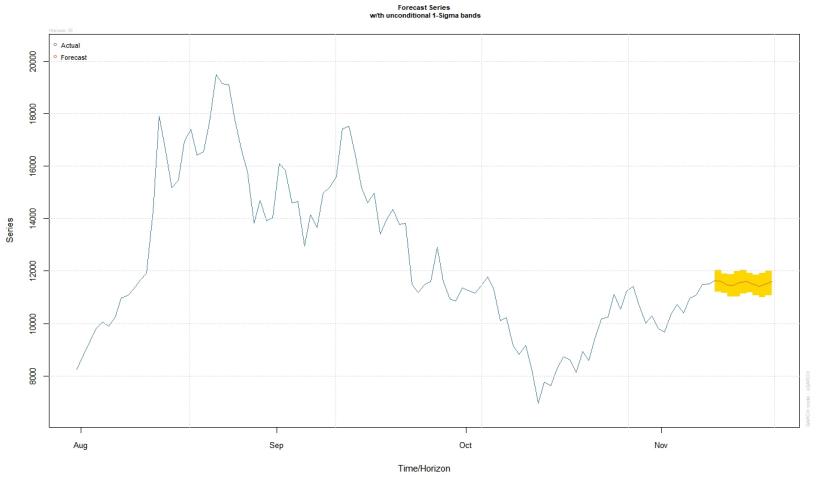


Figure 19: Forecast plot for the ARMA(4, 2) + GARCH(1, 1) model.

6 Mean Absolute Scaled Error

The Mean Absolute Scaled Error (MASE) was utilised in order to compare both the results of the fitted model to the dataset and the results of the forecasts from each of the candidate models and compare it to actual observed values over the forecasting period. The MASE R code function can be seen in Appendix Two.

Listing 17 shows the code snippets for each of the MASE calculations for the three candidate models regarding their forecasts. Table 3 below illustrates the results.

```
# ARMA(2, 1) + GARCH(1, 1)
MASE(as.vector(btc.forecasts), as.vector(forc.11.21@forecast$seriesFor))
# ARMA(3, 2) + GARCH(1, 1)
MASE(as.vector(btc.forecasts), as.vector(forc.11.32@forecast$seriesFor))
# ARMA(4, 2) + GARCH(1, 1)
MASE(as.vector(btc.forecasts), as.vector(forc.11.42@forecast$seriesFor))
```

Listing 17: Code snippet for the MASE values of the candidate models regarding forecasting.

Table 3: Candidate models' MASE forecast results.

Fitted Model	**MASE**
ARMA(2, 1) + GARCH(1, 1)	3.61
ARMA(3, 2) + GARCH(1, 1)	3.67
ARMA(4, 2) + GARCH(1, 1)	3.71

From this, it can be seen that the candidate model with the most accurate forecasting results according to the MASE is the ARMA(2, 1) + GARCH(1, 1) model.

Simultaneously, Listing 18 shows the code snippets for each of the MASE calculations for the three candidate models regarding their model fitting. Table 4 below illustrates the results.

```
# ARMA(2, 1) + GARCH(1, 1)
MASE(as.vector(bitcoin.ts),as.vector(m.11.21@fit$fitted.values))
# ARMA(3, 2) + GARCH(1, 1)
MASE(as.vector(bitcoin.ts),as.vector(m.11.32@fit$fitted.values))
# ARMA(4, 2) + GARCH(1, 1)
MASE(as.vector(bitcoin.ts),as.vector(m.11.42@fit$fitted.values))
```

Listing 18: Code snippet for the MASE values of the candidate models regarding model fitting.

Table 4: Candidate models' MASE model fitting results.

Fitted Model	**MASE**
ARMA(2, 1) + GARCH(1, 1)	1.00
ARMA(3, 2) + GARCH(1, 1)	1.02
ARMA(4, 2) + GARCH(1, 1)	1.01

From this, it can be seen that the candidate model with the most accurate fitting results according to the MASE is the ARMA(2, 1) + GARCH(1, 1) model. However, each of the models scored very closely and the difference between the lowest MASE value and the highest is 0.2.

From all of this, it can be said that the most accurate model overall, regarding both the model fitting accuracy to the original dataset and the accuracy of their forecasts was the ARMA(2, 1) + GARCH(1, 1) model.

7 Conclusion

After converting the received bitcoin dataset into a time series type, a number of ARMA + GARCH models was fit to the data given the occurances of ARCH effects that were seen through an analysis into the dataset.

It was determined that a GARCH(1, 1) model sufficiently accounted for the conditional variance in the dataset and after this a number of candidate models of an ARMA and GARCH combination were proposed. These models were: ARMA(2, 1) + GARCH(1, 1), ARMA(3, 2) + GARCH(1, 1), ARMA(4, 2) + GARCH(1, 1). These models were further determined to be the best set of candidate models as fitting an ARMA(5, 2) + GARCH(1, 1) model was deemed to be an overfit in the AR component and and ARMA(4, 3) + GARCH(1, 1) model was an overfit in the MA component.

After determining their significance through their respective model summaries, examining their residuals, and comparing the models through their information criteria values, the ideal model was determined to be the ARMA(3, 2) + GARCH(1, 1) model.

Each of the candidate models were utilised for forecasting and it was determined, that the most accurate model in terms of the MASE was determined to be the ARMA(2, 1) + GARCH(1, 1) model. This was also true for the model fitting accuracy.

ARMA(2, 1) + GARCH(1, 1)	**MASE**
Fitted Values	1.00
Forecasts	3.61

Appendix One - R Code

```
#--------------------IMPORT PACKAGES--------------------------#
library(TSA)
library(fUnitRoots)
library(AID)
library(lmtest)
library(forecast)
library(rugarch)
library(fGarch)
library(readxl)
library(FitAR)

#-----------------TIME SERIES CONVERSION-----------------------#
# Import the data set.
bitcoin <- read.csv("~/RMIT/2018 Semester One/Time Series Analysis/Final
    Project/Bitcoin/Bitcoin_Historical_Price.csv")

# Create a daily date object for time series conversion.
date <- seq(as.Date("2013-04-27"), as.Date("2018-03-03"), by="day")

# Convert to a ts object.
bitcoin.ts <- ts(bitcoin$Close, start = c(2013, as.numeric(format(date[1], "%j"))),
    frequency = 365)

# Plot the time series data.
plot(bitcoin.ts, ylab='Closing Price ($)', main="Time Series Plot of Bitcoin Closing
    Price Data")

#---------------------RETURN SERIES---------------------------#
# Convert the series to a return series.
r.bitcoin <- diff(log(bitcoin.ts))

# Plot the return series.
plot(r.bitcoin, main = "Daily Return Series for Bitcoin Price Data", ylab = 'Closing
    Price ($)')
```

```
#-----------------RETURN SERIES NORMALITY TESTS-----------------#
# QQ Plot.
qqnorm(r.bitcoin, main = "QQ Plot of Daily Return Bitcoin Price Series.")
qqline(r.bitcoin, col = 2)

# ACF/PACF Plots.
par(mfrow=c(1,2))
acf(r.bitcoin, main = "The Sample ACF Plot for the Daily Return Bitcoin Price Series.")
pacf(r.bitcoin, main = "The Sample PACF Plot for the Daily Return Bitcoin Price Series.")
par(mfrow=c(1,1))

# EACF Plot.
eacf(r.bitcoin)

#McLeod-Li Test.
McLeod.Li.test(y=r.bitcoin, main = "McLeod-Li Test Statistics for Daily Return Series.")

#----------ABSOLUTE AND SQUARED SERIES TRANSFORMATION----------#
# Compute the absolute value and squared series of the return.
abs.r.bitcoin = abs(r.bitcoin)
sq.r.bitcoin = r.bitcoin^2

# ACF and PACF of the squared series.
par(mfrow=c(1,2))
acf(sq.r.bitcoin, ci.type="ma", main = "The Sample ACF Plot for the Squared Bitcoin
    Price Series.")
pacf(sq.r.bitcoin, main = "The Sample PACF Plot for the Squared Bitcoin Price Series.")
par(mfrow=c(1,1))

# EACF of the squared series.
eacf(sq.r.bitcoin)

# ACF and PACF of the absolute value series.
par(mfrow=c(1,2))
acf(abs.r.bitcoin, ci.type="ma", main = "The Sample ACF Plot for the Bitcoin Price
    Series.")
pacf(abs.r.bitcoin, main = "The Sample PACF Plot for the Bitcoin Price Series.")
par(mfrow=c(1,1))
```

```
# EACF of the absolute value series.
eacf(abs.r.bitcoin)

#---------------GARCH MODEL FITTING---------------------#
# GARCH(1, 1)
model.11 = garch(r.bitcoin, order= c(1,1), trace = FALSE)
# Remove NaN values.
model.11$residuals <- na.omit(model.11$residuals)
# Produce model summary.
summary(model.11)

# GARCH(1, 1)
model.11_2 = garchFit(formula ~garch(1,1), data = bitcoin.ts)
summary(model.11_2)

# GARCH(2, 2)
model.22 = garch(r.bitcoin, order = c(2,2), trace = FALSE)
# Produce model summary.
summary(model.22)

# GARCH(2, 2)
model.22_2 = garchFit(formula ~garch(2,2), data = bitcoin.ts)
summary(model.22_2)

# GARCH(3, 3)
model.33 = garch(r.bitcoin, order = c(3,3), trace = FALSE)
# Produce model summary.
summary(model.33)

# GARCH(3, 3)
model.33_2 = garchFit(formula ~garch(3,3), data = bitcoin.ts)
summary(model.33_2)

# Rank GARCH models according to AIC.
sort.score(AIC(model.11, model.22, model.33), score = "aic")
```

```
# Plot and list GARCH(1, 1) residuals.
# Omit all NaN values.
model.11$residuals <- na.omit(model.11$residuals)
res.model.11 <- model.11$residuals

par(mfrow=c(3,2))
plot(res.model.11,type='o',ylab='Standardised residuals', main="Time series plot of
    standardised residuals")
abline(h=0)
hist(res.model.11,main="Histogram of standardised residuals")
acf(model.11$residuals, main="ACF of standardised residuals")
pacf(res.model.11,main="PACF of standardised residuals")
qqnorm(res.model.11,main="QQ plot of standardised residuals")
qqline(res.model.11, col = 2)
print(shapiro.test(res.model.11))
k=0
LBQPlot(res.model.11, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
par(mfrow=c(1,1))


# Plot and list GARCH(2, 2) residuals.
# Omit all NaN values.
model.22$residuals <- na.remove(model.22$residuals)
res.model.22 <- model.22$residuals

par(mfrow=c(3,2))
plot(res.model.22,type='o',ylab='Standardised residuals', main="Time series plot of
    standardised residuals")
abline(h=0)
hist(res.model.22,main="Histogram of standardised residuals")
acf(model.22$residuals, main="ACF of standardised residuals")
pacf(res.model.22,main="PACF of standardised residuals")
qqnorm(res.model.22,main="QQ plot of standardised residuals")
qqline(res.model.22, col = 2)
print(shapiro.test(res.model.22))
k=0
LBQPlot(res.model.22, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
par(mfrow=c(1,1))
```

```
# Plot and list GARCH(3, 3) residuals.
# Omit all NaN values.
model.33$residuals <- na.remove(model.33$residuals)
res.model.33 <- model.33$residuals

par(mfrow=c(3,2))
plot(res.model.33,type='o',ylab='Standardised residuals', main="Time series plot of
    standardised residuals")
abline(h=0)
hist(res.model.33,main="Histogram of standardised residuals")
acf(model.33$residuals, main="ACF of standardised residuals")
pacf(res.model.33,main="PACF of standardised residuals")
qqnorm(res.model.33,main="QQ plot of standardised residuals")
qqline(res.model.33, col = 2)
print(shapiro.test(res.model.33))
k=0
LBQPlot(res.model.33, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
par(mfrow=c(1,1))

#------------------ARMA + GARCH Model Fitting---------------------#
#ARMA(1,1) + GARCH(1,1)
model1 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,1)), distribution.model = "norm")
m.11.11 <- ugarchfit(spec=model1, data=bitcoin.ts)

#ARMA(1,2) + GARCH(1,1)
model2 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,2)), distribution.model = "norm")
m.11.12 <- ugarchfit(spec=model2, data=bitcoin.ts)

#ARMA(2, 1) + GARCH(1,1)
model3 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(2,1)), distribution.model = "norm")
m.11.21 <- ugarchfit(spec=model3, data=bitcoin.ts)

#ARMA(3, 1) + GARCH(1,1)
model4 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(3,1)), distribution.model = "norm")
```

```
m.11.31 <- ugarchfit(spec=model4, data=bitcoin.ts)

#ARMA(3, 2) + GARCH(1,1)
model5 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(3,2)), distribution.model = "norm")
m.11.32 <- ugarchfit(spec=model5, data=bitcoin.ts)

#ARMA(4, 2) + GARCH(1,1)
model6 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(4,2)), distribution.model = "norm")
m.11.42 <- ugarchfit(spec=model6, data=bitcoin.ts)

#ARMA(5, 2) + GARCH(1,1)
model7 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(5,2)), distribution.model = "norm")
m.11.52 <- ugarchfit(spec=model7, data=bitcoin.ts)

#ARMA(4, 3) + GARCH(1,1)
model8 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(4,3)), distribution.model = "norm")
m.11.43 <- ugarchfit(spec=model8, data=bitcoin.ts, solver.control = list(tol = 1e-5))

#-----------------------FORECASTING----------------------------#
# ARMA(2, 1) + GARCH(1, 1) forecast.
forc.11.21 = ugarchforecast(m.11.21, data = bitcoin.ts, n.ahead = 10)
# View the forecast values.
forc.11.21

# Plot the forecast.
plot(forc.11.21)

# ARMA(3, 2) + GARCH(1, 1) forecast.
forc.11.32 = ugarchforecast(m.11.32, data = bitcoin.ts, n.ahead = 10)
# View the forecast values.
forc.11.32

# Plot the forecast.
plot(forc.11.32)
```

```
# ARMA(4, 2) + GARCH(1, 1) forecast.
forc.11.42 = ugarchforecast(m.11.42, data = bitcoin.ts,n.ahead = 10)
# View the forecast values.
forc.11.42

# Plot the forecast.
plot(forc.11.42)

#----------------------MASE CALCULATIONS----------------------#
# Read in values to compare to forecasts.
btc.forecasts <- read_xlsx("~/RMIT/2018 Semester One/Time Series Analysis/Final
    Project/Bitcoin/Bitcoin_Prices_Forecasts.xlsx")
btc.forecasts <- btc.forecasts$`Closing price`

# ARMA(2, 1) + GARCH(1, 1) MASE: Forecast.
MASE(as.vector(btc.forecasts), as.vector(forc.11.21@forecast$seriesFor))
# ARMA(3, 2) + GARCH(1, 1) MASE: Forecast.
MASE(as.vector(btc.forecasts), as.vector(forc.11.32@forecast$seriesFor))
# ARMA(4, 2) + GARCH(1, 1) MASE: Forecast.
MASE(as.vector(btc.forecasts), as.vector(forc.11.42@forecast$seriesFor))

# ARMA(2, 1) + GARCH(1, 1) MASE: Fitted model.
MASE(as.vector(bitcoin.ts),as.vector(m.11.21@fit$fitted.values))
# ARMA(3, 2) + GARCH(1, 1) MASE: Fitted model.
MASE(as.vector(bitcoin.ts),as.vector(m.11.32@fit$fitted.values))
# ARMA(4, 2) + GARCH(1, 1) MASE: Fitted model.
MASE(as.vector(bitcoin.ts),as.vector(m.11.42@fit$fitted.values))
```

Appendix Two - Sort.Score and MASE R Functions

```
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")')
   }
}
```

Listing 19: Sort.Score

```
MASE = function(observed , fitted ){
# observed: Observed series on the forecast period
# fitted: Forecast values by your model
Y.t = observed
n = length(fitted)
e.t = Y.t - fitted
sum = 0

for (i in 2:n){
   sum = sum + abs(Y.t[i] - Y.t[i-1] )
}

q.t = e.t / (sum/(n-1))

MASE = data.frame( MASE = mean(abs(q.t)))

return(list(MASE = MASE))
}
```

Listing 20: MASE

Appendix Three - Fitted GARCH Model Summaries

```
##
## Call:
## garch(x = r.bitcoin, order = c(1, 1), trace = FALSE)
##
## Model:
## GARCH(1,1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.55811 -0.33740  0.06798  0.53194  4.76716
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 3.632e-05   3.816e-06    9.518   <2e-16 ***
## a1 1.485e-01   9.555e-03   15.546   <2e-16 ***
## b1 8.522e-01   7.025e-03  121.310   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 5626.9, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 1.7522, df = 1, p-value = 0.1856
```

Model summary for the fitted GARCH(1, 1) model.

```
##
## Call:
## garch(x = r.bitcoin, order = c(2, 2), trace = FALSE)
##
## Model:
## GARCH(2,2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.74488 -0.33250  0.06947  0.53368  4.60743
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 4.863e-05   6.861e-06    7.088 1.36e-12 ***
## a1 2.019e-01   1.825e-02   11.059  < 2e-16 ***
## a2 1.973e-02   2.304e-02    0.856    0.392
## b1 1.876e-01   8.065e-02    2.326    0.020 *
## b2 5.942e-01   6.731e-02    8.827  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 5506.8, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.026848, df = 1, p-value = 0.8698
```

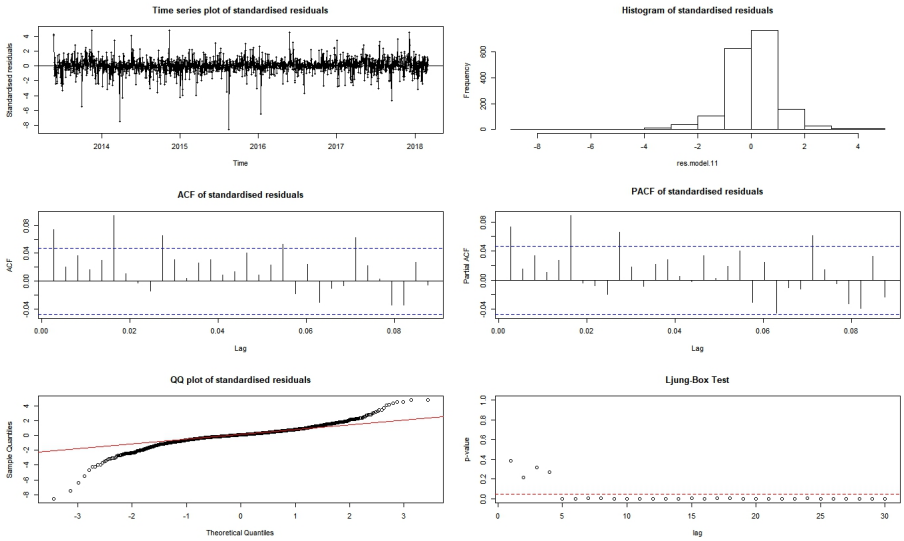
Model summary for the fitted GARCH(2, 2) model.

```
##
## Call:
## garch(x = r.bitcoin, order = c(3, 3), trace = FALSE)
##
## Model:
## GARCH(3,3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.53751 -0.32699  0.06485  0.53062  4.70153
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 8.965e-05   2.380e-05    3.766 0.000166 ***
## a1 1.681e-01   1.840e-02    9.132  < 2e-16 ***
## a2 8.004e-02   6.194e-02    1.292 0.196287
## a3 1.035e-01   3.456e-02    2.995 0.002742 **
## b1 1.008e-09   3.346e-01    0.000 1.000000
## b2 4.201e-01   1.245e-01    3.374 0.000740 ***
## b3 2.281e-01   2.097e-01    1.087 0.276867
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 5702.1, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.37393, df = 1, p-value = 0.5409
```

Model summary for the fitted GARCH(3, 3) model.

Appendix Four - GARCH Model Residuals and Code

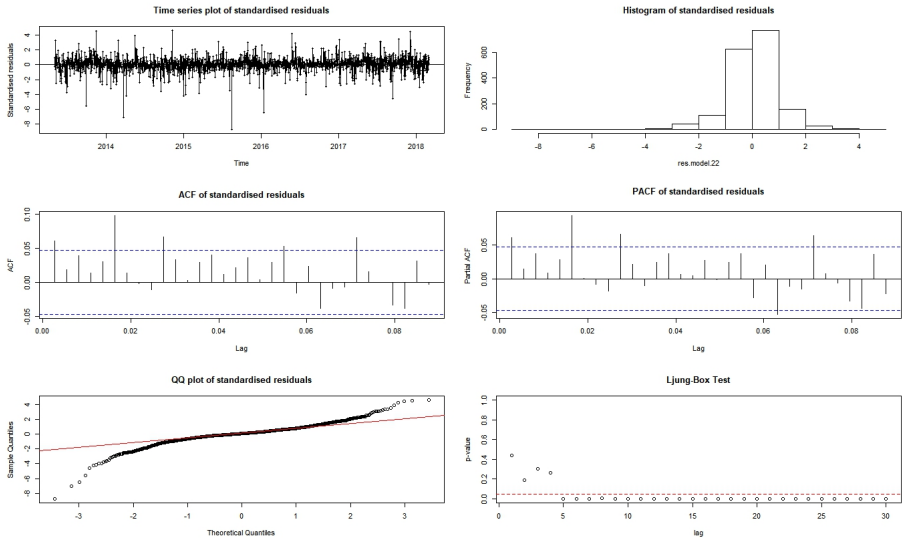
```
# Plot and list GARCH(1, 1) residuals.
res.model.11 <- model.11$residuals
par(mfrow=c(3,2))
plot(res.model.11,type='o',ylab='Standardised residuals', main="Time series plot of
    standardised residuals")
abline(h=0)
hist(res.model.11,main="Histogram of standardised residuals")
acf(model.11$residuals, main="ACF of standardised residuals")
pacf(res.model.11,main="PACF of standardised residuals")
qqnorm(res.model.11,main="QQ plot of standardised residuals")
qqline(res.model.11, col = 2)
print(shapiro.test(res.model.11))
k=0
LBQPlot(res.model.11, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
par(mfrow=c(1,1))
```




```
##
##  Shapiro-Wilk normality test
##
## data:  res.model.11
## W = 0.9021, p-value < 2.2e-16
```

GARCH(1, 1) model residuals.

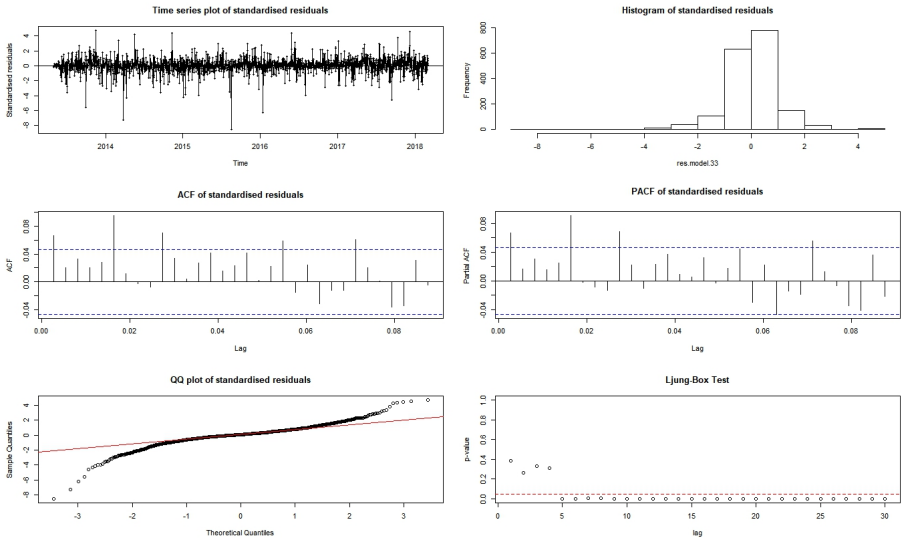
```
# Plot and list GARCH(2, 2) residuals.
# Omit all NaN values.
model.22$residuals <- na.remove(model.22$residuals)
res.model.22 <- model.22$residuals
par(mfrow=c(3,2))
plot(res.model.22,type='o',ylab='Standardised residuals', main="Time series plot of
    standardised residuals")
abline(h=0)
hist(res.model.22,main="Histogram of standardised residuals")
acf(model.22$residuals, main="ACF of standardised residuals")
pacf(res.model.22,main="PACF of standardised residuals")
qqnorm(res.model.22,main="QQ plot of standardised residuals")
qqline(res.model.22, col = 2)
print(shapiro.test(res.model.22))
k=0
LBQPlot(res.model.22, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
par(mfrow=c(1,1))
```




```
##
##  Shapiro-Wilk normality test
##
## data:  res.model.22
## W = 0.90588, p-value < 2.2e-16
```

GARCH(2, 2) model residuals.

```
# Plot and list GARCH(3, 3) residuals.
# Omit all NaN values.
model.33$residuals <- na.remove(model.33$residuals)
res.model.33 <- model.33$residuals
par(mfrow=c(3,2))
plot(res.model.33,type='o',ylab='Standardised residuals', main="Time series plot of
    standardised residuals")
abline(h=0)
hist(res.model.33,main="Histogram of standardised residuals")
acf(model.33$residuals, main="ACF of standardised residuals")
pacf(res.model.33,main="PACF of standardised residuals")
qqnorm(res.model.33,main="QQ plot of standardised residuals")
qqline(res.model.33, col = 2)
print(shapiro.test(res.model.33))
k=0
LBQPlot(res.model.33, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
par(mfrow=c(1,1))
```




```
##
##  Shapiro-Wilk normality test
##
## data:  res.model.33
## W = 0.90221, p-value < 2.2e-16
```

GARCH(3, 3) model residuals.

Appendix Five - Fitted ARMA + GARCH(1, 1) Model Summaries

```
#ARMA(1,1) + GARCH(1,1)
model1 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,1)), distribution.model = "norm")
m.11.11 <- ugarchfit(spec=model1, data=bitcoin.ts)
```

```
## *---------------------------------*
## *          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|)
## mu     134.047583   80.472114   1.66576 0.095760
## ar1      1.000000    0.001024 976.37950 0.000000
## ma1      0.009365    0.030876   0.30332 0.761644
## omega    5.175870    1.100804   4.70190 0.000003
## alpha1   0.193097    0.014375  13.43235 0.000000
## beta1    0.805903    0.015563  51.78286 0.000000
##
## Robust Standard Errors:
##          Estimate  Std. Error   t value Pr(>|t|)
## mu     134.047583   23.925732   5.60265 0.000000
## ar1      1.000000    0.001755 569.90211 0.000000
## ma1      0.009365    0.040540   0.23102 0.817303
## omega    5.175870    4.565958   1.13358 0.256972
## alpha1   0.193097    0.051487   3.75042 0.000177
## beta1    0.805903    0.068870  11.70186 0.000000
##
## LogLikelihood : -8182.592
##
## Information Criteria
## ------------------------------------
##
## Akaike       9.2422
## Bayes        9.2608
## Shibata      9.2422
## Hannan-Quinn 9.2491
```

ARMA(1, 1) + GARCH(1, 1) model summary.

```
#ARMA(1,2) + GARCH(1,1)
model2 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(1,2)), distribution.model = "norm")
m.11.12 <- ugarchfit(spec=model2, data=bitcoin.ts)
```

```
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(1,0,2)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##          Estimate  Std. Error   t value Pr(>|t|)
## mu     131.243625  262.395281   0.50018 0.616952
## ar1      1.000000    0.001126 888.37775 0.000000
## ma1      0.008792    0.030131   0.29180 0.770435
## ma2     -0.026594    0.030696  -0.86637 0.386286
## omega    5.212871    1.116311   4.66973 0.000003
## alpha1   0.193220    0.014435  13.38506 0.000000
## beta1    0.805780    0.015697  51.33445 0.000000
##
## Robust Standard Errors:
##          Estimate  Std. Error   t value Pr(>|t|)
## mu     131.243625  247.024542   0.53130  0.59521
## ar1      1.000000    0.002150 465.09548  0.00000
## ma1      0.008792    0.039295   0.22375  0.82295
## ma2     -0.026594    0.036214  -0.73438  0.46272
## omega    5.212871    4.647711   1.12160  0.26203
## alpha1   0.193220    0.051958   3.71876  0.00020
## beta1    0.805780    0.069739  11.55419  0.00000
##
## LogLikelihood : -8185.655
##
## Information Criteria
## ------------------------------------
##
## Akaike       9.2468
## Bayes        9.2684
## Shibata      9.2468
## Hannan-Quinn 9.2548
```

ARMA(1, 2) + GARCH(1, 1) model summary.

```
#ARMA(2, 1) + GARCH(1,1)
model3 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(2,1)), distribution.model = "norm")
m.11.21 <- ugarchfit(spec=model3, data=bitcoin.ts)
```

```
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(2,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## mu     131.31542   22.951399     5.7215    0e+00
## ar1      1.85484    0.000321  5776.8319    0e+00
## ar2     -0.85475    0.000210 -4068.6758    0e+00
## ma1     -0.87688    0.012969   -67.6115    0e+00
## omega    5.44384    1.145529     4.7522    2e-06
## alpha1   0.19795    0.014833    13.3451    0e+00
## beta1    0.80105    0.016065    49.8635    0e+00
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## mu     131.31542   12.113781    10.8402 0.000000
## ar1      1.85484    0.001384  1340.4867 0.000000
## ar2     -0.85475    0.000663 -1288.7364 0.000000
## ma1     -0.87688    0.028510   -30.7565 0.000000
## omega    5.44384    4.775818     1.1399 0.254338
## alpha1   0.19795    0.053064     3.7304 0.000191
## beta1    0.80105    0.071225    11.2468 0.000000
##
## LogLikelihood : -8184.705
##
## Information Criteria
## ------------------------------------
##
## Akaike       9.2457
## Bayes        9.2674
## Shibata      9.2457
## Hannan-Quinn 9.2537
```

ARMA(2, 1) + GARCH(1, 1) model summary.

```
#ARMA(3, 1) + GARCH(1,1)
model4 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(3,1)), distribution.model = "norm")
m.11.31 <- ugarchfit(spec=model4, data=bitcoin.ts)
```

```
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(3,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##           Estimate  Std. Error   t value Pr(>|t|)
## mu     -523.029991  180.905149   -2.8912 0.003838
## ar1       1.342290    0.000203 6607.4728 0.000000
## ar2      -0.384586    0.001845 -208.4155 0.000000
## ar3       0.035642    0.001443   24.7026 0.000000
## ma1      -0.057160    0.036324   -1.5736 0.115569
## omega  7695.537635  327.985674   23.4630 0.000000
## alpha1    0.897078    0.058220   15.4085 0.000000
## beta1     0.101921    0.012516    8.1434 0.000000
##
## Robust Standard Errors:
##           Estimate  Std. Error    t value Pr(>|t|)
## mu     -523.029991  1.3235e+03   -0.39519 0.692703
## ar1       1.342290  3.7000e-04 3623.18787 0.000000
## ar2      -0.384586  8.4950e-03  -45.26960 0.000000
## ar3       0.035642  1.4150e-03   25.19382 0.000000
## ma1      -0.057160  1.1778e-01   -0.48533 0.627440
## omega  7695.537635  4.9201e+02   15.64107 0.000000
## alpha1    0.897078  1.1881e-01    7.55049 0.000000
## beta1     0.101921  4.5349e-02    2.24751 0.024608
##
## LogLikelihood : -10422.93
##
## Information Criteria
## ------------------------------------
##
## Akaike       11.773
## Bayes        11.798
## Shibata      11.773
## Hannan-Quinn 11.782
```

ARMA(3, 1) + GARCH(1, 1) model summary.

```
#ARMA(3, 2) + GARCH(1,1)
model5 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(3,2)), distribution.model = "norm")
m.11.32 <- ugarchfit(spec=model5, data=bitcoin.ts)
```

```
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(3,0,2)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     139.12118    1.783237   78.0161        0
## ar1      1.28451    0.002983  430.5639        0
## ar2     -1.23336    0.003497 -352.6835        0
## ar3      0.94791    0.003706  255.7576        0
## ma1     -0.24840    0.004550  -54.5974        0
## ma2      0.96166    0.003023  318.1081        0
## omega    5.54532    1.047047    5.2962        0
## alpha1   0.21760    0.015271   14.2493        0
## beta1    0.78140    0.015550   50.2516        0
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     139.12118    0.596856  233.0901  0.00000
## ar1      1.28451    0.002620  490.2907  0.00000
## ar2     -1.23336    0.001549 -796.4679  0.00000
## ar3      0.94791    0.003167  299.3262  0.00000
## ma1     -0.24840    0.006968  -35.6475  0.00000
## ma2      0.96166    0.002816  341.5129  0.00000
## omega    5.54532    3.427033    1.6181  0.10564
## alpha1   0.21760    0.041703    5.2178  0.00000
## beta1    0.78140    0.051816   15.0803  0.00000
##
## LogLikelihood : -8173.493
##
## Information Criteria
## ------------------------------------
##
## Akaike       9.2353
## Bayes        9.2632
## Shibata      9.2353
## Hannan-Quinn 9.2456
```

ARMA(3, 2) + GARCH(1, 1) model summary.

```
#ARMA(4, 2) + GARCH(1,1)
model6 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(4,2)), distribution.model = "norm")
m.11.42 <- ugarchfit(spec=model6, data=bitcoin.ts)
```

```
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(4,0,2)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##          Estimate  Std. Error   t value Pr(>|t|)
## mu     140.143008    1.371580  102.1763    0e+00
## ar1      1.261935    0.003719  339.3393    0e+00
## ar2     -1.221929    0.003480 -351.1613    0e+00
## ar3      0.932316    0.004292  217.1994    0e+00
## ar4      0.026981    0.005891    4.5799    5e-06
## ma1     -0.258070    0.002078 -124.2186    0e+00
## ma2      0.970039    0.001173  827.1765    0e+00
## omega    5.575647    1.016432    5.4855    0e+00
## alpha1   0.219322    0.014958   14.6628    0e+00
## beta1    0.779678    0.014935   52.2043    0e+00
##
## Robust Standard Errors:
##          Estimate  Std. Error   t value Pr(>|t|)
## mu     140.143008    0.678916  206.4218 0.000000
## ar1      1.261935    0.005990  210.6665 0.000000
## ar2     -1.221929    0.003936 -310.4656 0.000000
## ar3      0.932316    0.005508  169.2677 0.000000
## ar4      0.026981    0.012950    2.0835 0.037203
## ma1     -0.258070    0.004709  -54.8064 0.000000
## ma2      0.970039    0.002920  332.2096 0.000000
## omega    5.575647    3.240360    1.7207 0.085307
## alpha1   0.219322    0.038739    5.6615 0.000000
## beta1    0.779678    0.047529   16.4042 0.000000
##
## LogLikelihood : -8174.571
##
## Information Criteria
## ------------------------------------
##
## Akaike       9.2377
## Bayes        9.2686
## Shibata      9.2376
## Hannan-Quinn 9.2491
```

ARMA(4, 2) + GARCH(1, 1) model summary.

```
#ARMA(5, 2) + GARCH(1,1)
model7 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(5,2)), distribution.model = "norm")
m.11.52 <- ugarchfit(spec=model7, data=bitcoin.ts)
```

```
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(5,0,2)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##          Estimate  Std. Error  t value Pr(>|t|)
## mu     128.480358    3.946690 32.55395 0.000000
## ar1     -0.208868    0.098784 -2.11439 0.034482
## ar2      0.291915    0.095171  3.06727 0.002160
## ar3      0.933063    0.021415 43.57110 0.000000
## ar4     -0.016826    0.053090 -0.31694 0.751292
## ar5      0.000119    0.048674  0.00244 0.998053
## ma1      1.220481    0.086299 14.14246 0.000000
## ma2      0.906430    0.021212 42.73130 0.000000
## omega    5.580567    1.160209  4.80997 0.000002
## alpha1   0.202012    0.015248 13.24849 0.000000
## beta1    0.796988    0.016433 48.49907 0.000000
##
## Robust Standard Errors:
##          Estimate  Std. Error   t value Pr(>|t|)
## mu     128.480358    9.411151 13.651929 0.000000
## ar1     -0.208868    0.419275 -0.498163 0.618369
## ar2      0.291915    0.408387  0.714799 0.474733
## ar3      0.933063    0.034762 26.841204 0.000000
## ar4     -0.016826    0.207738 -0.080997 0.935444
## ar5      0.000119    0.177266  0.000670 0.999466
## ma1      1.220481    0.373077  3.271392 0.001070
## ma2      0.906430    0.028406 31.910207 0.000000
## omega    5.580567    4.456127  1.252336 0.210448
## alpha1   0.202012    0.048775  4.141759 0.000034
## beta1    0.796988    0.064033 12.446507 0.000000
##
## LogLikelihood : -8190.408
##
## Information Criteria
## ------------------------------------
##
## Akaike       9.2567
## Bayes        9.2907
## Shibata      9.2566
## Hannan-Quinn 9.2692
```

ARMA(5, 2) + GARCH(1, 1) model summary.

```
#ARMA(4, 3) + GARCH(1,1)
model8 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),
mean.model = list(armaOrder = c(4,3)), distribution.model = "norm")
m.11.43 <- ugarchfit(spec=model8, data=bitcoin.ts, solver.control = list(tol = 1e-5))
```

```
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(4,0,3)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##           Estimate  Std. Error  t value Pr(>|t|)
## mu     1536.555339  175.314427   8.7646 0.000000
## ar1       0.219949    0.042685   5.1528 0.000000
## ar2       0.525565    0.017310  30.3624 0.000000
## ar3       0.492956    0.028528  17.2798 0.000000
## ar4      -0.237260    0.034369  -6.9033 0.000000
## ma1       0.640869    0.050494  12.6921 0.000000
## ma2       0.236821    0.046281   5.1170 0.000000
## ma3       0.039045    0.031447   1.2416 0.214389
## omega  9252.271181  402.411066  22.9921 0.000000
## alpha1    0.954914    0.063280  15.0903 0.000000
## beta1     0.044084    0.009026   4.8843 0.000001
##
## Robust Standard Errors:
##           Estimate  Std. Error  t value Pr(>|t|)
## mu     1536.555339  2.3268e+03  0.66037 0.509019
## ar1       0.219949  7.1127e-02  3.09234 0.001986
## ar2       0.525565  1.4566e-02 36.08054 0.000000
## ar3       0.492956  3.8482e-02 12.81012 0.000000
## ar4      -0.237260  5.3730e-02 -4.41578 0.000010
## ma1       0.640869  1.0450e-01  6.13291 0.000000
## ma2       0.236821  9.5863e-02  2.47040 0.013496
## ma3       0.039045  6.6890e-02  0.58372 0.559411
## omega  9252.271181  7.0015e+02 13.21468 0.000000
## alpha1    0.954914  1.3295e-01  7.18227 0.000000
## beta1     0.044084  2.0386e-02  2.16243 0.030585
##
## LogLikelihood : -10522.77
##
## Information Criteria
## ------------------------------------
##
## Akaike       11.889
## Bayes        11.923
## Shibata      11.889
## Hannan-Quinn 11.902
```

ARMA(4, 3) + GARCH(1, 1) model summary.