Code

Final Project-Time Series-Bitcoin Daily Closing Price

Vamika Pardeshi-s3701024 and Arpan Kumar-s3696599

May 24, 2019

1. INTRODUCTION

The topic that is given for this final project is "Bitcoin daily closing price" from the 27th of April 2013 to the 24th of February 2019. The data has been sourced from "https://coinmarketcap.com/ (https://coinmarketcap.com/)". Bitcoin was the first Cryptocurrency. The value of bitcoin can be evaluated by what people want to spend for it, and so, it is very unpredictable and fluctuates wildly on a daily basis. Hence, the study of bitcoin can be very challenging and includes most of the aspects of time series analysis. The requirements of this project are branched into sub-sections, and those sub-sections are descriptive analysis, data visualizations, model specification/fitting/selection and diagnostic checking. The objective, here, is to prepare a comprehensive report that scrutinizes the overall analysis of the given bitcoin data using the time series analysis methods and accurately predict the value of bitcoin for the next 10 days.

2. METHODOLOGY

The models that are taken into consideration are- 1. LINEAR MODEL and QUADRATIC MODEL- For analyzing the trends of the series and selecting the best fit trend model. 2. ARIMA model and GARCH processes- For model specifications 3. ARMA+GARCH- For finding out the best model that fits the given Bitcoin data using estimation of parameters and model diagnostic.

3. DATA IMPORT

- Imported the 'Bitcoin_Historical_Price' dataset into R, using read.csv(), and named it as 'Bitcoin_price'.
- Checked the class of the imported data and viewed the first few observations of the same.

Hide

```
setwd("C:\\Time series")
Bitcoin_price <- read.csv("Bitcoin_Historical_Price.csv")
class(Bitcoin_price)
```

```
[1] "data.frame"
```

Hide

```
head(Bitcoin_price)
```

	Date <fctr>	Close <dbl>
1	2013-04-27	134.21
2	2013-04-28	144.54

	Date <fctr>	Close <dbl>
3	2013-04-29	139.00
4	2013-04-30	116.99
5	2013-05-01	105.21
6	2013-05-02	97.75

6 rows

4. DATA PREPROCESSING

- Since, the class of the imported data was data frame, so converting to time series object first, in order to plot the time series of the given data.

Hide

```
setdate <-seq(as.Date("2013-04-27"), as.Date("2019-2-24"), by= "day")
Bitcoin_price_ts <- ts(as.vector(Bitcoin_price$Close), start= c(2013, as.numeric(format(setdate[1], "%j"))), frequency=365)
class(Bitcoin_price_ts)
```

```
[1] "ts"
```

Hide

```
head(Bitcoin_price_ts)
```

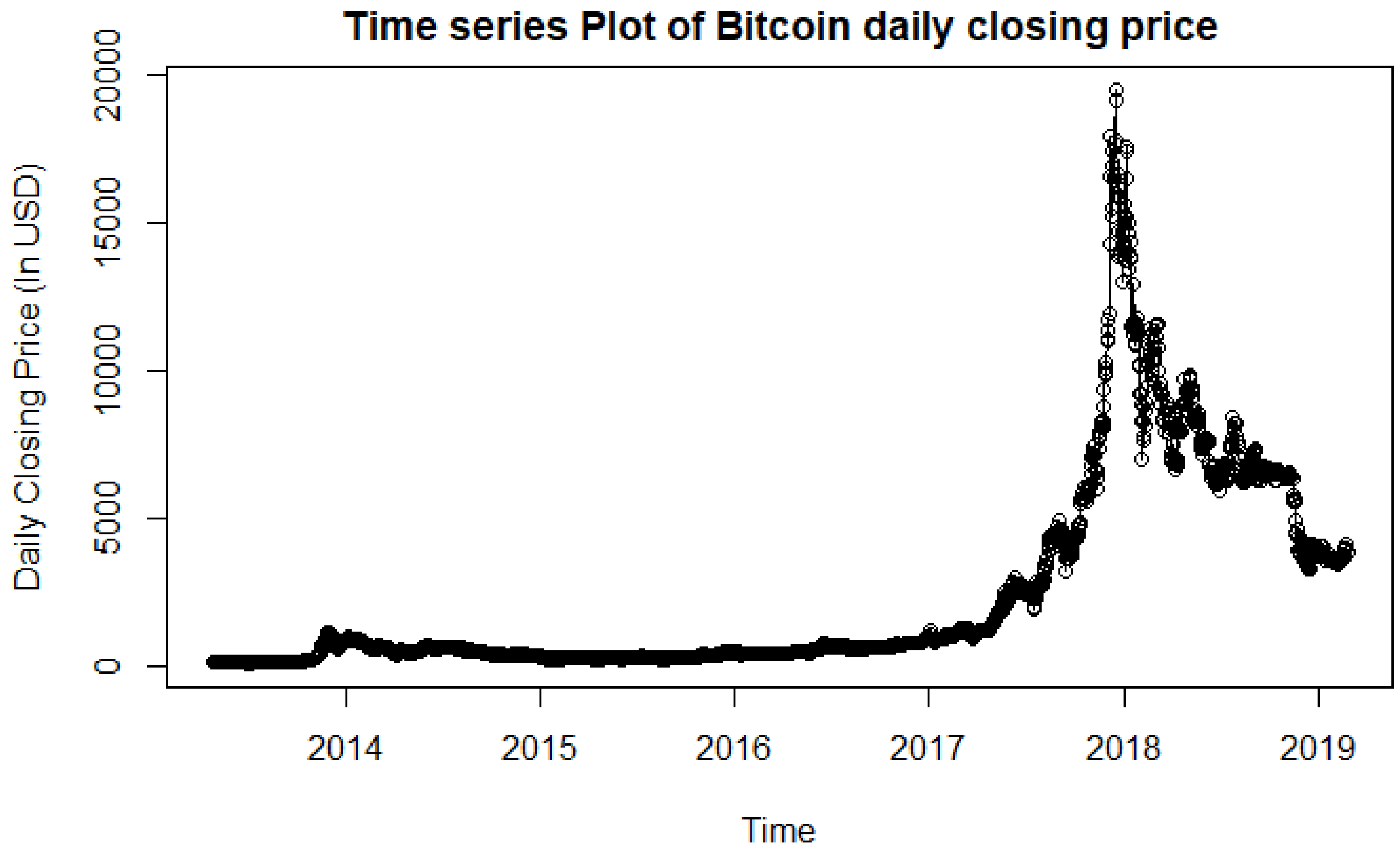
```
Time Series:
Start = c(2013, 117)
End = c(2013, 122)
Frequency = 365
[1] 134.21 144.54 139.00 116.99 105.21  97.75
```

5. ANALYZING THE DATA

5.1 Time series plot

Hide

```
plot(Bitcoin_price_ts,type='o',ylab='Daily Closing Price (In USD)',main = "Time series Plot of Bitcoin daily closing price")
```



- Based on the above time series plot, the following can be inferred-

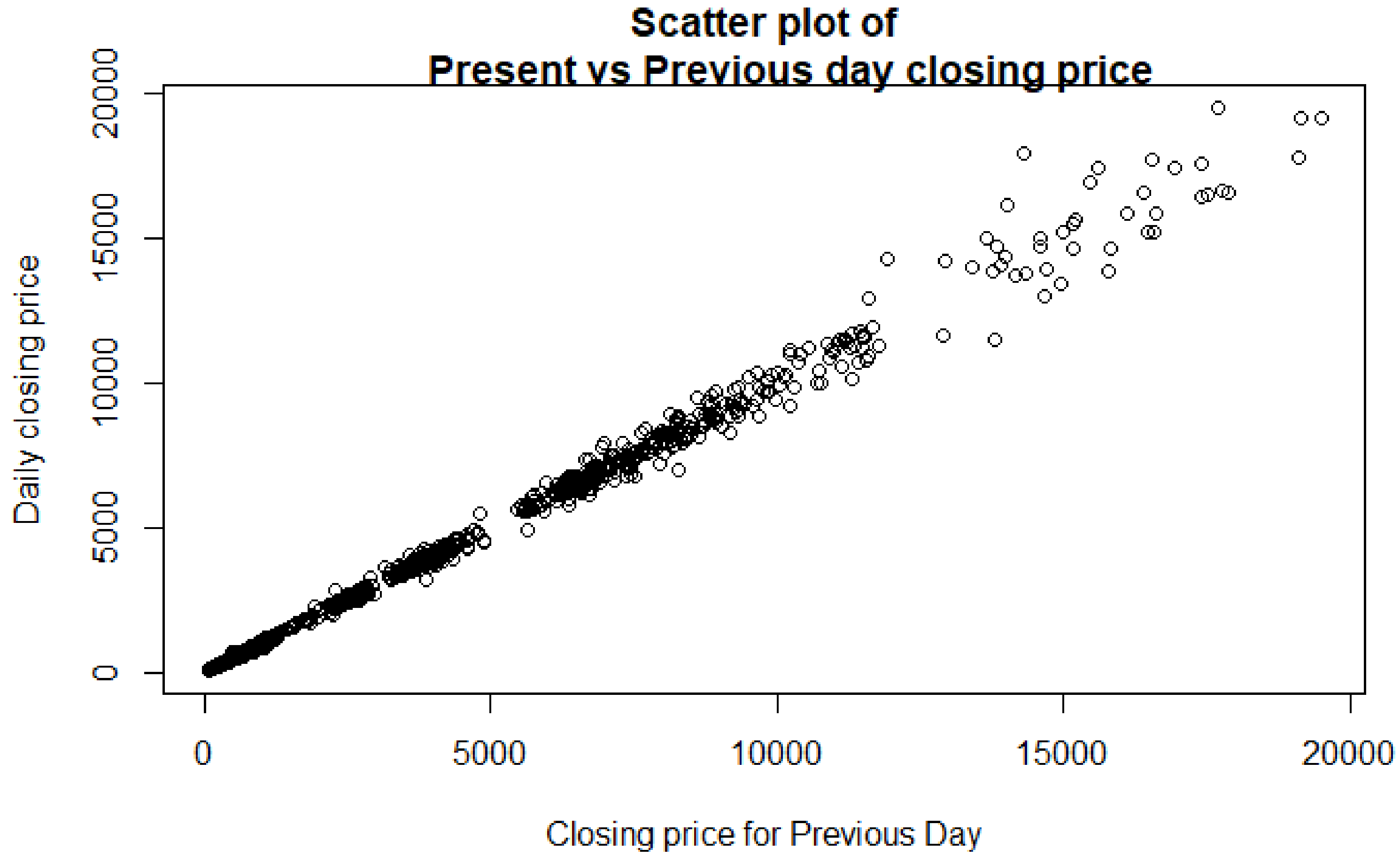
1. Trend- An overall upward/positive trend, stating the existence of non-stationarity.
2. Pattern- No seasonal patterns or cyclic movements.
3. Variance- Change in variance is present.
4. Behaviour- An auto-regressive behaviour, from the succeeding observations.

5.2 Scatter plot for neighbouring observations

- To check if there is any relationship between the consecutive years, the correlation between the observations has been investigated by plotting a scatter plot. This will help in forecasting next day's closing price of bitcoin by using the present day's bitcoin price.
- The below scatter plot shows a positive trend and it can be said that a high correlation is present between the consecutive observations.

Hide

```
plot(y=Bitcoin_price_ts,x=zlag(Bitcoin_price_ts),ylab = 'Daily closing price', xlab = 'Closing price for Previous Day', main
="Scatter plot of
     Present vs Previous day closing price")
```



6. BEST FITTING TREND MODEL

6.1 Linear Model

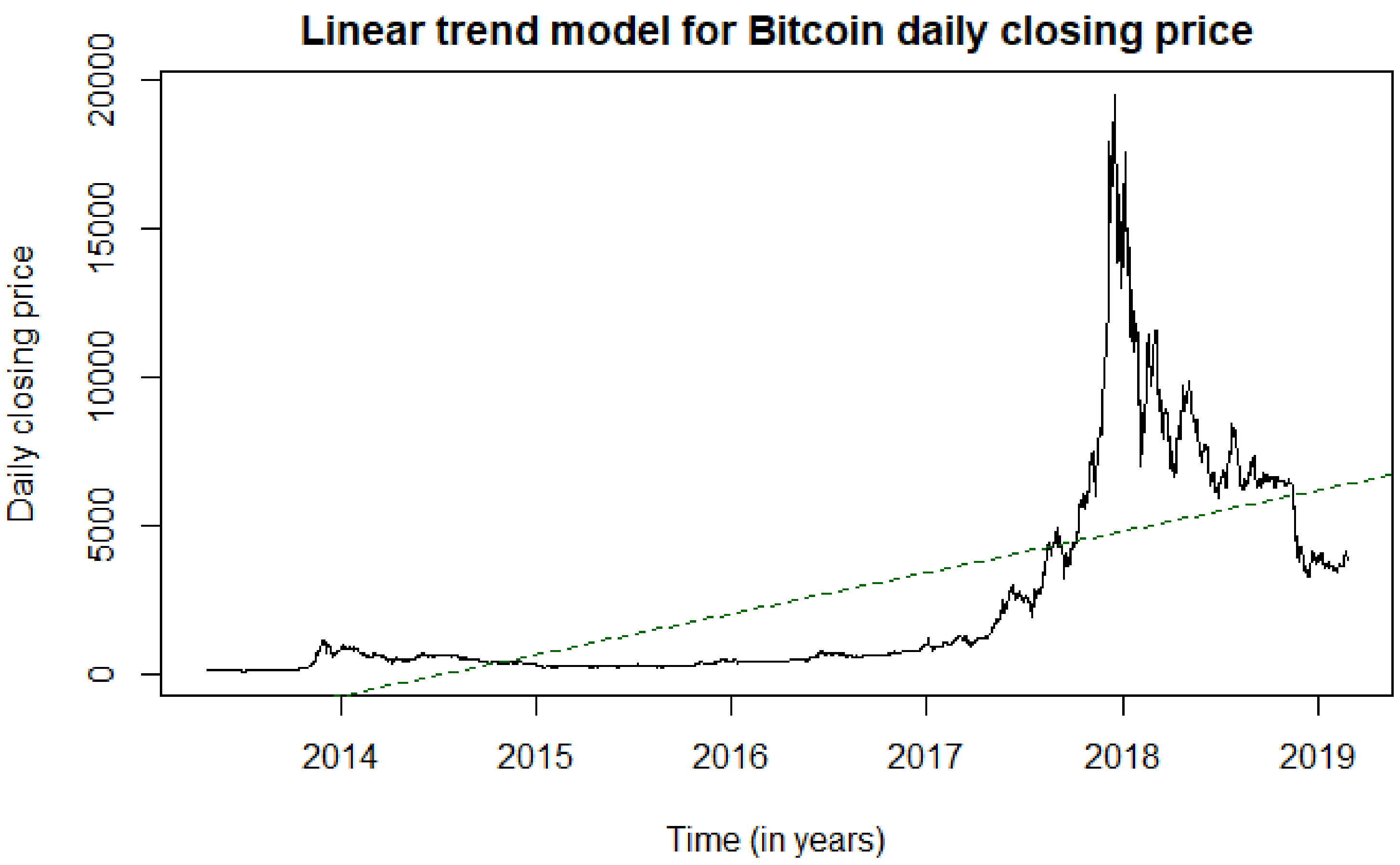
Hide

```
LinearModel <- lm(Bitcoin_price_ts~time(Bitcoin_price_ts))
summary(LinearModel)
```

```
Call:
lm(formula = Bitcoin_price_ts ~ time(Bitcoin_price_ts))

Residuals:
    Min      1Q  Median      3Q     Max 
-2907.7 -1865.4  -371.2  1212.0 14773.3 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)            -2.778e+06  6.383e+04  -43.52   <2e-16 ***
time(Bitcoin_price_ts)  1.379e+03  3.166e+01   43.56   <2e-16 ***
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1    1

Residual standard error: 2461 on 2128 degrees of freedom
Multiple R-squared:  0.4714,    Adjusted R-squared:  0.4711 
F-statistic:  1898 on 1 and 2128 DF,  p-value: < 2.2e-16
```

Hide

```
plot(Bitcoin_price_ts,ylab='Daily closing price', xlab='Time (in years)', main= "Linear trend model for Bitcoin daily closin
g price")
abline(LinearModel,lty=2, col="dark green")
```



Hide

```
par(mfrow=c(2,2))
Linear_residual = rstudent(LinearModel)
plot(y = Linear_residual, x = as.vector(time(Bitcoin_price_ts)),xlab = 'Time (in years)', ylab='Residuals (Standardized)',ty
pe='o', main = "Linear Model Residuals plot")
hist(rstudent(LinearModel), xlab = "Residuals (Standardized)")
```

Hide

```
qqnorm(Linear_residual)
qqline(Linear_residual, col = 2, lwd = 1, lty = 2)
```

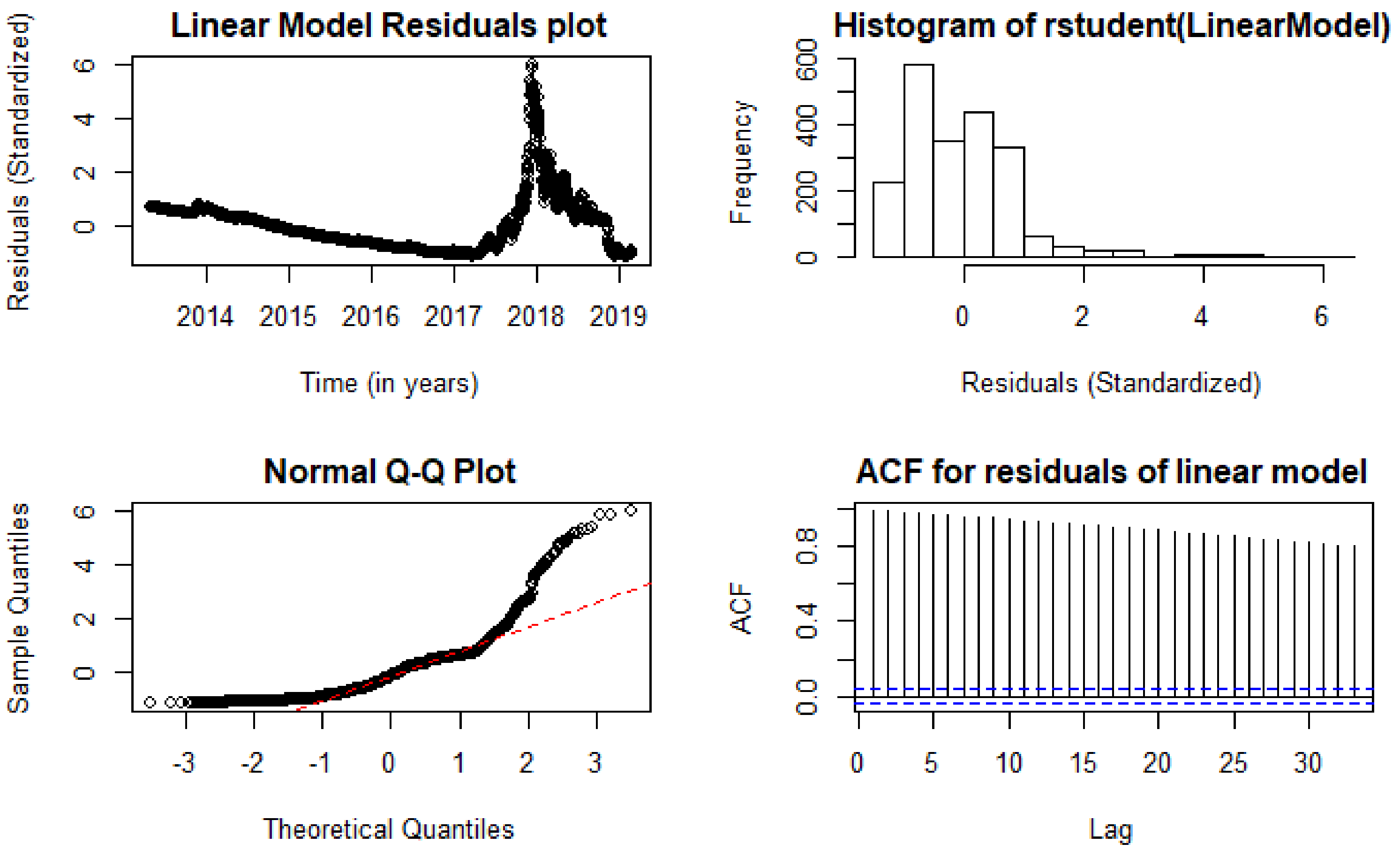
Hide

```
shapiro.test(Linear_residual)
```

```

	Shapiro-Wilk normality test

data:  Linear_residual
W = 0.8223, p-value < 2.2e-16
```

Hide

```
acf(Linear_residual, main="ACF for residuals of linear model")
par(mfrow=c(1,1))
```



6.2 Quadratic MOdel

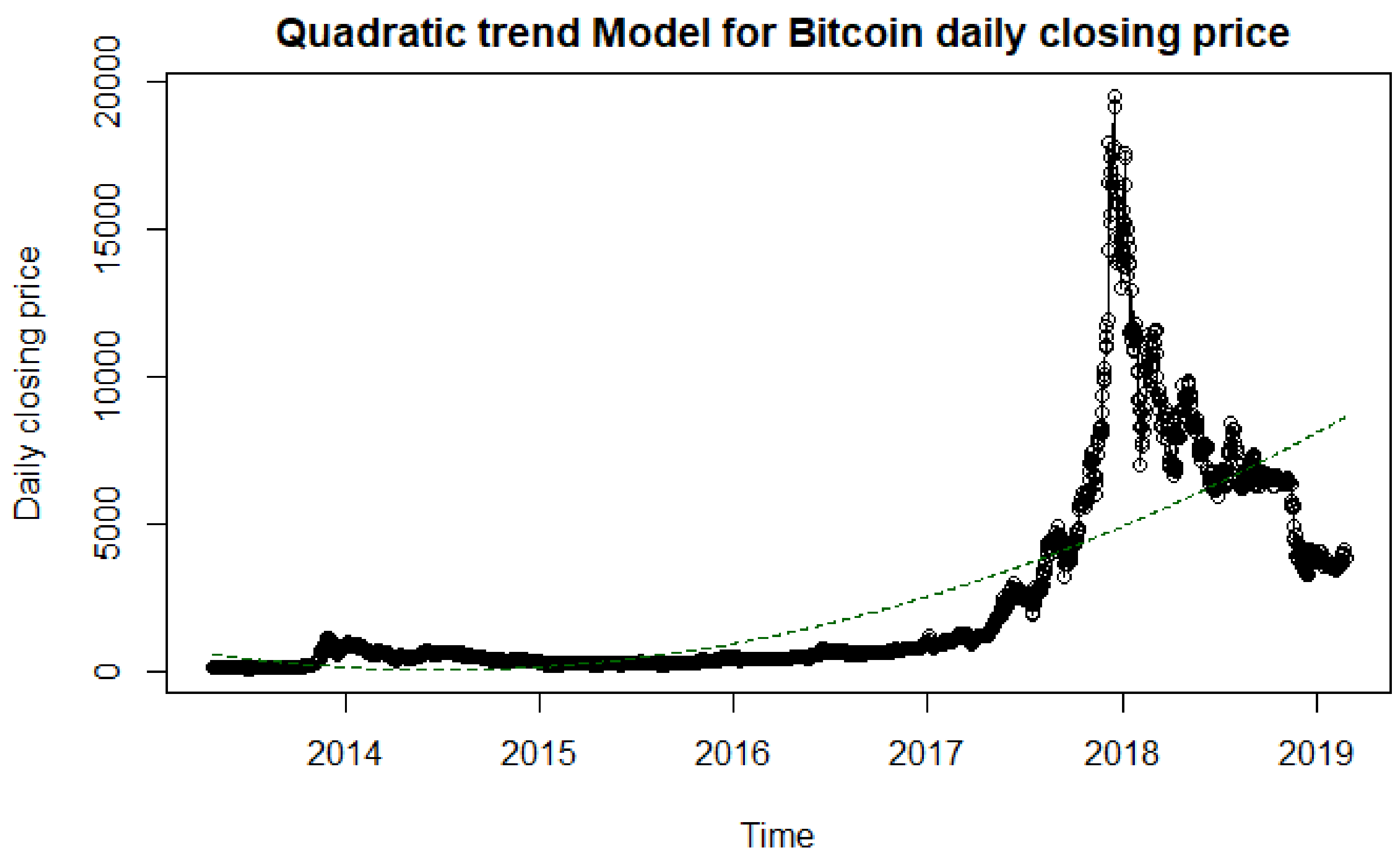
Hide

```
t = time(Bitcoin_price_ts)
t2 = t^2
QuadraticModel = lm(Bitcoin_price_ts~t + t2)
summary(QuadraticModel)
```

```
Call:
lm(formula = Bitcoin_price_ts ~ t + t2)

Residuals:
    Min      1Q  Median      3Q     Max 
-5044.0  -942.0  -196.7   435.7 14722.9 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.606e+09  7.798e+07   20.59   <2e-16 ***
t           -1.594e+06  7.736e+04  -20.61   <2e-16 ***
t2           3.957e+02  1.918e+01   20.63   <2e-16 ***
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1    1

Residual standard error: 2247 on 2127 degrees of freedom
Multiple R-squared:  0.5595,	Adjusted R-squared:  0.5591 
F-statistic:  1351 on 2 and 2127 DF,  p-value: < 2.2e-16
```

Hide

```
plot(Bitcoin_price_ts,type='o',ylab='Daily closing price',main = " Quadratic trend Model for Bitcoin daily closing price")
points(t,predict.lm(QuadraticModel), type="l", lty=2,col="dark green")
```



Hide

```
par(mfrow=c(2,2))
Quadratic_residual = rstudent(QuadraticModel)
plot(y = Quadratic_residual, x = as.vector(time(Bitcoin_price_ts)),xlab = 'Time (in years)', ylab='Residuals (Standardized)'
,type='o', main = "Quadratic Model Residuals plot")
hist(rstudent(QuadraticModel), xlab = "Residuals (Standardized)")
```

Hide

```
qqnorm(Quadratic_residual)
qqline(Quadratic_residual, col = 2, lwd = 1, lty = 2)
```

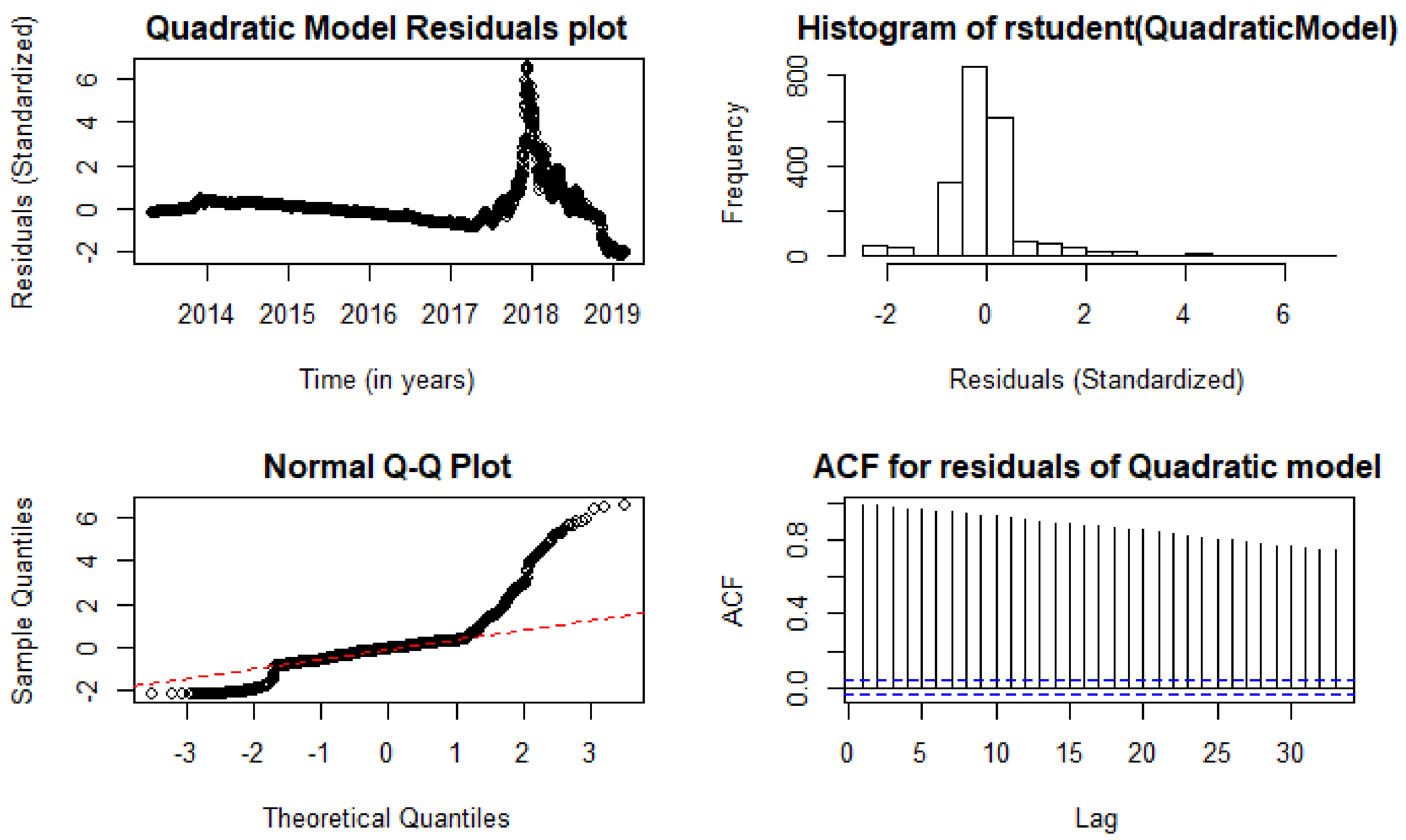
Hide

```
shapiro.test(Quadratic_residual)
```

```

	Shapiro-Wilk normality test

data:  Quadratic_residual
W = 0.74479, p-value < 2.2e-16
```

Hide

```
acf(Quadratic_residual, main="ACF for residuals of Quadratic model")
par(mfrow=c(1,1))
```



- With the R-squared values, it can be said that less than 55% of the variation in the data can be explained by both linear and quadratic models.
- The QQ-PLOTs for both the models are showing deviation from the normality.
- p-values are less than alpha for both the models, hence, the null hypothesis is rejected, indicating that data is not normally distributed.
- Residual plots are not meeting the expectation for randomness.
- ACF plots having all the significant lags.
- With all the above analysis, it can be concluded that time series do not fit into both the linear and quadratic models.
- The trend needs to be removed and the data needs to be made stationary.

7. DATA PREPARATION

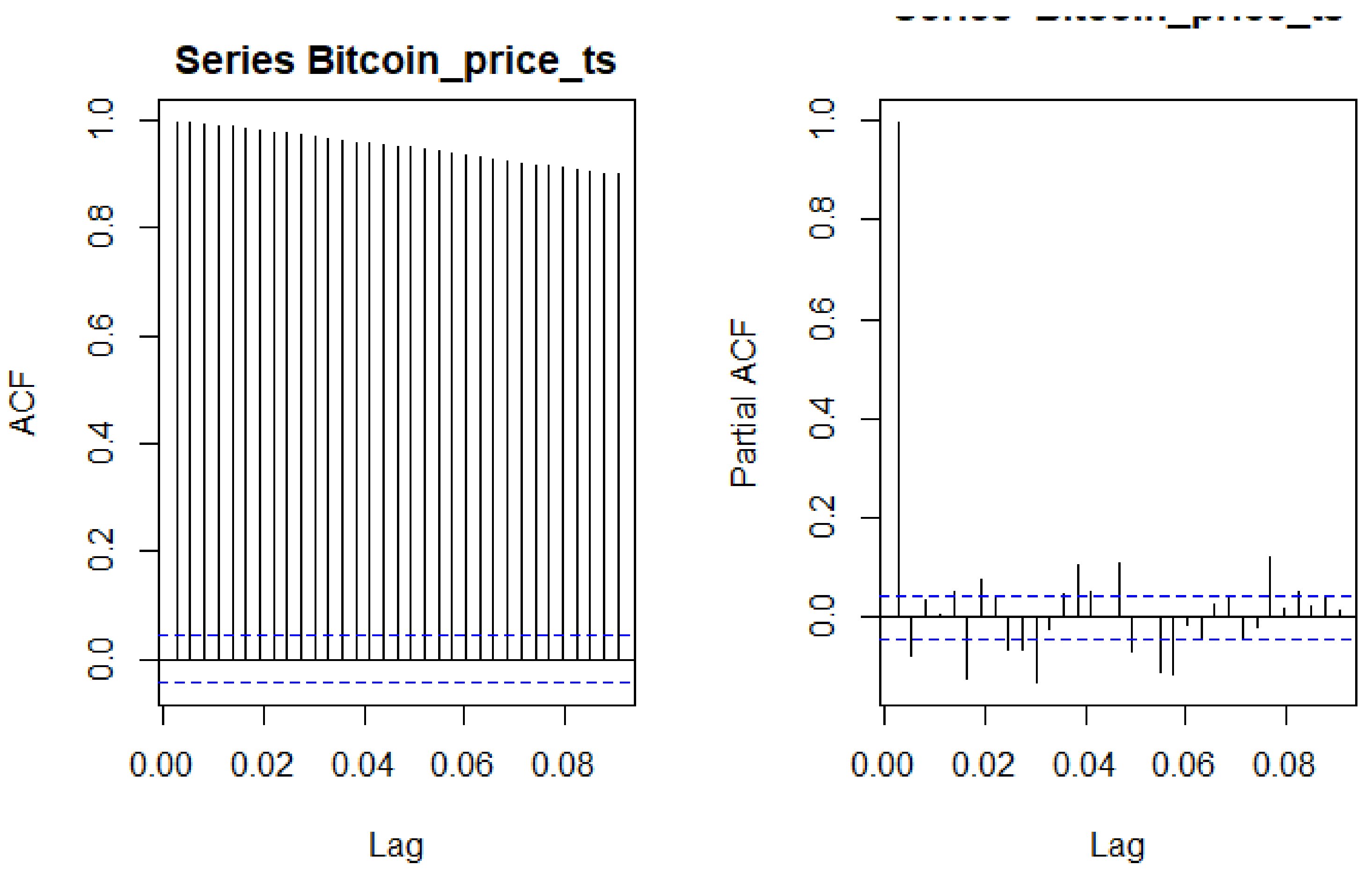
- ACF, PACF and ADF test have been applied to the data in order to confirm the presence of non-stationarity.

Hide

```
par(mfrow=c(1,2))
acf(Bitcoin_price_ts)
pacf(Bitcoin_price_ts)
```

Hide

```
par(mfrow=c(1,1))
```



Hide

```
adf.test(Bitcoin_price_ts)
```

```

	Augmented Dickey-Fuller Test

data:  Bitcoin_price_ts
Dickey-Fuller = -2.6913, Lag order = 12, p-value = 0.2856
alternative hypothesis: stationary
```

Hide

```
shapiro.test(Bitcoin_price_ts)
```

```

	Shapiro-Wilk normality test

data:  Bitcoin_price_ts
W = 0.68136, p-value < 2.2e-16
```

- In ACF, there is a slowly decaying pattern and in PACF, the first correlation is very high, that implies the presence of non-stationarity and trend.
- The p-value 0.2856 from the ADF test is greater than alpha (0.05), hence, null hypothesis can not be rejected, confirming the presence of non-stationarity in the series.
- In order to overcome the non-stationarity of the series and make the series stationary, transformation and differencing are needed.

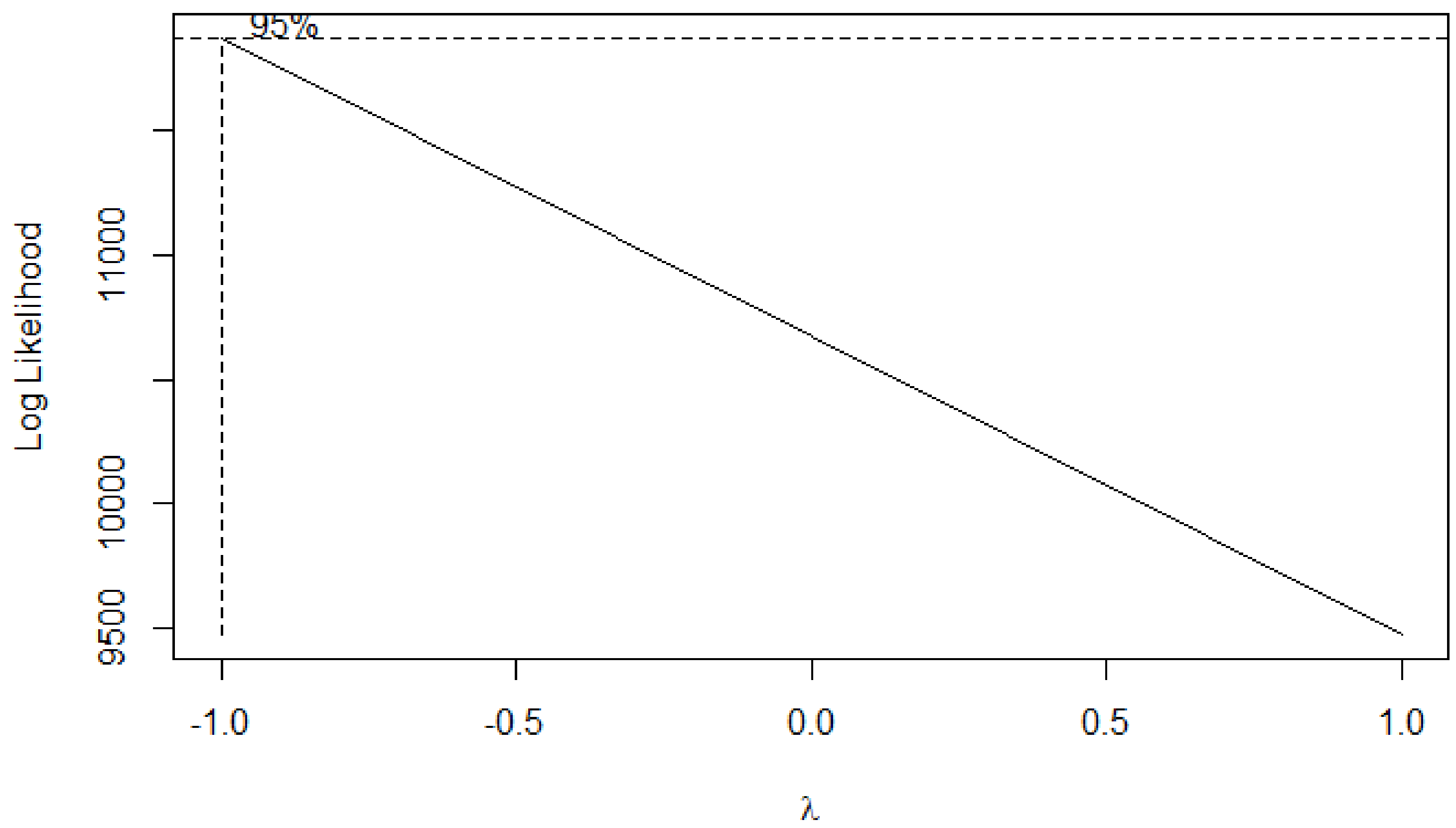
7.1 TRANSFORMATION

- Box-cox transformation has been applied in order to remove those components from the series, that are non-stationary.

Hide

```
Bitcoin_price_Trans = BoxCox.ar(Bitcoin_price_ts+abs(min(Bitcoin_price_ts))+0.1, lambda=c(-1,1))
```

```
possible convergence problem: optim gave code = 1
```



Hide

```
Bitcoin_price_Trans$ci
```

```
[1] -1 -1
```

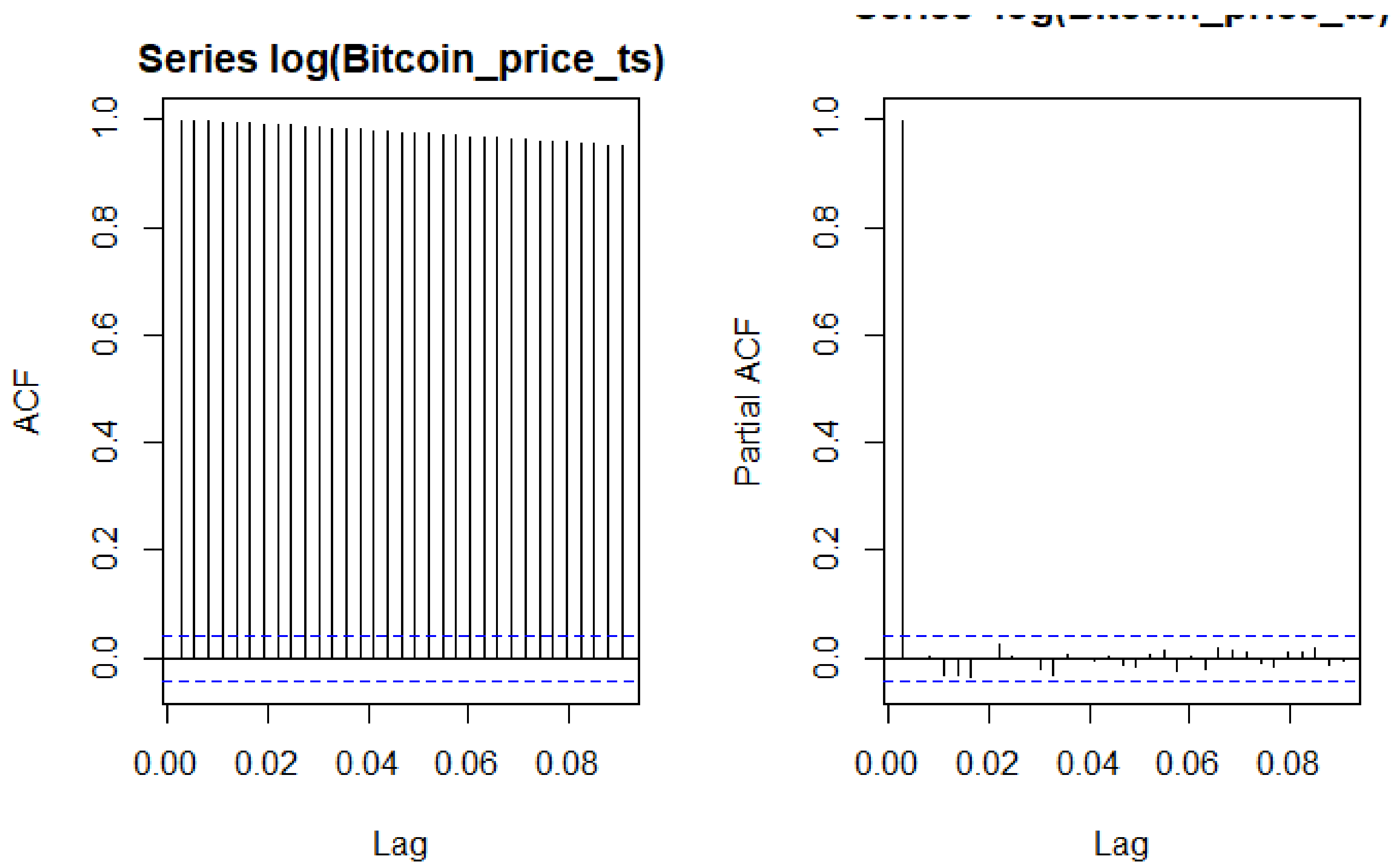
- The above plot of log likelihood is capturing ‘0’, therefore, log transformation can be applied to the bitcoin data.

Hide

```
par(mfrow=c(1,2))
acf(log(Bitcoin_price_ts))
pacf(log(Bitcoin_price_ts))
```

Hide

```
par(mfrow=c(1,1))
```



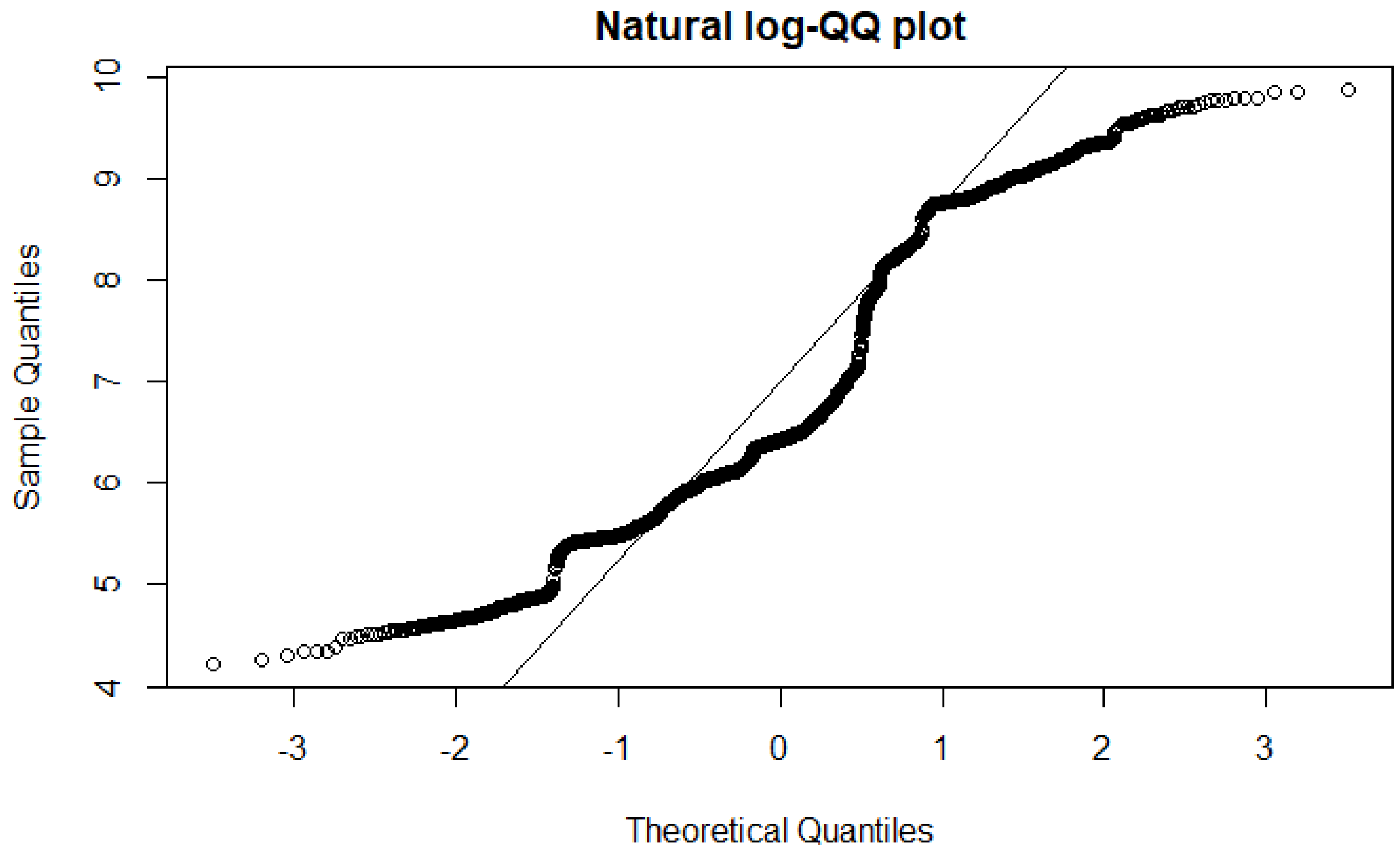
Hide

```
adf.test(log(Bitcoin_price_ts))
```

```

	Augmented Dickey-Fuller Test

data:  log(Bitcoin_price_ts)
Dickey-Fuller = -1.6272, Lag order = 12, p-value = 0.7362
alternative hypothesis: stationary
```

Hide

```
qqnorm(log(Bitcoin_price_ts), main = "Natural log-QQ plot")
qqline(log(Bitcoin_price_ts))
```



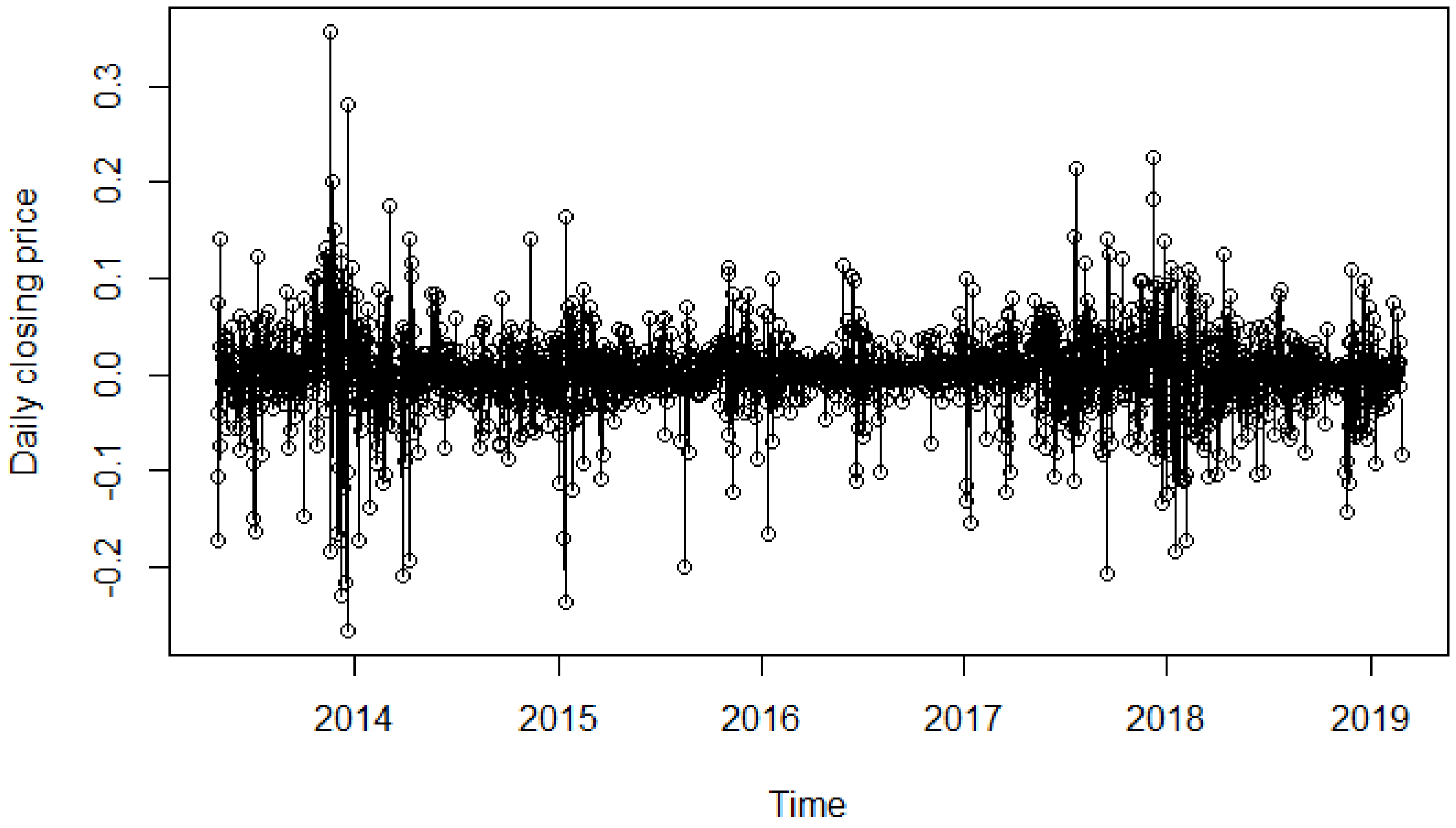
- Even after applying the log transformation, the ACF and PACF plots still have a slowly decaying pattern and high first correlation, proving that trend and non-stationarity are still present in the log transformed bitcoin series.
- The p-value from the adf test is 0.7362, that is greater than alpha, proving the series is still non-stationary.
- QQ-plot also showing deviation from the normality.
- Therefore, differencing needs to be applied on the transformed data.

7.2 DIFFERENCING

*CALCULATING THE FIRST DIFFERENCE

Hide

```
Bitcoin_price_diff =diff(log(Bitcoin_price_ts), difference =1)
plot(Bitcoin_price_diff, type = 'o', ylab = 'Daily closing price')
```



Hide

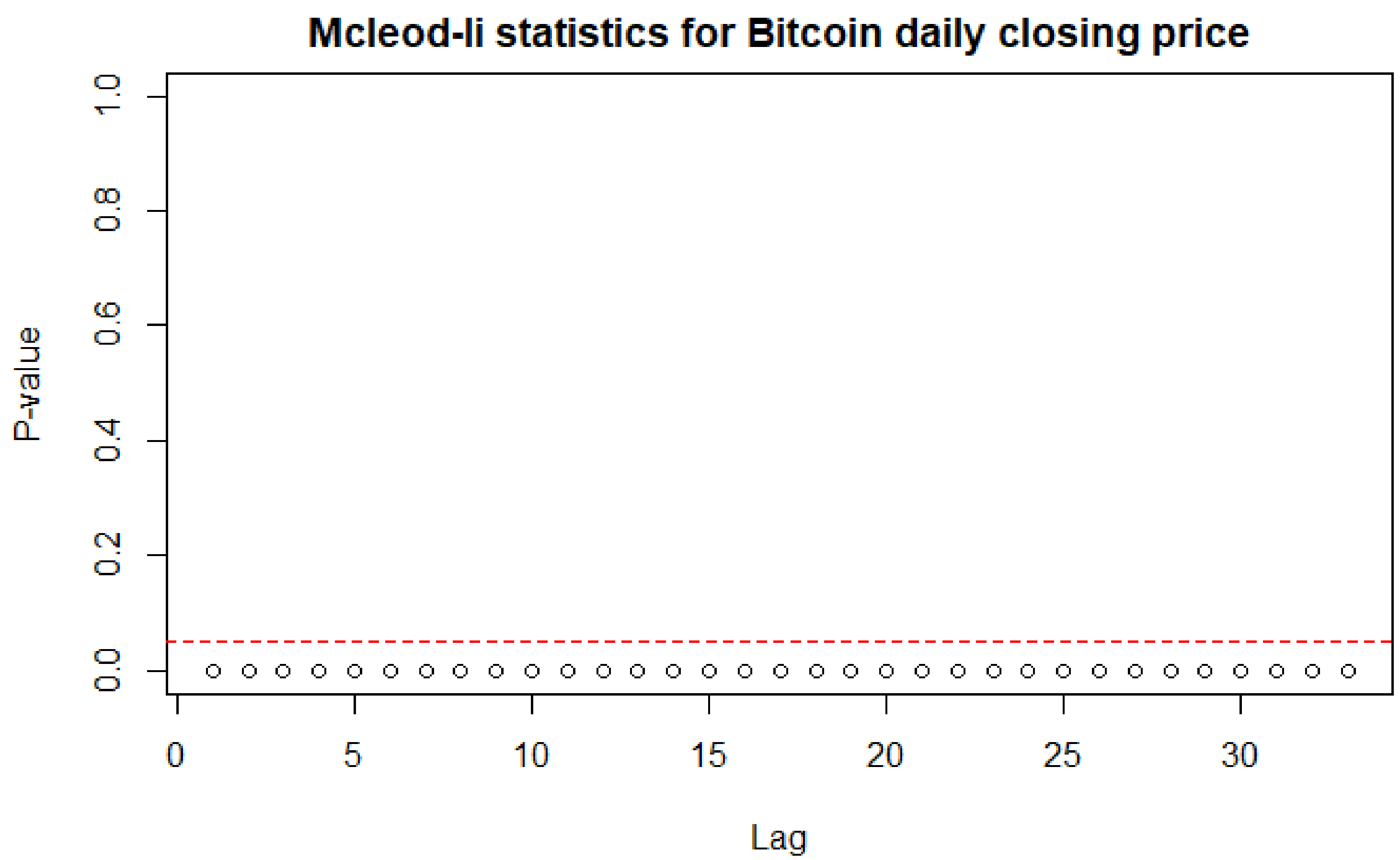
```
setorder = ar(diff(Bitcoin_price_diff))$order
adfTest(Bitcoin_price_diff, lags = setorder, title = NULL, description = NULL)
```

```
p-value smaller than printed p-value
```

```


Title:
 Augmented Dickey-Fuller Test

Test Results:
  PARAMETER:
    Lag Order: 32
  STATISTIC:
    Dickey-Fuller: -7.4259
  P VALUE:
    0.01

Description:
 Sat Jun 08 06:55:03 2019 by user: vamip
```

Hide

```
McLeod.Li.test(y=Bitcoin_price_diff, main="Mcleod-li statistics for Bitcoin daily closing price")
```



- The plot of the data after differencing, is showing that the trend has been removed. Although, change is variance is still present.
- The p-value of adf test is 0.01, that is less than alpha (0.05), therefore, the null hypothesis can be rejected, proving the data to be stationary now.
- The Mcleod-li test statistics is significant for all the lags at 5% level of significance., indicating the presence of volatility clustering, and giving the reason behind the change in variance.

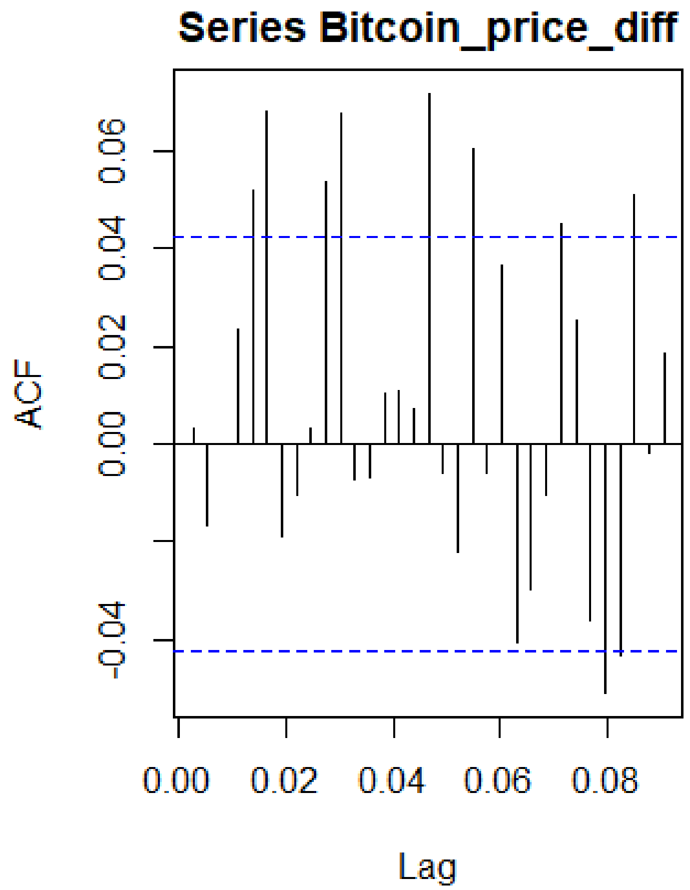
8. MODEL SPECIFICATION (ARIMA)

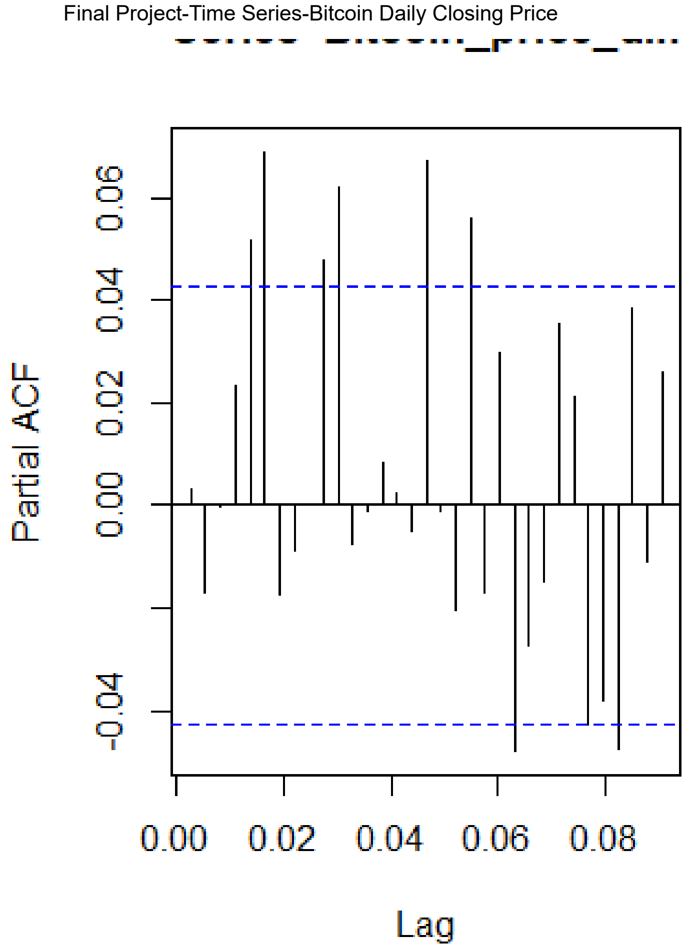
Hide

```
par(mfrow=c(1,2))
acf(Bitcoin_price_diff)
pacf(Bitcoin_price_diff)
```

Hide

```
par(mfrow= c(1,1))
```





- EACF (EXTENDED ACF)

Hide

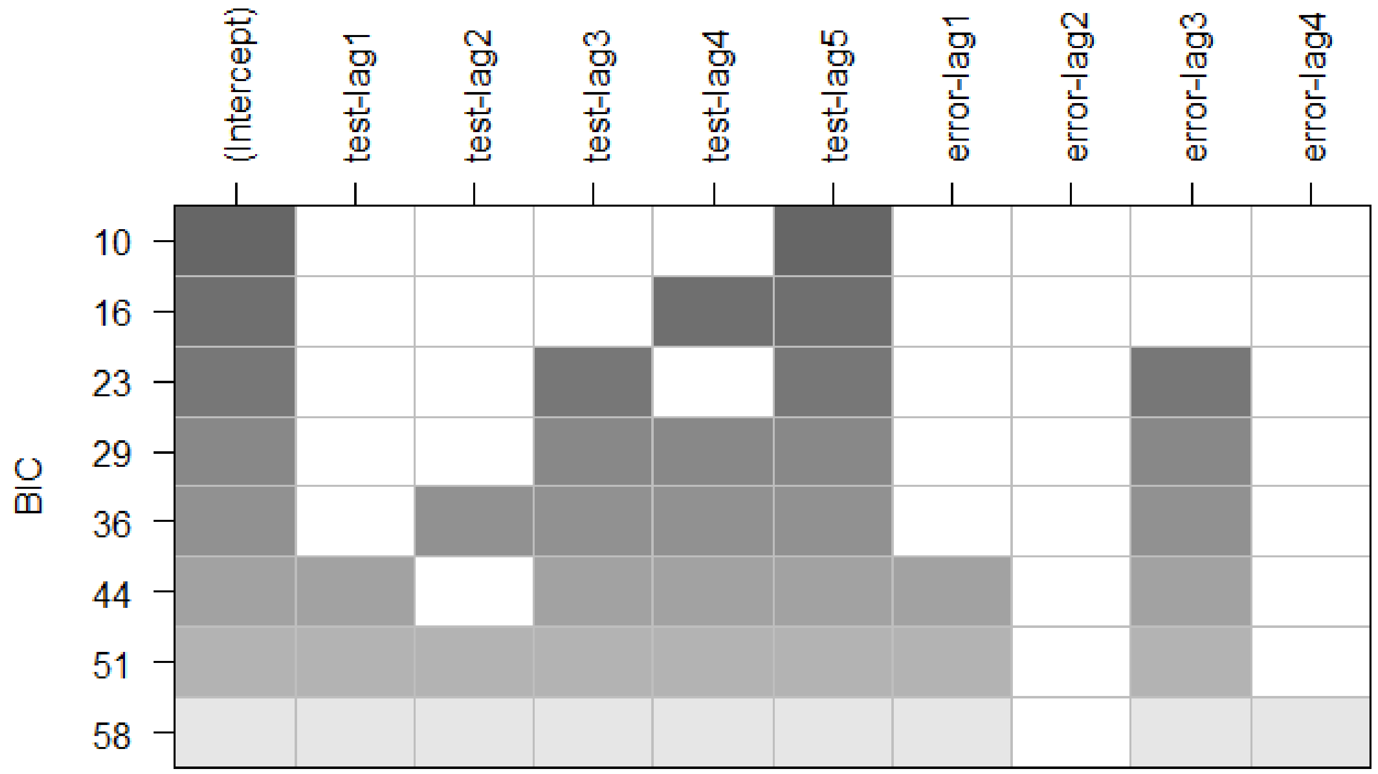
```
eacf(Bitcoin_price_diff, ar.max = 10, ma.max = 10)
```

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

*BAYESIAN INFORMATION CRITERION (BIC)

Hide

```
res_1 = armasubsets(y=Bitcoin_price_diff,nar=5,nma=4,y.name='test',ar.method='yw')
plot(res_1)
```



- To know the order of AR (auto-regressive) and MA (moving average) components of an ARMA model,Extended ACF(EACF) is used. From EACF results, the ARIMA models that can be taken into account are- ARIMA(1,1,2), ARIMA (1,1,4), ARIMA(2,1,3), ARIMA(2,1,4), ARIMA (2,1,0) and ARIMA(3,1,2)
- In the above BIC table, the shaded columns are corresponding to AR(3), AR(4) coefficients and there is one MA effect,i.e., MA(3). So, the models from the above output, that can be included in the set of possible models are ARIMA(3,1,3) and ARIMA(4,1,3).
- Hence, the set of all candidate models are- ARIMA(1,1,2), ARIMA (1,1,4), ARIMA(2,1,3), ARIMA(2,1,4), ARIMA (2,1,0), ARIMA(3,1,2), ARIMA(3,1,3) and ARIMA(4,1,3).

8.1 PARAMETER ESTIMATION (Model Testing)

- After it is ensured that the series is staionary, and the specifications of orders of the AR and MA elements for ARMA model have been calculated, the next step is the estimation of parameters of the above specified tentative models. For this least squares estimation (CSS) and maximum likelihood estimation (ML) will be applied. At last, the selelction of the best model will be established from AIC and BIC.
- ARIMA(1,1,2)

Hide

```
model_112_css = arima(log(Bitcoin_price_ts),order=c(1,1,2),method='CSS')
coeftest(model_112_css)
```

```
z test of coefficients:

     Estimate Std. Error z value Pr(>|z|)
ar1 -0.077929   0.247528 -0.3148   0.7529
ma1  0.082717   0.247621  0.3340   0.7383
ma2 -0.011566   0.021253 -0.5442   0.5863
```

Hide

```
model_112_ml = arima(log(Bitcoin_price_ts),order=c(1,1,2),method='ML')
coeftest(model_112_ml)
```

```
z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1  0.0022707  0.6764819  0.0034   0.9973
ma1  0.0023673  0.6760958  0.0035   0.9972
ma2 -0.0148271  0.0211290 -0.7017   0.4828
```

- AR(1) and both the components of MA,i.e., MA(1) and MA(2) are insignificant for both CSS and ML, since the p-values are greater than alpha (0.05).
- ARIMA (1,1,4)

Hide

```
model_114_css = arima(log(Bitcoin_price_ts),order=c(1,1,4),method='CSS')
coeftest(model_114_css)
```

```
z test of coefficients:

     Estimate Std. Error z value Pr(>|z|)
ar1  0.160174   0.287589  0.5570   0.5776
ma1 -0.158420   0.289146 -0.5479   0.5838
ma2 -0.017219   0.022350 -0.7704   0.4411
ma3  0.008396   0.021876  0.3838   0.7011
ma4  0.036093   0.024863  1.4517   0.1466
```

Hide

```
model_114_ml = arima(log(Bitcoin_price_ts),order=c(1,1,4),method='ML')
coeftest(model_114_ml)
```

```
z test of coefficients:

     Estimate Std. Error  z value Pr(>|z|)
ar1  0.913265   0.048424  18.8596   <2e-16 ***
ma1 -0.915329   0.052755 -17.3506   <2e-16 ***
ma2 -0.018482   0.029626  -0.6238   0.5327
ma3  0.019229   0.029521   0.6514   0.5148
ma4  0.031721   0.021832   1.4529   0.1462
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- All the components but AR(1) and MA(1) in ML, are insignificant.
- ARIMA (2,1,0)

Hide

```
model_210_css = arima(log(Bitcoin_price_ts),order=c(2,1,0),method='CSS')
coeftest(model_210_css)
```

```
z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1  0.0052939  0.0216847  0.2441   0.8071
ar2 -0.0155749  0.0216730 -0.7186   0.4724
```

Hide

```
model_210_ml = arima(log(Bitcoin_price_ts),order=c(2,1,0),method='ML')
coeftest(model_210_ml)
```

```
z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1  0.0045837  0.0216988  0.2112   0.8327
ar2 -0.0155873  0.0217011 -0.7183   0.4726
```

- Both the components of AR are insignificant for both CSS and ML.
- ARIMA(2,1,3)

Hide

```
model_213_css = arima(log(Bitcoin_price_ts),order=c(2,1,3),method='CSS')
coeftest(model_213_css)
```

```
z test of coefficients:

     Estimate Std. Error z value Pr(>|z|)
ar1  0.962895   0.054010  17.828   <2e-16 ***
ar2 -0.795055   0.058819 -13.517   <2e-16 ***
ma1 -0.968266   0.058019 -16.689   <2e-16 ***
ma2  0.765671   0.066824  11.458   <2e-16 ***
ma3  0.033840   0.022323   1.516   0.1295
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hide

```
model_213_ml = arima(log(Bitcoin_price_ts), order = c(2,1,3), method = 'ML')
coeftest(model_213_ml)
```

```
z test of coefficients:

      Estimate Std. Error  z value Pr(>|z|)
ar1  0.7472990  0.0140195  53.3044   <2e-16 ***
ar2 -0.9773468  0.0124041 -78.7919   <2e-16 ***
ma1 -0.7480633  0.0256522 -29.1617   <2e-16 ***
ma2  0.9557809  0.0234259  40.8001   <2e-16 ***
ma3 -0.0038779  0.0219544  -0.1766   0.8598
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- All the components of AR and MA but MA(3) are significant for both ML and CSS.
- ARIMA (2,1,4)

Hide

```
model_214_css = arima(log(Bitcoin_price_ts),order=c(2,1,4),method='CSS')
coeftest(model_214_css)
```

```
z test of coefficients:

      Estimate Std. Error  z value Pr(>|z|)
ar1  0.9412765  0.0653880  14.3953   <2e-16 ***
ar2 -0.7719471  0.0603820 -12.7844   <2e-16 ***
ma1 -0.9473336  0.0684839 -13.8329   <2e-16 ***
ma2  0.7679639  0.0677996  11.3270   <2e-16 ***
ma3  0.0003474  0.0316621   0.0110   0.9912
ma4  0.0347996  0.0228930   1.5201   0.1285
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hide

```
model_214_ml = arima(log(Bitcoin_price_ts), order = c(2,1,4), method = 'ML')
coeftest(model_214_ml)
```

```
z test of coefficients:

     Estimate Std. Error z value Pr(>|z|)
ar1  0.228327   0.357569  0.6386  0.52311
ar2  0.651728   0.338344  1.9262  0.05408 .
ma1 -0.230488   0.358335 -0.6432  0.52008
ma2 -0.673945   0.340079 -1.9817  0.04751 *
ma3  0.017083   0.022423  0.7619  0.44614
ma4  0.052693   0.024724  2.1313  0.03307 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- AR(1) and AR(2) are significant for CSS but not for ML, whereas MA(1) and MA(2) are significant for CSS and MA(2) and MA(4) are significant for ML.
- ARIMA(3,1,2)

Hide

```
model_312_css = arima(log(Bitcoin_price_ts),order=c(3,1,2),method='CSS')
coeftest(model_312_css)
```

```
z test of coefficients:

     Estimate Std. Error  z value Pr(>|z|)
ar1  0.904638   0.077770  11.6322  < 2e-16 ***
ar2 -0.709831   0.081648  -8.6938  < 2e-16 ***
ar3  0.042067   0.023384   1.7990  0.07203 .
ma1 -0.908918   0.075800 -11.9910  < 2e-16 ***
ma2  0.672523   0.078831   8.5312  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hide

```
model_312_ml = arima(log(Bitcoin_price_ts), order = c(3,1,2), method = 'ML')
coeftest(model_312_ml)
```

```
z test of coefficients:

      Estimate Std. Error  z value Pr(>|z|)
ar1  0.7424152  0.0302300  24.5589   <2e-16 ***
ar2 -0.9734566  0.0221515 -43.9455   <2e-16 ***
ar3 -0.0045663  0.0230023  -0.1985   0.8426
ma1 -0.7436814  0.0210925 -35.2582   <2e-16 ***
ma2  0.9522127  0.0182775  52.0976   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- All the components of AR and MA but AR(3) are significant for both CSS and ML.
- ARIMA(3,1,3)

Hide

```
model_313_css = arima(log(Bitcoin_price_ts),order=c(3,1,3),method='CSS')
coeftest(model_313_css)
```

```
z test of coefficients:

     Estimate Std. Error z value  Pr(>|z|)
ar1  0.087867   0.193871  0.4532 0.6503892
ar2  0.027591   0.179948  0.1533 0.8781417
ar3 -0.607158   0.162364 -3.7395 0.0001844 ***
ma1 -0.118734   0.190338 -0.6238 0.5327556
ma2 -0.054627   0.177761 -0.3073 0.7586115
ma3  0.611406   0.155628  3.9286 8.543e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hide

```
model_313_ml = arima(log(Bitcoin_price_ts),order=c(3,1,3),method='ML')
coeftest(model_313_ml)
```

```
z test of coefficients:

    Estimate Std. Error z value  Pr(>|z|)
ar1 -0.16114    0.17952 -0.8976   0.36938
ar2 -0.29820    0.13554 -2.2001   0.02780 *
ar3 -0.88765    0.17638 -5.0327 4.837e-07 ***
ma1  0.15977    0.18476  0.8647   0.38718
ma2  0.27966    0.13901  2.0118   0.04424 *
ma3  0.86090    0.17748  4.8508 1.230e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- AR(3) and MA(3) are significant for both CSS and ML.
- ARIMA (4,1,3)

Hide

```
model_413_css = arima(log(Bitcoin_price_ts),order=c(4,1,3),method='CSS')
coeftest(model_413_css)
```

```
z test of coefficients:

     Estimate Std. Error z value Pr(>|z|)
ar1  0.178863   0.362796  0.4930  0.62200
ar2  0.025244   0.347210  0.0727  0.94204
ar3 -0.525049   0.262301 -2.0017  0.04532 *
ar4  0.052326   0.022650  2.3102  0.02088 *
ma1 -0.182066   0.363703 -0.5006  0.61666
ma2 -0.052764   0.358636 -0.1471  0.88303
ma3  0.527366   0.263465  2.0017  0.04532 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hide

```
model_413_ml = arima(log(Bitcoin_price_ts),order=c(4,1,3),method='ML')
coeftest(model_413_ml)
```

```
z test of coefficients:

     Estimate Std. Error z value Pr(>|z|)  
ar1  0.415918   0.319303  1.3026  0.19272  
ar2  0.719807   0.350880  2.0514  0.04022 *
ar3 -0.294916   0.258969 -1.1388  0.25478  
ar4  0.049644   0.024759  2.0051  0.04495 *
ma1 -0.417112   0.319464 -1.3057  0.19167  
ma2 -0.742269   0.353158 -2.1018  0.03557 *
ma3  0.308013   0.258572  1.1912  0.23357  
---
Signif. codes:  0 □***□ 0.001 □**□ 0.01 □*□ 0.05 □.□ 0.1 □ □ 1
```

- AR(4) is significant for both CSS and ML methods.
- So, after applying CSS and ML methos too all the candidate models, it can concluded that ARIMA (2,1,3) and ARIMA(3,1,2) are the models with maximum number of significant components. AIC and BIC scores will further confirm the best model.

8.2 SORTING THE MODEL WITH AIC AND BIC

Hide

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

Hide

```
sc.AIC=AIC(model_112_ml,model_114_ml,model_210_ml,model_213_ml,model_214_ml,model_312_ml,model_313_ml,model_413_ml)
sc.BIC=AIC(model_112_ml,model_114_ml,model_210_ml,model_213_ml,model_214_ml,model_312_ml,model_313_ml,model_413_ml, k = log
(length(Bitcoin_price_ts)))
```

Hide

```
sort.score(sc.AIC, score = "aic")
```

	df <dbl>	AIC <dbl>
model_213_ml	6	-7321.132
model_312_ml	6	-7321.132
model_313_ml	7	-7319.343

	df <dbl>	AIC <dbl>
model_114_ml	6	-7306.098
model_214_ml	7	-7303.588
model_413_ml	8	-7302.444
model_210_ml	3	-7298.181
model_112_ml	4	-7296.156

8 rows

Hide

```
sort.score(sc.BIC, score = "aic")
```

	df <dbl>	AIC <dbl>
model_213_ml	6	-7287.149
model_312_ml	6	-7287.149
model_210_ml	3	-7281.189
model_313_ml	7	-7279.696
model_112_ml	4	-7273.501
model_114_ml	6	-7272.115
model_214_ml	7	-7263.941
model_413_ml	8	-7257.133

8 rows

- The AIC scores are the same and lowest for both the models, ARIMA(2,1,3) and ARIMA(3,1,2), so, the further procedures can be carried out with ARIMA(2,1,3).

8.3 MODEL DIAGNOSTICS

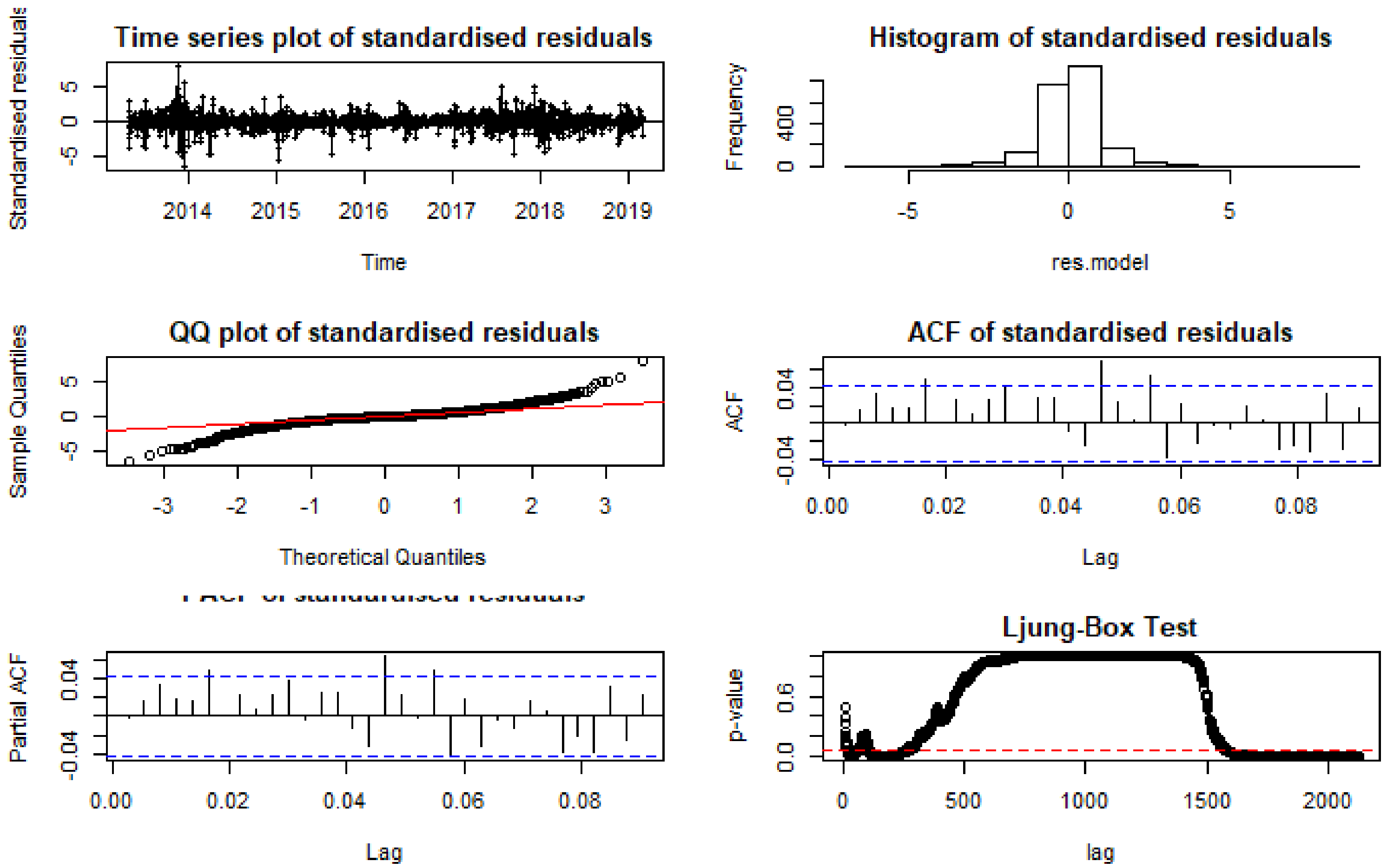
Hide

```
residual.analysis <- function(model, std = TRUE){
  library(TSA)
  library(FitAR)
  if (std == TRUE){
    res.model = rstandard(model)
  }else{
    res.model = residuals(model)
  }
  par(mfrow=c(3,2))
  plot(res.model,type='o',ylab='Standardised residuals', main="Time series plot of standardised residuals")
  abline(h=0)
  hist(res.model,main="Histogram of standardised residuals")
  qqnorm(res.model,main="QQ plot of standardised residuals")
  qqline(res.model, col = 2)
  acf(res.model,main="ACF of standardised residuals")
  pacf(res.model,main="PACF of standardised residuals")
  print(shapiro.test(res.model))
  k=0
  LBQPlot(res.model, lag.max = length(model$residuals)-1 , StartLag = k + 1, k = 0, SquaredQ = FALSE)
  par(mfrow=c(1,1))
}
```

Hide

```
residual.analysis(model = model_213_ml)
```

```

    Shapiro-Wilk normality test

data:  res.model
W = 0.89218, p-value < 2.2e-16
```

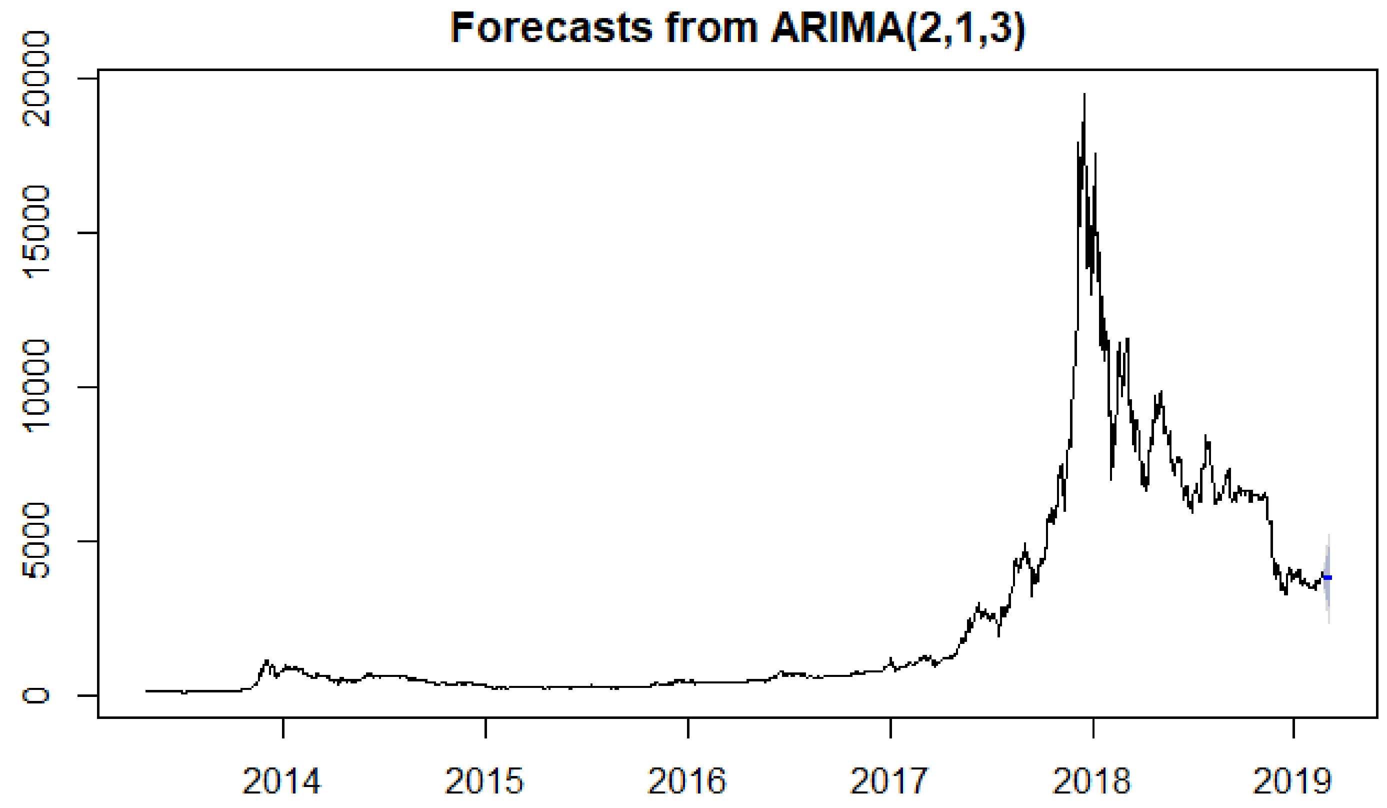


- The p-value from the Shapiro-wilk normality test is less than alpha, so the null hypothesis is rejected, stating that the stochastic components for this model are not normally distributed.
- The standardised residuals plot is showing that the trend is no longer present but change in variance still exists in the residuals.
- Both ends of the tail are deviating from the normality, hence, normality assumption can also be rejected. This deviation also indicating ARCH effect.
- Those lags that are significant in ACF plot, are confirming that autocorrelation is still present in the residuals.
- In the Ljung-Box test, all the observations are lying above the dashed line, hence null hypothesis can not be rejected, proving error terms are not correlated.
- So, on the basis of the above model diagnostic, it can be inferred that ARIMA(2,1,3) did not come out to be a good model in order to capture the dependency in the Bitcoin series.

8.4 ARIMA MODEL FORECAST

Hide

```
fit_forecast = Arima(Bitcoin_price_ts,c(2,1,3))
plot(forecast(fit_forecast,h=10))
```



Hide

```
forecast(fit_forecast,h=10)
```

```
          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2019.1534       3785.857 3489.814 4081.899 3333.098 4238.615
2019.1562       3798.957 3362.367 4235.547 3131.250 4466.664
2019.1589       3788.685 3254.798 4322.573 2972.175 4605.196
2019.1616       3796.419 3174.915 4417.922 2845.911 4746.926
2019.1644       3790.354 3095.766 4484.943 2728.073 4852.636
2019.1671       3794.919 3031.594 4558.245 2627.513 4962.325
2019.1699       3791.339 2966.778 4615.900 2530.282 5052.396
2019.1726       3794.034 2911.132 4676.936 2443.752 5144.316
2019.1753       3791.920 2855.238 4728.602 2359.389 5224.451
2019.1781       3793.511 2805.261 4781.760 2282.114 5304.908
```

- The forecast for ARIMA(2,1,3) model is very hard to interpret based on the above graph and also no valuable information is provided.
- Also, from the McLeod-Li test statistics, the series came out to be significant for all the lags, indicating the presence of volatility clustering. Additionally, even after removing the trend via ARIMA model, change in variance still exists.
- Hence the conditional variance will be dealt with the help of GARCH models.

9. GARCH MODEL

- The series that will be used for learning the GARCh effect is transformed with log and undergone the first differencing.

Hide

```
Bitcoin_price_g =diff(log(Bitcoin_price_ts))
```

9.1 ACF AND PACF PLOTS FOR LOG, ABSOLUTE VALUE AND SQUARED TRANSFORMATION

Hide

```
abs_bitcoin_g = abs(Bitcoin_price_g)
sq_bitcoin_g = Bitcoin_price_g^2
par(mfrow=c(3,2))
acf(Bitcoin_price_g, main="The sample ACF plot of LOG Bitcoin daily closing price")
pacf(Bitcoin_price_g, main="The sample PACF plot of LOG Bitcoin daily closing price")
```

Hide

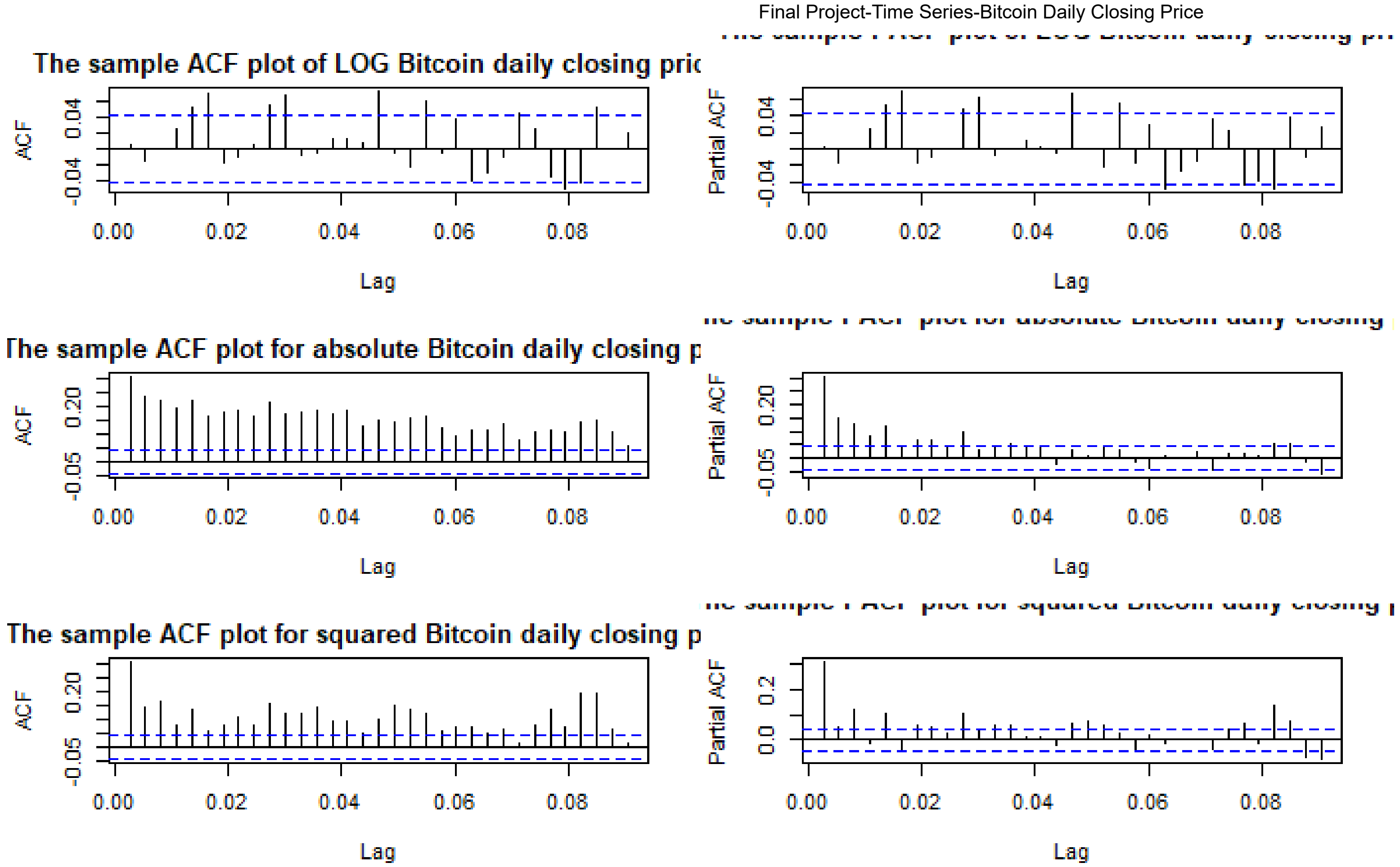
```
acf(abs_bitcoin_g, main="The sample ACF plot for absolute Bitcoin daily closing price")
pacf(abs_bitcoin_g, main="The sample PACF plot for absolute Bitcoin daily closing price")
```

Hide

```
acf(sq_bitcoin_g, main="The sample ACF plot for squared Bitcoin daily closing price")
pacf(sq_bitcoin_g, main="The sample PACF plot for squared Bitcoin daily closing price")
```

Hide

```
par(mfrow=c(1,1))
```

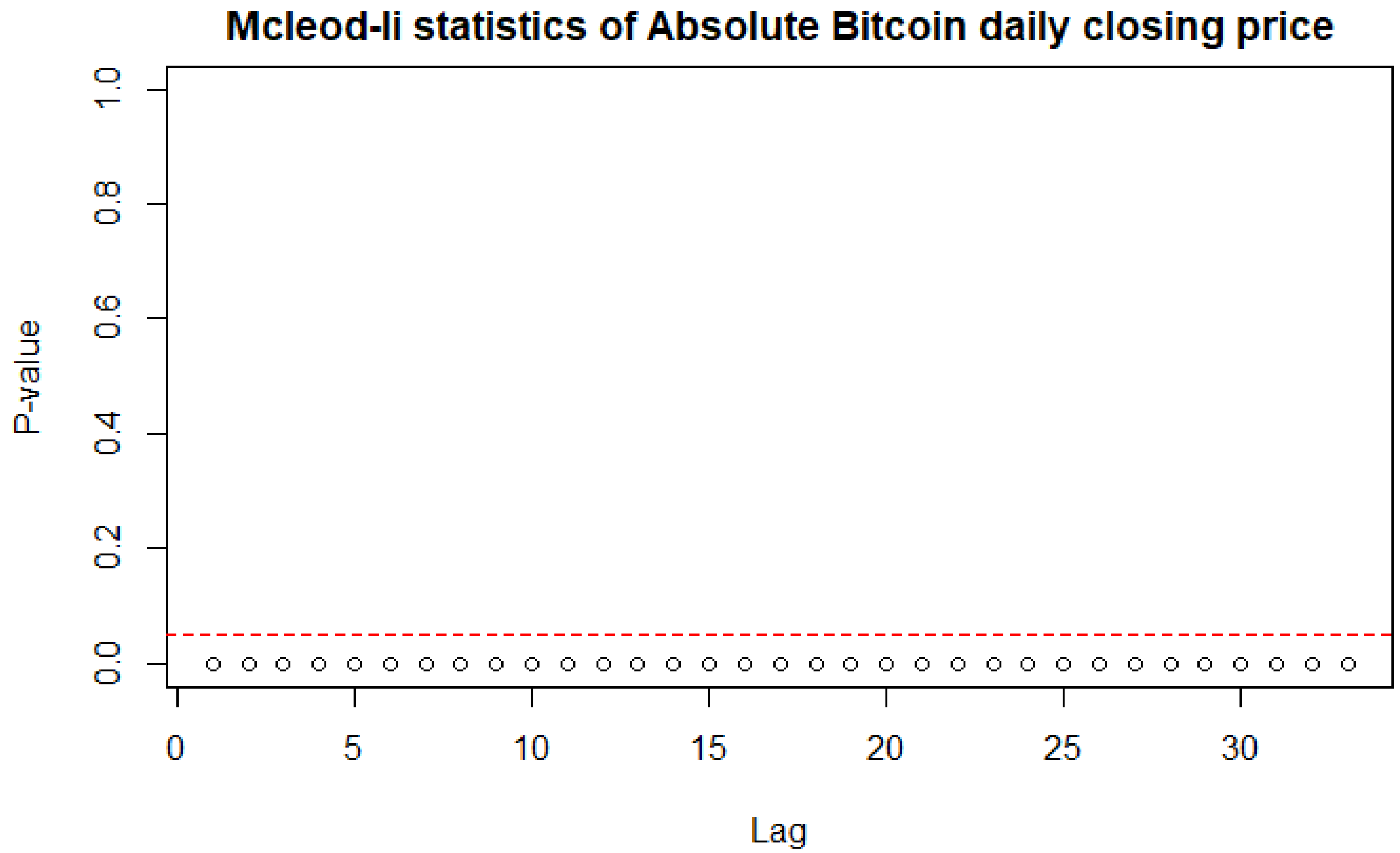


- The ACF and PACF plots of absolute value and squared transformation are showing many significant correlations with high volatile effect. This is indicating that Bitcoing daily closing prices are not identically distributed and are dependent.

9.1.1 ABSOLUTE VALUE TEST

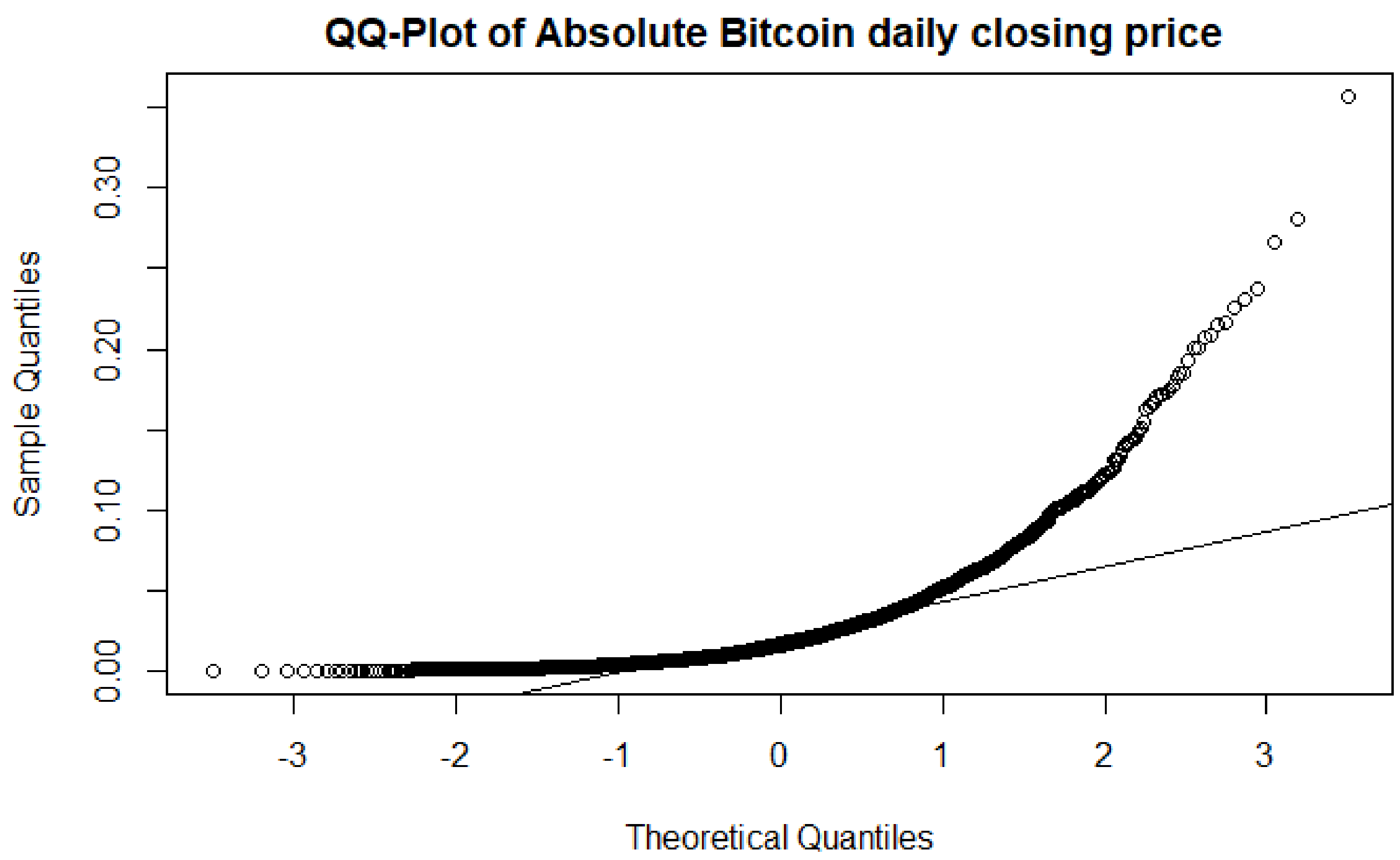
Hide

```
McLeod.Li.test(y=abs_bitcoin_g, main= "Mcleod-li statistics of Absolute Bitcoin daily closing price")
```



Hide

```
qqnorm(abs_bitcoin_g, main = "QQ-Plot of Absolute Bitcoin daily closing price ")
qqline(abs_bitcoin_g)
```



Hide

```
shapiro.test(abs_bitcoin_g)
```

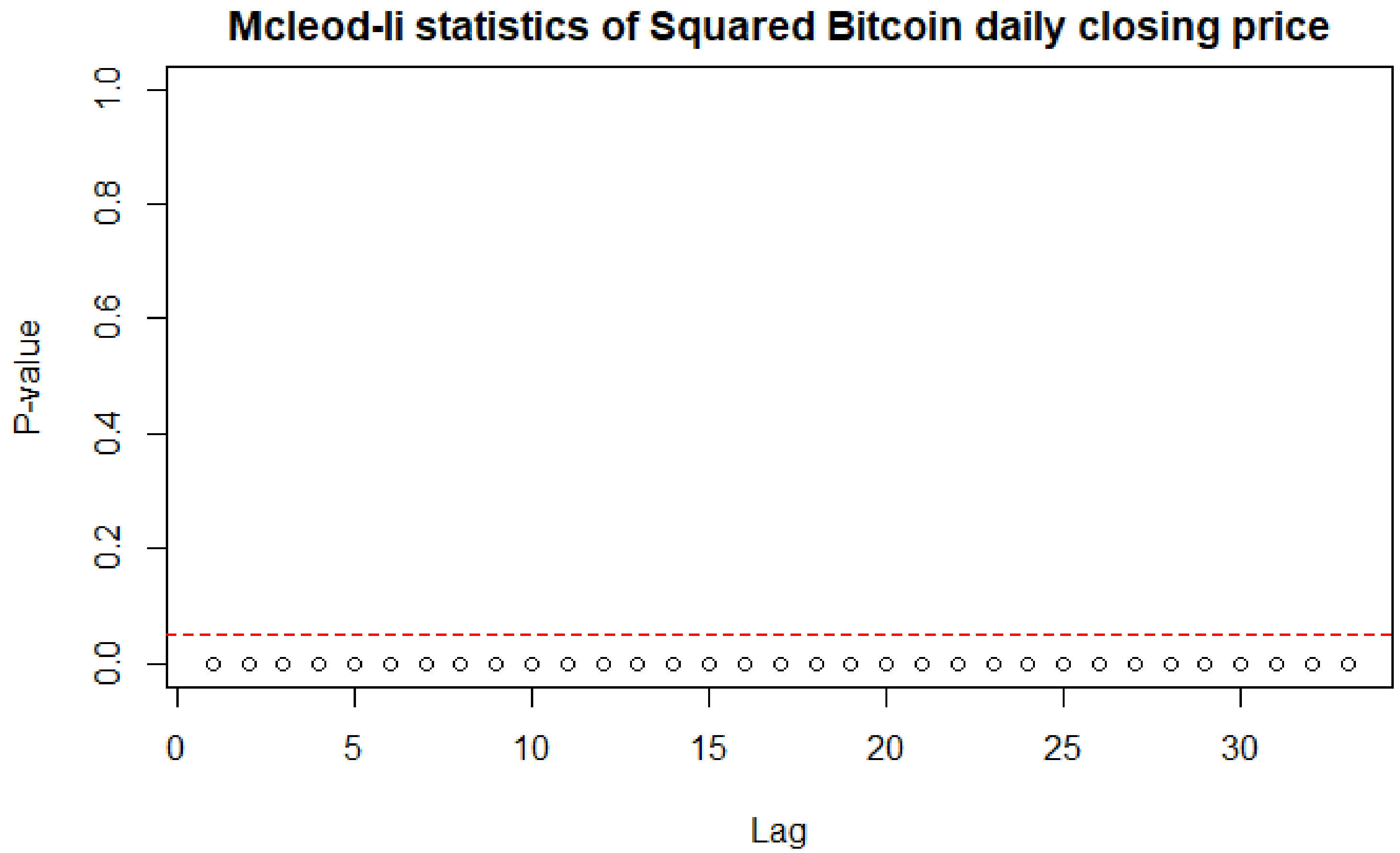
```
	Shapiro-Wilk normality test

data:  abs_bitcoin_g
W = 0.71321, p-value < 2.2e-16
```

- McLeod-Li test is showing significance for all the lags, providing evidence of volatility clustering, and that is confirmed by the tails of the QQ-plot. Also, the p-value from the Shapiro-wilk test is less than alpha, so the null hypothesis is rejected, proving the series to be not normally distributed.

9.1.2 SQUARED TRANSFORMATION TEST

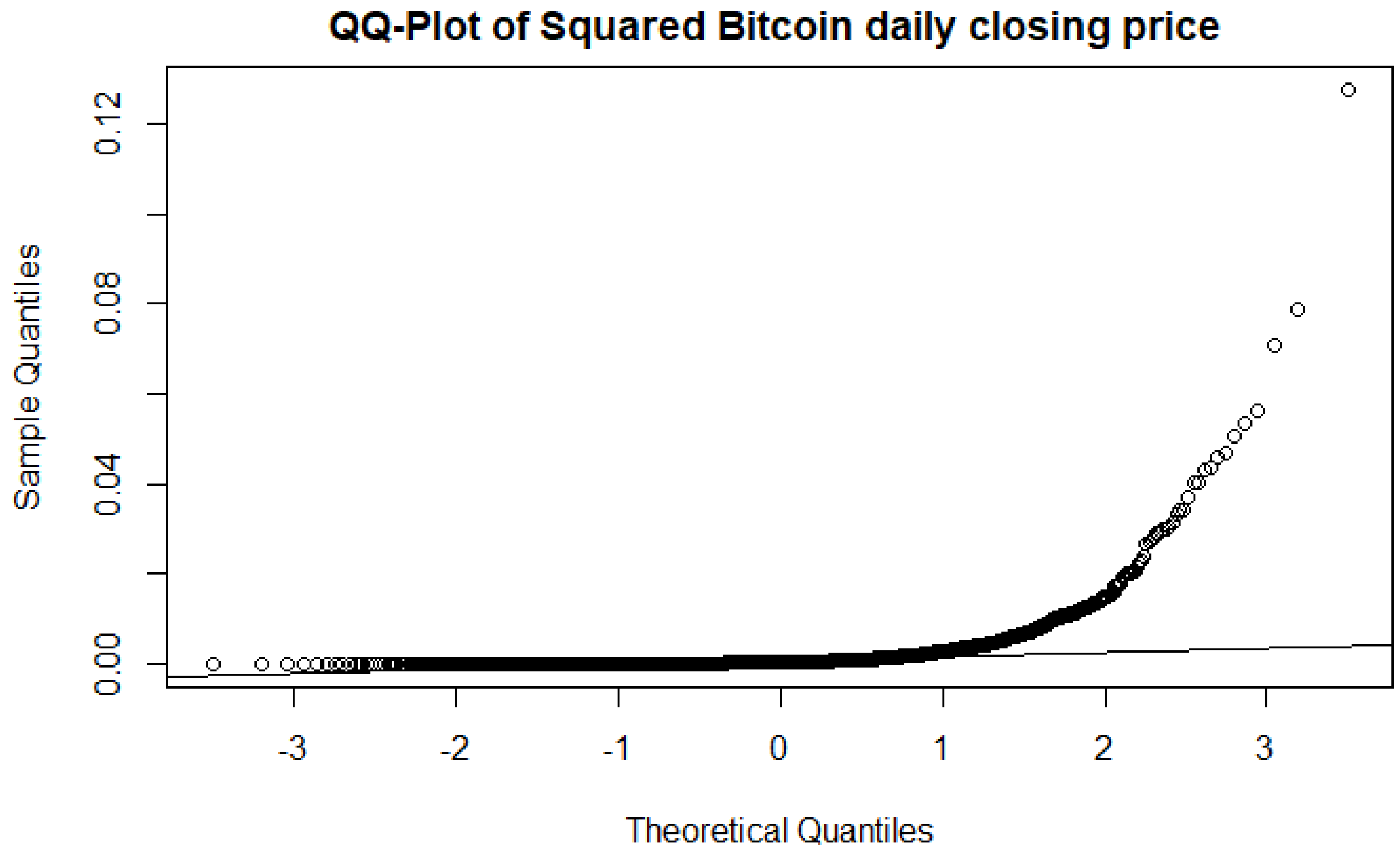
Hide

```
McLeod.Li.test(y=sq_bitcoin_g, main= "Mcleod-li statistics of Squared Bitcoin daily closing price")
```



Hide

```
qqnorm(sq_bitcoin_g, main = "QQ-Plot of Squared Bitcoin daily closing price ")
qqline(sq_bitcoin_g)
```



Hide

```
shapiro.test(sq_bitcoin_g)
```

```
	Shapiro-Wilk normality test

data:  sq_bitcoin_g
W = 0.31043, p-value < 2.2e-16
```

- The results for squared transformations are similar to that of the absolute values, i.e., McLeod-Li test is showing significance for all the lags, providing evidence of volatility clustering, and that is confirmed by the tails of the QQ-plot. Also, the p-value from the Shapiro-wilk test is less than alpha, so the null hypothesis is rejected, proving the series to be not normally distributed.

9.2 MODEL SPECIFICATION FOR GARCH MODEL

Hide

```
eacf(Bitcoin_price_g)
```

```
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 o x o o o x o o o o x  o  o  o
3 o x o 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 x x x o o o o o o  o  o  o
7 x x o x x x x o o o o  o  o  o
```

Hide

```
eacf(abs_bitcoin_g)
```

```
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 x o o x x o o o x o  o  o  o
2 x x o o o x o o o x o  o  o  o
3 x x x o o o o o o o o  o  o  o
4 x x x x o o o o o o o  o  o  o
5 x x x x x x 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 x x x x o o o o o o  o  o  o
```

Hide

```
eacf(sq_bitcoin_g)
```

```
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 x x x x x o x o x o  o  x  o
2 x x x 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 x x x x o o o o  o  o  o
```

- EACF plot for log transformed data is giving a series with white noise, providing the evidence for correlation.
- From the EACF of ABSOLUTE VALUE, ARMA(1,2) ARMA(1,3) models are identified. These models correspond to parameter settings of [max(1,2),1], [max(1,3),1]. HEnce, the corresponding tentative GARCH models are GARCH(2,1) GARCH(3,1).
- From the EACF of SQUARED TRANSFORMATION, ARMA(2,3) ARMA(2,4) models are identified. These models correspond to parameter settings of [max(2,3),2], [max(2,4),2]. HEnce, the corresponding tentative GARCH models are GARCH(3,2) GARCH(4,2).
- So all the tentative GARCH models are GARCH(2,1) GARCH(3,1), GARCH(3,2) and GARCH(4,2).

9.3 FITTING THE GARCH MODELS

- The 4 above specified GARCH models will be fitted and shall be checked for violation of residual assumption in order to find the best GARCH component order.
- GARCH (2,1)

Hide

```
m.21 = garch(Bitcoin_price_g,order=c(2,1),trace = FALSE)
summary(m.21)
```

```


Call:
garch(x = Bitcoin_price_g, order = c(2, 1), trace = FALSE)

Model:
GARCH(2,1)

Residuals:
     Min       1Q   Median       3Q      Max 
-8.39959 -0.36388  0.05552  0.50988  4.73268 

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)    
a0 5.656e-05   5.011e-06   11.287  < 2e-16 ***
a1 1.789e-01   1.157e-02   15.460  < 2e-16 ***
b1 2.244e-01   4.622e-02    4.855  1.2e-06 ***
b2 5.794e-01   4.146e-02   13.973  < 2e-16 ***
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1    1

Diagnostic Tests:
    Jarque Bera Test

data:  Residuals
X-squared = 5021.2, df = 2, p-value < 2.2e-16


    Box-Ljung test

data:  Squared.Residuals
X-squared = 0.061721, df = 1, p-value = 0.8038
```

Hide

```
m.21_2 = garchFit(formula = ~garch(2,1), data =Bitcoin_price_g )
```

```
Series Initialization:
 ARMA Model:                arma
 Formula Mean:              ~ arma(0, 0)
 GARCH Model:               garch
 Formula Variance:          ~ garch(2, 1)
 ARMA Order:                0 0
 Max ARMA Order:            0
 GARCH Order:               2 1
 Max GARCH Order:           2
 Maximum Order:             2
 Conditional Dist:          norm
 h.start:                   3
 llh.start:                 1
 Length of Series:          2129
 Recursion Init:            mci
 Series Scale:              0.04351601

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
                     U           V     params includes
    mu     -0.36117116   0.3611712 0.03611712     TRUE
    omega   0.00000100 100.0000000 0.10000000     TRUE
    alpha1  0.00000001   1.0000000 0.05000000     TRUE
    alpha2  0.00000001   1.0000000 0.05000000     TRUE
    gamma1 -0.99999999   1.0000000 0.10000000    FALSE
    gamma2 -0.99999999   1.0000000 0.10000000    FALSE
    beta1   0.00000001   1.0000000 0.80000000     TRUE
    delta   0.00000000   2.0000000 2.00000000    FALSE
    skew    0.10000000  10.0000000 1.00000000    FALSE
    shape   1.00000000  10.0000000 4.00000000    FALSE
 Index List of Parameters to be Optimized:
    mu  omega alpha1 alpha2  beta1
     1      2      3      4      7
 Persistence:                  0.9


--- START OF TRACE ---
Selected Algorithm: nlminb

R coded nlminb Solver:

  0:     2753.0159: 0.0361171 0.100000 0.0500000 0.0500000 0.800000
  1:     2724.8599: 0.0361159 0.0719500 0.0575288 0.0501946 0.786597
  2:     2714.1797: 0.0361141 0.0725709 0.0822143 0.0669892 0.798057
  3:     2699.2589: 0.0361132 0.0543973 0.0804901 0.0626676 0.788755
  4:     2691.9547: 0.0361102 0.0500782 0.0968282 0.0712460 0.797659
  5:     2689.9556: 0.0361093 0.0430630 0.0969127 0.0695648 0.795622
  6:     2688.4457: 0.0361067 0.0427216 0.101936 0.0698180 0.801170
```

```
  7:     2686.8096: 0.0360992 0.0334748 0.107653 0.0628389 0.808775
  8:     2684.7152: 0.0360893 0.0355155 0.112924 0.0517497 0.817129
  9:     2681.7168: 0.0360627 0.0322734 0.120429 0.0196303 0.832409
 10:     2678.7644: 0.0360286 0.0221826 0.122053 0.00425619 0.863886
 11:     2678.5455: 0.0360284 0.0214373 0.121767 0.00369286 0.863415
 12:     2678.3385: 0.0360259 0.0231344 0.121756 0.000394761 0.863551
 13:     2678.2999: 0.0360256 0.0221947 0.121718 1.00000e-08 0.863270
 14:     2678.2399: 0.0360100 0.0227064 0.122545 1.00000e-08 0.863079
 15:     2678.2272: 0.0359658 0.0218844 0.124078 1.00000e-08 0.862699
 16:     2678.1967: 0.0358119 0.0223593 0.124254 1.00000e-08 0.863232
 17:     2678.1765: 0.0356634 0.0218616 0.123376 1.00000e-08 0.864098
 18:     2678.1751: 0.0356613 0.0217838 0.123631 1.00000e-08 0.864220
 19:     2678.1747: 0.0356610 0.0217312 0.123627 1.00000e-08 0.864193
 20:     2678.1745: 0.0356594 0.0217496 0.123660 1.00000e-08 0.864183
 21:     2678.1677: 0.0355342 0.0220148 0.125022 1.00000e-08 0.862785
 22:     2678.1632: 0.0353903 0.0221437 0.125009 1.00000e-08 0.862935
 23:     2677.8546: 0.0250132 0.0210731 0.121596 1.00000e-08 0.866500
 24:     2677.8278: 0.0233013 0.0220510 0.123385 1.00000e-08 0.863653
 25:     2677.8273: 0.0222152 0.0218602 0.124964 1.00000e-08 0.863228
 26:     2677.8232: 0.0225588 0.0219241 0.124010 1.00000e-08 0.863623
 27:     2677.8232: 0.0225719 0.0219212 0.124006 1.00000e-08 0.863624
 28:     2677.8232: 0.0225707 0.0219208 0.124005 1.00000e-08 0.863627

Final Estimate of the Negative LLH:
 LLH:  -3995.796    norm LLH:  -1.876842
          mu        omega       alpha1       alpha2        beta1 
9.821859e-04 4.151011e-05 1.240046e-01 1.000000e-08 8.636266e-01 

R-optimhess Difference Approximated Hessian Matrix:
                 mu        omega        alpha1        alpha2         beta1
mu     -2161250.709     -4258414      7807.727      5196.389      3709.128
omega  -4258413.969 -75954462203 -36605456.263 -37640212.956 -67519674.784
alpha1     7807.727    -36605456    -35883.212    -35116.071    -48200.754
alpha2     5196.389    -37640213    -35116.071    -36756.116    -50240.981
beta1      3709.128    -67519675    -48200.754    -50240.981    -77586.942
attr(,"time")
Time difference of 0.05902314 secs

--- END OF TRACE ---


Time to Estimate Parameters:
 Time difference of 2.835877 secs
```

Hide

```
summary(m.21_2)
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(2, 1), data = Bitcoin_price_g)

Mean and Variance Equation:
 data ~ garch(2, 1)
<environment: 0x000000002a358df0>
 [data = Bitcoin_price_g]

Conditional Distribution:
 norm

Coefficient(s):
        mu       omega      alpha1      alpha2       beta1
0.00098219  0.00004151  0.12400465  0.00000001  0.86362655

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     9.822e-04   6.814e-04    1.441 0.149458
omega  4.151e-05   1.091e-05    3.804 0.000143 ***
alpha1 1.240e-01   2.078e-02    5.967 2.41e-09 ***
alpha2 1.000e-08   2.985e-02    0.000 1.000000
beta1  8.636e-01   2.232e-02   38.696  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log Likelihood:
 3995.796    normalized:  1.876842

Description:
 Sat Jun 08 06:55:22 2019 by user: vamip


Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  5099.268  0
 Shapiro-Wilk Test  R    W      0.9095853 0
 Ljung-Box Test     R    Q(10)  40.74306  1.252929e-05
 Ljung-Box Test     R    Q(15)  44.76348  8.343757e-05
 Ljung-Box Test     R    Q(20)  54.25097  5.309314e-05
 Ljung-Box Test     R^2  Q(10)  11.06538  0.3524474
 Ljung-Box Test     R^2  Q(15)  14.55051  0.4842525
 Ljung-Box Test     R^2  Q(20)  16.16431  0.7063798
 LM Arch Test       R    TR^2   11.43366  0.4921642

Information Criterion Statistics:
```

```
      AIC       BIC       SIC      HQIC 
-3.748987 -3.735686 -3.748998 -3.744118 
```

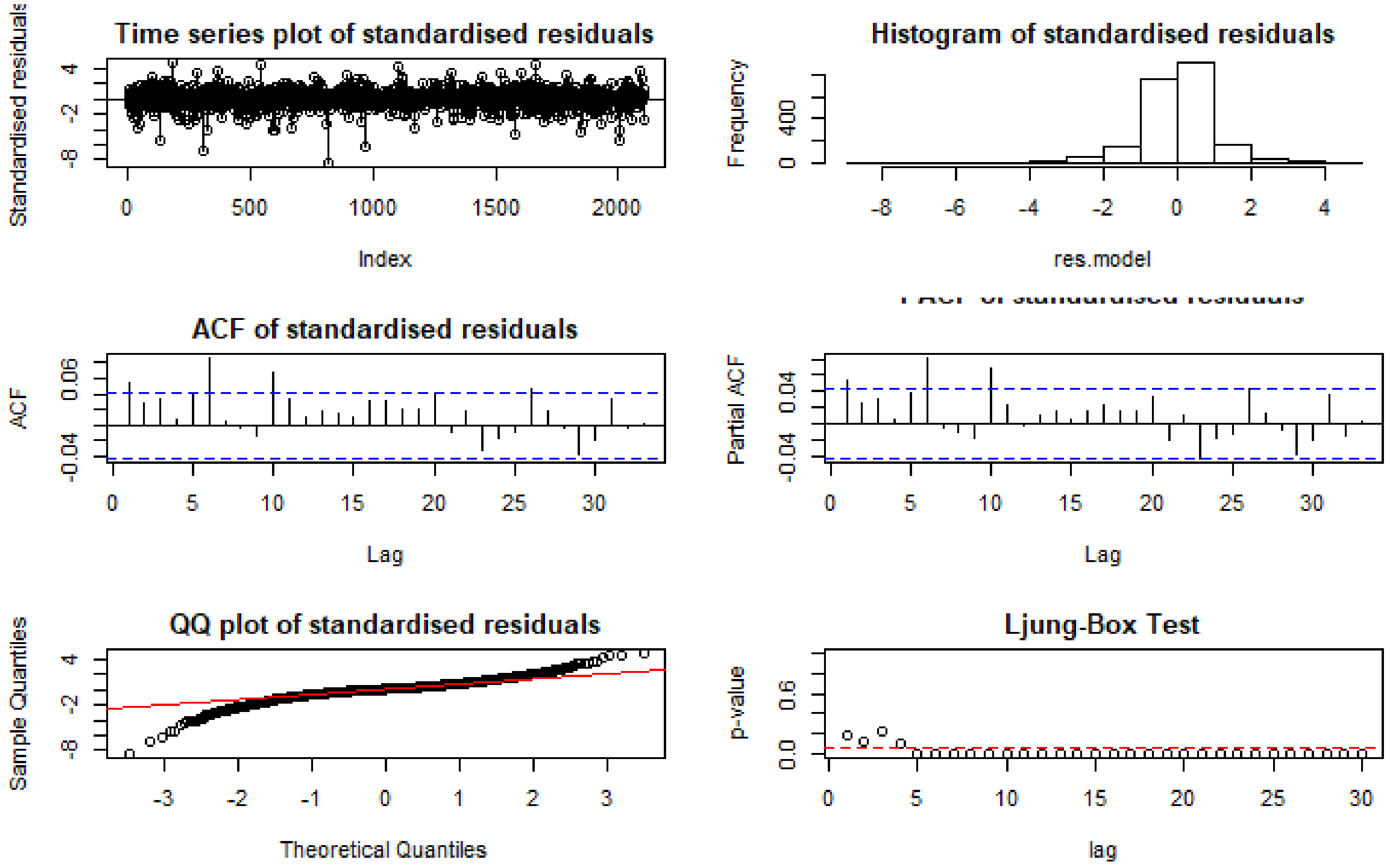
Hide

```
residual.analysis <- function(model, std = TRUE,start = 2, class = c("ARIMA","GARCH","ARMA-GARCH")[1]){
  # 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' ")
  }
  par(mfrow=c(3,2))
  plot(res.model,type='o',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)
  print(shapiro.test(res.model))
  k=0
  LBQPlot(res.model, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
}
```

Hide

```
residual.analysis(m.21,class="GARCH",start=25)
```

```
	Shapiro-Wilk normality test

data:  res.model
W = 0.91066, p-value < 2.2e-16
```



- For GARCH (2,1), all the components are significant with p-value less than 0.05. With the p-value of shapiro-wilk test, it can be said that normality does not exist. The autocorrelation is no longer present in the residuals.
- GARCH (3.1)

Hide

```
m.31 = garch(Bitcoin_price_g,order=c(3,1),trace = FALSE)
summary(m.31)
```

```
Call:
garch(x = Bitcoin_price_g, order = c(3, 1), trace = FALSE)

Model:
GARCH(3,1)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3567 -0.3605  0.0567  0.5055  4.6630

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 6.856e-05   7.806e-06    8.783  < 2e-16 ***
a1 2.082e-01   2.083e-02    9.997  < 2e-16 ***
b1 3.398e-01   8.276e-02    4.106 4.02e-05 ***
b2 4.379e-01   7.642e-02    5.731 1.00e-08 ***
b3 1.401e-08   7.256e-02    0.000        1
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1     1

Diagnostic Tests:
    Jarque Bera Test

data:  Residuals
X-squared = 5622.7, df = 2, p-value < 2.2e-16


    Box-Ljung test

data:  Squared.Residuals
X-squared = 0.013398, df = 1, p-value = 0.9079
```

Hide

```
m.31_2 = garchFit(formula = ~garch(3,1), data =Bitcoin_price_g )
```

```
Series Initialization:
 ARMA Model:                arma
 Formula Mean:              ~ arma(0, 0)
 GARCH Model:               garch
 Formula Variance:          ~ garch(3, 1)
 ARMA Order:                0 0
 Max ARMA Order:            0
 GARCH Order:               3 1
 Max GARCH Order:           3
 Maximum Order:             3
 Conditional Dist:          norm
 h.start:                   4
 llh.start:                 1
 Length of Series:          2129
 Recursion Init:            mci
 Series Scale:              0.04351601

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
                     U           V     params includes
    mu     -0.36117116   0.3611712 0.03611712     TRUE
    omega   0.00000100 100.0000000 0.10000000     TRUE
    alpha1  0.00000001   1.0000000 0.03333333     TRUE
    alpha2  0.00000001   1.0000000 0.03333333     TRUE
    alpha3  0.00000001   1.0000000 0.03333333     TRUE
    gamma1 -0.99999999   1.0000000 0.10000000    FALSE
    gamma2 -0.99999999   1.0000000 0.10000000    FALSE
    gamma3 -0.99999999   1.0000000 0.10000000    FALSE
    beta1   0.00000001   1.0000000 0.80000000     TRUE
    delta   0.00000000   2.0000000 2.00000000    FALSE
    skew    0.10000000  10.0000000 1.00000000    FALSE
    shape   1.00000000  10.0000000 4.00000000    FALSE
 Index List of Parameters to be Optimized:
    mu  omega alpha1 alpha2 alpha3  beta1
     1      2      3      4      5      9
 Persistence:                  0.9


--- START OF TRACE ---
Selected Algorithm: nlminb

R coded nlminb Solver:

  0:     2761.2265: 0.0361171 0.100000 0.0333333 0.0333333 0.0333333 0.800000
  1:     2733.2207: 0.0361162 0.0812585 0.0413092 0.0337180 0.0332420 0.791052
  2:     2698.4953: 0.0361130 0.0550049 0.0766734 0.0486409 0.0475810 0.791261
  3:     2694.9636: 0.0361126 0.0507427 0.0767641 0.0474825 0.0464671 0.789551
  4:     2693.3841: 0.0361117 0.0463376 0.0783910 0.0466526 0.0458006 0.788846
```

```
  5:     2691.4749: 0.0361089 0.0442039 0.0866727 0.0475279 0.0474114 0.793115
  6:     2689.7352: 0.0360985 0.0363370 0.0955423 0.0357087 0.0386887 0.797858
  7:     2685.8747: 0.0360864 0.0412869 0.108693 0.0256025 0.0324882 0.804244
  8:     2681.6270: 0.0360283 0.0243854 0.121677 1.00000e-08 0.0138400 0.842917
  9:     2680.7524: 0.0360060 0.0258763 0.121843 1.00000e-08 0.0119411 0.857259
 10:     2678.6064: 0.0359841 0.0266698 0.128746 1.00000e-08 0.000280954 0.852034
 11:     2678.5022: 0.0359832 0.0253402 0.129313 1.00000e-08 0.000263405 0.852602
 12:     2678.4133: 0.0359802 0.0257241 0.129814 1.00000e-08 1.00000e-08 0.853982
 13:     2678.3220: 0.0359721 0.0246702 0.129147 1.00000e-08 1.00000e-08 0.854880
 14:     2678.2397: 0.0359497 0.0241673 0.128043 1.00000e-08 1.00000e-08 0.857672
 15:     2678.1930: 0.0359039 0.0228468 0.126376 1.00000e-08 1.00000e-08 0.859548
 16:     2678.1479: 0.0358146 0.0228555 0.125462 1.00000e-08 1.00000e-08 0.861191
 17:     2678.1336: 0.0357060 0.0224920 0.125010 1.00000e-08 1.00000e-08 0.861715
 18:     2677.9374: 0.0305200 0.0211502 0.120354 1.00000e-08 1.00000e-08 0.867715
 19:     2677.7704: 0.0253313 0.0219167 0.124474 1.00000e-08 1.00000e-08 0.863146
 20:     2677.7620: 0.0241109 0.0223714 0.123906 1.00000e-08 1.00000e-08 0.863019
 21:     2677.7519: 0.0228902 0.0221189 0.124252 1.00000e-08 1.00000e-08 0.863291
 22:     2677.7497: 0.0228902 0.0220017 0.124213 1.00000e-08 1.00000e-08 0.863231
 23:     2677.7496: 0.0228853 0.0220237 0.124210 1.00000e-08 1.00000e-08 0.863245
 24:     2677.7495: 0.0228763 0.0220354 0.124099 1.00000e-08 1.00000e-08 0.863260
 25:     2677.7494: 0.0228564 0.0220425 0.124092 1.00000e-08 1.00000e-08 0.863280
 26:     2677.7483: 0.0222422 0.0219619 0.123775 1.00000e-08 1.00000e-08 0.863604
 27:     2677.7483: 0.0221588 0.0219740 0.123874 1.00000e-08 1.00000e-08 0.863537
 28:     2677.7483: 0.0221611 0.0219736 0.123870 1.00000e-08 1.00000e-08 0.863539

Final Estimate of the Negative LLH:
 LLH:  -3995.871    norm LLH:  -1.876877
          mu        omega       alpha1       alpha2       alpha3        beta1 
9.643607e-04 4.161011e-05 1.238702e-01 1.000000e-08 1.000000e-08 8.635392e-01 

R-optimhess Difference Approximated Hessian Matrix:
                  mu        omega        alpha1        alpha2        alpha3        beta1
mu     -2161275.6953     -4300492      7598.248      5798.643  2.458677e+02      3788.13
omega  -4300492.4289 -75871686536 -36614986.025 -37622227.460 -3.955236e+07 -67446181.13
alpha1     7598.2480    -36614986    -35939.444    -35111.610 -3.625128e+04    -48172.55
alpha2     5798.6431    -37622227    -35111.610    -36648.370 -3.796008e+04    -50142.79
alpha3      245.8677    -39552356    -36251.283    -37960.078 -4.046583e+04    -53003.75
beta1      3788.1296    -67446181    -48172.546    -50142.794 -5.300375e+04    -77429.76
attr(,"time")
Time difference of 0.08233714 secs

--- END OF TRACE ---


Time to Estimate Parameters:
 Time difference of 3.534589 secs
```

Hide

```
summary(m.31_2)
```

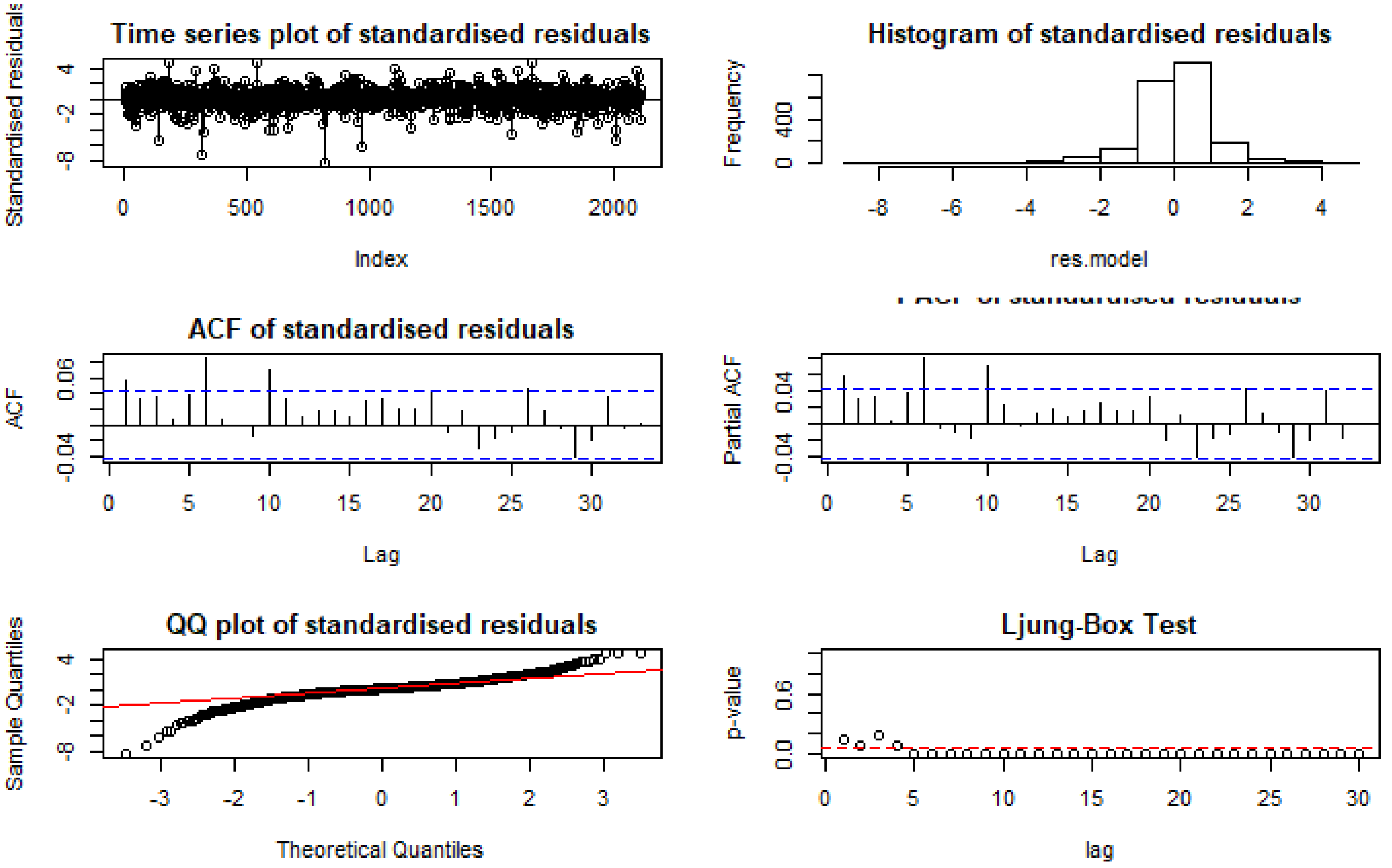
```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(3, 1), data = Bitcoin_price_g)

Mean and Variance Equation:
 data ~ garch(3, 1)
<environment: 0x000000000486ef50>
 [data = Bitcoin_price_g]

Conditional Distribution:
 norm

Coefficient(s):
        mu       omega      alpha1      alpha2      alpha3       beta1
0.00096436  0.00004161  0.12387024  0.00000001  0.00000001  0.86353921

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     9.644e-04   6.875e-04    1.403  0.16070
omega  4.161e-05   1.240e-05    3.356  0.00079 ***
alpha1 1.239e-01   2.096e-02    5.909 3.43e-09 ***
alpha2 1.000e-08   3.869e-02    0.000  1.00000
alpha3 1.000e-08   3.573e-02    0.000  1.00000
beta1  8.635e-01   2.690e-02   32.103  < 2e-16 ***
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1     1

Log Likelihood:
 3995.871    normalized:  1.876877

Description:
 Sat Jun 08 06:55:29 2019 by user: vamip


Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  5094.658  0
 Shapiro-Wilk Test  R    W      0.9096243 0
 Ljung-Box Test     R    Q(10)  40.72215  1.263639e-05
 Ljung-Box Test     R    Q(15)  44.74102  8.412008e-05
 Ljung-Box Test     R    Q(20)  54.23053  5.34658e-05
 Ljung-Box Test     R^2  Q(10)  11.06178  0.3527249
 Ljung-Box Test     R^2  Q(15)  14.5586   0.483654
 Ljung-Box Test     R^2  Q(20)  16.16849  0.7061181
 LM Arch Test       R    TR^2   11.4242   0.4929566
```

```
Information Criterion Statistics:
      AIC       BIC       SIC      HQIC 
-3.748118 -3.732157 -3.748133 -3.742276 
```

Hide

```
residual.analysis(m.31,class="GARCH",start=20)
```

```
    Shapiro-Wilk normality test

data:  res.model
W = 0.90762, p-value < 2.2e-16
```



- All the components are not significant from both the above estimation methods, with few of the components having p-value greater than 0.05. The p-value from shapiro wilk test is also less than 0.05, confirming that series is not normal, and auto-correlation is not present in the residuals.
- GARCH (3,2)

Hide

```
m.32 = garch(Bitcoin_price_g,order=c(3,2),trace = FALSE)
summary(m.32)
```

```
Call:
garch(x = Bitcoin_price_g, order = c(3, 2), trace = FALSE)

Model:
GARCH(3,2)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.4214 -0.3570  0.0541  0.4887  4.9302 

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)    
a0 7.095e-05   2.058e-05    3.447 0.000567 ***
a1 2.410e-01   2.384e-02   10.109  < 2e-16 ***
a2 1.697e-14   8.061e-02    0.000 1.000000    
b1 3.574e-01   3.162e-01    1.130 0.258401    
b2 1.779e-01   1.373e-01    1.296 0.195091    
b3 2.212e-01   1.393e-01    1.588 0.112234    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Diagnostic Tests:
    Jarque Bera Test

data:  Residuals
X-squared = 6541.5, df = 2, p-value < 2.2e-16


    Box-Ljung test

data:  Squared.Residuals
X-squared = 0.13409, df = 1, p-value = 0.7142
```

Hide

```
m.32_2 = garchFit(formula = ~garch(3,2), data =Bitcoin_price_g )
```

```
Series Initialization:
 ARMA Model:                arma
 Formula Mean:              ~ arma(0, 0)
 GARCH Model:               garch
 Formula Variance:          ~ garch(3, 2)
 ARMA Order:                0 0
 Max ARMA Order:            0
 GARCH Order:               3 2
 Max GARCH Order:           3
 Maximum Order:             3
 Conditional Dist:          norm
 h.start:                   4
 llh.start:                 1
 Length of Series:          2129
 Recursion Init:            mci
 Series Scale:              0.04351601

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
                     U           V     params includes
    mu     -0.36117116   0.3611712 0.03611712     TRUE
    omega   0.00000100 100.0000000 0.10000000     TRUE
    alpha1  0.00000001   1.0000000 0.03333333     TRUE
    alpha2  0.00000001   1.0000000 0.03333333     TRUE
    alpha3  0.00000001   1.0000000 0.03333333     TRUE
    gamma1 -0.99999999   1.0000000 0.10000000    FALSE
    gamma2 -0.99999999   1.0000000 0.10000000    FALSE
    gamma3 -0.99999999   1.0000000 0.10000000    FALSE
    beta1   0.00000001   1.0000000 0.40000000     TRUE
    beta2   0.00000001   1.0000000 0.40000000     TRUE
    delta   0.00000000   2.0000000 2.00000000    FALSE
    skew    0.10000000  10.0000000 1.00000000    FALSE
    shape   1.00000000  10.0000000 4.00000000    FALSE
 Index List of Parameters to be Optimized:
    mu  omega alpha1 alpha2 alpha3  beta1  beta2
     1      2      3      4      5      9     10
 Persistence:                  0.9


--- START OF TRACE ---
Selected Algorithm: nlminb

R coded nlminb Solver:

  0:     2761.7391: 0.0361171 0.100000 0.0333333 0.0333333 0.0333333 0.400000 0.400000
  1:     2732.9621: 0.0361159 0.0734236 0.0424958 0.0327784 0.0322276 0.385315 0.385218
  2:     2706.1684: 0.0361142 0.0697932 0.0690381 0.0480534 0.0473036 0.390123 0.390170
  3:     2696.3064: 0.0361115 0.0409873 0.0808151 0.0485876 0.0478853 0.378755 0.378874
```

```
  4:     2681.9094: 0.0360993 0.0439119 0.114260 0.0519221 0.0556707 0.381085 0.383514
  5:     2677.2603: 0.0360722 0.0389528 0.139284 0.0306119 0.0456013 0.377014 0.385088
  6:     2675.4670: 0.0360350 0.0424100 0.151138 0.00421752 0.0324068 0.381503 0.398559
  7:     2675.2132: 0.0359801 0.0264419 0.159081 1.00000e-08 0.0135435 0.388491 0.417977
  8:     2674.8331: 0.0359456 0.0277872 0.165147 1.00000e-08 0.00544243 0.393813 0.430082
  9:     2673.0342: 0.0358918 0.0284251 0.163587 1.00000e-08 0.00646412 0.386030 0.428259
 10:     2672.9121: 0.0358486 0.0268504 0.164724 1.00000e-08 0.000597286 0.385423 0.433737
 11:     2672.8791: 0.0357684 0.0284119 0.165046 1.00000e-08 1.00000e-08 0.379982 0.439347
 12:     2672.7273: 0.0356861 0.0281859 0.165481 1.00000e-08 1.00000e-08 0.373368 0.443910
 13:     2672.1473: 0.0347339 0.0286835 0.188062 1.00000e-08 1.00000e-08 0.301916 0.499864
 14:     2671.6354: 0.0337881 0.0330485 0.194340 1.00000e-08 1.00000e-08 0.231038 0.560473
 15:     2671.5397: 0.0337876 0.0322399 0.193998 1.00000e-08 1.00000e-08 0.230665 0.560113
 16:     2671.4114: 0.0337530 0.0306951 0.185830 1.00000e-08 1.00000e-08 0.234622 0.565561
 17:     2671.4063: 0.0330702 0.0286811 0.184806 1.00000e-08 1.00000e-08 0.227675 0.572677
 18:     2671.2815: 0.0326872 0.0295797 0.184828 1.00000e-08 1.00000e-08 0.229003 0.572115
 19:     2671.2546: 0.0323221 0.0297060 0.184204 1.00000e-08 1.00000e-08 0.231308 0.569387
 20:     2671.0773: 0.0250965 0.0284519 0.183977 1.00000e-08 1.00000e-08 0.206647 0.595534
 21:     2670.9765: 0.0178340 0.0291377 0.180689 1.00000e-08 1.00000e-08 0.229498 0.576834
 22:     2670.9369: 0.0155581 0.0286944 0.180815 1.00000e-08 1.00000e-08 0.233320 0.570154
 23:     2670.8902: 0.0163934 0.0303133 0.183591 1.00000e-08 1.00000e-08 0.226543 0.573468
 24:     2670.8698: 0.0172776 0.0299548 0.181635 1.00000e-08 1.00000e-08 0.228054 0.574300
 25:     2670.8609: 0.0181648 0.0298256 0.180904 1.00000e-08 1.00000e-08 0.227253 0.575570
 26:     2670.8576: 0.0188689 0.0294878 0.179427 1.00000e-08 1.00000e-08 0.228378 0.575802
 27:     2670.8576: 0.0189332 0.0295527 0.179629 1.00000e-08 1.00000e-08 0.227982 0.575970
 28:     2670.8576: 0.0189130 0.0295462 0.179615 1.00000e-08 1.00000e-08 0.228045 0.575926
 29:     2670.8576: 0.0189137 0.0295462 0.179614 1.00000e-08 1.00000e-08 0.228045 0.575927

Final Estimate of the Negative LLH:
 LLH:  -4002.762    norm LLH:  -1.880114
          mu        omega       alpha1       alpha2       alpha3        beta1        beta2
8.230482e-04 5.594999e-05 1.796141e-01 1.000000e-08 1.000000e-08 2.280453e-01 5.759269e-01

R-optimhess Difference Approximated Hessian Matrix:
                 mu         omega        alpha1       alpha2        alpha3         beta1         beta2
mu     -2182126.031 -2.764658e+04      5697.532     10751.20      2077.975      5774.317      4382.364
omega    -27646.576 -3.805605e+10 -18290900.211 -19864018.34 -19905448.251 -33386358.041 -33850909.075
alpha1     5697.532 -1.829090e+07    -17739.842    -18719.72    -18241.635    -24050.027    -24208.838
alpha2    10751.196 -1.986402e+07    -18719.715    -22869.06    -19850.128    -27111.181    -26800.342
alpha3     2077.975 -1.990545e+07    -18241.635    -19850.13    -20666.344    -26534.822    -26961.467
beta1      5774.317 -3.338636e+07    -24050.027    -27111.18    -26534.822    -38727.338    -39034.874
beta2      4382.364 -3.385091e+07    -24208.838    -26800.34    -26961.467    -39034.874    -39648.501
attr(,"time")
Time difference of 0.111104 secs

--- END OF TRACE ---


Time to Estimate Parameters:
 Time difference of 3.741881 secs
```

Hide

```
summary(m.32_2)
```

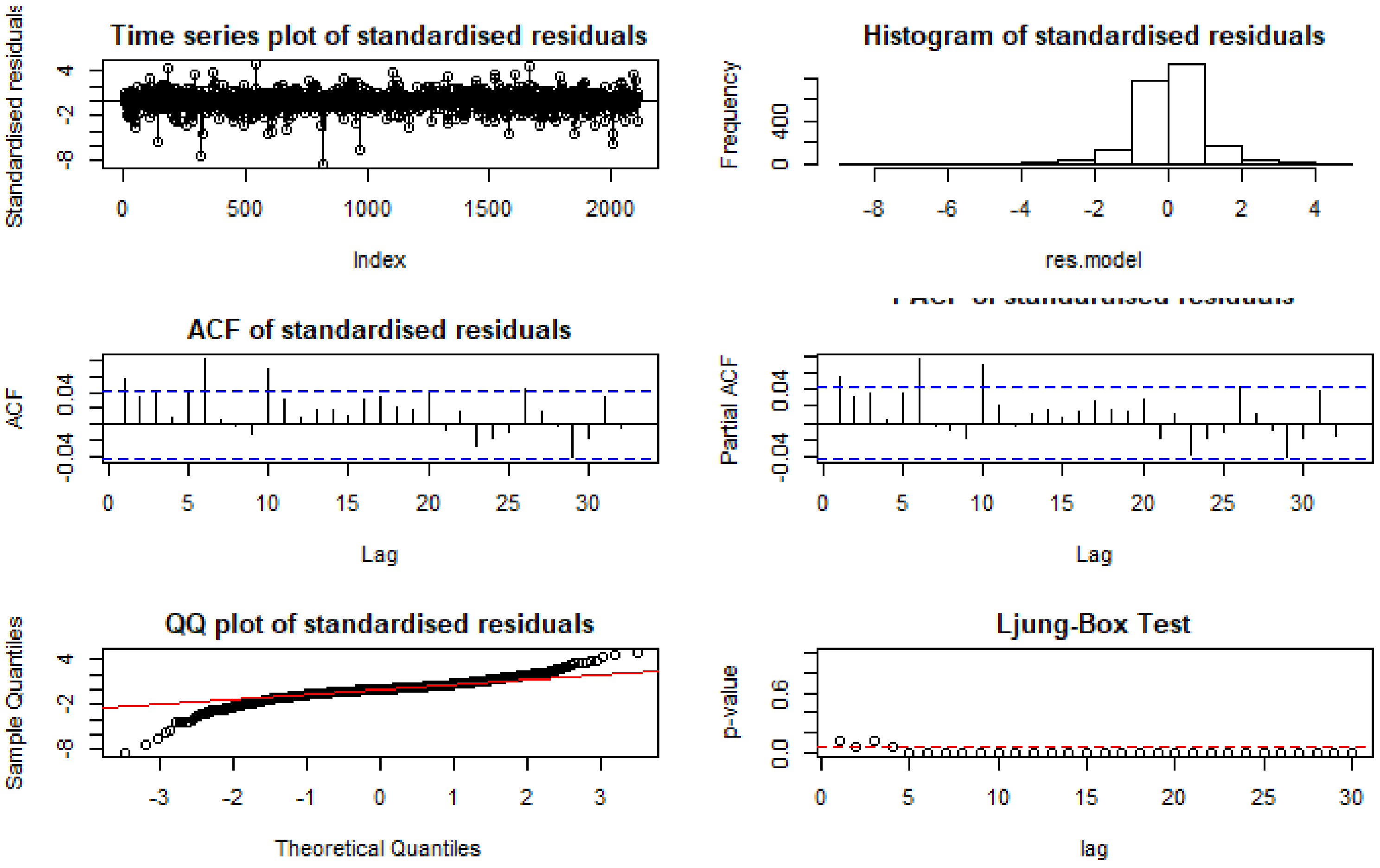
```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(3, 2), data = Bitcoin_price_g)

Mean and Variance Equation:
 data ~ garch(3, 2)
<environment: 0x000000002a76b600>
 [data = Bitcoin_price_g]

Conditional Distribution:
 norm

Coefficient(s):
        mu       omega      alpha1      alpha2      alpha3       beta1       beta2
0.00082305  0.00005595  0.17961414  0.00000001  0.00000001  0.22804528  0.57592687

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     8.230e-04   6.807e-04    1.209 0.226587
omega  5.595e-05   1.504e-05    3.720 0.000199 ***
alpha1 1.796e-01   2.839e-02    6.327 2.50e-10 ***
alpha2 1.000e-08   2.667e-02    0.000 1.000000
alpha3 1.000e-08   3.474e-02    0.000 1.000000
beta1  2.280e-01   8.500e-02    2.683 0.007299 **
beta2  5.759e-01   7.942e-02    7.252 4.11e-13 ***
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1    1

Log Likelihood:
 4002.762    normalized:  1.880114

Description:
 Sat Jun 08 06:55:35 2019 by user: vamip


Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  4937.99   0
 Shapiro-Wilk Test  R    W      0.910874  0
 Ljung-Box Test     R    Q(10)  39.73046  1.890007e-05
 Ljung-Box Test     R    Q(15)  43.95205  0.0001118809
 Ljung-Box Test     R    Q(20)  53.41482  7.06058e-05
 Ljung-Box Test     R^2  Q(10)  9.687263  0.4683458
 Ljung-Box Test     R^2  Q(15)  12.73511  0.6227498
 Ljung-Box Test     R^2  Q(20)  14.73241  0.791509
 LM Arch Test       R    TR^2   10.11356  0.6059984
```

```
Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-3.753651 -3.735030 -3.753673 -3.746836
```

Hide

```
residual.analysis(m.32,class="GARCH",start=20)
```

```
	Shapiro-Wilk normality test

data:  res.model
W = 0.90396, p-value < 2.2e-16
```



- All the components are not significant with p-value greater than 0.05.
- GARCH(4,2)

Hide

```
m.42 = garch(Bitcoin_price_g,order=c(4,2),trace = FALSE)
summary(m.42)
```

```
Call:
garch(x = Bitcoin_price_g, order = c(4, 2), trace = FALSE)

Model:
GARCH(4,2)

Residuals:
     Min       1Q   Median       3Q      Max 
-8.13287 -0.35965  0.05775  0.50077  5.05541 

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)    
a0 7.485e-05   9.091e-06    8.234 2.22e-16 ***
a1 2.218e-01   2.129e-02   10.417  < 2e-16 ***
a2 1.200e-02   3.410e-02    0.352  0.72485    
b1 3.432e-01   1.192e-01    2.880  0.00398 ** 
b2 6.556e-03   7.499e-02    0.087  0.93033    
b3 1.588e-07   5.278e-02    0.000  1.00000    
b4 3.925e-01   5.274e-02    7.443 9.88e-14 ***
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1    1

Diagnostic Tests:
    Jarque Bera Test

data:  Residuals
X-squared = 4990.7, df = 2, p-value < 2.2e-16


    Box-Ljung test

data:  Squared.Residuals
X-squared = 0.087162, df = 1, p-value = 0.7678
```

Hide

```
m.42_2 = garchFit(formula = ~garch(4.2), data =Bitcoin_price_g)
```

```
Series Initialization:
 ARMA Model:                arma
 Formula Mean:              ~ arma(0, 0)
 GARCH Model:               garch
 Formula Variance:          ~ garch(4.2)
 ARMA Order:                0 0
 Max ARMA Order:            0
 GARCH Order:               4.2 0
 Max GARCH Order:           4.2
 Maximum Order:             4.2
 Conditional Dist:          norm
 h.start:                   5.2
 llh.start:                 1
 Length of Series:          2129
 Recursion Init:            mci
 Series Scale:              0.04351601

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
                     U           V     params includes
    mu     -0.36117116   0.3611712 0.03611712     TRUE
    omega   0.00000100 100.0000000 0.10000000     TRUE
    alpha1  0.00000001   1.0000000 0.02380952     TRUE
    alpha2  0.00000001   1.0000000 0.02380952     TRUE
    alpha3  0.00000001   1.0000000 0.02380952     TRUE
    alpha4  0.00000001   1.0000000 0.02380952     TRUE
    gamma1 -0.99999999   1.0000000 0.10000000    FALSE
    gamma2 -0.99999999   1.0000000 0.10000000    FALSE
    gamma3 -0.99999999   1.0000000 0.10000000    FALSE
    gamma4 -0.99999999   1.0000000 0.10000000    FALSE
    delta   0.00000000   2.0000000 2.00000000    FALSE
    skew    0.10000000  10.0000000 1.00000000    FALSE
    shape   1.00000000  10.0000000 4.00000000    FALSE
 Index List of Parameters to be Optimized:
    mu  omega alpha1 alpha2 alpha3 alpha4
     1      2      3      4      5      6
 Persistence:                  0.0952381


--- START OF TRACE ---
Selected Algorithm: nlminb

R coded nlminb Solver:

  0:     5383.3504: 0.0361171 0.100000 0.0238095 0.0238095 0.0238095 0.0238095
  1:     3008.7502: 0.0361041 0.757945 0.477483 0.389952 0.383164 0.337008
  2:     2946.4996: 0.0359208 0.451061 0.825023 0.479465 0.492680 0.214511
  3:     2839.4626: -0.0349419 0.270158 0.615423 0.210992 0.502627 0.174236
```

```
  4:     2829.7527: -0.0346156 0.265348 0.460279 0.0661991 0.409537 0.0331423
  5:     2808.7009: -0.0424758 0.380497 0.431686 0.136531 0.392196 0.116953
  6:     2799.0117: -0.0518601 0.353588 0.384378 0.121165 0.339939 0.100426
  7:     2796.5101: -0.0427729 0.340549 0.326422 0.116555 0.274022 0.0502969
  8:     2791.1938: -0.0440429 0.391295 0.313472 0.128412 0.221007 0.101320
  9:     2789.7382: -0.0407057 0.382298 0.292262 0.128457 0.225906 0.0803148
 10:     2788.6428: -0.0337234 0.398122 0.292593 0.140380 0.244929 0.0830929
 11:     2787.6320: -0.0267051 0.382225 0.292379 0.132505 0.244540 0.0753390
 12:     2786.9917: -0.0126371 0.395494 0.304632 0.132407 0.236241 0.0960513
 13:     2786.0234: -0.0150149 0.388299 0.273913 0.139582 0.212970 0.0826704
 14:     2785.9207: -0.0142392 0.407441 0.266655 0.155997 0.228752 0.0793687
 15:     2785.5133: -0.0130493 0.393619 0.262777 0.150925 0.228971 0.0697589
 16:     2785.3563: -0.0117372 0.398197 0.262001 0.149366 0.230261 0.0777733
 17:     2785.2059: -0.00903697 0.394018 0.263747 0.149602 0.229749 0.0724232
 18:     2785.1116: -0.00647593 0.400097 0.267130 0.156077 0.224721 0.0749302
 19:     2785.0056: -0.00698231 0.396738 0.256161 0.150998 0.216953 0.0772809
 20:     2784.9583: -0.00636569 0.401212 0.254111 0.151299 0.227113 0.0781794
 21:     2784.9192: -0.00567283 0.396175 0.253436 0.149411 0.229597 0.0741485
 22:     2784.8700: -0.00494791 0.398693 0.255173 0.150685 0.227847 0.0758557
 23:     2784.8157: -0.00351735 0.397246 0.254046 0.151011 0.217507 0.0760491
 24:     2784.7322: -0.00205947 0.398580 0.255467 0.152607 0.224105 0.0758428
 25:     2784.7320: 0.000846494 0.394205 0.242439 0.150053 0.230392 0.0729226
 26:     2784.5922: 0.00228580 0.399192 0.250244 0.151843 0.231366 0.0740330
 27:     2784.5757: 0.00360697 0.396429 0.249488 0.159492 0.214811 0.0746749
 28:     2784.5163: 0.00504448 0.401112 0.253920 0.153058 0.216393 0.0777883
 29:     2784.5000: 0.00560078 0.400820 0.252103 0.155454 0.218583 0.0722257
 30:     2784.4948: 0.00562573 0.402295 0.250185 0.156741 0.219932 0.0761747
 31:     2784.4850: 0.00577157 0.399940 0.249370 0.155354 0.219549 0.0752715
 32:     2784.4812: 0.00594924 0.400533 0.249932 0.155354 0.220065 0.0757900
 33:     2784.4742: 0.00669686 0.399655 0.250919 0.154308 0.220484 0.0753525
 34:     2784.4637: 0.00727952 0.401816 0.247227 0.155101 0.218612 0.0744887
 35:     2784.4626: 0.00728687 0.402360 0.247973 0.155960 0.218885 0.0754238
 36:     2784.4609: 0.00733610 0.401587 0.247889 0.155962 0.218817 0.0750866
 37:     2784.4558: 0.00798379 0.400835 0.247518 0.157573 0.220406 0.0751174
 38:     2784.4439: 0.00939913 0.401777 0.249053 0.157431 0.217820 0.0744819
 39:     2784.4432: 0.00970905 0.401111 0.248068 0.156244 0.217657 0.0758352
 40:     2784.4412: 0.0100273 0.401656 0.248323 0.156686 0.217847 0.0755564
 41:     2784.4401: 0.0103448 0.401608 0.248253 0.157074 0.217892 0.0746274
 42:     2784.4394: 0.0109895 0.401646 0.248405 0.157323 0.217638 0.0748426
 43:     2784.4394: 0.0109958 0.401693 0.248380 0.157354 0.217695 0.0747918
 44:     2784.4394: 0.0109949 0.401690 0.248383 0.157356 0.217690 0.0747933

Final Estimate of the Negative LLH:
 LLH:  -3889.18    norm LLH:  -1.826764 
          mu        omega       alpha1       alpha2       alpha3       alpha4 
0.0004784542 0.0007606582 0.2483827596 0.1573561675 0.2176901289 0.0747933058 

R-optimhess Difference Approximated Hessian Matrix:
                  mu         omega       alpha1       alpha2       alpha3       alpha4
mu     -1741215.0135     1125214.4     652.6678    6182.3936   -1554.2891   -1114.8643
omega   1125214.4186 -744473937.1 -295810.0167 -368284.1670 -428690.6956 -515380.3661
alpha1      652.6678    -295810.0    -999.7471    -268.7737    -166.5061    -291.4361
```

```
alpha2     6182.3936    -368284.2    -268.7737   -1173.6651    -339.7147    -554.3930
alpha3    -1554.2891    -428690.7    -166.5061    -339.7147    -712.6106    -359.2000
alpha4    -1114.8643    -515380.4    -291.4361    -554.3930    -359.2000   -1938.3817
attr(,"time")
Time difference of 0.07687998 secs

--- END OF TRACE ---


Time to Estimate Parameters:
 Time difference of 4.434766 secs
```

Hide

```
summary(m.42_2)
```

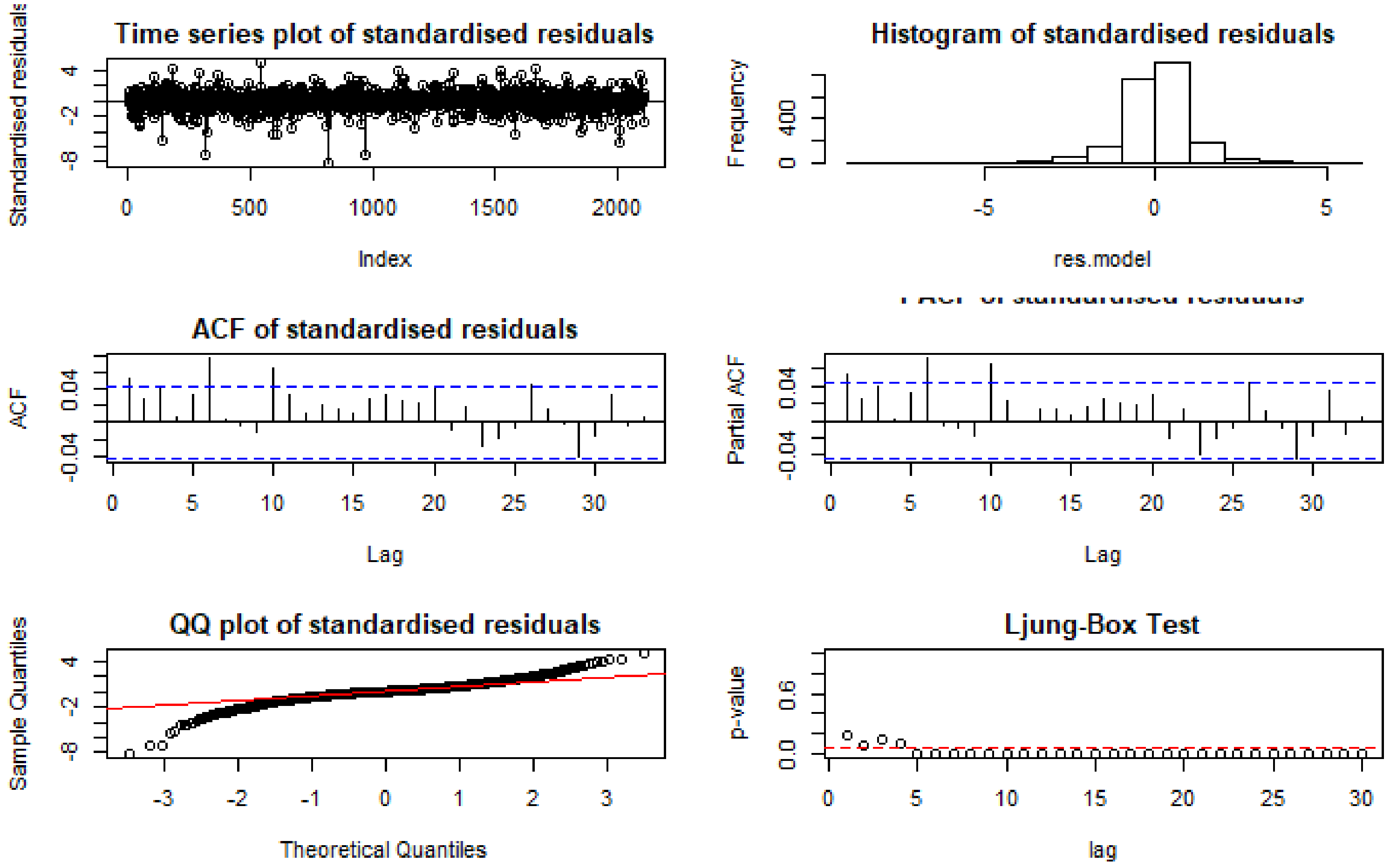
```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(4.2), data = Bitcoin_price_g)

Mean and Variance Equation:
 data ~ garch(4.2)
<environment: 0x00000000275a36b8>
 [data = Bitcoin_price_g]

Conditional Distribution:
 norm

Coefficient(s):
        mu       omega      alpha1      alpha2      alpha3      alpha4
0.00047845  0.00076066  0.24838276  0.15735617  0.21769013  0.07479331

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     4.785e-04   7.714e-04    0.620  0.53512
omega  7.607e-04   4.994e-05   15.232  < 2e-16 ***
alpha1 2.484e-01   3.401e-02    7.302 2.83e-13 ***
alpha2 1.574e-01   3.394e-02    4.637 3.54e-06 ***
alpha3 2.177e-01   4.748e-02    4.585 4.54e-06 ***
alpha4 7.479e-02   2.599e-02    2.877  0.00401 **
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1     1

Log Likelihood:
 3889.18    normalized:  1.826764

Description:
 Sat Jun 08 06:55:41 2019 by user: vamip


Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  5034.913  0
 Shapiro-Wilk Test  R    W      0.8985237 0
 Ljung-Box Test     R    Q(10)  35.29971  0.0001109792
 Ljung-Box Test     R    Q(15)  39.73223  0.0004976711
 Ljung-Box Test     R    Q(20)  44.75164  0.001192302
 Ljung-Box Test     R^2  Q(10)  25.00412  0.0053377
 Ljung-Box Test     R^2  Q(15)  40.99751  0.0003200729
 Ljung-Box Test     R^2  Q(20)  50.24743  0.0002041125
 LM Arch Test       R    TR^2   25.44339  0.01285604
```

```
Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-3.647891 -3.631930 -3.647907 -3.642049
```

Hide

```
residual.analysis(m.42,class="GARCH",start=20)
```

```

    Shapiro-Wilk normality test

data:  res.model
W = 0.91299, p-value < 2.2e-16
```



- Most of the components are significant for GARCh (4,2)

Hide

```
sort.score(AIC(m.21,m.31,m.32,m.42), score = "aic")
```

	df	AIC
	<dbl>	<dbl>
m.42	7	-8003.902
m.31	5	-7993.943

	df	AIC
	<dbl>	<dbl>
m.21	4	-7990.400
m.32	6	-7979.514

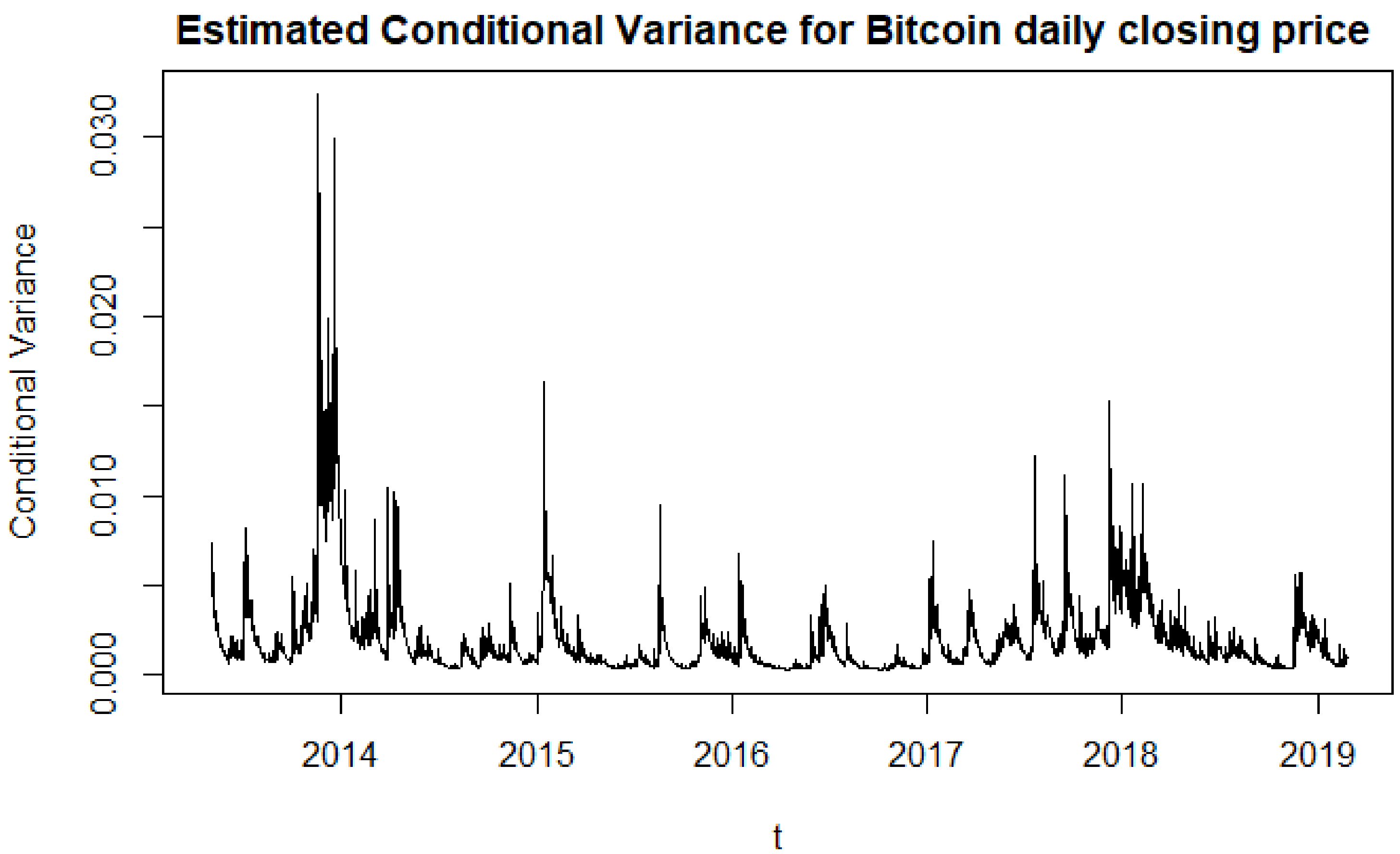
4 rows

- Since, all the tentative GARCH models are showing similar analysis for the residuals. AIC score will be a better way in order to find the best GARCh model among the 4 models.
- With the lowest AIC score, GARCH(4,2) came out to be the best model among the 4 tentative models. Also, most of the components were significant for this model.
- So, GARCH (4,2) will be used in order to make predctions for estimated conditional variances.

9.4 PREDICTION OF ESTIMATED CONDITIONAL VARIANCE

Hide

```
par(mfrow=c(1,1))
plot((fitted(m.42)[,1])^2,type='l',ylab='Conditional Variance',xlab='t',main="Estimated Conditional Variance for Bitcoin dai
ly closing price ")
```

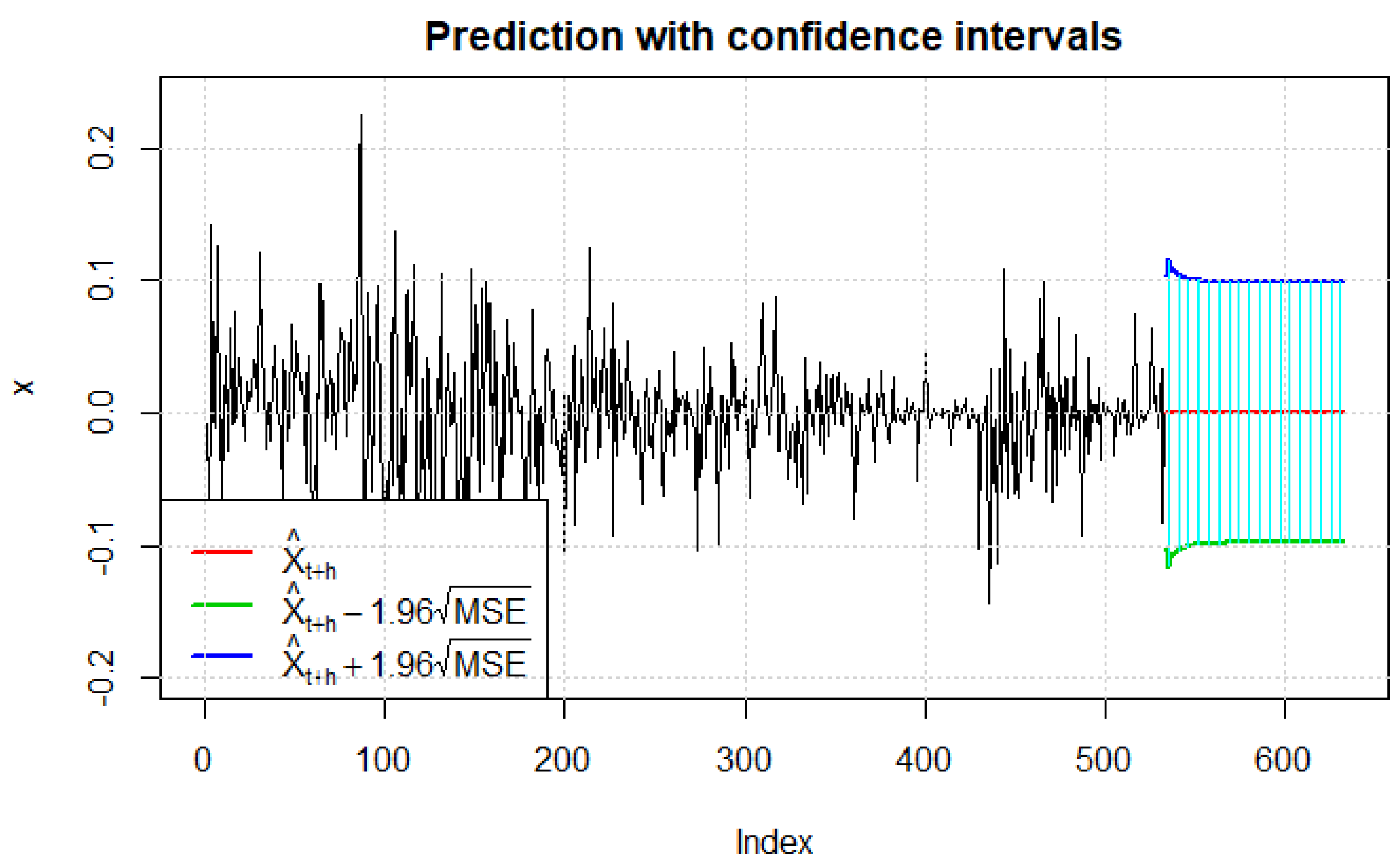


Hide

```
fGarch::predict(m.42_2,n.ahead=100,trace=FALSE,plot=TRUE)
```

meanForecast <dbl>	meanError <dbl>	standardDeviation <dbl>	lowerInterval <dbl>	upperInterval <dbl>
0.0004784542	0.05227231	0.05227231	-0.10197339	0.10293030
0.0004784542	0.05293029	0.05293029	-0.10326300	0.10421991
0.0004784542	0.05921396	0.05921396	-0.11557877	0.11653568
0.0004784542	0.05652885	0.05652885	-0.11031606	0.11127297
0.0004784542	0.05404030	0.05404030	-0.10543858	0.10639549
0.0004784542	0.05442136	0.05442136	-0.10618545	0.10714236
0.0004784542	0.05397873	0.05397873	-0.10531792	0.10627483
0.0004784542	0.05315211	0.05315211	-0.10369776	0.10465467
0.0004784542	0.05276379	0.05276379	-0.10293667	0.10389358
0.0004784542	0.05246440	0.05246440	-0.10234988	0.10330679

1-10 of 100 rows Previous 1 2 3 4 5 6 ... 10 Next



- Change in conditional variance is observed at the beginning of the series for the year 2014 and then the conditional variance settles down.

10. FITTING ARMA INTO GARCH FOR FINAL MODEL SELECTION

10.1 Estimation of parameters

10.1.1 ARMA(1,1)+ GARCH(2,1)

Hide

```
bit_model1<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
                   mean.model = list(armaOrder = c(1, 1), include.mean = FALSE),
                   distribution.model = "norm")
m.11_21<-ugarchfit(spec = bit_model1, data = Bitcoin_price_g, out.sample = 100)
m.11_21
```

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

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(2,1)
Mean Model  : ARFIMA(1,0,1)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
ar1     0.765579    0.407066  1.880725 0.060009
ma1    -0.733582    0.429832 -1.706674 0.087883
omega   0.000038    0.000011  3.329977 0.000869
alpha1  0.121392    0.020349  5.965541 0.000000
alpha2  0.000000    0.032629  0.000001 0.999999
beta1   0.867839    0.025166 34.485217 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
ar1     0.765579    0.836013  0.91575 0.359797
ma1    -0.733582    0.879737 -0.83386 0.404357
omega   0.000038    0.000041  0.91143 0.362068
alpha1  0.121392    0.030921  3.92587 0.000086
alpha2  0.000000    0.092478  0.00000 1.000000
beta1   0.867839    0.093249  9.30672 0.000000

LogLikelihood : 3816.466

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

Akaike       -3.7560
Bayes        -3.7394
Shibata      -3.7560
Hannan-Quinn -3.7499

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic p-value
Lag[1]                      2.700 0.10036
Lag[2*(p+q)+(p+q)-1][5]     2.981 0.48407
Lag[4*(p+q)+(p+q)-1][9]     9.732 0.01197
d.o.f=2
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                         statistic p-value
```

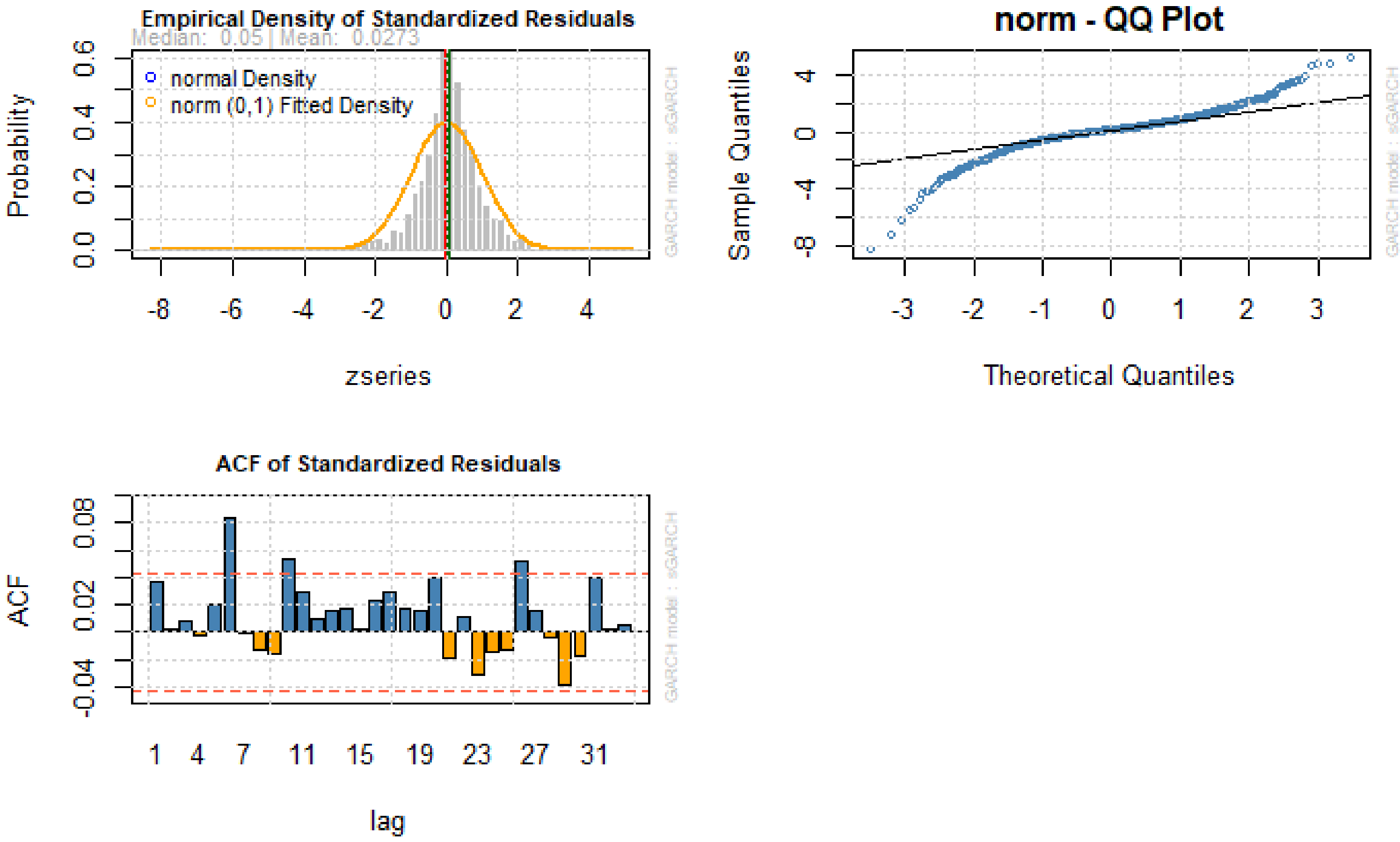
```
Lag[1]                       1.655  0.1982
Lag[2*(p+q)+(p+q)-1][8]      2.930  0.6984
Lag[4*(p+q)+(p+q)-1][14]     4.970  0.7732
d.o.f=3

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[4]     1.435 0.500 2.000  0.2310
ARCH Lag[6]     1.523 1.461 1.711  0.6047
ARCH Lag[8]     1.763 2.368 1.583  0.7891

Nyblom stability test
------------------------------------
Joint Statistic:  1.2647
Individual Statistics:
ar1    0.02841
ma1    0.03228
omega  0.15363
alpha1 0.09204
alpha2 0.08511
beta1  0.12240

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         1.49 1.68 2.12
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias           1.3117 0.1898
Negative Sign Bias  0.3114 0.7555
Positive Sign Bias  0.4741 0.6355
Joint Effect        2.8624 0.4133


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     347.2    3.952e-62
2    30     380.6    6.328e-63
3    40     397.7    5.859e-61
4    50     407.8    4.606e-58


Elapsed time : 1.163033
```

Hide

```
par(mfrow=c(2,2))
plot(m.11_21, which=8)
plot(m.11_21, which=9)
```

Hide

```
plot(m.11_21, which=10)
```



- The AR and MA components are not significant for this model.
- With significant lags in ACF, it can be said that the residuals does not possess white noise. In the QQ-plot, the deviation from the normality can be observed.
- Hence, the order of AR is reduced by 1, to check the improvement in the residuals assumptions.

10.1.2 ARMA(0,1)+GARCH(2,1)

Hide

```
bit_model2<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
                    mean.model = list(armaOrder = c(0, 1), include.mean = FALSE),
                    distribution.model = "norm")
m.01_21<-ugarchfit(spec = bit_model2, data = Bitcoin_price_g, out.sample = 100)
m.01_21
```

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

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(2,1)
Mean Model  : ARFIMA(0,0,1)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
ma1     0.041116    0.025333  1.623003 0.104589
omega   0.000038    0.000011  3.308212 0.000939
alpha1  0.121738    0.020504  5.937270 0.000000
alpha2  0.000000    0.032616  0.000001 0.999999
beta1   0.867743    0.025088 34.587877 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
ma1     0.041116    0.024492  1.67873 0.093206
omega   0.000038    0.000041  0.91772 0.358766
alpha1  0.121738    0.031185  3.90370 0.000095
alpha2  0.000000    0.089108  0.00000 1.000000
beta1   0.867743    0.090100  9.63088 0.000000

LogLikelihood : 3815.986

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

Akaike       -3.7565
Bayes        -3.7427
Shibata      -3.7565
Hannan-Quinn -3.7514

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic p-value
Lag[1]                      2.075 0.14973
Lag[2*(p+q)+(p+q)-1][2]     2.707 0.06199
Lag[4*(p+q)+(p+q)-1][5]     4.319 0.17909
d.o.f=1
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                         statistic p-value
Lag[1]                       1.575  0.2094
Lag[2*(p+q)+(p+q)-1][8]      2.864  0.7104
```

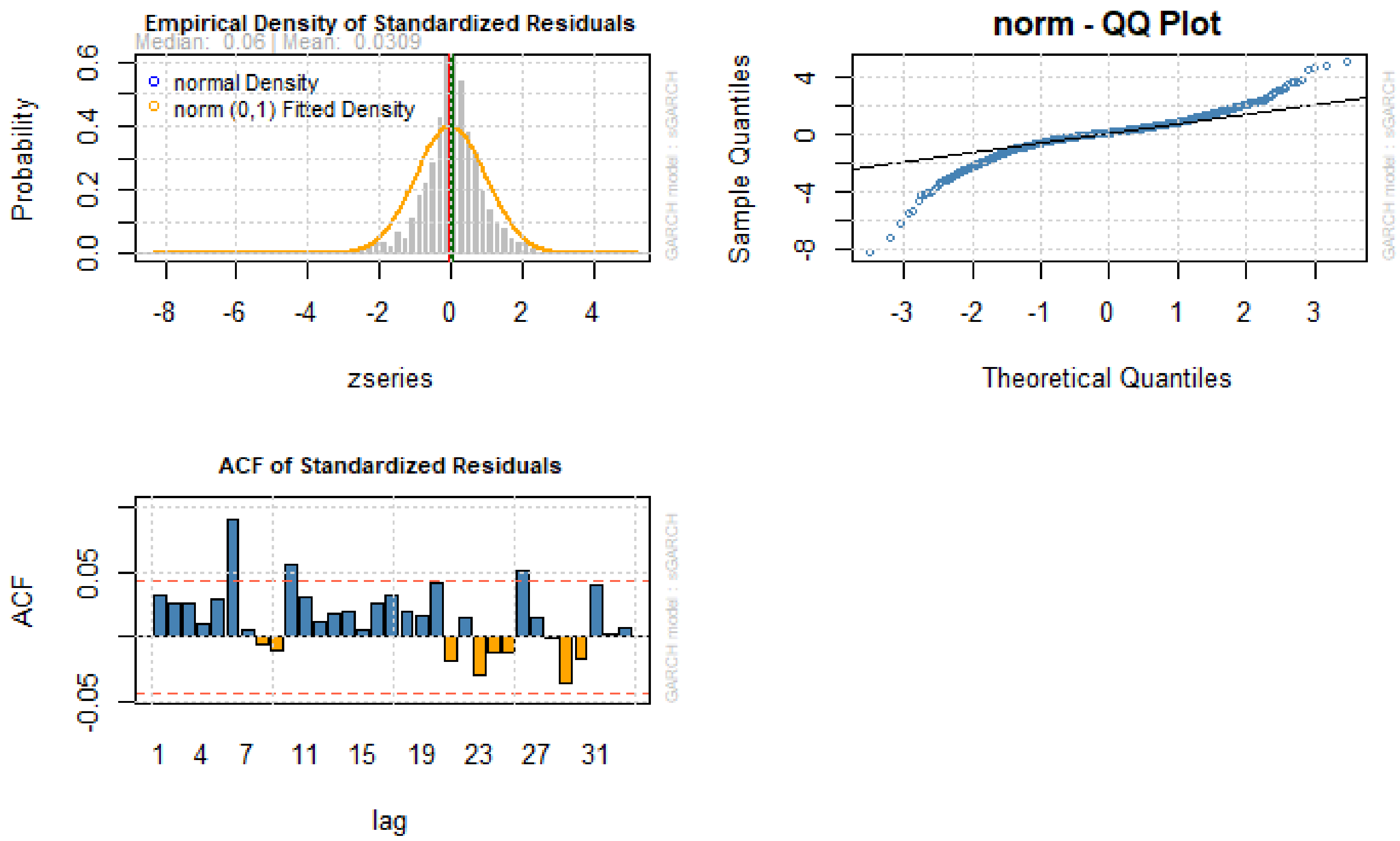
```
Lag[4*(p+q)+(p+q)-1][14]     4.964  0.7740
d.o.f=3

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[4]     1.449 0.500 2.000  0.2287
ARCH Lag[6]     1.515 1.461 1.711  0.6067
ARCH Lag[8]     1.765 2.368 1.583  0.7886

Nyblom stability test
------------------------------------
Joint Statistic:  1.0847
Individual Statistics:
ma1    0.02080
omega  0.15596
alpha1 0.09079
alpha2 0.08340
beta1  0.12150

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         1.28 1.47 1.88
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias           1.0870 0.2772
Negative Sign Bias  0.3942 0.6935
Positive Sign Bias  0.3956 0.6924
Joint Effect        2.2598 0.5203


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     336.9    5.163e-60
2    30     366.0    5.486e-60
3    40     386.2    1.041e-58
4    50     398.5    2.891e-56


Elapsed time : 0.8551331
```

Hide

```
par(mfrow=c(2,2))
plot(m.01_21, which=8)
plot(m.01_21, which=9)
```

Hide

```
plot(m.01_21, which=10)
```



- By reducing the order of AR by 1, the residual analysis still came out to be the same as earlier, i.e., the MA component is not significant. With significant lags in ACF, it can be said that the residuals does not possess white noise. In the QQ-plot, the deviation from the normality can be observed.
- This time the order of MA is increased by 1 and making the order of AR back to 1.

10.1.3 ARMA(1,2)+GARCH(2,1)

Hide

```
bit_model3<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
                  mean.model = list(armaOrder = c(1, 2), include.mean = FALSE),
                  distribution.model = "norm")
m.12_21<-ugarchfit(spec = bit_model3, data = Bitcoin_price_g, out.sample = 100)
m.12_21
```

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

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(2,1)
Mean Model  : ARFIMA(1,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error    t value Pr(>|t|)
ar1     0.925364    0.040143  23.051604 0.000000
ma1    -0.884421    0.026736 -33.080077 0.000000
ma2    -0.024886    0.004749  -5.240092 0.000000
omega   0.000038    0.000011   3.381088 0.000722
alpha1  0.121268    0.020252   5.988039 0.000000
alpha2  0.000000    0.031882   0.000004 0.999997
beta1   0.868142    0.024357  35.642165 0.000000

Robust Standard Errors:
        Estimate  Std. Error    t value Pr(>|t|)
ar1     0.925364    0.025736  35.956235 0.000000
ma1    -0.884421    0.025893 -34.157294 0.000000
ma2    -0.024886    0.005319  -4.678912 0.000003
omega   0.000038    0.000039   0.961222 0.336441
alpha1  0.121268    0.030697   3.950438 0.000078
alpha2  0.000000    0.084623   0.000001 0.999999
beta1   0.868142    0.085127  10.198237 0.000000

LogLikelihood : 3816.75

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

Akaike       -3.7553
Bayes        -3.7359
Shibata      -3.7553
Hannan-Quinn -3.7482

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                         statistic   p-value
Lag[1]                       1.718 1.900e-01
Lag[2*(p+q)+(p+q)-1][8]      7.710 5.681e-06
Lag[4*(p+q)+(p+q)-1][14]    14.233 3.862e-03
d.o.f=3
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
```

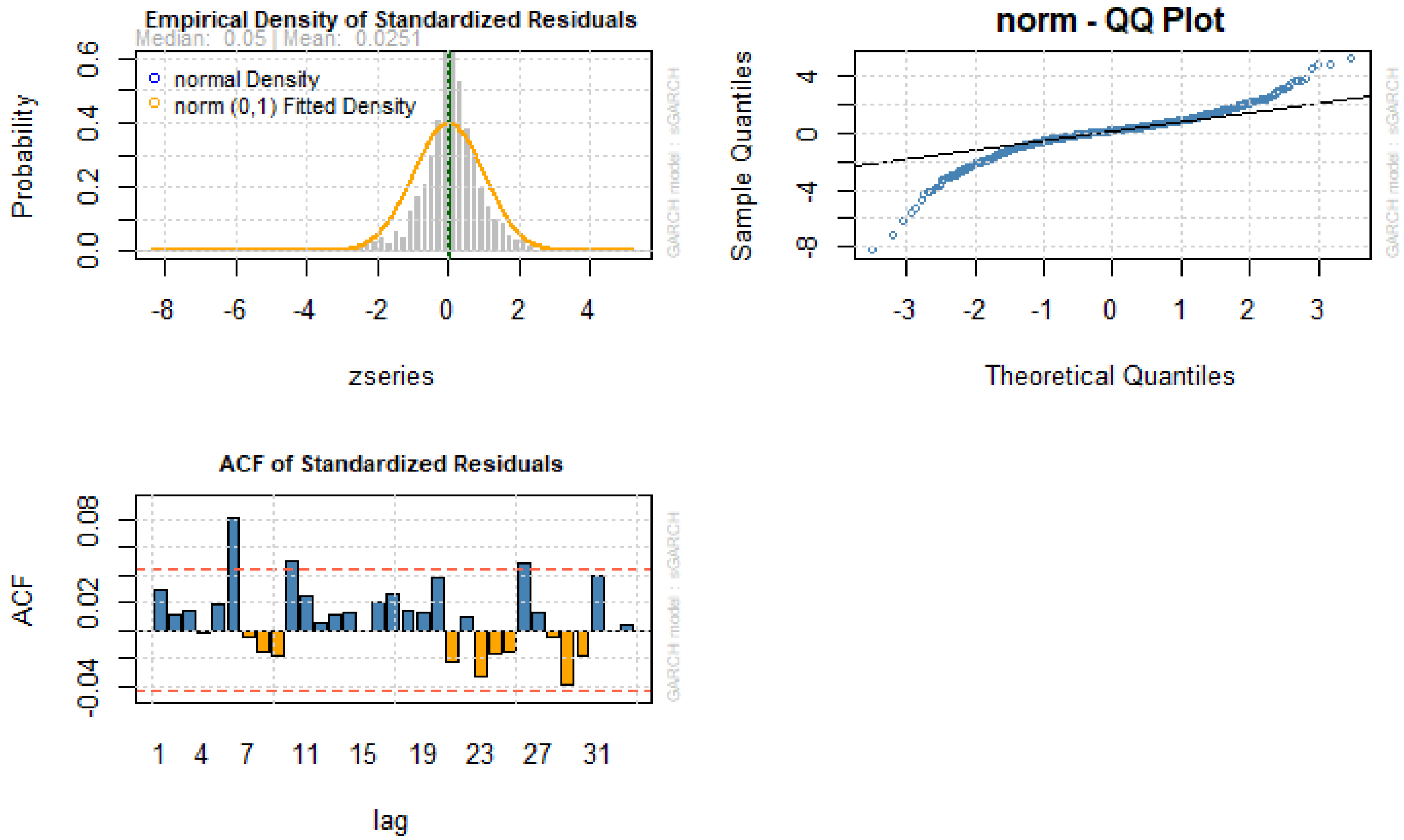
```
------------------------------------
                         statistic p-value
Lag[1]                       1.758  0.1849
Lag[2*(p+q)+(p+q)-1][8]      3.060  0.6746
Lag[4*(p+q)+(p+q)-1][14]     5.128  0.7535
d.o.f=3

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[4]     1.460 0.500 2.000  0.2269
ARCH Lag[6]     1.549 1.461 1.711  0.5981
ARCH Lag[8]     1.799 2.368 1.583  0.7818

Nyblom stability test
------------------------------------
Joint Statistic:  1.2429
Individual Statistics:
ar1    0.02008
ma1    0.01856
ma2    0.02344
omega  0.15353
alpha1 0.09221
alpha2 0.08409
beta1  0.12180

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         1.69 1.9 2.35
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias           1.3235 0.1858
Negative Sign Bias  0.3108 0.7560
Positive Sign Bias  0.4879 0.6257
Joint Effect        2.8793 0.4106


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     354.9    9.896e-64
2    30     382.1    3.140e-63
3    40     405.5    1.660e-62
4    50     420.3    1.866e-60


Elapsed time : 0.351665
```

Hide

```
par(mfrow=c(2,2))
plot(m.12_21, which=8)
plot(m.12_21, which=9)
```

Hide

```
plot(m.12_21, which=10)
```



- By increasing the order of MA by 1, it can be noticed that, all the components of AR and MA are now significant. Therefore, ARMA(1,2) is better.
- Now using ARMA(1,2) with the other tentative GARCH models.

10.1.4 ARMA(1,2)+GARCH(3,1)

Hide

```
bit_model4<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(3, 1)),
                       mean.model = list(armaOrder = c(1, 2), include.mean = FALSE),
                       distribution.model = "norm")
m.12_31<-ugarchfit(spec = bit_model4, data = Bitcoin_price_g, out.sample = 100)
m.12_31
```

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

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(3,1)
Mean Model  : ARFIMA(1,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error    t value Pr(>|t|)
ar1     0.925089    0.041439  22.323901 0.000000
ma1    -0.885892    0.027850 -31.809122 0.000000
ma2    -0.023278    0.004887  -4.762994 0.000002
omega   0.000038    0.000013   2.815038 0.004877
alpha1  0.120913    0.020398   5.927828 0.000000
alpha2  0.000000    0.037524   0.000001 0.999999
alpha3  0.000000    0.036893   0.000001 0.999999
beta1   0.868256    0.031476  27.584764 0.000000

Robust Standard Errors:
        Estimate  Std. Error   t value Pr(>|t|)
ar1     0.925089    0.026048  35.51413 0.000000
ma1    -0.885892    0.026094 -33.95000 0.000000
ma2    -0.023278    0.005740  -4.05565 0.000050
omega   0.000038    0.000052   0.72679 0.467355
alpha1  0.120913    0.030765   3.93026 0.000085
alpha2  0.000000    0.070967   0.00000 1.000000
alpha3  0.000000    0.086280   0.00000 1.000000
beta1   0.868256    0.120091   7.22997 0.000000

LogLikelihood : 3816.747

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

Akaike       -3.7543
Bayes        -3.7322
Shibata      -3.7543
Hannan-Quinn -3.7462

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic   p-value
Lag[1]                      1.881 1.702e-01
Lag[2*(p+q)+(p+q)-1][8]     7.875 2.358e-06
Lag[4*(p+q)+(p+q)-1][14]   14.398 3.308e-03
d.o.f=3
H0 : No serial correlation
```

```
Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                         statistic p-value
Lag[1]                       1.763  0.1843
Lag[2*(p+q)+(p+q)-1][11]     4.137  0.7125
Lag[4*(p+q)+(p+q)-1][19]     6.687  0.8138
d.o.f=4

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[5]   0.04693 0.500 2.000  0.8285
ARCH Lag[7]   0.22326 1.473 1.746  0.9656
ARCH Lag[9]   1.18604 2.402 1.619  0.9043

Nyblom stability test
------------------------------------
Joint Statistic:  2.1208
Individual Statistics:
ar1    0.02177
ma1    0.02039
ma2    0.02356
omega  0.15241
alpha1 0.08843
alpha2 0.07742
alpha3 0.06235
beta1  0.11981

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         1.89 2.11 2.59
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias           1.3223 0.1862
Negative Sign Bias  0.3168 0.7514
Positive Sign Bias  0.4921 0.6227
Joint Effect        2.8846 0.4098


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     356.0    5.850e-64
2    30     383.1    1.914e-63
3    40     408.2    4.905e-63
4    50     423.9    3.769e-61


Elapsed time : 0.46245
```

- All the components of AR and MA are significant with p-value less than 0.05.

10.1.5 ARMA(1,2)+GARCH(3,2)

Hide

```
bit_model5<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(3, 2)),
                   mean.model = list(armaOrder = c(1, 2), include.mean = FALSE),
                   distribution.model = "norm")
m.12_32<-ugarchfit(spec = bit_model5, data = Bitcoin_price_g, out.sample = 100)
m.12_32
```

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

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(3,2)
Mean Model  : ARFIMA(1,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
ar1     0.920028    0.161100  5.710908 0.000000
ma1    -0.881879    0.157549 -5.597489 0.000000
ma2    -0.020734    0.035031 -0.591887 0.553926
omega   0.000051    0.000016  3.270211 0.001075
alpha1  0.173425    0.027531  6.299320 0.000000
alpha2  0.000000    0.028866  0.000003 0.999997
alpha3  0.000000    0.036736  0.000001 0.999999
beta1   0.245223    0.099767  2.457957 0.013973
beta2   0.566302    0.088943  6.367049 0.000000

Robust Standard Errors:
        Estimate  Std. Error   t value Pr(>|t|)
ar1     0.920028    0.211893  4.341948 0.000014
ma1    -0.881879    0.197120 -4.473826 0.000008
ma2    -0.020734    0.043131 -0.480737 0.630704
omega   0.000051    0.000051  1.000352 0.317140
alpha1  0.173425    0.043419  3.994209 0.000065
alpha2  0.000000    0.051477  0.000002 0.999999
alpha3  0.000000    0.081432  0.000000 1.000000
beta1   0.245223    0.107374  2.283826 0.022382
beta2   0.566302    0.086114  6.576213 0.000000

LogLikelihood : 3823.535

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

Akaike       -3.7600
Bayes        -3.7351
Shibata      -3.7601
Hannan-Quinn -3.7509

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                         statistic   p-value
Lag[1]                       1.890 1.692e-01
Lag[2*(p+q)+(p+q)-1][8]      7.699 6.030e-06
Lag[4*(p+q)+(p+q)-1][14]    14.125 4.272e-03
```

```
d.o.f=3
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.2548  0.6137
Lag[2*(p+q)+(p+q)-1][14]    3.1500  0.9469
Lag[4*(p+q)+(p+q)-1][24]    6.0987  0.9662
d.o.f=5

Weighted ARCH LM Tests
------------------------------------
             Statistic Shape Scale P-Value
ARCH Lag[6]    0.01676 0.500 2.000  0.8970
ARCH Lag[8]    0.22034 1.480 1.774  0.9676
ARCH Lag[10]   2.80278 2.424 1.650  0.6185

Nyblom stability test
------------------------------------
Joint Statistic:  1.7957
Individual Statistics:
ar1    0.01868
ma1    0.01817
ma2    0.02414
omega  0.13793
alpha1 0.08563
alpha2 0.10247
alpha3 0.07430
beta1  0.11195
beta2  0.11780

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         2.1 2.32 2.82
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias          1.34721 0.1781
Negative Sign Bias 0.28045 0.7792
Positive Sign Bias 0.05403 0.9569
Joint Effect       2.45394 0.4837


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     352.7    2.832e-63
2    30     383.3    1.738e-63
3    40     404.7    2.419e-62
4    50     411.3    9.777e-59
```

```
Elapsed time : 0.471199
```

- MA(2) is not significant for this model.

10.1.6 ARMA(1,2)+GARCH(4,2)

Hide

```
bit_model6<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(4, 2)),
                   mean.model = list(armaOrder = c(1, 2), include.mean = FALSE),
                   distribution.model = "norm")
m.12_42<-ugarchfit(spec = bit_model6, data = Bitcoin_price_g, out.sample = 100)
m.12_42
```

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

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(4,2)
Mean Model  : ARFIMA(1,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error    t value Pr(>|t|)
ar1     0.930407    0.019232  48.377858 0.000000
ma1    -0.891687    0.009570 -93.173454 0.000000
ma2    -0.023739    0.004931  -4.814027 0.000001
omega   0.000049    0.000023   2.168365 0.030131
alpha1  0.171867    0.027225   6.312856 0.000000
alpha2  0.000000    0.033393   0.000006 0.999995
alpha3  0.000000    0.028424   0.000004 0.999997
alpha4  0.000000    0.045172   0.000000 1.000000
beta1   0.255251    0.173960   1.467292 0.142297
beta2   0.559130    0.123742   4.518510 0.000006

Robust Standard Errors:
        Estimate  Std. Error    t value Pr(>|t|)
ar1     0.930407    0.014049  66.227725 0.000000
ma1    -0.891687    0.009131 -97.657350 0.000000
ma2    -0.023739    0.002246 -10.569382 0.000000
omega   0.000049    0.000099   0.496401 0.619611
alpha1  0.171867    0.048557   3.539470 0.000401
alpha2  0.000000    0.095806   0.000002 0.999998
alpha3  0.000000    0.045965   0.000002 0.999998
alpha4  0.000000    0.171524   0.000000 1.000000
beta1   0.255251    0.561153   0.454868 0.649204
beta2   0.559130    0.293327   1.906164 0.056629

LogLikelihood : 3822.46

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

Akaike       -3.7580
Bayes        -3.7303
Shibata      -3.7580
Hannan-Quinn -3.7478

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic   p-value
Lag[1]                      2.130 1.445e-01
```

```
Lag[2*(p+q)+(p+q)-1][8]      8.081 7.596e-07
Lag[4*(p+q)+(p+q)-1][14]    14.531 2.916e-03
d.o.f=3
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                         statistic p-value
Lag[1]                      0.4225  0.5157
Lag[2*(p+q)+(p+q)-1][17]    4.2416  0.9424
Lag[4*(p+q)+(p+q)-1][29]    7.7828  0.9687
d.o.f=6

Weighted ARCH LM Tests
------------------------------------
             Statistic Shape Scale P-Value
ARCH Lag[7]    0.08835 0.500 2.000  0.7663
ARCH Lag[9]    1.42366 1.485 1.796  0.6572
ARCH Lag[11]   3.56690 2.440 1.677  0.4968

Nyblom stability test
------------------------------------
Joint Statistic:  8.5527
Individual Statistics:
ar1    0.01506
ma1    0.01437
ma2    0.01949
omega  0.13697
alpha1 0.09371
alpha2 0.11335
alpha3 0.09335
alpha4 0.16533
beta1  0.11885
beta2  0.12341

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         2.29 2.54 3.05
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias          1.26548 0.2058
Negative Sign Bias 0.04671 0.9628
Positive Sign Bias 0.04986 0.9602
Joint Effect       2.46786 0.4811


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     346.6    5.091e-62
```

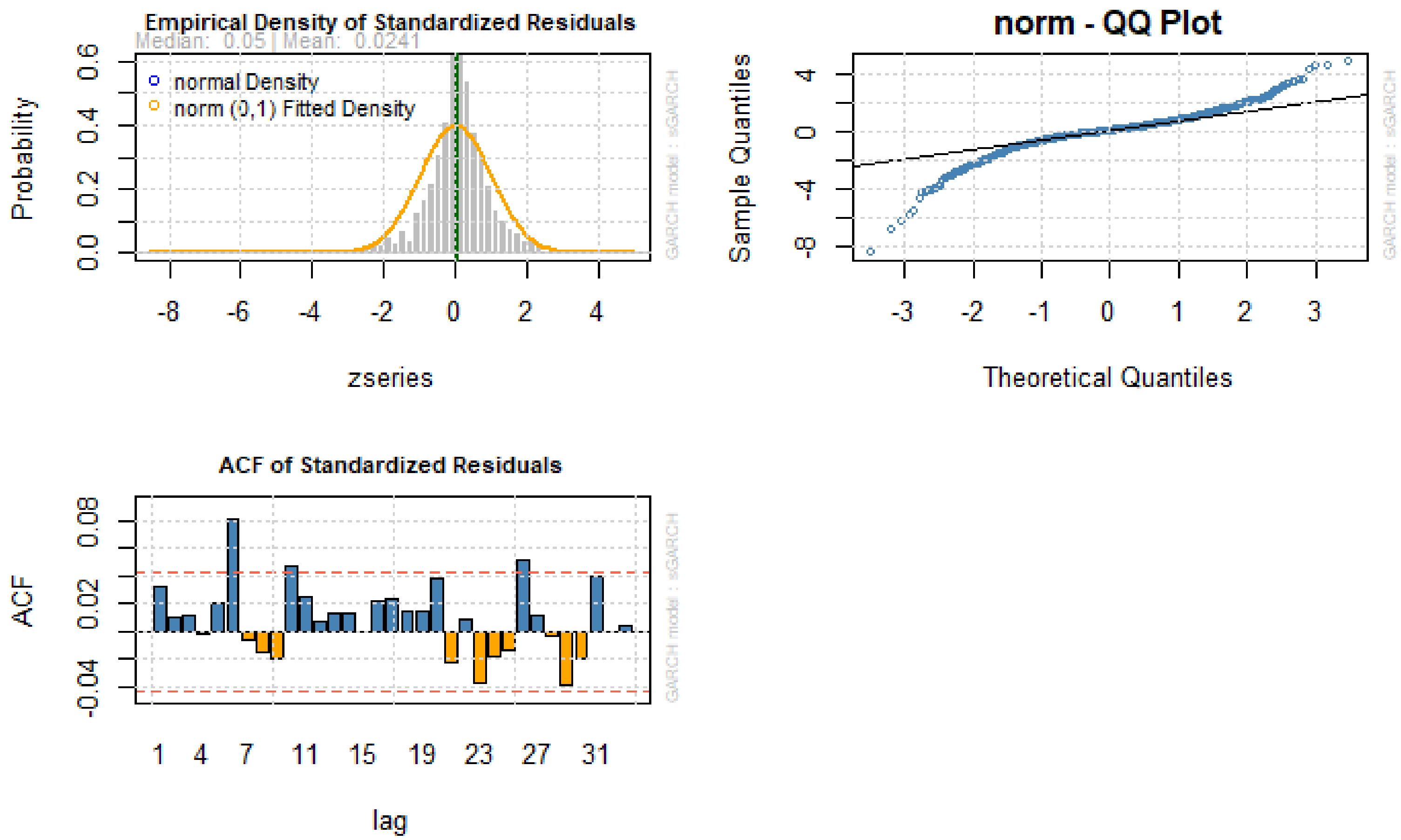
```
2    30      374.0     1.335e-61
3    40      405.4     1.784e-62
4    50      407.8     4.708e-58


Elapsed time : 0.4979849
```

Hide

```
par(mfrow=c(2,2))
plot(m.12_42, which=8)
plot(m.12_42, which=9)
```

Hide

```
plot(m.12_42, which=10)
```



- All the components of AR and MA are significant.
- The residual assumptions did not improve for any of the combinations. Although, ARMA(1,2)+GARCH(2,1), ARMA(1,2)+GARCH(3,1) and ARMA(1,2)+GARCH(4,2) are having all the significant components with p-value less than 0.05. So, other parameters are being compared now.

10.2 SELECTING THE BEST MODEL BASED ON SUMMARY ANALYSIS

- Results from the summary of all ARMA+GARCH models

- Model m.11_21: Akaike: -3.7560, Bayes: -3.7394, Shibata: -3.7560, Hannan-Quinn: -3.7499
- Model m.01_21: Akaike: -3.7565, Bayes: -3.7427, Shibata: -3.7565, Hannan-Quinn: -3.7514
- Model m.12_21: Akaike: -3.7553, Bayes: -3.7359, Shibata: -3.7553, Hannan-Quinn: -3.7482
- Model m.12_31: Akaike: -3.7543, Bayes: -3.7322, Shibata: -3.7543, Hannan-Quinn: -3.7462
- Model m.12_32: Akaike: -3.7600, Bayes: -3.7351, Shibata: -3.7600, Hannan-Quinn: -3.7509
- Model m.12_42: Akaike: -3.7580, Bayes: -3.7303, Shibata: -3.7580, Hannan-Quinn: -3.7478

- It can be seen that ARMA(1,2)+GARCH(3,2) is having the lowest values, but also it’s MA component was not significant. So, the next lowest values are of ARMA(1,2)+GARCH(4,2).

- Hence, the final best model is ARMA(1,2)+GARCH(3,2).

10.3 FORECAST FOR ARMA(1,2)+GARCH(4,2) MODEL

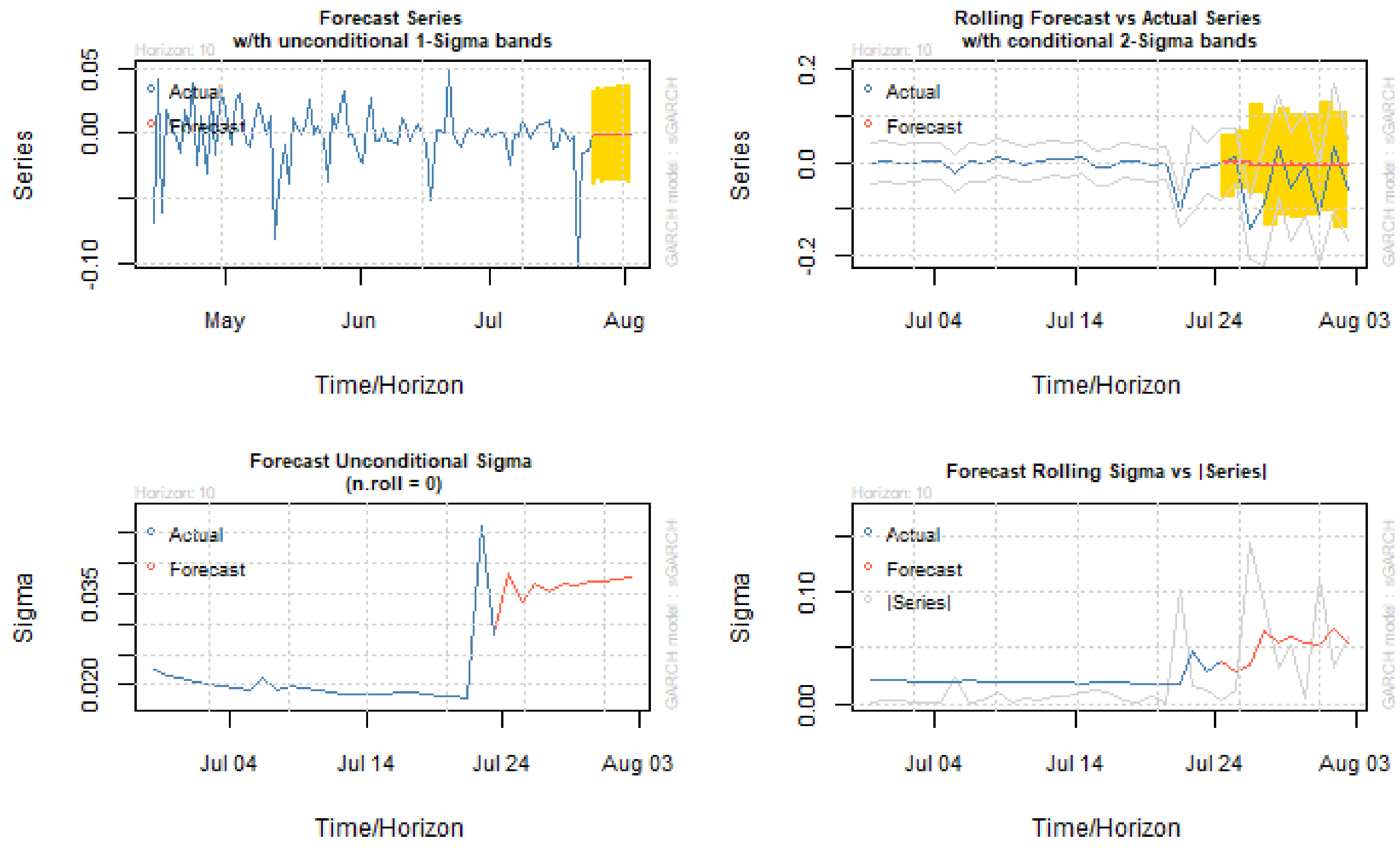
Hide

```
forc.12_42 = ugarchforecast(m.12_42, data = Bitcoin_price_g, n.ahead = 10, n.roll = 10)
forc.12_42
```

```

*------------------------------------*
*       GARCH Model Forecast         *
*------------------------------------*
Model: sGARCH
Horizon: 10
Roll Steps: 10
Out of Sample: 10

0-roll forecast [T0=1975-07-23 10:00:00]:
         Series   Sigma
T+1  -0.0018589 0.03822
T+2  -0.0014665 0.03343
T+3  -0.0013645 0.03665
T+4  -0.0012695 0.03533
T+5  -0.0011812 0.03652
T+6  -0.0010990 0.03628
T+7  -0.0010225 0.03684
T+8  -0.0009513 0.03695
T+9  -0.0008851 0.03730
T+10 -0.0008235 0.03751
```

Hide

```
plot(forc.12_42, which = "all")
```



- From the first plot, it can be inferred that the series might go through a downward (negative) trend FOR THE NEXT 10 DAYS.
- From the rolling forecast, long term prediction can be done. Here, it can be noticed that the conditional variance will increase or in other words will go through an upward trend in the near future. It also shows that the Bitcoin closing price might be fluctuating in the future and also, will be eventually increasing.

11. MASE

- Imported the given data for calculating MASE into R and named it as “Bitcoin_price_mase”.

Hide

```
Bitcoin_price_mase <- read.csv("Bitcoin_Prices_Forecasts.csv")
head(Bitcoin_price_mase)
```

	Date <fctr>	Closing.price <dbl>
1	2019-02-25	3882.70
2	2019-02-26	3854.36
3	2019-02-27	3851.05
4	2019-02-28	3854.79
5	2019-03-01	3859.58

	Date <fctr>	Closing.price <dbl>
6	2019-03-02	3864.42

6 rows

- The Mean Absolute Scaled Error (MASE) is a measure of the accuracy for the forecasted values.
- Hence, the MASE values are calculated for all the candidate ARMA+GARCH models.

Hide

```
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))
}
```

Hide

```
Bitcoin_price_mase$Closing.price <- as.numeric(as.character(Bitcoin_price_mase$Closing.price))
Bitcoin_price_ts <- log(Bitcoin_price_ts)
```

Hide

```
forc.12_21 =ugarchforecast(m.12_21, data = Bitcoin_price_g, n.ahead = 10, n.roll = 10)
forc.01_21 = ugarchforecast(m.01_21, data = Bitcoin_price_g, n.ahead = 10, n.roll = 10)
forc.11_21 = ugarchforecast(m.11_21, data = Bitcoin_price_g, n.ahead = 10, n.roll = 10)
forc.12_31 = ugarchforecast(m.12_31, data = Bitcoin_price_g, n.ahead = 10, n.roll = 10)
forc.12_32 = ugarchforecast(m.12_32, data = Bitcoin_price_g, n.ahead = 10, n.roll = 10)
```

11.1 MASE for ARMA(0,1)+ GARCH(2,1)

Hide

```
## MASE over fitted values
fitted.value =fitted(m.01_21)
o_1 =diffinv(as.vector(fitted.value), xi =  8.245497)
data_logback= exp(o_1)
data_logback_fit = data_logback[2:11]
MASE(observed = as.vector(Bitcoin_price_mase$Closing.price), fitted = as.vector(data_logback_fit))
```

```
$MASE
```

MASE
<dbl>
2.225645

1 row

Hide

```
## MASE over forecast
forecast_bit = forc.01_21@forecast$seriesFor
data_logback =diffinv(as.vector(forecast_bit), xi= Bitcoin_price_ts[2130])
data_logback =exp(data_logback)
data_logback = data_logback[2:11]
MASE(observed =as.vector(Bitcoin_price_mase$Closing.price), fitted= as.vector(data_logback))
```

```
$MASE
```

MASE
<dbl>
1.813842

1 row

```
NA
```

- MASE over fitted values= 2.225645
- MASE over forecast= 1.813842

11.2 MASE for ARMA(1,1)+ GARCH(2,1)

Hide

```
fitted.value =fitted(m.11_21)
o_1 =diffinv(as.vector(fitted.value), xi = Bitcoin_price_ts[2130])
data_logback= exp(o_1)
data_logback_fit = data_logback[2:11]
MASE(observed = as.vector(Bitcoin_price_mase$Closing.price), fitted = as.vector(data_logback_fit))
```

```
$MASE
```

MASE
<dbl>
2.616602

1 row

Hide

```
forecast_bit = forc.11_21@forecast$seriesFor
data_logback =diffinv(as.vector(forecast_bit), xi= Bitcoin_price_ts[2130])
data_logback =exp(data_logback)
data_logback = data_logback[2:11]
MASE(observed =as.vector(Bitcoin_price_mase$Closing.price), fitted= as.vector(data_logback))
```

```
$MASE
```

MASE <dbl>
2.456549

1 row

```
NA
```

- MASE over fitted values= 2.616602
- MASE over forecast= 2.456549

11.3 MASE for ARMA(1,2)+ GARCH(2,1)

Hide

```
fitted.value =fitted(m.12_21)
o_1 =diffinv(as.vector(fitted.value), xi = Bitcoin_price_ts[2130])
data_logback= exp(o_1)
data_logback_fit = data_logback[2:11]
MASE(observed = as.vector(Bitcoin_price_mase$Closing.price), fitted = as.vector(data_logback_fit))
```

```
$MASE
```

MASE <dbl>
2.504902

1 row

Hide

```
forecast_bit = forc.12_21@forecast$seriesFor
data_logback =diffinv(as.vector(forecast_bit), xi= Bitcoin_price_ts[2130])
data_logback =exp(data_logback)
data_logback = data_logback[2:11]
MASE(observed =as.vector(Bitcoin_price_mase$Closing.price), fitted= as.vector(data_logback))
```

```
$MASE
```

MASE
<dbl>
2.414436

1 row

```
NA
```

- MASE over fitted values= 2.504902
- MASE over forecast= 2.414436

11.4 MASE for ARMA(1,2)+ GARCH(3,1)

Hide

```
fitted.value =fitted(m.12_31)
o_1 =diffinv(as.vector(fitted.value), xi = Bitcoin_price_ts[2130])
data_logback= exp(o_1)
data_logback_fit = data_logback[2:11]
MASE(observed = as.vector(Bitcoin_price_mase$Closing.price), fitted = as.vector(data_logback_fit))
```

```
$MASE
```

MASE
<dbl>
2.480932

1 row

Hide

```
forecast_bit = forc.12_31@forecast$seriesFor
data_logback =diffinv(as.vector(forecast_bit), xi= Bitcoin_price_ts[2130])
data_logback =exp(data_logback)
data_logback = data_logback[2:11]
MASE(observed =as.vector(Bitcoin_price_mase$Closing.price), fitted= as.vector(data_logback))
```

```
$MASE
```

MASE
<dbl>
2.412144

1 row

```
NA
```

- MASE over fitted values= 2.480932
- MASE over forecast= 2.412144

11.5 MASE for ARMA(1,2)+ GARCH(3,2)

Hide

```
fitted.value =fitted(m.12_32)
o_1 =diffinv(as.vector(fitted.value), xi = Bitcoin_price_ts[2130])
data_logback= exp(o_1)
data_logback_fit = data_logback[2:11]
MASE(observed = as.vector(Bitcoin_price_mase$Closing.price), fitted = as.vector(data_logback_fit))
```

```
$MASE
```

MASE
<dbl>
2.511007

1 row

Hide

```
forecast_bit = forc.12_32@forecast$seriesFor
data_logback =diffinv(as.vector(forecast_bit), xi= Bitcoin_price_ts[2130])
data_logback =exp(data_logback)
data_logback = data_logback[2:11]
MASE(observed =as.vector(Bitcoin_price_mase$Closing.price), fitted= as.vector(data_logback))
```

```
$MASE
```

MASE
<dbl>
2.461447

1 row

```
NA
```

- MASE over fitted values= 2.511007
- MASE over forecast= 2.461447

11.6 MASE for ARMA(1,2)+ GARCH(4,2)

Hide

```
fitted.value =fitted(m.12_42)
o_1 =diffinv(as.vector(fitted.value), xi = Bitcoin_price_ts[2130])
data_logback= exp(o_1)
data_logback_fit = data_logback[2:11]
MASE(observed = as.vector(Bitcoin_price_mase$Closing.price), fitted = as.vector(data_logback_fit))
```

```
$MASE
```

MASE <dbl>
2.453739

1 row

Hide

```
forecast_bit = forc.12_42@forecast$seriesFor
data_logback =diffinv(as.vector(forecast_bit), xi= Bitcoin_price_ts[2130])
data_logback =exp(data_logback)
data_logback = data_logback[2:11]
MASE(observed =as.vector(Bitcoin_price_mase$Closing.price), fitted= as.vector(data_logback))
```

```
$MASE
```

MASE <dbl>
2.39247

1 row

```
NA
```

- MASE over fitted values= 2.453739
- MASE over forecast= 2.39247

- So, after calculating MASE values over fitted values and over forecasts for all the tentative ARMA+GARCH models, ARMA(0,1)+ GARCH(2,1) came out to be the best model with the lowest MASE values over forecasts and fitted values.

- CRITERION:Min.MASE
- MODEL: ARMA(0,1)+ GARCH(2,1)
- MODEL FIT: 2.225645
- FORECASTS: 1.813842

CONCLUSION

The very first and most important step for any time series analysis,is to make the series stationary. So, the series is made to be stationary by log transformation and first differencing. The next step was the specification of the ARIMA model, and after the parameter estimation and AIC scores, ARIMA(2,1,3) model was the best. But, the forecast for the ARIMA was not valuable and so, the need for GARCH models came into the picture. GARCH models are identified by applying EACF to the absolute values and the squared transformations and then, all the candidate GARCH models were fitted. Although the residual analysis for all the models were similar, but with the help of AIC score GARCH(4,2) came out to be the best. In order to find the final best model, ARMA is fitted to the GARCH, and ARMA(1,2)+GARCH(4,2) was found out to be the best fitted model. After forecasting ARMA(1,2)+GARCH(4,2), it could be inferred that the bitcoin daily closing price will go through a little downward (negative) trend in the next 10 days. However, the prediction also gave an idea that the bitcoin prices will be increasing in the near fututre.

Finally, the Mean Absolute Scaled Error (MASE) was calculated for the given forecast bitcoin data. According to the MASE values ARMA(0,1)+ GARCH(2,1) was the best model with the lowest MASE over fitted value of 2.225645 and MASE over forecast value of 1.813842.