Analysis of Bitcoin Daily Closing Price

Akshay Sharma [redacted] *& Vishesh Jain* [redacted]

26 May 2018

Contents

INTRODUCTION

Any form of currency that only exists digitally relying on cryptography to prevent counterfeiting and fraudulent transactions is defined as cryptocurrency. Bitcoin was the very first Cryptocurrency. It was invented in 2009 by an anonymous person, or group of people, who referred to themselves as Satoshi Nakamoto.

In April 2013, the value of 1 bitcoin (BTC) was around 100 USD. At the beginning of 2017 its value was 1,022 USD and by the 15th of December it was worth 19,497 USD. As of the 3rd of March 2018, 1 BTC sells for 11,513 USD. So, the time series analysis of bitcoin series is very challenging.

The challenge is to find the best fitting model to the given cryptocurrency series and Predict the value of bitcoin for next **10-days**. The data-set used is the daily closing price of bitcoin from **27-April-2013** to **03-March-2018**. The data has been gathered from (https://coinmarketcap.com/)

NOTE: The performance of the model will be measured using **Mean-Absolute Scaled Error (MASE)**. We will use the real values of bitcoin for 10 days of forecast period **(4 to 13-March-2018)**.

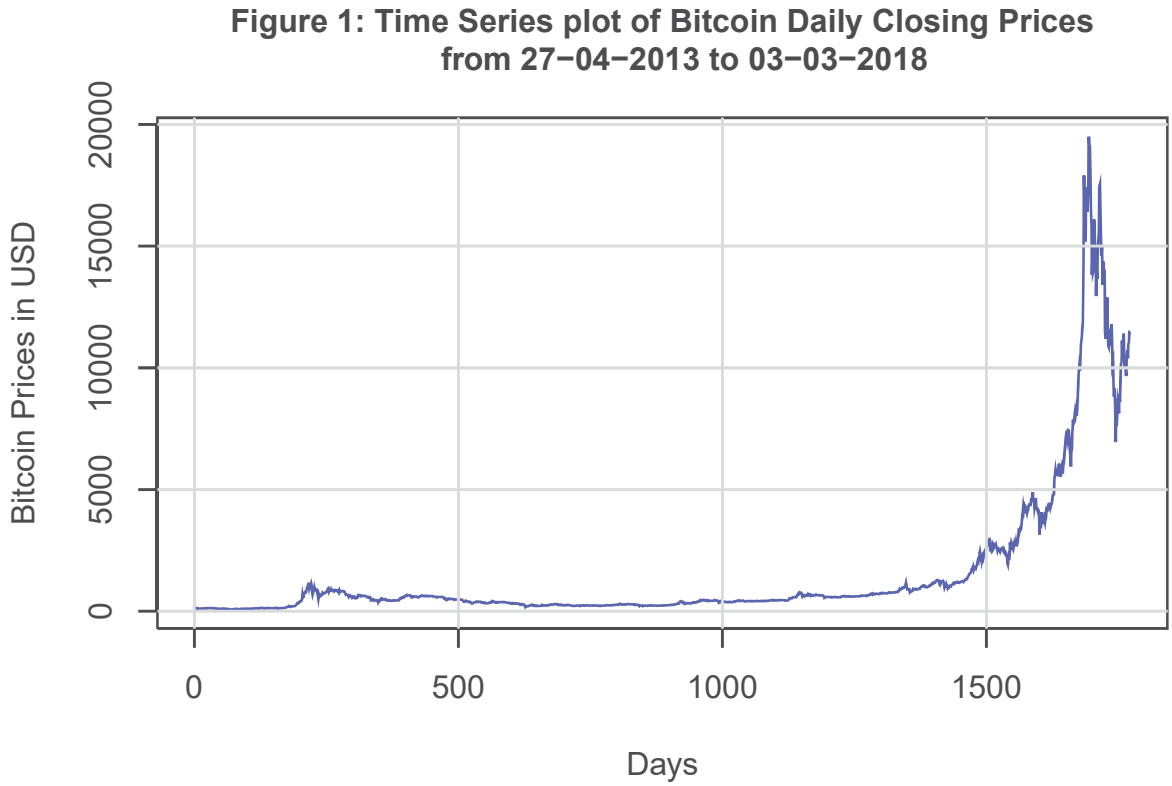
```
BoxCoxSearch = function(y, lambda=seq(-3,3,0.01),
                        m= c("sf", "sw","ad" ,"cvm", "pt", "lt", "jb"),
                        plotit = T, verbose = T){
  N = length(m)
  BC.y = array(NA,N)
  BC.lam = array(NA,N)
  for (i in 1:N){
    if (m[i] == "sf"){
      wrt = "Shapiro-Francia Test"
    } else if (m[i] == "sw"){
      wrt = "Shapiro-Wilk  Test"
    } else if (m[i] == "ad"){
      wrt = "Anderson-Darling Test"
    } else if (m[i] == "cvm"){
      wrt = "Cramer-von Mises Test"
    } else if (m[i] == "pt"){
      wrt = "Pearson Chi-square Test"
    } else if (m[i] == "lt"){
      wrt = "Lilliefors Test"
    } else if (m[i] == "jb"){
      wrt = "Jarque-Bera Test"
    }

    print(paste0("------------- ",wrt," -------------"))
    out = tryCatch({boxcoxnc(y, method = m[i], lam = lambda,
                             lambda2 = NULL, plot = plotit, alpha = 0.05,
                             verbose = verbose)
                    BC.lam[i] = as.numeric(out$lambda.hat)},
                    error = function(e) print("No results for this test!"))

  }
  return(list(lambda = BC.lam,p.value = BC.y))
}
```

```
library(TSA)
library(dplyr)
library(FSAdata)
library(readr)
library(knitr)
```

```r
library(tseries)
library(AID) # For the boxcoxnc() function
library(nortest)
library(lmtest)
library(forecast)
library(fGarch)

#Reading Data in R-Workspace
coin = read.csv("~/Bitcoin_Historical_Price.csv")
```

DATA EXPLORATION - TIME SERIES PLOT

```r
bitcoin = coin$Close # Taking closing Price from the data frame
plot(bitcoin, type = "l",xlab="Days",ylab = "Bitcoin Prices in USD",
     main = "Figure 1: Time Series plot of Bitcoin Daily Closing Prices
     from 27-04-2013 to 03-03-2018",
     col="blue", cex.main = 1)
grid()
```

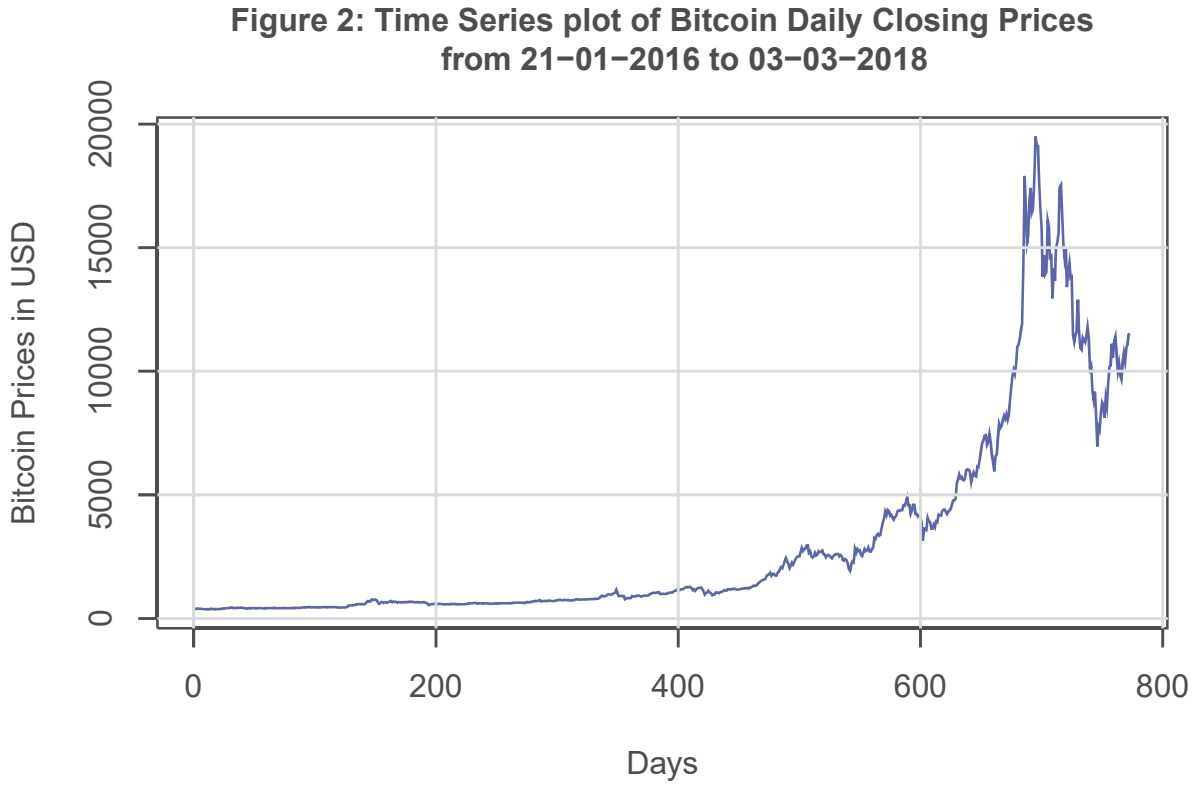


As we can see in this plot, there can be two different time series in the data

The purpose of this project is to predict future values, therefore, we will take the series after 1000 days i.e. from 21-January-2016.

```
bitcoin = bitcoin[1000:1772] # Taking closing prices from 1000 day (21-01-2016)
```

```
plot(bitcoin, type = "l",xlab="Days",ylab = "Bitcoin Prices in USD",
     main = "Figure 2: Time Series plot of Bitcoin Daily Closing Prices
     from 21-01-2016 to 03-03-2018",
     col="blue", cex.main = 1)
grid()
```



NORMALITY DISTRIBUTION OF SERIES

We can clearly see through figures 3 & 4, that the distribution of the series is not normal, this is a right-skewed distribution.

```
par(mfrow = c(1,2))
qqnorm(bitcoin,ylab="Sample Quantiles",
       main = "Figure 3: Normal Q-Q Plot",
       cex.main = 0.8, pch=20, col="blue")
qqline(bitcoin,col = 2, lwd = 1, lty = 2)
grid()

hist(bitcoin, freq = F, breaks = 7,
     main = "Figure 4: Histogram Plot",
     xlab = "bitcoins", col="lightblue", cex.main = 0.8)
grid()
```

Figure 3: Normal Q−Q Plot

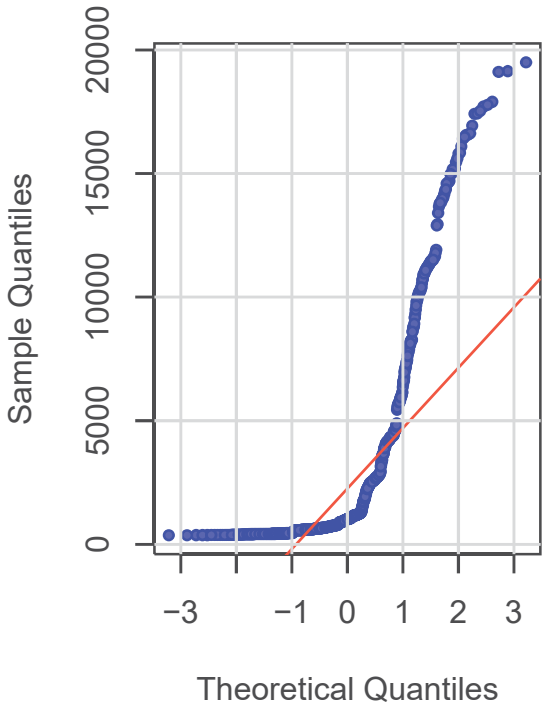
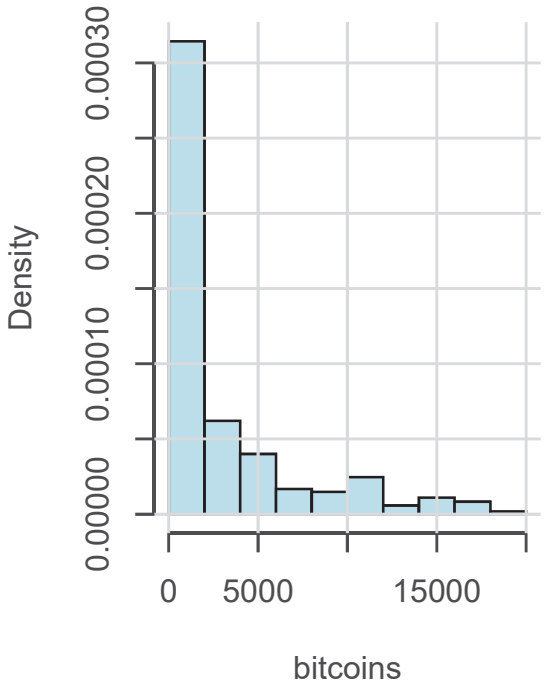


Figure 4: Histogram Plot



Therefore, we will employ a Box-Cox Search method to find the best value of lambda and transform the series using:

```
BoxCoxSearch(bitcoin, plotit = F) # Using normality tests to find lambda
```

```
## [1] "------------- Shapiro-Francia Test -------------"
##
##   Box-Cox power transformation
## ---------------------------------------------------------------
##   data : y
##
##   lambda.hat : -0.57
##
##
##   Shapiro-Francia normality test for transformed data (alpha = 0.05)
## ---------------------------------------------------------------
##
##   statistic  : 0.932774
##   p.value    : 3.626003e-16
##
##   Result     : Transformed data are not normal.
## ---------------------------------------------------------------
##
## [1] "No results for this test!"
## [1] "------------- Shapiro-Wilk  Test -------------"
##
##   Box-Cox power transformation
```

```
## ------------------------------------------------------------------
##   data : y
##
##   lambda.hat : -0.57
##
##
##   Shapiro-Wilk normality test for transformed data (alpha = 0.05)
## ------------------------------------------------------------------
##
##   statistic  : 0.9309022
##   p.value    : 2.086395e-18
##
##   Result     : Transformed data are not normal.
## ------------------------------------------------------------------
##
## [1] "No results for this test!"
## [1] "------------- Anderson-Darling Test -------------"
##
##   Box-Cox power transformation
## ------------------------------------------------------------------
##   data : y
##
##   lambda.hat : -0.63
##
##
##   Anderson-Darling normality test for transformed data (alpha = 0.05)
## ------------------------------------------------------------------
##
##   statistic  : 16.20394
##   p.value    : 3.7e-24
##
##   Result     : Transformed data are not normal.
## ------------------------------------------------------------------
##
## [1] "No results for this test!"
## [1] "------------- Cramer-von Mises Test -------------"
##
##   Box-Cox power transformation
## ------------------------------------------------------------------
##   data : y
##
##   lambda.hat : -0.72
##
##
##   Cramer-von Mises normality test for transformed data (alpha = 0.05)
## ------------------------------------------------------------------
##
##   statistic  : 2.376265
##   p.value    : 7.37e-10
##
##   Result     : Transformed data are not normal.
## ------------------------------------------------------------------
##
## [1] "No results for this test!"
```

```
## [1] "------------- Pearson Chi-square Test -------------"
##
##   Box-Cox power transformation
## ---------------------------------------------------------------
##   data : y
##
##   lambda.hat : -0.43
##
##
##   Pearson Chi-square normality test for transformed data (alpha = 0.05)
## ---------------------------------------------------------------
##
##   statistic  : 495.0091
##   p.value    : 3.752329e-88
##
##   Result     : Transformed data are not normal.
## ---------------------------------------------------------------
##
## [1] "No results for this test!"
## [1] "------------- Lilliefors Test -------------"
##
##   Box-Cox power transformation
## ---------------------------------------------------------------
##   data : y
##
##   lambda.hat : -0.51
##
##
##   Lilliefors normality test for transformed data (alpha = 0.05)
## ---------------------------------------------------------------
##
##   statistic  : 0.1006391
##   p.value    : 1.225191e-20
##
##   Result     : Transformed data are not normal.
## ---------------------------------------------------------------
##
## [1] "No results for this test!"
## [1] "------------- Jarque-Bera Test -------------"
##
##   Box-Cox power transformation
## ---------------------------------------------------------------
##   data : y
##
##   lambda.hat : -0.8
##
##
##   Jarque-Bera normality test for transformed data (alpha = 0.05)
## ---------------------------------------------------------------
##
##   statistic  : 54.87259
##   p.value    : 1.215028e-12
##
##   Result     : Transformed data are not normal.
```

```
## ----------------------------------------------------------------
##
## [1] "No results for this test!"
```

```
## $lambda
## [1] NA NA NA NA NA NA NA
##
## $p.value
## [1] NA NA NA NA NA NA NA
```

Through above search, we found lambda to be 0.05

We will transform the series using the given formula:

```
bit = (bitcoin^(-0.5)-1)/(-0.5)
```

We can see that the distribution of the series has considerably improved after the transformation. As per figure 5, data is aligned with the normality line except for fat tails which is expected in a financial time-series.

```
par(mfrow = c(1,2))
qqnorm(bit,ylab="Sample Quantiles",
       main = "Figure 5: Normal Q-Q Plot",
       cex.main = 0.8, pch=20, col="blue")
qqline(bit,col = 2, lwd = 1, lty = 2)
grid()

hist(bit, freq = F, breaks=5,
     main = "Figure 6: Histogram Plot",
     xlab = "bitcoins", col="lightblue"
     , cex.main = 0.8)
grid()
```

Figure 5: Normal Q−Q Plot

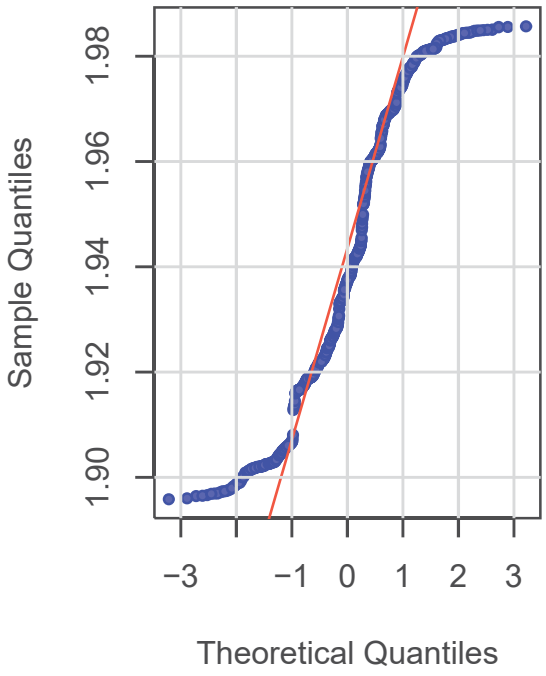
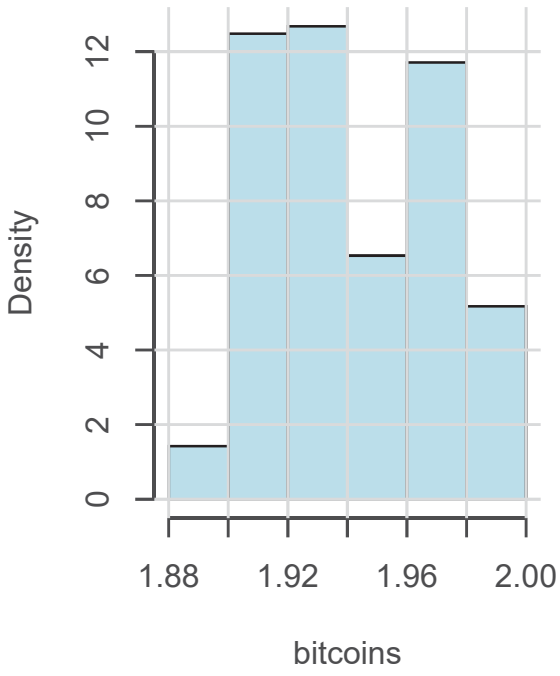
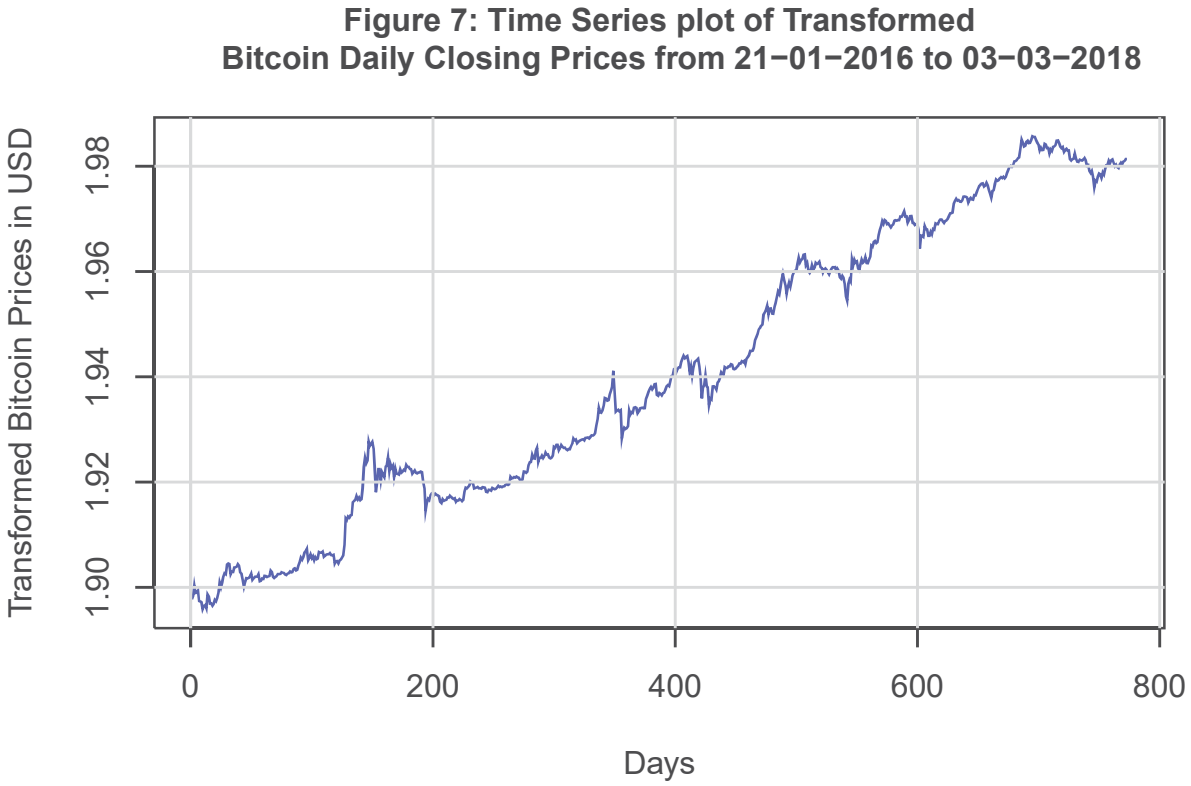


Figure 6: Histogram Plot



BOX-COX TRANSFORMED SERIES

```
plot(bit, type = "l",xlab="Days",
     ylab = "Transformed Bitcoin Prices in USD",
     main = "Figure 7: Time Series plot of Transformed
     Bitcoin Daily Closing Prices from 21-01-2016 to 03-03-2018",
     col="blue", cex.main = 1)
grid()
```



Observations:

- *Trend*: There is a clear upward trend in the series
- *Seasonality*: Seasonality is not clear in the series
- *Behavior*: This could be auto-regressive or moving-average or both
- *Variance*: Change in variance might not be present

Following the given observations, we will move ahead with ARIMA/ SARIMA modeling of the series.

Using ts() function from TSA package, the given series is converted to a time series object.

```
bit1 = ts(log(bit)) #Log is used to control the change in variance (if any)
```

STATIONARITY & CORRELATION IN THE SERIES

As seen in figure 8 & 9, there is ordinary trend in the series. Therefore, we will proceed with first difference of the series.

```
# No-Differencing
par(mfrow=c(1,2))
acf(bit1, lag.max = 50,
    main="Figure 8: ACF Plot of \n Transformed Series",
    cex.main = 1)
acf(bit1, lag.max = 50, type = 'partial',
    main="Figure 9: PACF of \n Transformed Series",
    ylab = 'PACF', cex.main = 1)
```

Figure 8: ACF Plot of Transformed Series

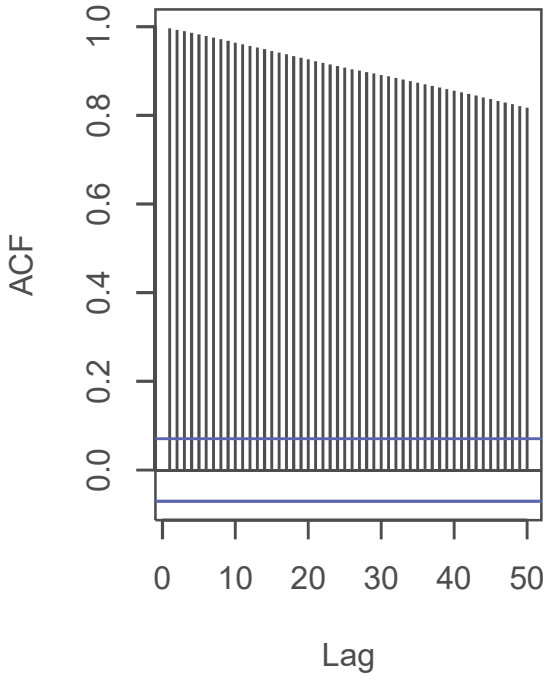
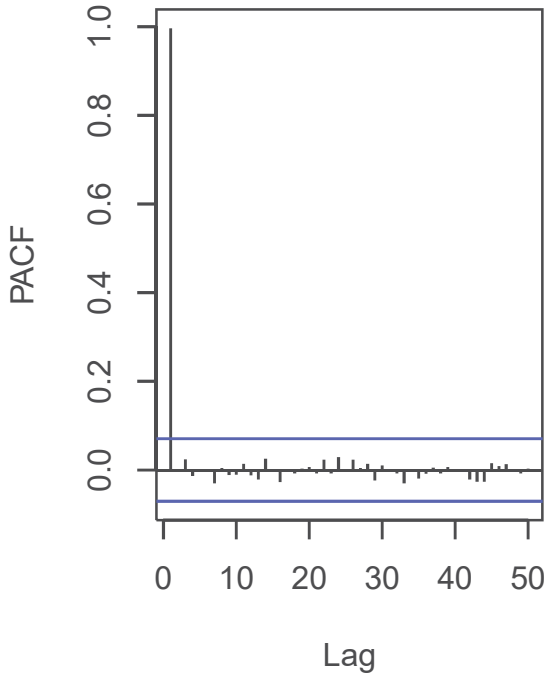


Figure 9: PACF of Transformed Series




```
bitd = diff(bit1)
```

From figure 10 & 11, we can confirm that there is no ordinary trend left in the series. Also, there are no significant AR/MA lags. However, we can see significant lags at period 6, which is hinting towards weak seasonality.

```
# No-Differencing
par(mfrow=c(1,2))
acf(bitd,lag.max=24,
    main="Figure 10: ACF Plot of \n Transformed Series"
    ,cex.main = 1)
acf(bitd, lag.max=24, type = 'partial',
    main="Figure 11: PACF of \n Transformed Series",
    cex.main = 1, ylab = 'PACF')
```

Figure 10: ACF Plot of Transformed Series

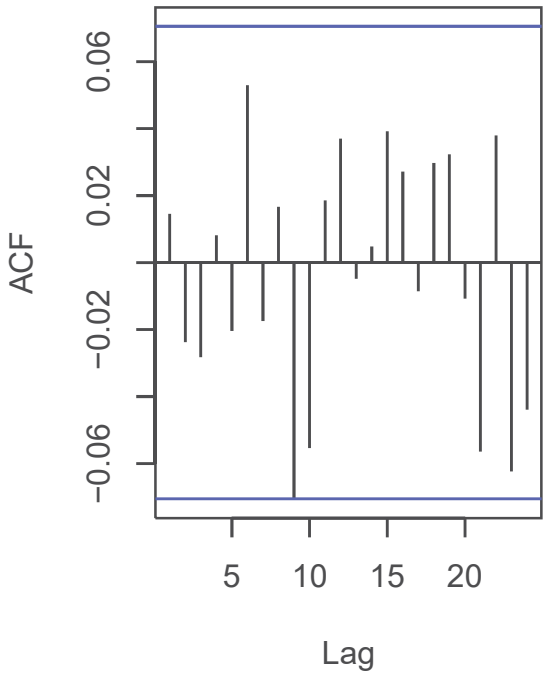
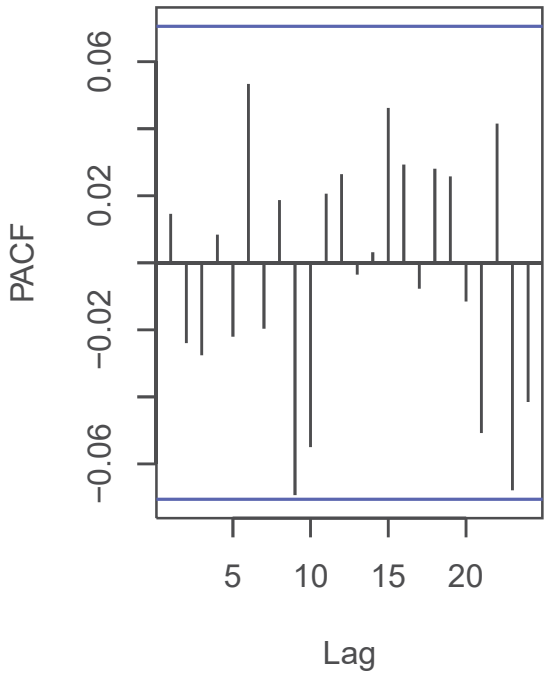


Figure 11: PACF of Transformed Series




```
# First Difference
adf.test(bitd)
```

```
## Warning in adf.test(bitd): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  bitd
## Dickey-Fuller = -9.6357, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(bitd, null = "Trend", lshort = F)
```

```
## Warning in kpss.test(bitd, null = "Trend", lshort = F): p-value greater
## than printed p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  bitd
## KPSS Trend = 0.046841, Truncation lag parameter = 19, p-value =
## 0.1
```

p-value of 0.01 for ADF Test proves that the series is difference stationary. p-value of 0.1 for KPSS proves that the series is trend stationary.

```
Box.test(bitd,  type = "Ljung-Box")
```

```
##
```

```
##  Box-Ljung test
##
## data:  bitd
## X-squared = 0.16323, df = 1, p-value = 0.6862
```

Ljung-box proves that there is auto correlation in the series.

ARIMA/ SARIMA MODELING

No-Differencing : ADF and KPSS test shows that the series is not stationary

First-Difference : ADF and KPSS test shows that the series is stationary now

Remember in ACF & PACF plots of the first-differenced series, there were no significant lags. Therefore, we will use **EACF** and **BIC** to find the suitable candidate models.

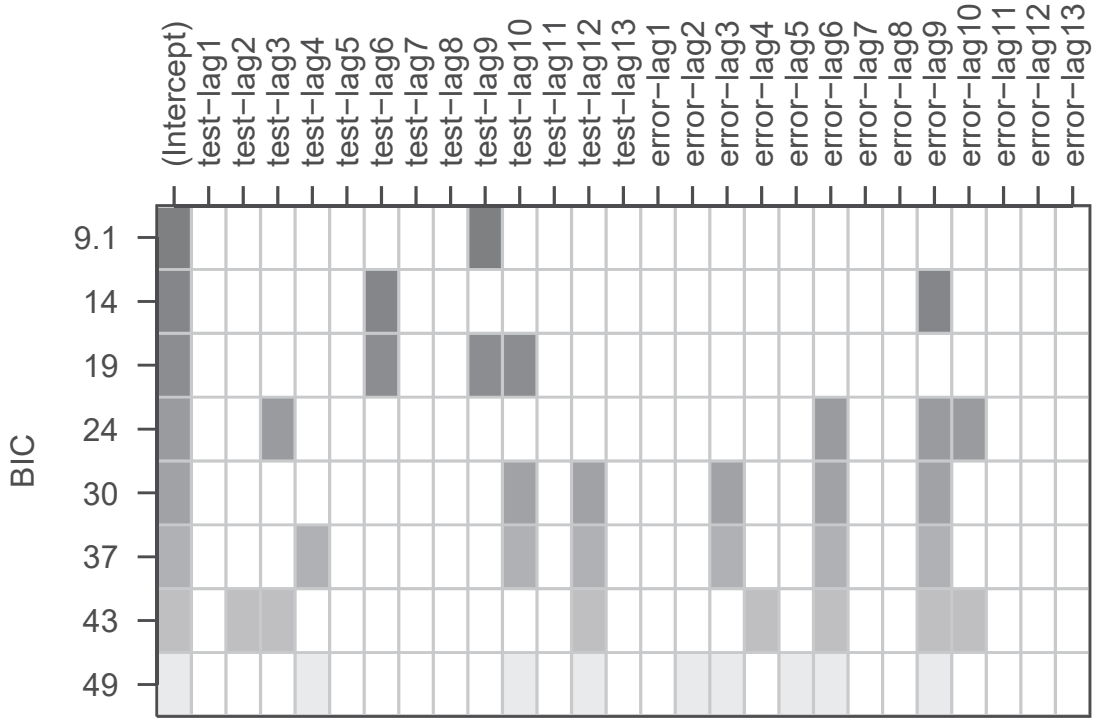
```
eacf(bitd)
```

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

EACF - Possible candidate models are: $ARIMA(1,1,1), ARIMA(0,1,1) ARIMA(2,1,2)$

```
BICplot = armasubsets(y=bitd,nar=13,nma=13 ,
                      y.name='test',ar.method='ols')
plot(BICplot)
title("Figure 12: BIC Plot of the differenced Series",
      line = 6)
```

Figure 12: BIC Plot of the differenced Series



BIC - Possible models are: $ARIMA(3,1,6), ARIMA(6,1,6), ARIMA(2,1,3), ARIMA(2,1,6)$

MODEL FITTING

Let's explore more by fitting all the candidate models.

Through model fitting and coefficient testing, we found that ARIMA(2,1,2) is the best fit model.

After conducting the coefficient test fo ARIMA(2,1,2), we found 'NA' in the standard error and z-value of the model, which means indicates towards seasonality in the data.

We will next explore the seaonal model.

```
for(i in c(0,1,2,3,6)){
  for(j in c(1,2,3,6)){
    print(paste("#############ARIMA", "(",j,",1,",i,")","################",sep = ""))
    print(coeftest(arima(bit1, order = c(i,1,j))))
  }
}
```

```
## [1] "#############ARIMA(1,1,0)################"
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ma1 0.027938   0.036299  0.7697   0.4415
##
```

```
## [1] "#############ARIMA(2,1,0)################"
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ma1  0.0273423  0.0360047  0.7594   0.4476
## ma2 -0.0093611  0.0352780 -0.2654   0.7907
##
## [1] "#############ARIMA(3,1,0)################"
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ma1  0.028162   0.036021  0.7818   0.4343
## ma2 -0.010034   0.035282 -0.2844   0.7761
## ma3 -0.014558   0.034011 -0.4280   0.6686
##
## [1] "#############ARIMA(6,1,0)################"
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ma1  0.0298144  0.0359709  0.8288  0.40719
## ma2 -0.0158915  0.0359630 -0.4419  0.65857
## ma3 -0.0063764  0.0364344 -0.1750  0.86107
## ma4  0.0252391  0.0355759  0.7094  0.47805
## ma5 -0.0109699  0.0382970 -0.2864  0.77454
## ma6  0.0639244  0.0350244  1.8251  0.06798 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "#############ARIMA(1,1,1)################"
## Warning in sqrt(diag(se)): NaNs produced
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.41431         NA      NA       NA
## ma1 -0.39480         NA      NA       NA
##
## [1] "#############ARIMA(2,1,1)################"
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1  0.424619   1.621556  0.2619   0.7934
## ma1 -0.396867   1.620350 -0.2449   0.8065
## ma2 -0.020495   0.036410 -0.5629   0.5735
##
## [1] "#############ARIMA(3,1,1)################"
## Warning in sqrt(diag(se)): NaNs produced
##
```

```
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1  0.4557733         NA      NA       NA
## ma1 -0.4275969         NA      NA       NA
## ma2 -0.0217341  0.0335777 -0.6473   0.5175
## ma3  0.0025458  0.0152634  0.1668   0.8675
##
## [1] "#############ARIMA(6,1,1)################"
##
## z test of coefficients:
##
##        Estimate  Std. Error z value Pr(>|z|)
## ar1 -5.2972e-01  3.4745e-01 -1.5246   0.1274
## ma1  5.5991e-01  3.4852e-01  1.6065   0.1082
## ma2 -7.6112e-05  4.3232e-02 -0.0018   0.9986
## ma3 -2.0323e-02  4.2146e-02 -0.4822   0.6297
## ma4  2.0308e-02  4.0920e-02  0.4963   0.6197
## ma5  6.4266e-03  4.5945e-02  0.1399   0.8888
## ma6  5.5171e-02  3.9840e-02  1.3848   0.1661
##
## [1] "#############ARIMA(1,1,2)################"
## Warning in sqrt(diag(se)): NaNs produced
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1 -0.3458264         NA      NA       NA
## ar2  0.0092652         NA      NA       NA
## ma1  0.3731888         NA      NA       NA
##
## [1] "#############ARIMA(2,1,2)################"
## Warning in sqrt(diag(se)): NaNs produced
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 -1.65207         NA      NA       NA
## ar2 -0.77162         NA      NA       NA
## ma1  1.68510         NA      NA       NA
## ma2  0.81629         NA      NA       NA
##
## [1] "#############ARIMA(3,1,2)################"
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -1.486266   0.498955 -2.9788 0.002894 **
## ar2 -0.630348   0.463629 -1.3596 0.173958
## ma1  1.517078   0.500114  3.0335 0.002418 **
## ma2  0.657023   0.497505  1.3206 0.186623
## ma3 -0.019053   0.054152 -0.3518 0.724952
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "#############ARIMA(6,1,2)################"
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1 -1.0952011  0.9234736 -1.1860   0.2356
## ar2 -0.4115926  0.6452933 -0.6378   0.5236
## ma1  1.1263465  0.9243847  1.2185   0.2230
## ma2  0.4309517  0.6795031  0.6342   0.5259
## ma3 -0.0173682  0.0571433 -0.3039   0.7612
## ma4  0.0023730  0.0599163  0.0396   0.9684
## ma5  0.0073469  0.0583850  0.1258   0.8999
## ma6  0.0380208  0.0672544  0.5653   0.5719
##
## [1] "#############ARIMA(1,1,3)################"
## Warning in sqrt(diag(se)): NaNs produced
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1 -0.0820778         NA      NA       NA
## ar2 -0.0078558  0.0208975 -0.3759  0.70698
## ar3 -0.0180166  0.0087964 -2.0482  0.04054 *
## ma1  0.1098279         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "#############ARIMA(2,1,3)################"
## Warning in sqrt(diag(se)): NaNs produced
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -1.366120         NA      NA       NA
## ar2 -0.824981         NA      NA       NA
## ar3 -0.001594   0.026237 -0.0608   0.9516
## ma1  1.395700         NA      NA       NA
## ma2  0.842252         NA      NA       NA
##
## [1] "#############ARIMA(3,1,3)################"
## Warning in sqrt(diag(se)): NaNs produced
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 -1.02330         NA      NA       NA
## ar2 -0.30830         NA      NA       NA
## ar3  0.45155         NA      NA       NA
```

```
## ma1  1.05516         NA      NA       NA
## ma2  0.33402         NA      NA       NA
## ma3 -0.44415         NA      NA       NA
##
## [1] "#############ARIMA(6,1,3)################"
```

```
## Warning in stats:::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : possible convergence problem: optim gave code = 1
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1 -0.7333969  0.8623423 -0.8505   0.3951
## ar2 -0.0489103  0.8359084 -0.0585   0.9533
## ar3  0.1911807  0.4131206  0.4628   0.6435
## ma1  0.7654329  0.8635709  0.8864   0.3754
## ma2  0.0590462  0.8449602  0.0699   0.9443
## ma3 -0.2131270  0.4251347 -0.5013   0.6161
## ma4  0.0068806  0.0475343  0.1448   0.8849
## ma5  0.0192062  0.0500021  0.3841   0.7009
## ma6  0.0563343  0.0514466  1.0950   0.2735
##
## [1] "#############ARIMA(1,1,6)################"
##
## z test of coefficients:
##
##        Estimate  Std. Error z value Pr(>|z|)
## ar1 -0.40352096  0.48669077 -0.8291   0.4070
## ar2 -0.00061533  0.04127999 -0.0149   0.9881
## ar3 -0.01856400  0.03922091 -0.4733   0.6360
## ar4  0.01612552  0.03952843  0.4079   0.6833
## ar5 -0.00130661  0.04037327 -0.0324   0.9742
## ar6  0.05996013  0.04028607  1.4884   0.1367
## ma1  0.43444178  0.48770403  0.8908   0.3730
##
## [1] "#############ARIMA(2,1,6)################"
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1 -0.1834401  0.4511972 -0.4066   0.6843
## ar2  0.2175060  0.3545227  0.6135   0.5395
## ar3 -0.0222690  0.0384463 -0.5792   0.5624
## ar4  0.0218463  0.0386430  0.5653   0.5718
## ar5 -0.0028817  0.0398044 -0.0724   0.9423
## ar6  0.0617022  0.0389519  1.5841   0.1132
## ma1  0.2137562  0.4515343  0.4734   0.6359
## ma2 -0.2255666  0.3608605 -0.6251   0.5319
##
## [1] "#############ARIMA(3,1,6)################"
```

```
## Warning in stats:::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : possible convergence problem: optim gave code = 1
```

```
##
```

```
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.443280   0.153465  2.8885  0.003871 **
## ar2  0.138696   0.200955  0.6902  0.490079
## ar3 -0.868242   0.141090 -6.1538 7.564e-10 ***
## ar4  0.064974   0.041921  1.5499  0.121157
## ar5 -0.029203   0.050456 -0.5788  0.562731
## ar6  0.025147   0.040429  0.6220  0.533932
## ma1 -0.419755   0.147378 -2.8481  0.004397 **
## ma2 -0.160579   0.191023 -0.8406  0.400558
## ma3  0.865666   0.129936  6.6622 2.697e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "#############ARIMA(6,1,6)################"
```

```
## Warning in stats:::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : possible convergence problem: optim gave code = 1
```

```
## Warning in stats:::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : NaNs produced
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1 -0.472955   0.072531  -6.5208 6.994e-11 ***
## ar2  0.415373   0.039761  10.4468 < 2.2e-16 ***
## ar3  0.044519         NA       NA        NA
## ar4 -0.400983   0.044587  -8.9932 < 2.2e-16 ***
## ar5  0.524701         NA       NA        NA
## ar6  0.888441   0.072251  12.2966 < 2.2e-16 ***
## ma1  0.487766   0.079811   6.1116 9.867e-10 ***
## ma2 -0.409151   0.047869  -8.5474 < 2.2e-16 ***
## ma3 -0.074510         NA       NA        NA
## ma4  0.402144   0.051501   7.8085 5.786e-15 ***
## ma5 -0.525613         NA       NA        NA
## ma6 -0.869022   0.080293 -10.8231 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
m.arima = arima(bit1, order = c(2,1,2))
res.arima = m.arima$residuals
coeftest(m.arima)
```

```
## Warning in sqrt(diag(se)): NaNs produced
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 -1.65207         NA      NA       NA
## ar2 -0.77162         NA      NA       NA
## ma1  1.68510         NA      NA       NA
## ma2  0.81629         NA      NA       NA
```

From the models above we saw that AR and MA 3,3 are relevant with AR2 and MA2 coming to be insignificant. We would try ARIMA(2,1,2), however all the coefficiants are coming to be NA. After doing some online research we found out that we could be missing one seasonal parameter which could lead to NA in the coefficiants.

The model of ARIMA(6,1,6) also seems interesting. Could this be as we found seasonal trends in ACF PACF plot. Let's explore some more.

However, for the sake of parsimony we would not go the higher orders such as 6.

Let's explore ARIMA(2,1,2) with seasonal component at period 6.

```
m1 = Arima(bit1, order = c(2,1,2),
           seasonal = list(order = c(0,0,1),
                           period =6))
coeftest(m1)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1  -0.214893   0.026120  -8.2271 < 2.2e-16 ***
## ar2  -0.967585   0.029948 -32.3089 < 2.2e-16 ***
## ma1   0.229169   0.037480   6.1144 9.692e-10 ***
## ma2   0.943695   0.038219  24.6916 < 2.2e-16 ***
## sma1  0.095924   0.037659   2.5472   0.01086 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
m1
```

```
## Series: bit1
## ARIMA(2,1,2)(0,0,1)[6]
##
## Coefficients:
##           ar1      ar2     ma1     ma2    sma1
##       -0.2149  -0.9676  0.2292  0.9437  0.0959
## s.e.   0.0261   0.0299  0.0375  0.0382  0.0377
##
## sigma^2 estimated as 2.353e-07:  log likelihood=4799.09
## AIC=-9586.17   AICc=-9586.06   BIC=-9558.28
```

All the parameteres are found to be significant at 5% significance level.

```
m1 = Arima(bit1, order = c(2,1,2),
           seasonal = list(order = c(0,0,1),
                           period =6))
coeftest(m1)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1  -0.214893   0.026120  -8.2271 < 2.2e-16 ***
## ar2  -0.967585   0.029948 -32.3089 < 2.2e-16 ***
## ma1   0.229169   0.037480   6.1144 9.692e-10 ***
## ma2   0.943695   0.038219  24.6916 < 2.2e-16 ***
## sma1  0.095924   0.037659   2.5472   0.01086 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
m1
```

```
## Series: bit1
## ARIMA(2,1,2)(0,0,1)[6]
##
## Coefficients:
##           ar1      ar2     ma1     ma2    sma1
##       -0.2149  -0.9676  0.2292  0.9437  0.0959
## s.e.   0.0261   0.0299  0.0375  0.0382  0.0377
##
## sigma^2 estimated as 2.353e-07:  log likelihood=4799.09
## AIC=-9586.17   AICc=-9586.06   BIC=-9558.28
```

After trying SAR1, SMA1 and SARMA(1,1) we found out that SMA1 is the most relevant order for seansonality. Hence we can go with SARIMA(2,1,2)X(0,0,1)_6

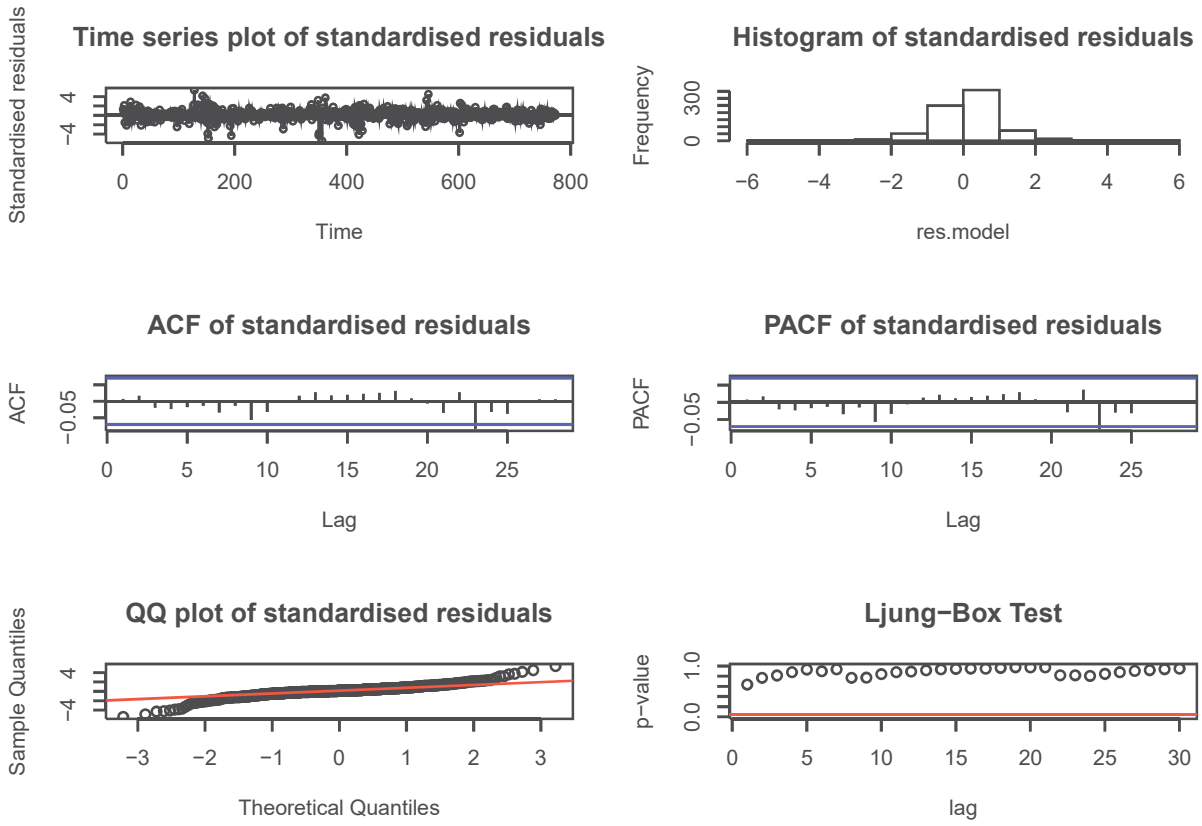
Let's look at residual analysis of this model.

RESIDUAL ANALYSIS

```
residual.analysis <- function(model, std = TRUE,
                              start = 2,
                              class = c("ARIMA","GARCH",
                                        "ARMA-GARCH")[1]){
  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")
  acf(res.model,main="PACF of standardised residuals",
      type = 'partial', ylab = "PACF")
  qqnorm(res.model,main="QQ plot of standardised residuals")
  qqline(res.model, col = 2)
  k=0
  LBQPlot(res.model, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
}
```

```
residual.analysis(m1)
```

```
## Loading required package: lattice
## Loading required package: ltsa
## Loading required package: bestglm
##
## Attaching package: 'FitAR'
## The following object is masked from 'package:forecast':
##
##     BoxCox
```



The residual analysis shows that there is no auto correlation in the residuals, however qq plot shows fat tails and the movements in the residuals hints towards volatile clustering.

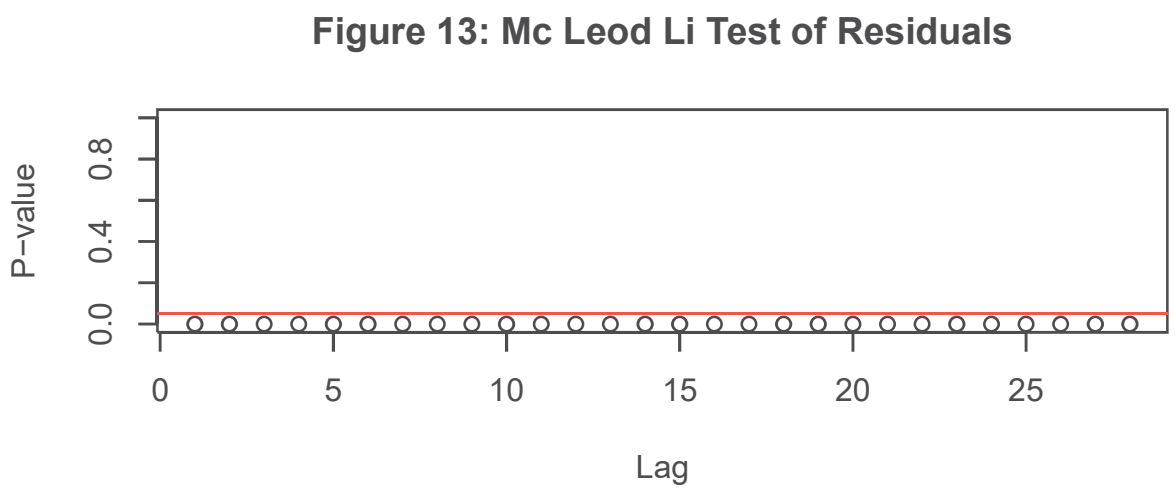
```
shapiro.test(residuals(m1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(m1)
## W = 0.90241, p-value < 2.2e-16
```

GARCH MODELING

Although, the residuals of the fitted model shows no auto-correlation, there might be presence of **volatility-clustering** in the series. We will perform a test to check the presence of this clustering.

```
bitr = residuals(m1)
McLeod.Li.test(y =bitr)
title("Figure 13: Mc Leod Li Test of Residuals")
```



The McLeod Li Test confirms the presence of volatility clustering at all lags. To model this clustering, we will use absolute and squared residuals and analyse ACF, PACF & EACF for possible GARCH models.

ACF & PACF - ABSOLUTE & SQUARE RESIDUALS

```
par(mfrow = c(2,2))
acf(abs(bitr),
    main="Figure 14: Absolute Residuals ACF")
acf(abs(bitr), type = "partial",
    main="Figure 15: Absolute Residuals PACF")
acf((bitr)^2,
    main="Figure 16: Squared Residuals ACF")
acf((bitr)^2, type = "partial",
    main="Figure 17: Squared Residuals PACF")
```

Figure 14: Absolute Residuals ACF

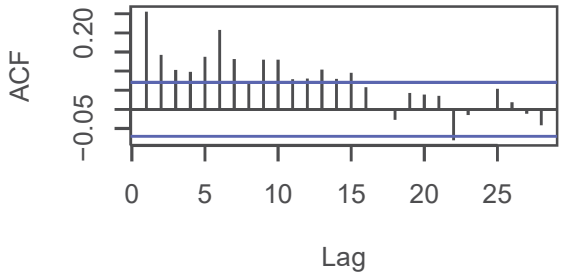


Figure 15: Absolute Residuals PACF

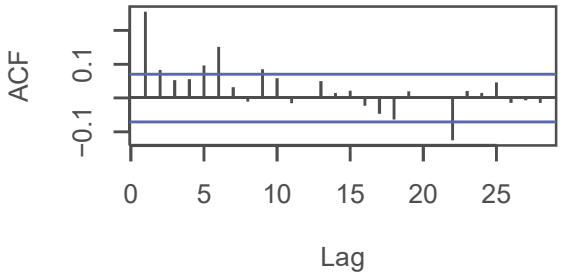


Figure 16: Squared Residuals ACF

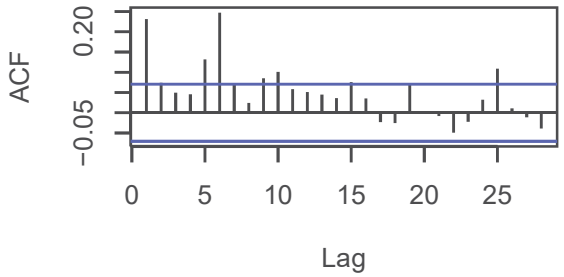
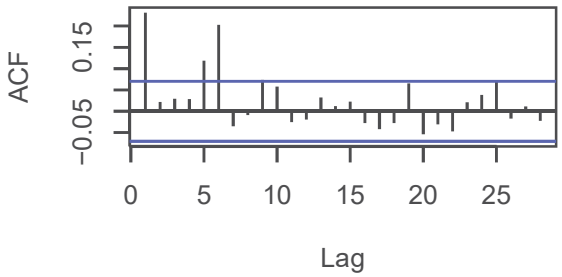


Figure 17: Squared Residuals PACF



We can see many significant lags from the ACF & PACF plots. The p value found from EACF of absolute residuals is 1,2 and from EACF of squared residuals is 2,3.

```
print("----Absolute Residuals EACF-----")
```

```
## [1] "----Absolute Residuals EACF-----"
```

```
eacf(abs(bitr)) #p = 1,2
```

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

```
print("----Squared Residuals EACF-----")
```

```
## [1] "----Squared Residuals EACF-----"
```

```
eacf((bitr)^2) #p = 2,3
```

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

```
## 1 x o o o o x o o o o o  o  o  o
## 2 x o o o o x o x o o o  o  o  o
## 3 x o o o o x x o o o o  o  o  o
## 4 x x x o o x o x o o o  o  o  o
## 5 x x x o x x x o o o o  o  o  o
## 6 x o x x o x o o o o o  o  o  o
## 7 x o x x x x o o o o o  o  o  o
```

GARCH MODELS

Through ACF, PACF & EACF of the absolute and squared residuals, we found possible models as; GARCH(1,2), GARCH(2,2), GARCH(1,3), GARCH(2,3)

```
for( i in c(1:3)){
  for(j in c(1:2)){
    print(paste("###########GARCH", "(",j,",",i,")","##########",sep = ""))
    print(summary(garch(bitr*100, order = c(i,j), trace = F)))
  }
}
```

```
## [1] "###########GARCH(1,1)##########"
##
## Call:
## garch(x = bitr * 100, order = c(i, j), trace = F)
##
## Model:
## GARCH(1,1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7292 -0.2930  0.1287  0.6322  4.4370
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 1.861e-04   2.718e-05    6.846 7.57e-12 ***
## a1 2.084e-01   2.077e-02   10.035  < 2e-16 ***
## b1 7.227e-01   2.452e-02   29.470  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 309.23, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 3.2876, df = 1, p-value = 0.06981
##
## [1] "###########GARCH(2,1)##########"
```

```
##
## Call:
## garch(x = bitr * 100, order = c(i, j), trace = F)
##
## Model:
## GARCH(1,2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8349 -0.2999  0.1306  0.6434  4.8850
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 1.465e-04   3.009e-05    4.866 1.14e-06 ***
## a1 1.537e-01   1.776e-02    8.657  < 2e-16 ***
## a2 9.721e-07   2.618e-02    0.000        1
## b1 7.810e-01   3.046e-02   25.638  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 350.15, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 6.7365, df = 1, p-value = 0.009446
##
## [1] "############GARCH(1,2)###########"
##
## Call:
## garch(x = bitr * 100, order = c(i, j), trace = F)
##
## Model:
## GARCH(2,1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5359 -0.2882  0.1288  0.6220  4.1059
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 2.348e-04   3.816e-05    6.153 7.62e-10 ***
## a1 2.585e-01   2.893e-02    8.936  < 2e-16 ***
## b1 3.493e-01   1.034e-01    3.377 0.000732 ***
## b2 3.022e-01   8.464e-02    3.570 0.000356 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
```

```
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 288.94, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 1.0474, df = 1, p-value = 0.3061
##
## [1] "###########GARCH(2,2)##########"
##
## Call:
## garch(x = bitr * 100, order = c(i, j), trace = F)
##
## Model:
## GARCH(2,2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5940 -0.2948  0.1287  0.6334  4.2500
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 2.172e-04   8.358e-05    2.598  0.00937 **
## a1 2.349e-01   2.757e-02    8.522  < 2e-16 ***
## a2 2.918e-06   8.286e-02    0.000  0.99997
## b1 3.844e-01   3.537e-01    1.087  0.27718
## b2 2.919e-01   2.494e-01    1.170  0.24187
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 298.06, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 1.6975, df = 1, p-value = 0.1926
##
## [1] "###########GARCH(1,3)##########"
##
## Call:
## garch(x = bitr * 100, order = c(i, j), trace = F)
##
## Model:
## GARCH(3,1)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -4.4399 -0.2869  0.1302  0.6202  3.9689
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 2.505e-04   4.081e-05    6.137 8.41e-10 ***
## a1 2.853e-01   3.354e-02    8.504  < 2e-16 ***
## b1 2.870e-01   9.227e-02    3.111  0.00187 **
## b2 1.904e-01   9.183e-02    2.074  0.03812 *
## b3 1.417e-01   7.784e-02    1.821  0.06860 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 291.77, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.39748, df = 1, p-value = 0.5284
##
## [1] "###########GARCH(2,3)##########"
##
## Call:
## garch(x = bitr * 100, order = c(i, j), trace = F)
##
## Model:
## GARCH(3,2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5841 -0.2978  0.1317  0.6491  4.3306
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 0.0002958   0.0001783    1.659   0.0972 .
## a1 0.2237629   0.0262672    8.519   <2e-16 ***
## a2 0.0789730   0.1849678    0.427   0.6694
## b1 0.0014598   0.8085392    0.002   0.9986
## b2 0.3284391   0.3418109    0.961   0.3366
## b3 0.2323626   0.2379062    0.977   0.3287
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 315.97, df = 2, p-value < 2.2e-16
##
```

```
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 1.8307, df = 1, p-value = 0.176
```

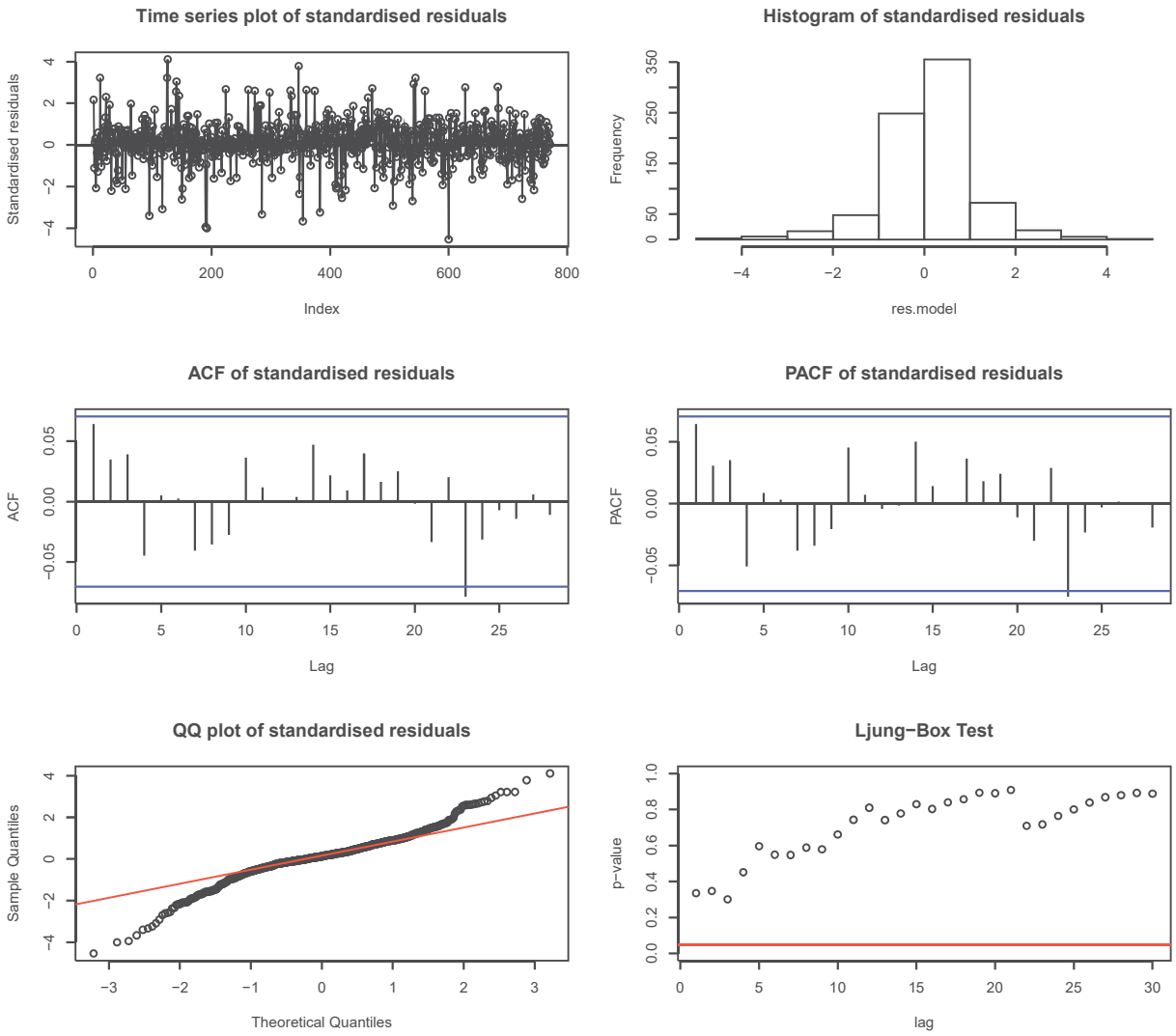
Two models can extracted from the tests above. GARCH(1,2) and GARCH(1,3). However, we can see that beta3 coefficiant is not exactly relevant. Let's test it with AIC to see which one is better.

```
m2<- garch(bitr*100, order = c(2,1), trace = F)
m3<- garch(bitr*100, order = c(3,1), trace = F)
AIC(m2,m3)
```

```
##    df       AIC
## m2  4 -2663.847
## m3  5 -2664.152
```

After, comparing significance of the coefficients and AIC values of the possible candidate models, we found GARCH(1,2) with **AIC=-2663.847** as the best fitted model for the residuals, although the AIC for GARCH(1,3) is minimum as the difference is not too much and beta3 coefficiant is not exactly within our significance level.

```
residual.analysis(m2, class = "GARCH", start = 3)
```



Residual analysis proves that GARCH(1,2) fits well on the residuals of SARIMA model. We can see that model is stationary and now the tails of qq plot are also thin.

SARIMA(2,1,2)X(0,0,1)_6 + GARCH(1,2) FORECASTING

To proceed with this, we will first predict the conditional variance for the next 10 days.

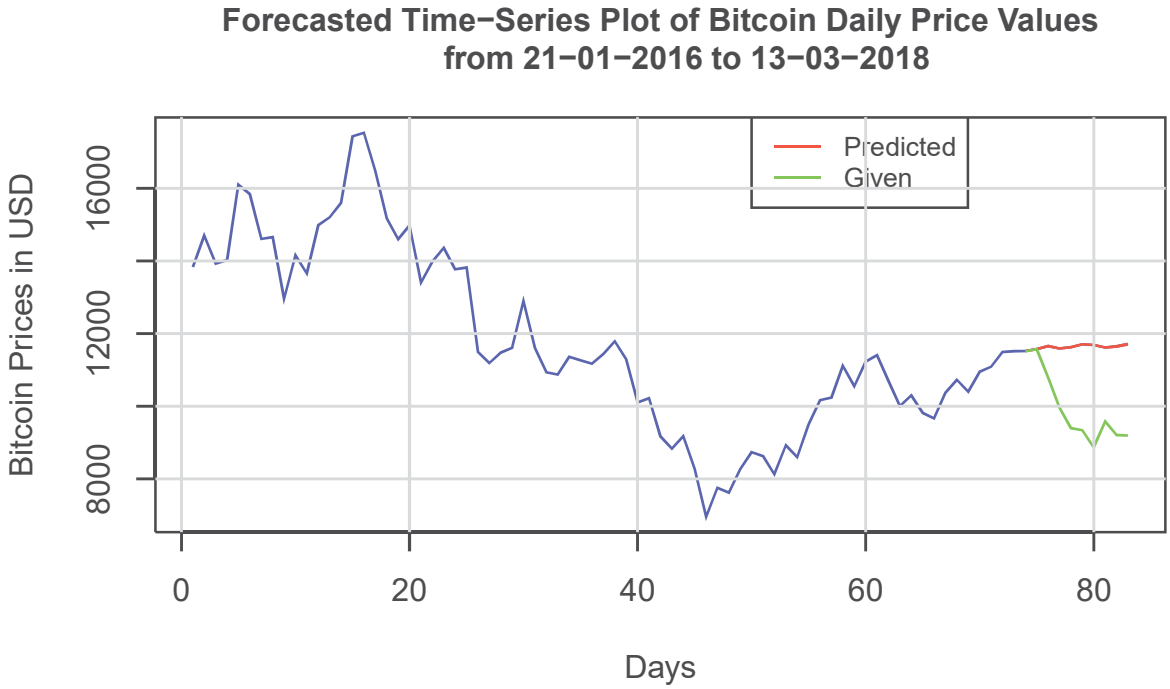
```
g = garchFit(~garch(1,3), bitr*10,
             include.mean = F , trace=FALSE) # GARCH(1,2) fitting
gpred <- predict(g, n.ahead = 10) # Predicating 10-ahead values
omega = g@fit$matcoef[1,1]
alpha = g@fit$matcoef[2,1]
beta = g@fit$matcoef[3,1]
beta2 = g@fit$matcoef[4,1]
bitrl = as.numeric(bitr)

# Finding the 10-ahead conditional variance using omeaga, alpha and beta using a for-loop
```

```
for(i in c(1:10)){
  bitrl[773 + i] = omega + alpha * bitrl[772 + i]^2*gpred$meanError[i] +
    beta*ifelse(is.na(g@h.t[772 + i]),
                gpred$standardDeviation[i-1]^2,g@h.t[772 + i]) +
    beta2*ifelse(is.na(g@h.t[771 + i]),
                 gpred$standardDeviation[i-2]^2,g@h.t[771 + i])
}
```

Next, we will forecast the 10-ahead values from SARIMA model and add the conditional variance found earlier. Then we will untransform the predicted values.

```
bitr1 <- as.numeric(bit1)

# Predicting the 10-ahead values from SARIMA model
kk <- forecast(m1, h = 10)
bitr1 <- c(bitr1, kk$mean)

# Adding conditional variance to the forecasted values from SARIMA model
bitr1[774:783]<- bitrl[774:783]/10 + bitr1[774:783]

# Untransforming the forecasted values
bitrl.1<-exp(bitr1) # For Log Transformation
bitrl.2<-(bitrl.1*(-0.5) + 1)^(-2) # For Box-Cox Transformation

realVal = read.csv("~/Bitcoin_Prices_Forecasts.csv")
```

```
plot(ts(bitrl.2[701:783], start = 1), col="blue",
     ylab="Bitcoin Prices in USD", xlab="Days")
lines(ts(as.vector(bitrl.2[774:783]), start = 74),
      col="red", type="l",lwd=3)
lines(ts(as.vector(realVal$Closing.price), start = 74),
      col="green", type="l",lwd=3)
title("Forecasted Time-Series Plot of Bitcoin Daily Price Values
      from 21-01-2016 to 13-03-2018"
      , cex.main=1)
legend(50, 18000, legend=c("Predicted", "Given"),
       col=c("red", "green"), lty=1:1,lwd = 3:1, cex = 0.8)
grid()
```



The figure above shows the difference between the real values(green) and predicted values(red). We can clearly see that there is much difference between predicted values from our model and real values.

PERFORMANCE MEASURE

As mentioned earlier, **MASE** will be used to check the performance of the model. The series as found during the analysis is seasonal, therefore, **MASE** function for seasonal series will also be used.

```
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))
}
```

```
MASE_SEASONAL = function(observed , fitted ){

  Y.t = observed
  n = length(fitted)
  e.t = Y.t - fitted
```

```
  sum = 0
  for (i in 7:n){
    sum = sum + abs(Y.t[i] - Y.t[i-6] )
  }
  q.t = e.t / (sum*(n-7)/(n-1))
  MASE = data.frame( MASE = mean(abs(q.t)))
  return(list(MASE = MASE))
}
```

```
observed = as.numeric(realVal$Closing.price)
fitted = as.numeric(bitrl.2[774:783])

MASE_value  =  MASE(observed, fitted)
print(paste("Value of MASE: ",MASE_value$MASE, sep = ""))
```

```
## [1] "Value of MASE: 3.93777358235943"
```

```
MASE_SEASONAL_value  =  MASE_SEASONAL(observed, fitted)
print(paste("Value of Seasonal MASE: ",
            MASE_SEASONAL_value$MASE, sep = ""))
```

```
## [1] "Value of Seasonal MASE: 0.725999358282658"
```

SUMMARY

- We explored the bitcoin series and found out that there was trend and a clear high movement after 1500 days.
- We tried to fit various ARIMA and SARIMA models to the tranformed series of Bitcoin prices and found out that SARIMA(2,1,2)X(0,0,1)_6 was the best fitting model using residual analysis.
- We found volatile clustering in the residuals of our model and added GARCH(1,2) model on the residuals of the SARIMA model.
- Finally we used SARIMA(2,1,2)X(0,0,1)_6 + GARCH(1,2) for forecasting the values of next 10 days.
- Visually we found that our model prediction was little far from the actual value.
- We used, MASE and MASE seasonal for calculating the preformance of our model which came out to be 3.93 and 0.72 respectively.