Final Project

Namita Chhibba (S3631442), Ohood Alhasiri (S3550844), Soumyashree Giri (S3640845) and Siddhant Gehlot (S3620089)

27 May 2018

Introduction

Bitcoin is the very first Cryptocurrency and one of the most popular around the world which was invented in 2009. The value of Bitcoin is determined by what the people are willing to pay and hence is very volatile and fluctuates wildly every day.

The dataset used for this analysis consists of the daily closing price of bitcoin from April 27,2013 to March 3,2018 and is gathered from `coinmarketcap.cpm` .

The aim of the project is to do the time series analysis on the Bitcoin Historical Prices and find the best fitting model to the given series to forecast the future values.

Loading Packages

```
library(readr)
library(TSA)
```

```
## Loading required package: leaps
```

```
## Loading required package: locfit
```

```
## locfit 1.5-9.1    2013-03-22
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-23. For overview type 'help("mgcv-package")'.
```

```
## Loading required package: tseries
```

```
##
## Attaching package: 'TSA'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
library(tseries)
library(fUnitRoots)
```

```
## Loading required package: timeDate
```

```
##
## Attaching package: 'timeDate'
```

```
## The following objects are masked from 'package:TSA':
##
##     kurtosis, skewness
```

```
## Loading required package: timeSeries
```

```
## Loading required package: fBasics
```

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following object is masked from 'package:timeSeries':
##
##     time<-
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(knitr)
library(forecast)
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:nlme':
##
##     getResponse
```

```
library(AID)
library(fGarch)
library(rugarch)
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

Importing Dataset

```
## Parsed with column specification:
## cols(
##   Date = col_character(),
##   Close = col_double()
## )
```

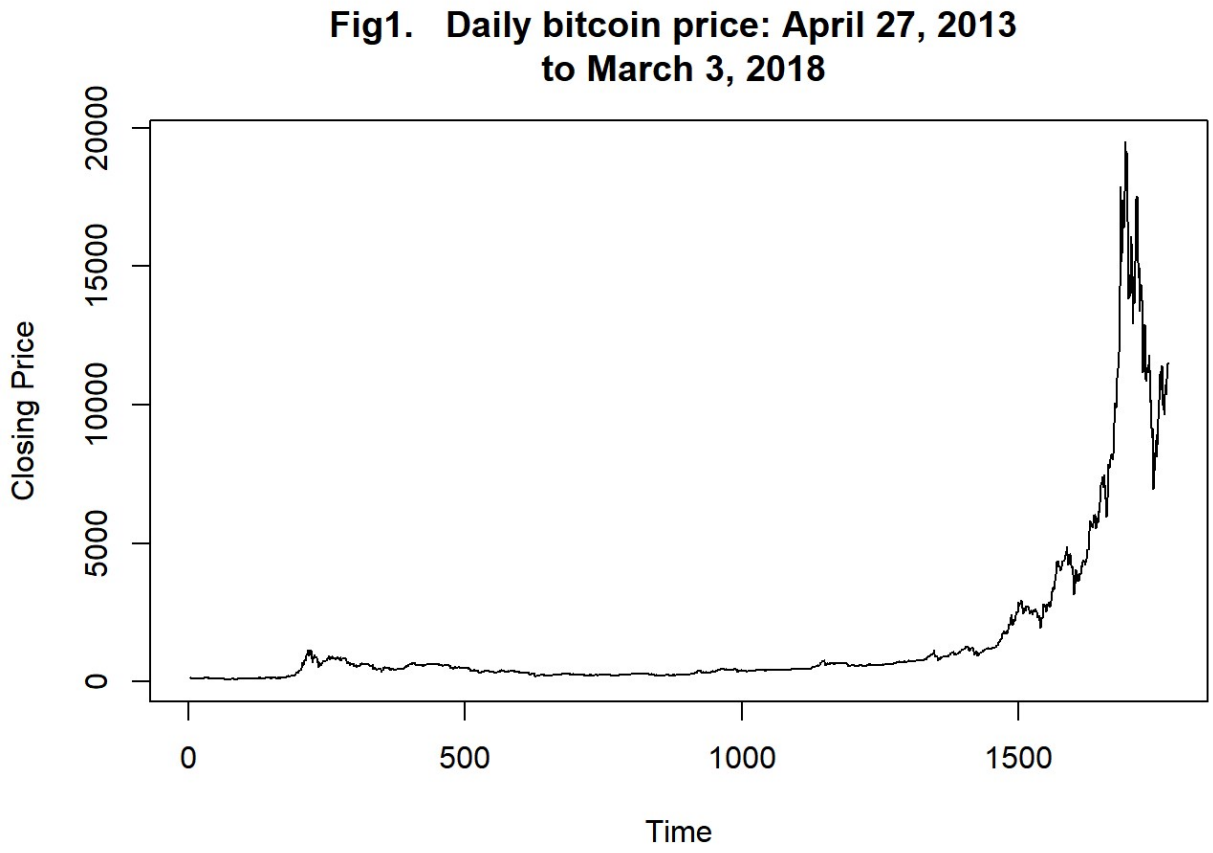
```
## # A tibble: 1,772 x 2
##    Date       Close
##    <chr>      <dbl>
##  1 27-04-2013 134.
##  2 28-04-2013 145.
##  3 29-04-2013 139.
##  4 30-04-2013 117.
##  5 01-05-2013 105.
##  6 02-05-2013  97.8
##  7 03-05-2013 112.
##  8 04-05-2013 116.
##  9 05-05-2013 112.
## 10 06-05-2013 112.
## # ... with 1,762 more rows
```

Converting data to time series object

```
bhp.ts <-ts(as.vector(bhp$Close), frequency=1)
par(mfrow=c(1,1))
```

Descriptive Analysis

```
plot(bhp.ts, type='l', ylab='Closing Price', main="Fig1.   Daily bitcoin price: April
27, 2013 \n to March 3, 2018")
```



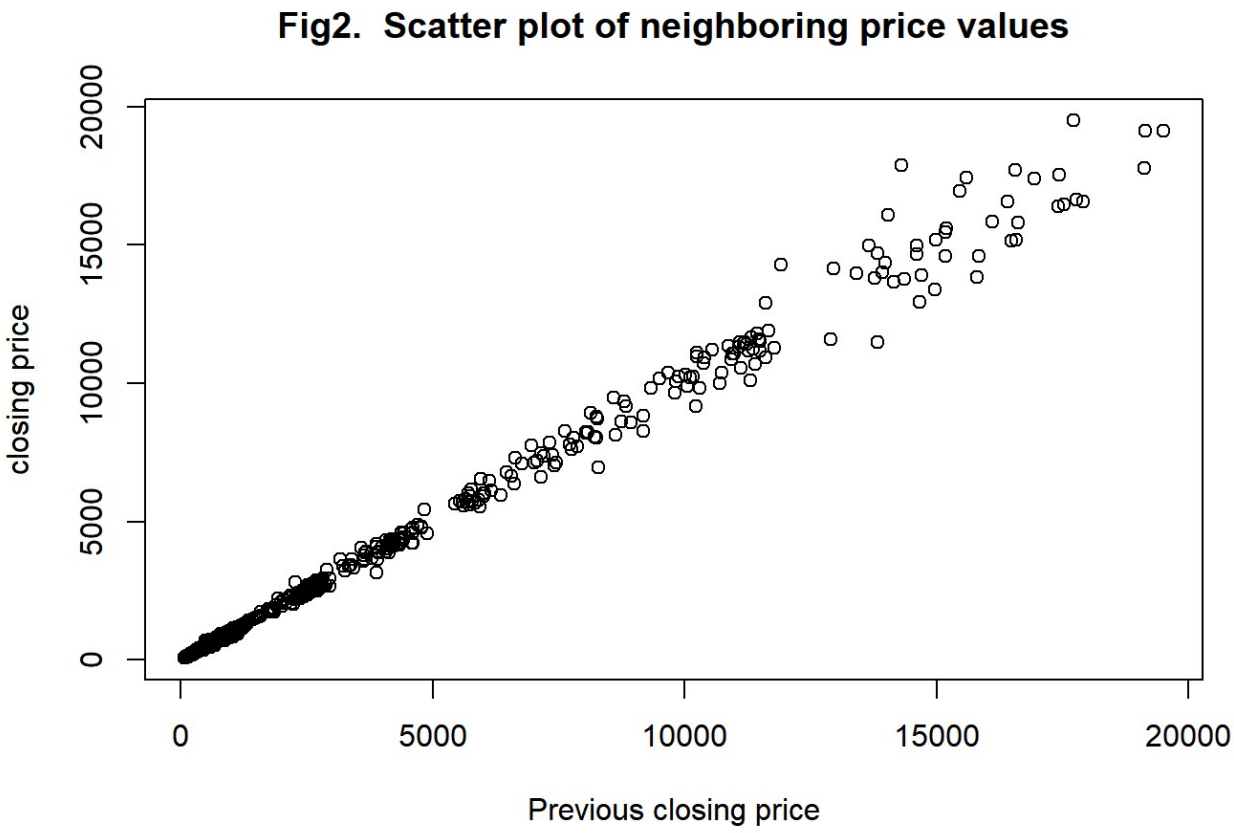
Interpretation:

As we can see the class is time series object and thereby the graph plotted is the time series plot.The discussion of the produced plot is based on 4 aspects.

1. Trend: There is a clear display of upward trend for major part of the series, implying that price of bitcoin has increased over the period of time first slowly and then showing disinguishable abrupt rise during the later stage in the series. But next we see a significant fall in the price of bitcoin though showing recovery at last.
2. Seasonality: There is no clear indication of seasonality. As there is no repeating pattern seen.
3. Changing variance: Changing variance is obvious. In some parts it's too low, in some very high and in some intermediate.
4. Autoregressive behaviour or Moving Average: It is very clear that succeeding measurements are related to one another as the values that are neighbors in time tend to be similar in size. So we can use chosen value of bitcoin to help forecast next subsequent value. But there is no presence of moving average in the major part of series but as we reach the end of the series we witness some hazy appearance of moving average.

The following code chunk generates a scatter plot to investigate the relationship between pairs of consecutive price values:

```
plot(y=bhp.ts,x=zlag(bhp.ts),ylab='closing price', xlab='Previous closing price', main
= "Fig2.  Scatter plot of neighboring price values")
```



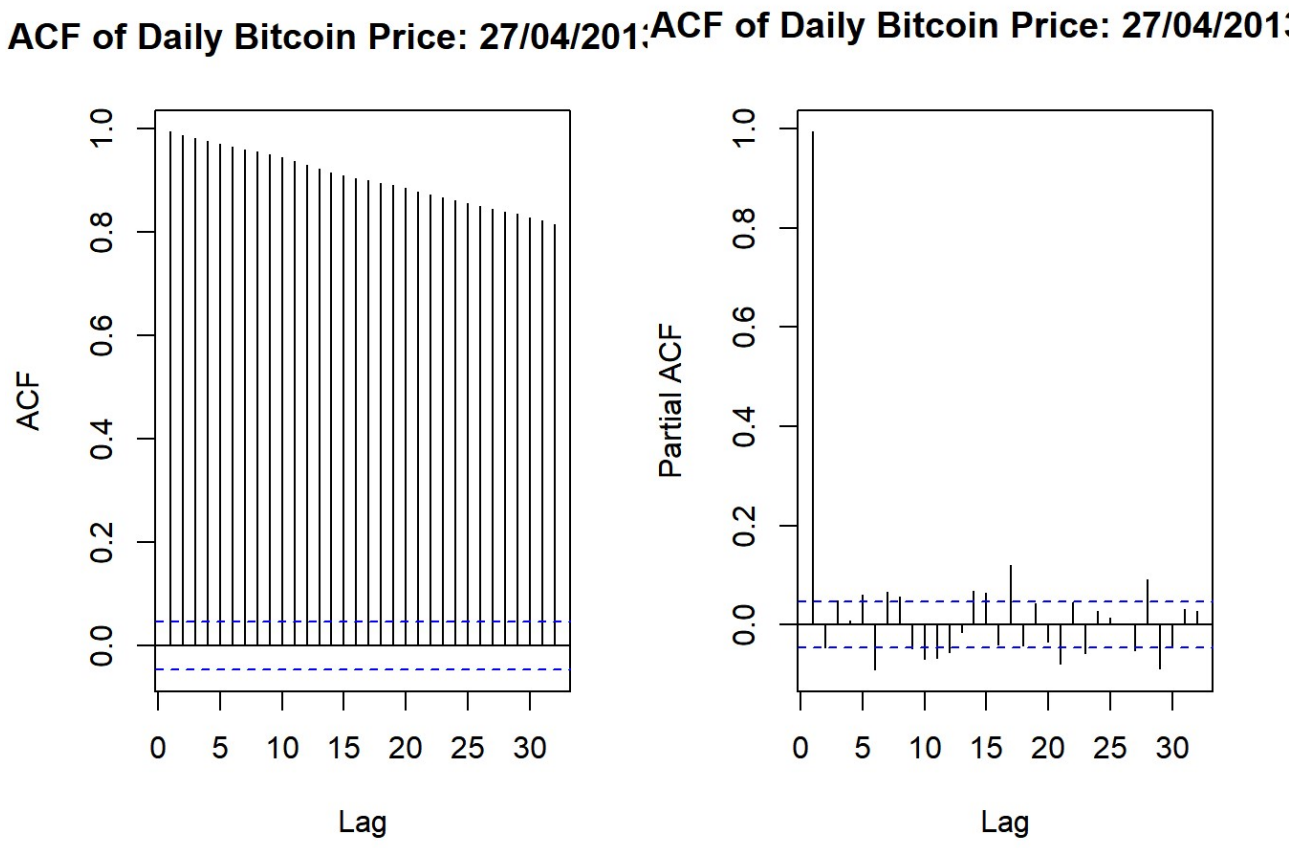
We observe a very neat upward trend. So there is very clear indication of dependency of subsequent values on each other. Corrrelation seems to be very strong and can be calculated as follows:

```
y = bhp.ts
x = zlag(bhp.ts)
index = 2:length(x)
cor(y[index],x[index])
```

```
## [1] 0.9970915
```

As expected, we observe a significantly high correlation (0.997) between present bitcoin price with subsequent bitcoin price.

```
par(mfrow=c(1,2))
acf(bhp.ts, main="Sample ACF of Daily Bitcoin Price: 27/04/2013 to 3/3/2017")
pacf(bhp.ts, main="Sample ACF of Daily Bitcoin Price: 27/04/2013 to 3/3/2017")
```




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

Slowly decaying pattern in ACF and and very high first correlation in PACF implies the existence of trend and non-stationarity.

Apply ADF test: Hypothesis test to decide on the existence of a trend in the series. To do that first we will find the order.

```
ar(diff(bhp.ts))
```

```
##
## Call:
## ar(x = diff(bhp.ts))
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.0708  -0.1226   0.0082  -0.0449   0.1723  -0.0366   0.0019   0.1417
##       9       10       11       12       13       14       15       16
##  0.0394   0.1845   0.0194  -0.0553  -0.1031  -0.1073  -0.0604  -0.1872
##      17       18       19       20       21       22       23       24
##  0.0963  -0.0581   0.1588   0.1660   0.0259   0.1276  -0.0166   0.0118
##      25       26       27       28       29       30       31       32
##  0.0110   0.0963  -0.1750  -0.0207  -0.0695  -0.0801  -0.0872  -0.0414
##
## Order selected 32  sigma^2 estimated as  41686
```

The order comes out to be 32. So we will use it to apply adf test on the series now.

```
adfTest(bhp.ts, lags=32, title=NULL, description=NULL)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 32
##   STATISTIC:
##     Dickey-Fuller: 1.2146
##   P VALUE:
##     0.9416
##
## Description:
##  Mon May 28 11:44:50 2018 by user: user
```

With p-value of 0.9416, we fail to reject the null hypothesis stating that the series is non- stationary.

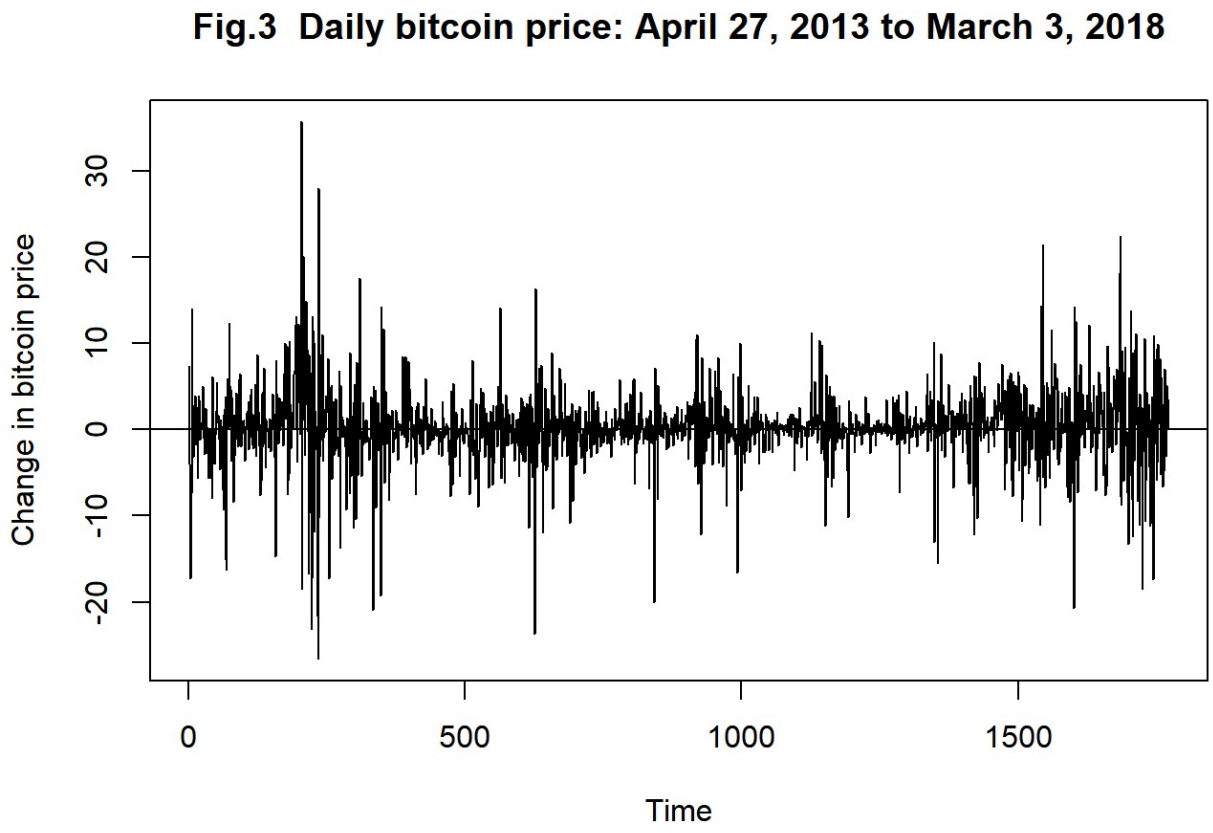
Now, we will try to overcome the non-stationarity nature of the series by using suitable tools.

Because there is variance in the series so we take the log of the series and then to tackle the non-stationarity we do the first differencing. Since after log and differencing we get very small values so we multiply the resultant series with 100.

```
bhp.log <- log(bhp.ts)
r.bhp =diff(log(bhp.ts))*100
head(r.bhp)
```

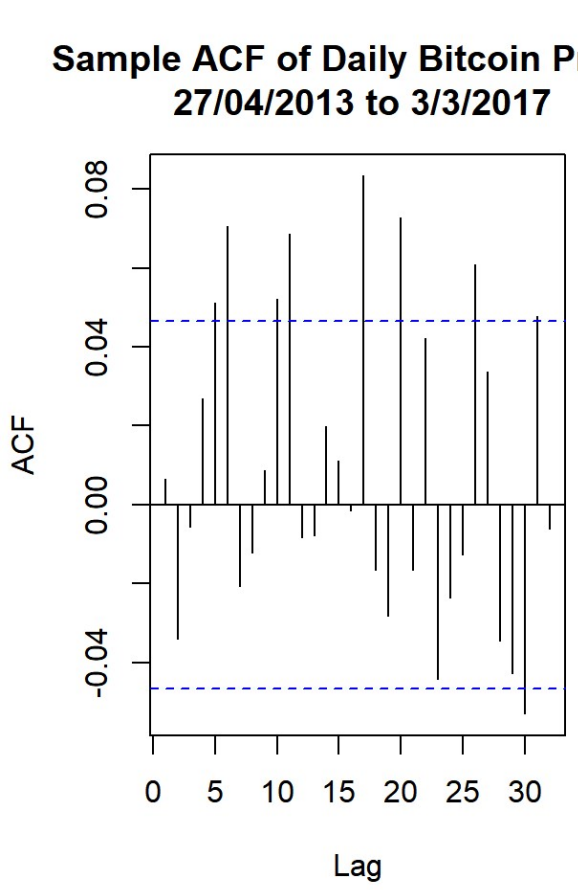
```
## Time Series:
## Start = 2
## End = 7
## Frequency = 1
## [1]   7.415055  -3.908235 -17.238547 -10.613011  -7.354515  14.054002
```

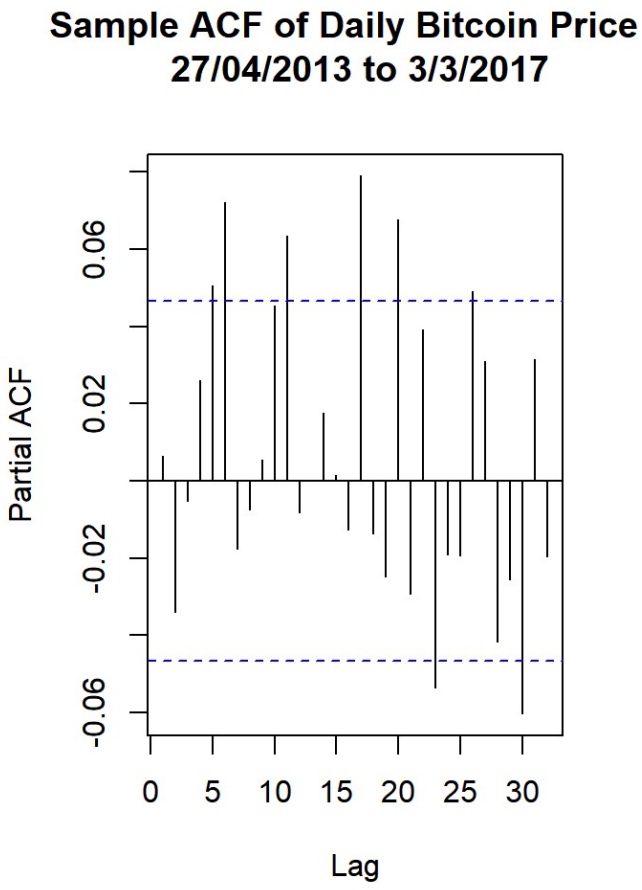
```
par(mfrow=c(1,1))
plot(r.bhp, ylab='Change in bitcoin price', type='l', main=" Fig.3  Daily bitcoin pric
e: April 27, 2013 to March 3, 2018 ")
abline(h=0)
```



Checking the ACF & PACF of transformed series above.

```
par(mfrow=c(1,2))
acf(r.bhp, main="Sample ACF of Daily Bitcoin Price: \n 27/04/2013 to 3/3/2017")
pacf(r.bhp, main="Sample ACF of Daily Bitcoin Price: \n 27/04/2013 to 3/3/2017")
```






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

The acf and pacf plots for the transformed series show many significant lags randomly. To check the stationarity we will perform the adf test on the transformed series.

Checking lag order

```
ar(diff(r.bhp))
```

```
##
## Call:
## ar(x = diff(r.bhp))
##
## Coefficients:
##       1        2        3        4        5        6        7        8
## -0.9459  -0.9388  -0.8974  -0.8356  -0.7694  -0.6769  -0.6772  -0.6464
##       9       10       11       12       13       14       15       16
## -0.6229  -0.5462  -0.4704  -0.4565  -0.4310  -0.3972  -0.3720  -0.3779
##      17       18       19       20       21       22       23       24
## -0.2763  -0.2670  -0.2656  -0.1793  -0.1994  -0.1436  -0.1806  -0.1704
##      25       26       27       28       29       30       31       32
## -0.1676  -0.0955  -0.0475  -0.0715  -0.0742  -0.1116  -0.0520  -0.0492
##
## Order selected 32  sigma^2 estimated as  20.58
```

Applying ADF test on the above series using lag value =32

```
adfTest(r.bhp, lags=32, title=NULL, description=NULL)
```

```
## Warning in adfTest(r.bhp, lags = 32, 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: -6.5432
##   P VALUE:
##     0.01
##
## Description:
##  Mon May 28 11:44:51 2018 by user: user
```

As the p value is less than 0.05, so, the ADF test suggests the stationarity of the transformed series.

Since it's difficult to identify AR and MA orders from the ACF and PACF plots, so, we use EACF to identify the order of autoregressive and moving average components of ARMA models.

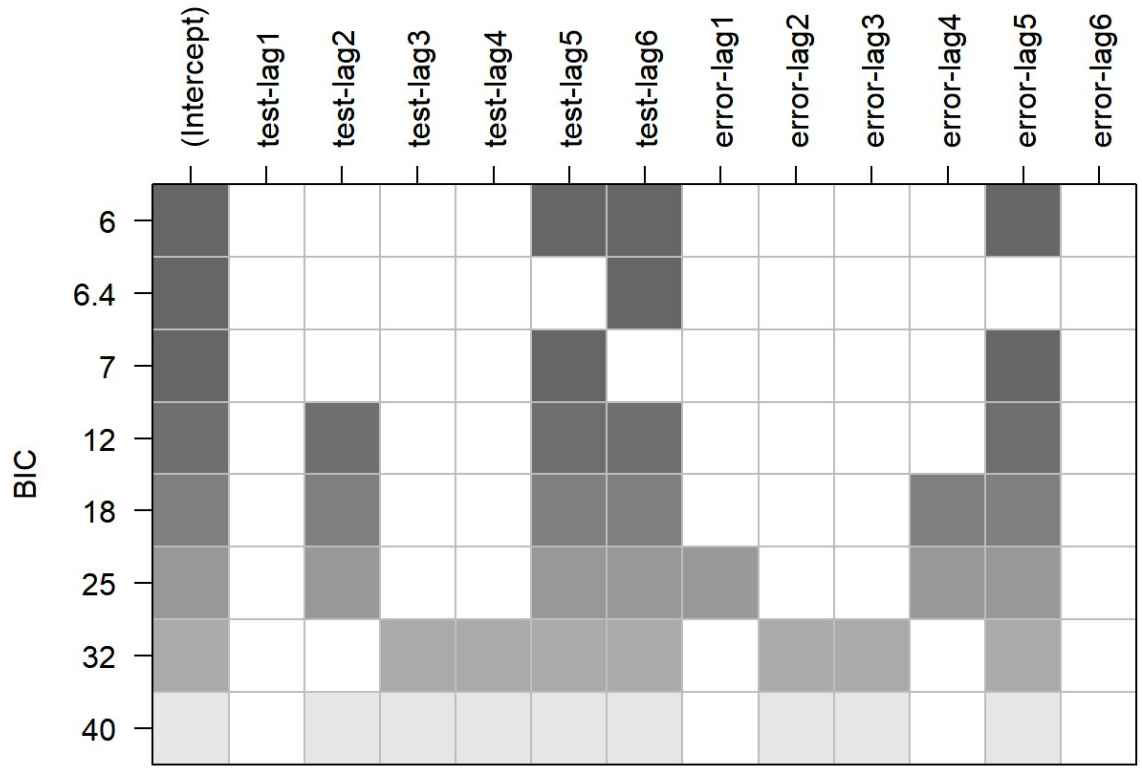
```
eacf(r.bhp)
```

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

In EACF, the vertex includes the orders of p=1 and q=1 and p=2, q=2. Consequently, the set of candidate models chosen from EACF are: ARIMA(1,1,1), ARMA(0,1,1), ARMA(1,1,2); ARIMA(2,1,2), ARIMA(2,1,3), ARIMA(3,1,2)

Further let's display the BIC table, to arrive at some helpful tentative models.

```
par(mfrow=c(1,1))
res= armasubsets(y=r.bhp, nar=6, nma=6, y.name='test', ar.method='ols')
plot(res)
```



In the BIC table shaded columns correspond to AR(5) and AR(6) and MA(5) coefficients. So here we can include ARIMA(5,1,5) and ARIMA(6,1,5). But because these models include a large number of parameters it is not worth including in the set of tentative models.

Now we will fit these models and select the best one according to the selection measures.

I) ARIMA(1,1,1)

ARIMA(1,1,1)-CSS method

```
model_111_css = arima(bhp.log,order=c(1,1,1),method='CSS') #NS
coeftest(model_111_css)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 -0.12498    0.27185 -0.4598   0.6457
## ma1  0.13881    0.27821  0.4989   0.6178
```

ARIMA(1,1,1)-MLE method

```
model_111_mle = arima(bhp.log,order=c(1,1,1),method='ML')  #NS
coeftest(model_111_mle)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 -0.37866    0.52360 -0.7232   0.4696
## ma1  0.39596    0.52317  0.7569   0.4491
```

Both AR(1) and MA(1) coefficients are insignificant for both CSS & MLE methods for ARIMA(1,1,1).

II) ARIMA(0,1,1)

ARIMA(0,1,1)-CSS

```
model_011_css = arima(bhp.log,order=c(0,1,1),method='CSS') #NS
coeftest(model_011_css)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ma1 0.010240   0.024526  0.4175   0.6763
```

ARIMA(0,1,1)-MLE

```
model_011_mle = arima(bhp.log,order=c(0,1,1),method='ML')  #NS
coeftest(model_011_mle)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ma1 0.010251   0.024539  0.4177   0.6761
```

MA(1) component is insignificant for both MLE and CSS methods for ARIMA(0,1,1).

III) ARIMA(1,1,2)

ARIMA(1,1,2)-CSS

```
model_112_css = arima(bhp.log,order=c(1,1,2),method='CSS') #NS
coeftest(model_112_css)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.057253   0.233120 -0.2456   0.8060
## ma1  0.067643   0.232955  0.2904   0.7715
## ma2 -0.025280   0.023225 -1.0884   0.2764
```

ARIMA(1,1,2)-mle

```
model_112_mle = arima(bhp.log,order=c(1,1,2),method='ML') #NS
coeftest(model_112_mle)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1  0.0049591  0.4402458  0.0113   0.9910
## ma1  0.0049531  0.4395671  0.0113   0.9910
## ma2 -0.0289824  0.0231665 -1.2510   0.2109
```

None of the coefficients is significant for ARIMA(1,1,2) model for both MLE and CSS methods.

IV) ARIMA(2,1,2)

ARIMA(2,1,2)-CSS

```
model_212_css = arima(bhp.log,order=c(2,1,2),method='CSS') #S
coeftest(model_212_css)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.954612   0.061743  15.461 < 2.2e-16 ***
## ar2 -0.810779   0.053255 -15.225 < 2.2e-16 ***
## ma1 -0.985525   0.063667 -15.479 < 2.2e-16 ***
## ma2  0.809654   0.056937  14.220 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(2,1,2)-MLE

```
model_212_mle = arima(bhp.log,order=c(2,1,2),method='ML')  #S
coeftest(model_212_mle)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.744968   0.016816  44.301 < 2.2e-16 ***
## ar2 -0.972707   0.014831 -65.588 < 2.2e-16 ***
## ma1 -0.741698   0.024191 -30.661 < 2.2e-16 ***
## ma2  0.944688   0.020423  46.255 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All the components viz. AR(1), AR(2), MA(1) and MA(2) are significant for both CSS and MLE methods for ARIMA(2,1,2) model.

V) ARIMA(2,1,3)

ARIMA(2,1,3)-CSS

```
model_213_css = arima(bhp.log,order=c(2,1,3),method='CSS') # overfitted
coeftest(model_213_css)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value Pr(>|z|)
## ar1  0.954259   0.061565  15.5001  < 2e-16 ***
## ar2 -0.783078   0.065340 -11.9847  < 2e-16 ***
## ma1 -0.955099   0.065833 -14.5079  < 2e-16 ***
## ma2  0.742778   0.075301   9.8641  < 2e-16 ***
## ma3  0.040836   0.024662   1.6558  0.09776 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(2,1,3)-MLE

```
model_213_mle = arima(bhp.log,order=c(2,1,3),method='ML')  # overfitted
coeftest(model_213_mle)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error  z value Pr(>|z|)
## ar1  0.7450210  0.0175523  42.4459   <2e-16 ***
## ar2 -0.9722219  0.0150791 -64.4749   <2e-16 ***
## ma1 -0.7408283  0.0295269 -25.0899   <2e-16 ***
## ma2  0.9432768  0.0278886  33.8230   <2e-16 ***
## ma3  0.0010487  0.0247171   0.0424   0.9662
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All coefficients are significant for ARIMA(2,1,3) using CSS method but MA(3) component is not significant with MLE method.

VI) ARIMA(3,1,2)

ARIMA(3,1,2)-CSS

```
model_312_css = arima(bhp.log,order=c(3,1,2),method='CSS') # Selected but one lag at
1% significance
coeftest(model_312_css)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.903003   0.083010  10.8783 < 2.2e-16 ***
## ar2 -0.709912   0.084040  -8.4473 < 2.2e-16 ***
## ar3  0.049970   0.025730   1.9421   0.05213 .
## ma1 -0.902888   0.080866 -11.1652 < 2.2e-16 ***
## ma2  0.659995   0.082431   8.0066 1.179e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA(3,1,2)-MLE

```
model_312_mle = arima(bhp.log,order=c(3,1,2),method='ML')  # S
coeftest(model_312_mle)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  1.072143   0.171779  6.2414 4.336e-10 ***
## ar2 -0.828499   0.105062 -7.8858 3.125e-15 ***
## ar3  0.055350   0.025513  2.1694   0.03005 *
## ma1 -1.065010   0.171326 -6.2163 5.091e-10 ***
## ma2  0.775862   0.113980  6.8070 9.964e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All coefficients are significant for ARIMA(3,1,2) using both CSS and MLE methods.

So after model fitting and checking for the significance of coefficients we choose ARIMA(2,1,2) and ARIMA(3,1,2) as our selected models.

Further to choose the best model out of these two selected models we use AIC and BIC score.

```
sort.score <- function(x, score = c("bic", "aic")){
  if (score == "aic"){
    x[with(x, order(AIC)),]
  } else if (score == "bic") {
    x[with(x, order(BIC)),]
  } else {
    warning('score = "x" only accepts valid arguments ("aic","bic")')
  }
}

sort.score(AIC(model_212_mle,model_312_mle), score = "aic")
```

```
##               df       AIC
## model_212_mle  5 -5970.630
## model_312_mle  6 -5954.661
```

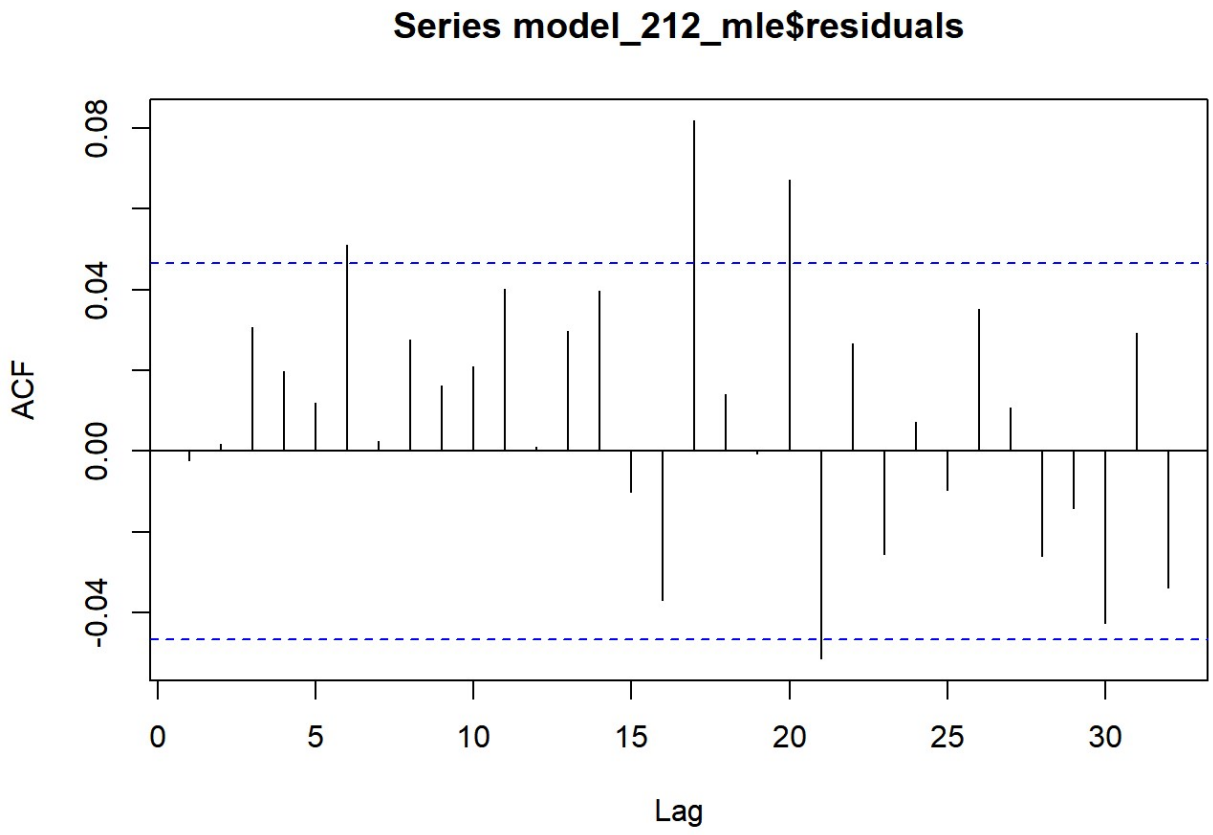
```
sort.score(BIC(model_212_mle,model_312_mle), score = "bic")
```

```
##               df       BIC
## model_212_mle  5 -5943.233
## model_312_mle  6 -5921.785
```

AIC and BIC scores suggest ARIMA(2,1,2) to be a better model over ARIMA(3,1,2).

We now make ACF plot for the residuals of ARIMA(2,1,2) to check for any autocorrelation in residuals.

```
par(mfrow=c(1,1))
acf(model_212_mle$residuals)
```



The acf plot of residuals of the model shows almost a white noise.

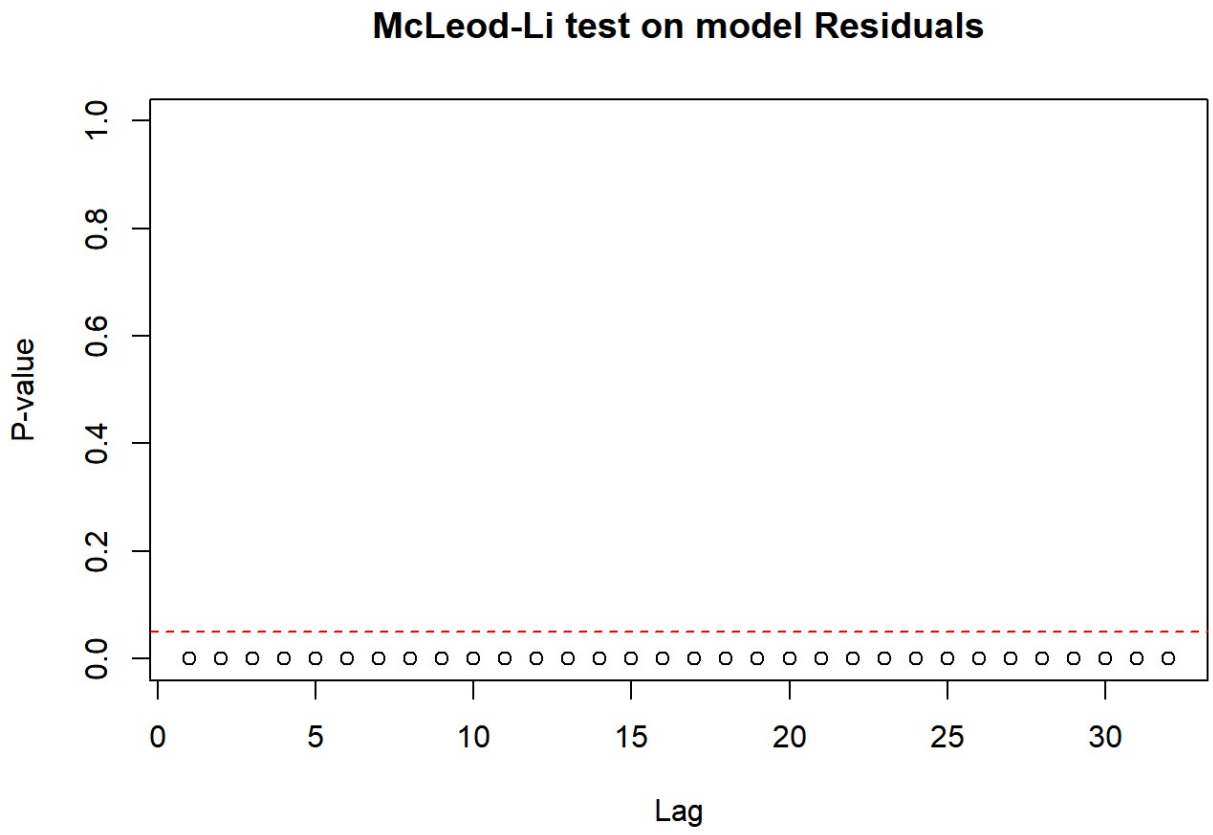
Checking for eacf for the model residuals we observe that:

```
eacf(model_212_mle$residuals)
```

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

EACF also suggests ARMA(0,0) presence. But when we apply McLeod-Li test on the residuals we observe the following:

```
McLeod.Li.test(y=model_212_mle$residuals, main="McLeod-Li test on model Residuals")
```

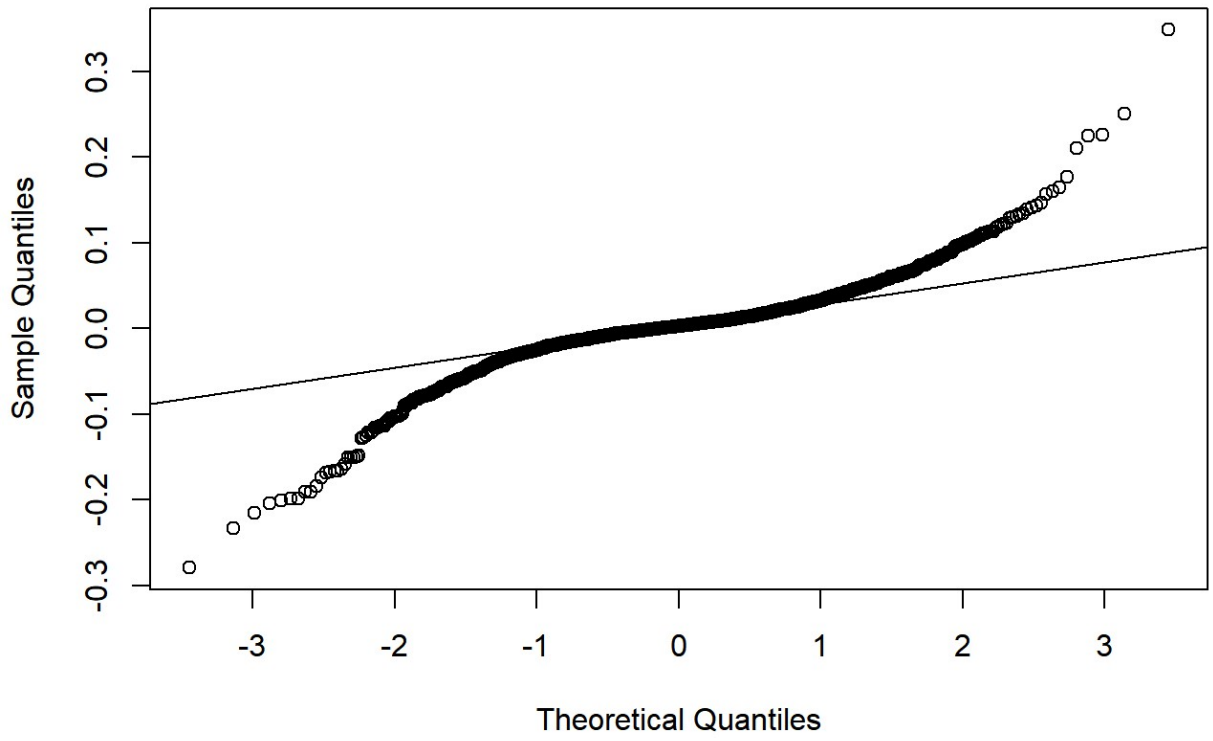


For the model residuals, all the lags are significant. This implies the presence of ARCH effect (Autoregressive Conditional Heteroscedastic effect).

Let's check for the QQ plot for normality of residuals.

```
qqnorm(model_212_mle$residuals, main="Fig.4  QQ Normal plot of model residuals")
qqline(model_212_mle$residuals)
```

Fig.4 QQ Normal plot of model residuals

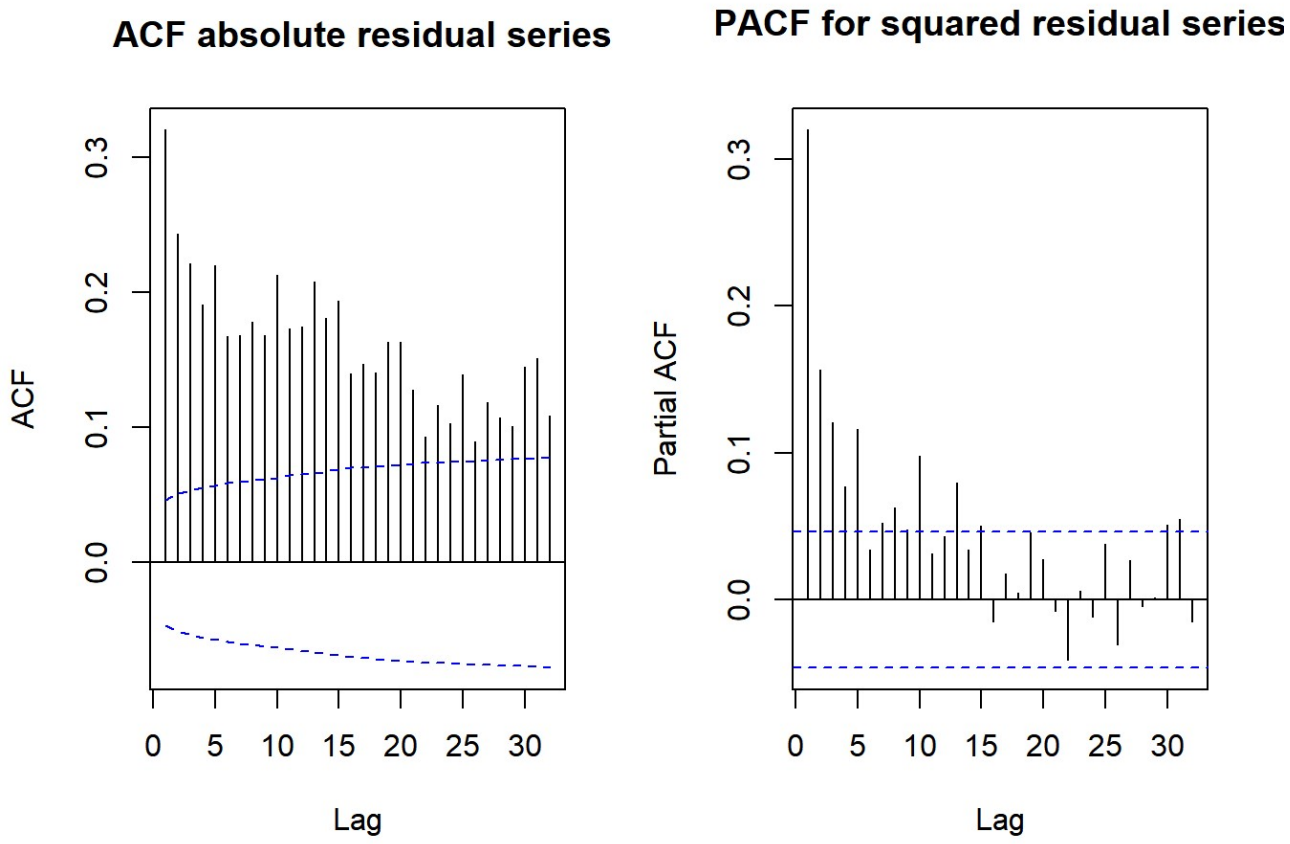


The tick tails in QQ plot is the other characteristic of ARCH effect.

Further to confirm the ARCH effect we will plot the ACF & PACF plot for the squared and absolute series.

```
abs.residuals <- abs(model_212_mle$residuals)
sq.residuals <- (model_212_mle$residuals)^2
```

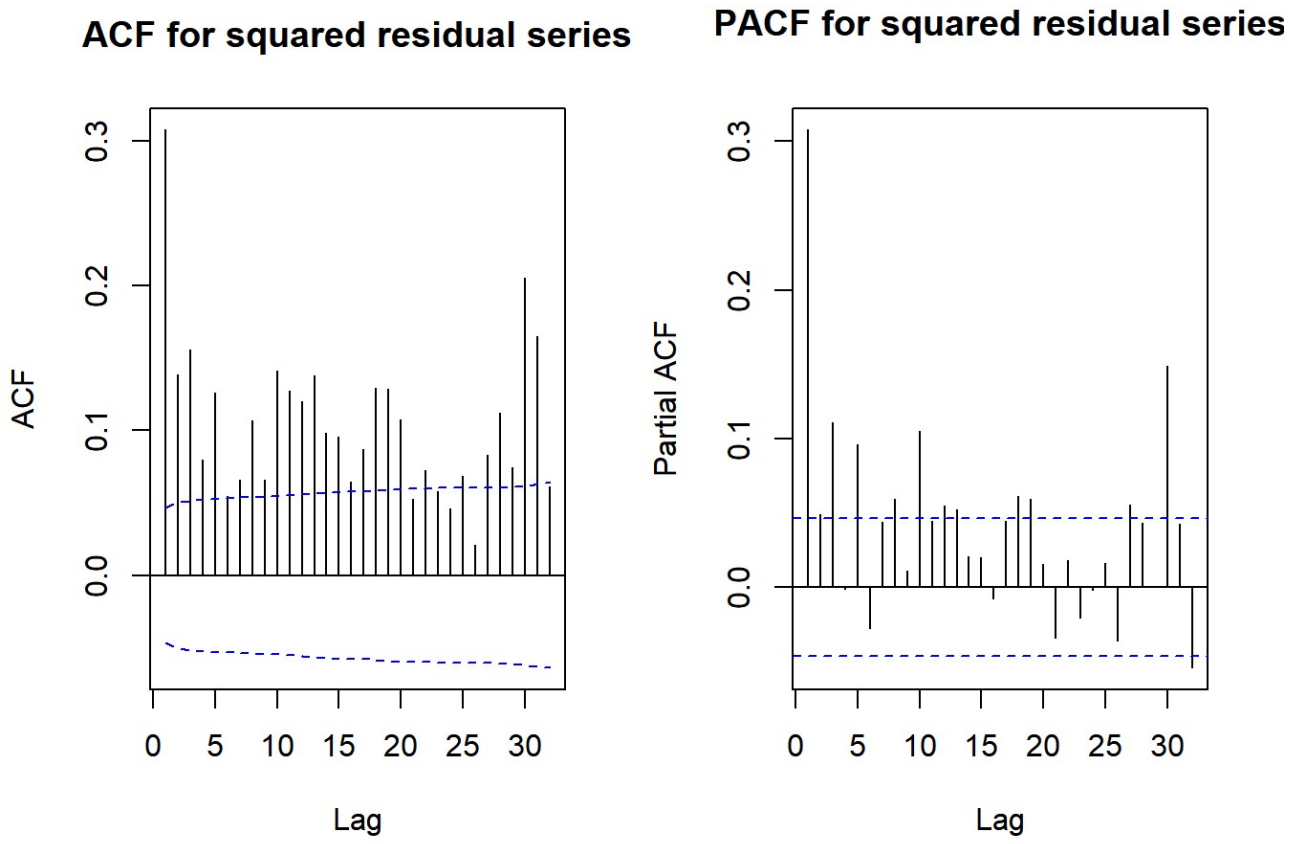
```
par(mfrow=c(1,2))
acf(abs.residuals, ci.type="ma", main=" ACF absolute residual series")
pacf(abs.residuals,  main="PACF for squared residual series")
```



We see that there are lots of significant lags in both ACF and PACF of absolute series.

Similar is the picture for ACF and PACF of squared residuals. As can be seen below:

```
par(mfrow=c(1,2))
acf(sq.residuals, ci.type="ma", main="ACF for squared residual series")
pacf(sq.residuals,  main="PACF for squared residual series")
```



So we are sure of the presence of ARCH effect in the given series considering all above proofs.

Now for model specification we will check for the EACF of both absolute and squared residual series.

```
par(mfrow=c(1,2))
eacf(abs.residuals)
```

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

```
eacf(sq.residuals)
```

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

From the EACF, we can identify ARMA(1,1), ARMA(1,2), ARMA(2,1), and ARMA(2,2), ARMA(2,3), ARMA(3,3) models for absolute and squared series.

These models correspond to parameter settings of [max(1,1),1], [max(1,2),2], [max(2,2),2] . [[max (2,3),3], [max(3,3),3].

So the corresponding tentative GARCH models are GARCH(1,1), GARCH(2,2), GARCH(3,3).

So, now we will fit these models on the residual series and select the best model of all the candidate models.

1. GARCH(1,1)

```
m.11=garch(model_212_mle$residuals, order=c(1,1), trace=FALSE)
```

```
## Warning in sqrt(pred$e): NaNs produced
```

```
summary(m.11)
```

```
##
## Call:
## garch(x = model_212_mle$residuals, order = c(1, 1), trace = FALSE)
##
## Model:
## GARCH(1,1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.4571 -0.3594  0.0874  0.5380  4.7753
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 3.076e-05   3.323e-06    9.256   <2e-16 ***
## a1 1.373e-01   8.502e-03   16.143   <2e-16 ***
## b1 8.643e-01   6.031e-03  143.307   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 5153.2, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 1.0114, df = 1, p-value = 0.3146
```

The effect of both squared returns and conditional variance is significant. Normality is violated. There is no problem of auto correlations.

2. GARCH(2,2)

```
m.22=garch(model_212_mle$residuals, order=c(2,2), trace=FALSE) # NS
```

```
## Warning in garch(model_212_mle$residuals, order = c(2, 2), trace = FALSE):
## singular information
```

```
summary(m.22)
```

```
##
## Call:
## garch(x = model_212_mle$residuals, order = c(2, 2), trace = FALSE)
##
## Model:
## GARCH(2,2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7.97004 -0.36649  0.08904  0.52967  4.97594
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 0.0001546          NA       NA       NA
## a1 0.1963836          NA       NA       NA
## a2 0.0850459          NA       NA       NA
## b1 0.2694337          NA       NA       NA
## b2 0.3781848          NA       NA       NA
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 4646.8, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.10501, df = 1, p-value = 0.7459
```

As there are NAs in the results, therefore the above model doesn't fit the data.

3. GARCH(3,3)

```
m.33=garch(model_212_mle$residuals, order=c(3,3), trace=FALSE) # NS
summary(m.33)
```

```
##
## Call:
## garch(x = model_212_mle$residuals, order = c(3, 3), trace = FALSE)
##
## Model:
## GARCH(3,3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.07428 -0.35707  0.08686  0.52069  5.37141
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 1.834e-04   2.348e-05    7.811 5.77e-15 ***
## a1 9.395e-02   1.134e-02    8.283 2.22e-16 ***
## a2 1.459e-01   2.132e-02    6.844 7.73e-12 ***
## a3 1.731e-01   2.641e-02    6.556 5.53e-11 ***
## b1 4.456e-07   1.096e-01    0.000    1.000
## b2 1.885e-01   1.162e-01    1.622    0.105
## b3 3.256e-01   7.899e-02    4.122 3.76e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 4270.2, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 4.0506, df = 1, p-value = 0.04416
```

Coefficients for previous conditional variance are not significant but coefficients for previous squared returns are significant. Normality is violated but autocorrelation is still present.

We include GARCH(1,2) and GARCH(2,1) as overfit models.

4. GARCH(1,2)

```
m.12=garch(model_212_mle$residuals, order=c(1,2), trace=FALSE) # NS
```

```
## Warning in sqrt(pred$e): NaNs produced
```

```
summary(m.12)
```

```
##
## Call:
## garch(x = model_212_mle$residuals, order = c(1, 2), trace = FALSE)
##
## Model:
## GARCH(1,2)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -15.74212  -0.35011   0.08386   0.50546   4.69605
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 4.760e-05   5.656e-06    8.416   <2e-16 ***
## a1 2.185e-01   2.430e-02    8.991   <2e-16 ***
## a2 1.514e-07   2.791e-02    0.000        1
## b1 8.060e-01   1.286e-02   62.679   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 94764, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.012777, df = 1, p-value = 0.91
```

The output of the model confirms that GARCH(1,2) is an overfit model as we get a2 coefficient as insignificant.

5. GARCH(2,1)

```
m.21=garch(model_212_mle$residuals, order=c(2,1), trace=FALSE)
```

```
## Warning in sqrt(pred$e): NaNs produced
```

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

```
##
## Call:
## garch(x = model_212_mle$residuals, order = c(2, 1), trace = FALSE)
##
## Model:
## GARCH(2,1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.67766 -0.35629  0.08901  0.53556  4.63502
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 3.901e-05   4.912e-06    7.942 2.00e-15 ***
## a1 1.976e-01   1.362e-02   14.511  < 2e-16 ***
## b1 2.335e-01   5.154e-02    4.530 5.89e-06 ***
## b2 5.734e-01   4.512e-02   12.709  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 5053.8, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.28312, df = 1, p-value = 0.5947
```

Coefficients for both squared returns and variance are significant. Normality is violated. Autocorrelations are insignificant.

Therefore, from ARCH/GARCH models GARCH(1,1) and GARCH(2,1) are the candidate models based on significance of the coefiicients.

Checking the AIC score for the two GARCH models we get:

```
AIC(m.21, m.11)
```

```
##      df       AIC
## m.21  4 -6605.385
## m.11  3 -6603.002
```

GARCH(2,1) as a better model in comparison to GARCH(1,1) in terms of AIC score.

```
residual.analysis <- function(model, std = TRUE,start = 2, class = c("ARIMA","GARC
H","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 sta
ndardised 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 = 50, StartLag = k + 1, k = 0, SquaredQ = FALSE)
}
```

Comparison of (GARCH(1,1) & GARCH(2,1)) on the basis of Residual Analysis.

1. Residual Analysis of GARCH(1,1)

```
residual.analysis(m.11, class="GARCH", start=8)
```

```
## Loading required package: lattice
```

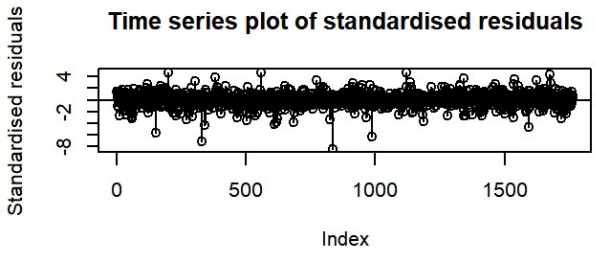
```
## Loading required package: ltsa
```

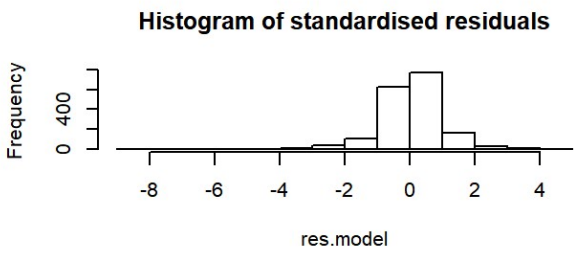
```
## Loading required package: bestglm
```

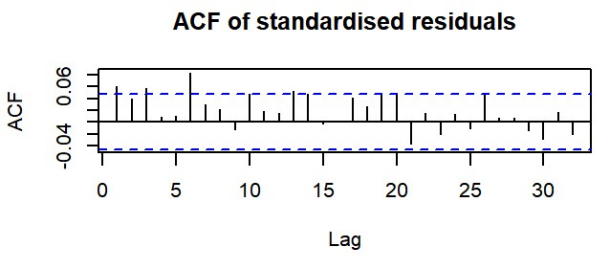
```
##
## Attaching package: 'FitAR'
```

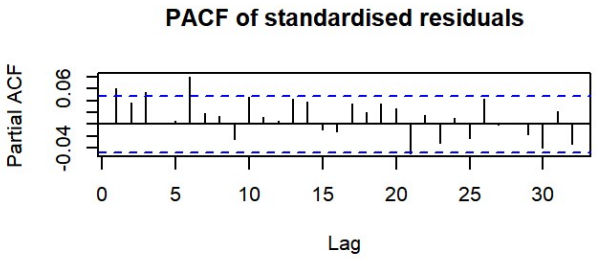
```
## The following object is masked from 'package:forecast':
##
##     BoxCox
```

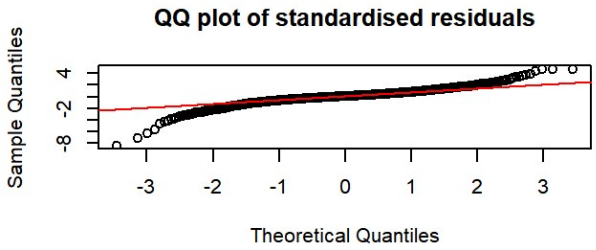
```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.90942, p-value < 2.2e-16
```







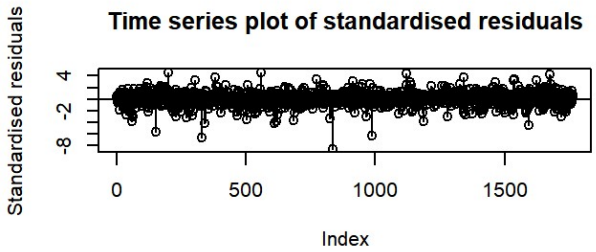


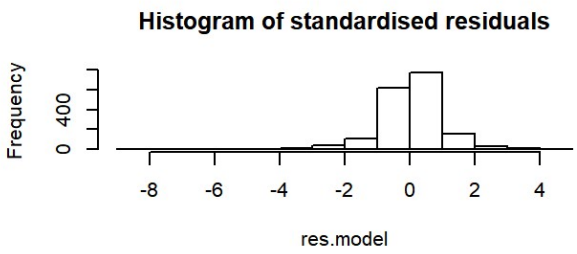


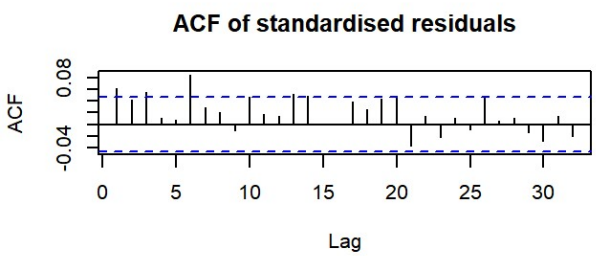
Residual Analysis for GARCH(2,1)

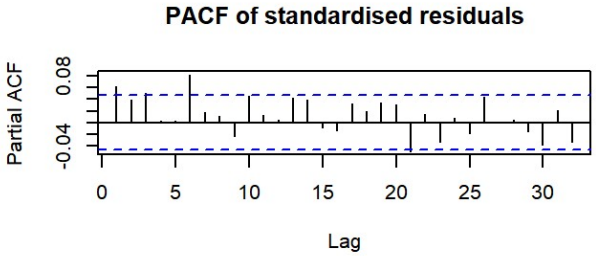
```
residual.analysis(m.21, class="GARCH", start=8)
```

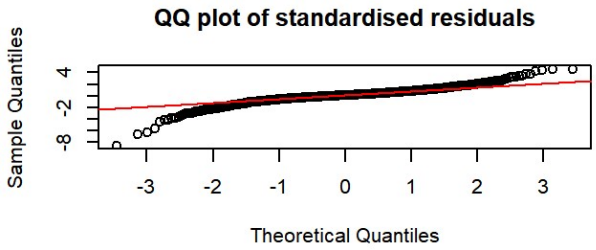
```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.9096, p-value < 2.2e-16
```











Residual Analysis of GARCH(1,1) & GARCH(2,1) models is almost similar.

1. Time series plot shows almost uniform distribution about mean.
2. Histogram of residuals is almost symmetric.
3. ACF and PACF plots depict white noise.
4. Normality is violated

So based on Residual Analysis both GARCH(1,1) and GARCH(2,1) are equally valid.

The summary of the GARCH(1,1) model :

```
m.11_2 = garchFit(formula = ~garch(1,1), data =model_212_mle$residuals )
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(0, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 1)
##  ARMA Order:                0 0
##  Max ARMA Order:            0
##  GARCH Order:               1 1
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          norm
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          1772
##  Recursion Init:            mci
##  Series Scale:              0.04463863
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                       U           V     params includes
##     mu     -0.57543282   0.5754328 0.05754328     TRUE
##     omega   0.00000100 100.0000000 0.10000000     TRUE
##     alpha1  0.00000001   1.0000000 0.10000000     TRUE
##     gamma1 -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  beta1
##      1      2      3      5
##  Persistence:                 0.9
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     2275.6135: 0.0575433 0.100000 0.100000 0.800000
##   1:     2251.8559: 0.0575403 0.0740222 0.103568 0.787789
##   2:     2235.8418: 0.0575343 0.0685107 0.129416 0.799544
##   3:     2227.7327: 0.0575319 0.0562743 0.131275 0.796468
##   4:     2210.4033: 0.0575131 0.0331968 0.170286 0.819873
```

```
##   5:     2208.8078: 0.0575118 0.0292683 0.169687 0.818760
##   6:     2208.1047: 0.0575004 0.0226589 0.172565 0.824653
##   7:     2207.1597: 0.0574455 0.0217807 0.164510 0.841398
##   8:     2206.1113: 0.0573161 0.0144717 0.153533 0.854358
##   9:     2205.8801: 0.0566136 0.0140622 0.144378 0.865032
##  10:     2205.5471: 0.0555748 0.0153188 0.143026 0.860835
##  11:     2205.3519: 0.0513028 0.0139791 0.138990 0.865493
##  12:     2205.3274: 0.0470171 0.0156210 0.139687 0.864524
##  13:     2205.1310: 0.0448776 0.0148373 0.140958 0.862752
##  14:     2205.0999: 0.0427354 0.0152722 0.141929 0.861897
##  15:     2205.0935: 0.0422709 0.0152218 0.140298 0.862655
##  16:     2205.0892: 0.0417957 0.0152473 0.140711 0.862523
##  17:     2205.0892: 0.0417955 0.0151392 0.140726 0.862555
##  18:     2205.0885: 0.0417954 0.0151805 0.140744 0.862590
##  19:     2205.0881: 0.0417950 0.0151563 0.140787 0.862692
##  20:     2205.0879: 0.0417821 0.0151204 0.140785 0.862688
##  21:     2205.0849: 0.0410656 0.0151322 0.140897 0.862611
##  22:     2205.0829: 0.0407796 0.0150017 0.139834 0.863499
##  23:     2205.0822: 0.0404828 0.0149601 0.139937 0.863541
##  24:     2205.0821: 0.0402415 0.0149483 0.139998 0.863532
##  25:     2205.0821: 0.0402515 0.0149455 0.139998 0.863538
##  26:     2205.0821: 0.0402509 0.0149460 0.139998 0.863537
##
## Final Estimate of the Negative LLH:
##  LLH:  -3304.342    norm LLH:  -1.864753
##           mu        omega       alpha1        beta1
## 1.796743e-03 2.978145e-05 1.399983e-01 8.635372e-01
##
## R-optimhess Difference Approximated Hessian Matrix:
##                  mu        omega        alpha1         beta1
## mu     -1955149.591     -7705194      3850.765     -2496.522
## omega  -7705193.930 -73161160370 -33280952.187 -61811546.321
## alpha1     3850.765    -33280952    -27485.458    -40073.060
## beta1     -2496.522    -61811546    -40073.060    -67745.491
## attr(,"time")
## Time difference of 0.02797699 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.2700548 secs
```

```
summary(m.11_2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = model_212_mle$residuals)
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x0000000006e9a5b8>
##  [data = model_212_mle$residuals]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1       beta1
## 1.7967e-03  2.9781e-05  1.4000e-01  8.6354e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.797e-03   7.166e-04    2.507 0.012162 *
## omega  2.978e-05   8.477e-06    3.513 0.000443 ***
## alpha1 1.400e-01   1.788e-02    7.832 4.88e-15 ***
## beta1  8.635e-01   1.593e-02   54.214  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  3304.342    normalized:  1.864753
##
## Description:
##  Mon May 28 11:45:01 2018 by user: user
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  4948.229  0
##  Shapiro-Wilk Test  R    W      0.9096566 0
##  Ljung-Box Test     R    Q(10)  35.02078  0.0001238498
##  Ljung-Box Test     R    Q(15)  45.24199  7.012147e-05
##  Ljung-Box Test     R    Q(20)  57.10771  1.976797e-05
##  Ljung-Box Test     R^2  Q(10)  7.909834  0.6376439
##  Ljung-Box Test     R^2  Q(15)  11.36662  0.7261989
##  Ljung-Box Test     R^2  Q(20)  13.54638  0.8527311
```

```
##  LM Arch Test       R    TR^2   8.758876  0.723376
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.724991 -3.712621 -3.725001 -3.720421
```

The summary of the GARCH(2,1) model :

```
m.21_2 = garchFit(formula = ~garch(2,1), data =model_212_mle$residuals )
```

```
##
## 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:          1772
##  Recursion Init:            mci
##  Series Scale:              0.04463863
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                       U           V     params includes
##     mu     -0.57543282   0.5754328 0.05754328     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:     2286.1728: 0.0575433 0.100000 0.0500000 0.0500000 0.800000
##   1:     2256.5075: 0.0575397 0.0719507 0.0588737 0.0507220 0.786787
##   2:     2239.7554: 0.0575340 0.0665086 0.0846880 0.0669083 0.795876
```

```
##   3:     2228.5146: 0.0575315 0.0523521 0.0849971 0.0646294 0.790046
##   4:     2216.8009: 0.0575162 0.0358148 0.113405 0.0783616 0.803226
##   5:     2215.0180: 0.0575143 0.0305653 0.112998 0.0767397 0.801434
##   6:     2214.3498: 0.0575053 0.0306187 0.116305 0.0747717 0.805762
##   7:     2212.9910: 0.0574843 0.0258901 0.119610 0.0666190 0.811625
##   8:     2206.8364: 0.0573300 0.0223698 0.147457 0.0105983 0.839782
##   9:     2206.6875: 0.0573290 0.0212596 0.147416 0.0102022 0.839696
##  10:     2206.5569: 0.0573242 0.0211608 0.148564 0.00976072 0.841772
##  11:     2206.2021: 0.0573080 0.0187542 0.147165 0.00819256 0.845386
##  12:     2205.8926: 0.0572836 0.0183642 0.146191 0.00642039 0.849737
##  13:     2205.2886: 0.0572052 0.0164166 0.143961 0.000473433 0.856623
##  14:     2205.1554: 0.0568738 0.0155360 0.142614 1.00000e-08 0.861857
##  15:     2205.1086: 0.0564018 0.0146014 0.142230 1.00000e-08 0.862150
##  16:     2205.0565: 0.0559272 0.0151293 0.142127 1.00000e-08 0.861870
##  17:     2204.8793: 0.0508694 0.0161107 0.141214 1.00000e-08 0.859366
##  18:     2204.6490: 0.0458102 0.0155709 0.142263 1.00000e-08 0.861035
##  19:     2204.6439: 0.0453175 0.0148735 0.139813 1.00000e-08 0.863114
##  20:     2204.6325: 0.0447980 0.0150366 0.138753 1.00000e-08 0.864341
##  21:     2204.6140: 0.0442829 0.0150643 0.140317 1.00000e-08 0.863099
##  22:     2204.5933: 0.0421718 0.0150649 0.140743 1.00000e-08 0.863090
##  23:     2204.5928: 0.0415555 0.0146915 0.139703 1.00000e-08 0.863805
##  24:     2204.5865: 0.0412453 0.0148675 0.139754 1.00000e-08 0.863775
##  25:     2204.5843: 0.0409449 0.0150850 0.140663 1.00000e-08 0.862787
##  26:     2204.5837: 0.0406346 0.0150768 0.140519 1.00000e-08 0.862889
##  27:     2204.5835: 0.0403397 0.0150686 0.140418 1.00000e-08 0.862971
##  28:     2204.5835: 0.0403407 0.0150690 0.140417 1.00000e-08 0.862971
##
## Final Estimate of the Negative LLH:
##  LLH:  -3304.84    norm LLH:  -1.865034
##           mu        omega       alpha1       alpha2        beta1
## 1.800751e-03 3.002655e-05 1.404166e-01 1.000000e-08 8.629708e-01
##
## R-optimhess Difference Approximated Hessian Matrix:
##                  mu        omega        alpha1        alpha2        beta1
## mu     -1955061.275     -7703522      3821.572      2265.989     -2505.98
## omega  -7703521.553 -72520805870 -33003845.076 -33786162.572 -61330878.82
## alpha1     3821.572    -33003845    -27265.778    -26677.670    -39726.82
## alpha2     2265.989    -33786163    -26677.670    -27897.329    -41449.85
## beta1     -2505.980    -61330879    -39726.824    -41449.847    -67165.44
## attr(,"time")
## Time difference of 0.04689407 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.531867 secs
```

```
summary(m.11_2)
```

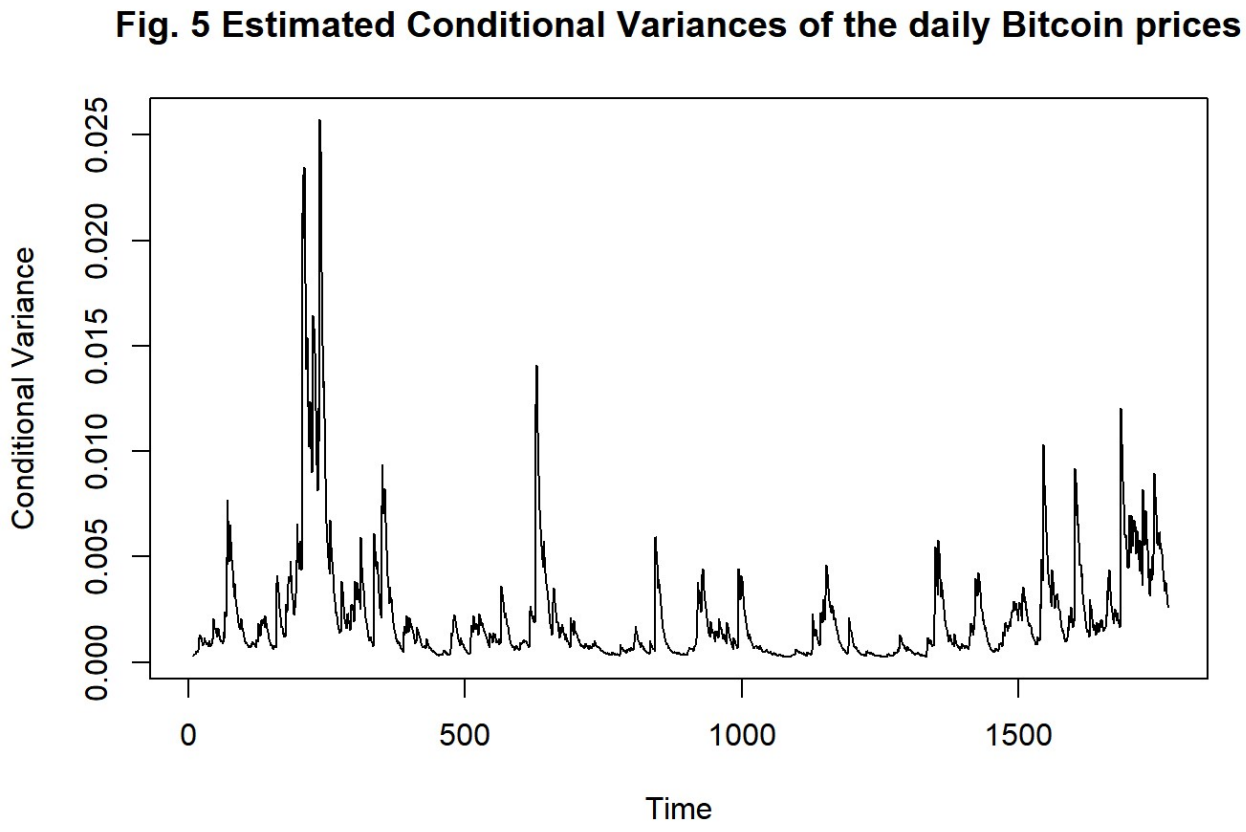
```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = model_212_mle$residuals)
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x0000000006e9a5b8>
##  [data = model_212_mle$residuals]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1       beta1
## 1.7967e-03  2.9781e-05  1.4000e-01  8.6354e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.797e-03   7.166e-04    2.507 0.012162 *
## omega  2.978e-05   8.477e-06    3.513 0.000443 ***
## alpha1 1.400e-01   1.788e-02    7.832 4.88e-15 ***
## beta1  8.635e-01   1.593e-02   54.214  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  3304.342    normalized:  1.864753
##
## Description:
##  Mon May 28 11:45:01 2018 by user: user
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  4948.229  0
##  Shapiro-Wilk Test  R    W      0.9096566 0
##  Ljung-Box Test     R    Q(10)  35.02078  0.0001238498
##  Ljung-Box Test     R    Q(15)  45.24199  7.012147e-05
##  Ljung-Box Test     R    Q(20)  57.10771  1.976797e-05
##  Ljung-Box Test     R^2  Q(10)  7.909834  0.6376439
##  Ljung-Box Test     R^2  Q(15)  11.36662  0.7261989
##  Ljung-Box Test     R^2  Q(20)  13.54638  0.8527311
```

```
##  LM Arch Test       R    TR^2   8.758876  0.723376
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.724991 -3.712621 -3.725001 -3.720421
```

The folowing code shows the within sample estimates of the conditional variances, which capture several periods of high volatility.

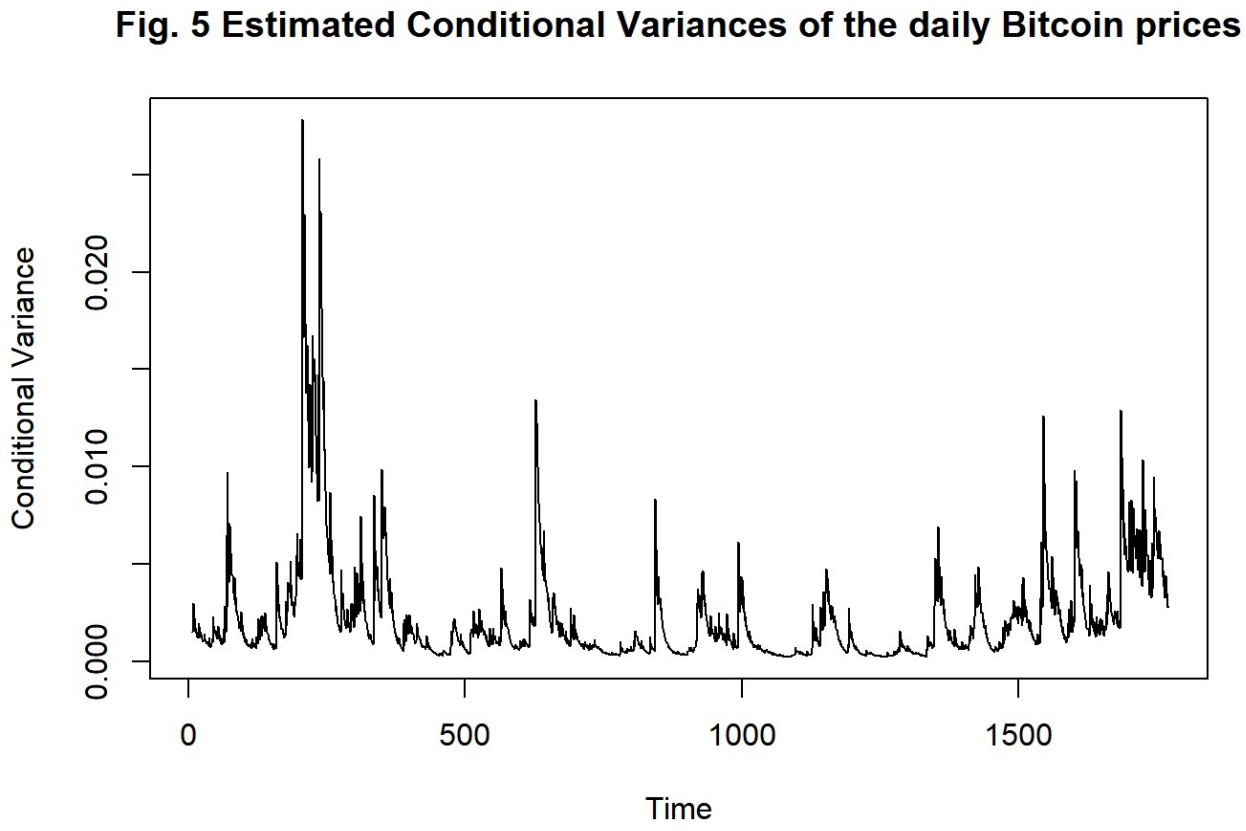
1. GARCH(1,1)

```
plot((fitted(m.11)[,1])^2, type='l', ylab='Conditional Variance', main="Fig. 5 Estimat
ed Conditional Variances of the daily Bitcoin prices")
```



2. GARCH(2,1)

```
plot((fitted(m.21)[,1])^2, type='l', ylab='Conditional Variance', main="Fig. 5 Estimat
ed Conditional Variances of the daily Bitcoin prices")
```

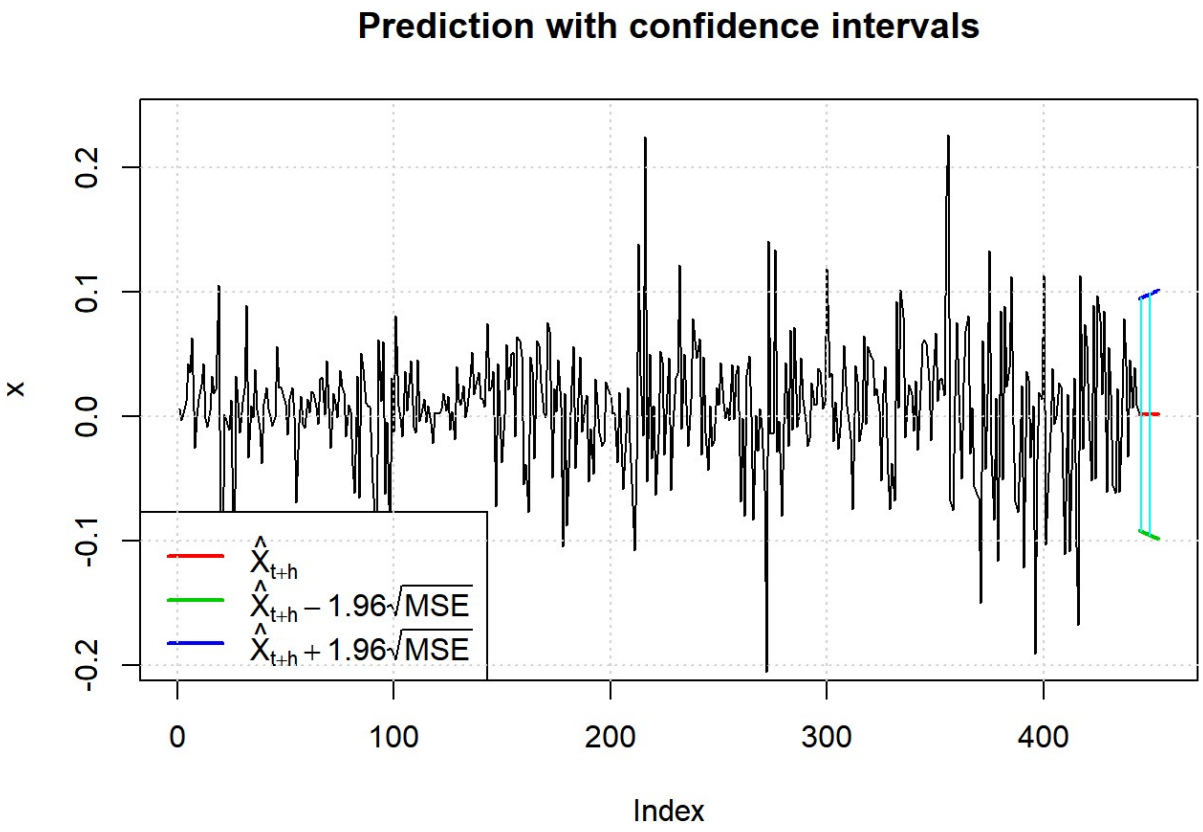


Changes in conditional variance occur at the beginning of the series and between observations 300 and 400, then the conditional variance settles down.

Forecasts for the confidance limits are based on the forecasts of conditional variance.

1. GARCH(1,1)

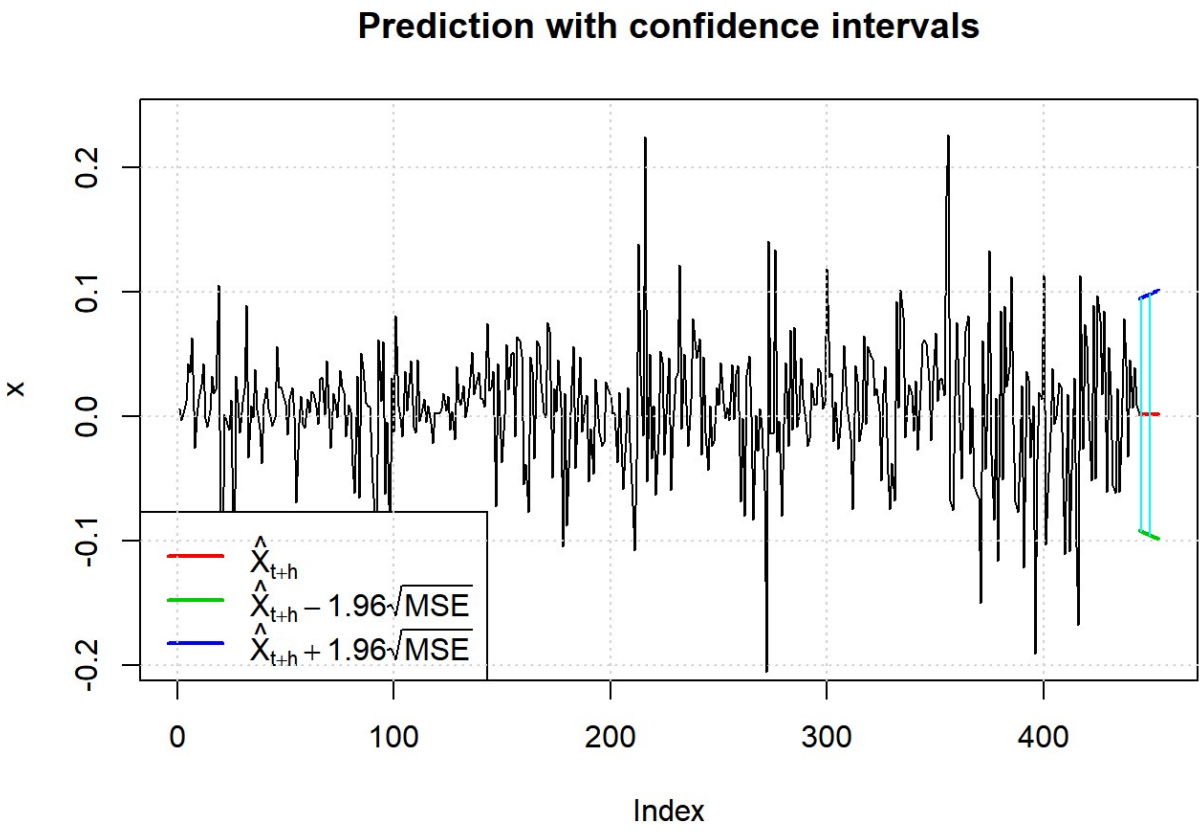
```
fGarch::predict(m.11_2,n.ahead=10,trace=FALSE,plot=TRUE)
```




```
##    meanForecast  meanError standardDeviation lowerInterval upperInterval
## 1   0.001796743 0.04758117        0.04758117   -0.09146065    0.09505413
## 2   0.001796743 0.04797660        0.04797660   -0.09223566    0.09582915
## 3   0.001796743 0.04837017        0.04837017   -0.09300705    0.09660054
## 4   0.001796743 0.04876194        0.04876194   -0.09377491    0.09736840
## 5   0.001796743 0.04915196        0.04915196   -0.09453933    0.09813281
## 6   0.001796743 0.04954027        0.04954027   -0.09530040    0.09889389
## 7   0.001796743 0.04992692        0.04992692   -0.09605821    0.09965170
## 8   0.001796743 0.05031194        0.05031194   -0.09681285    0.10040634
## 9   0.001796743 0.05069539        0.05069539   -0.09756439    0.10115788
## 10  0.001796743 0.05107730        0.05107730   -0.09831292    0.10190641
```

2. GARCH(2,1)

```
fGarch::predict(m.21_2,n.ahead=10,trace=FALSE,plot=TRUE)
```




```
##    meanForecast  meanError standardDeviation lowerInterval upperInterval
## 1   0.001800751 0.04749926        0.04749926   -0.09129608    0.09489759
## 2   0.001800751 0.04789414        0.04789414   -0.09207004    0.09567154
## 3   0.001800751 0.04828712        0.04828712   -0.09284026    0.09644176
## 4   0.001800751 0.04867823        0.04867823   -0.09360683    0.09720833
## 5   0.001800751 0.04906754        0.04906754   -0.09436986    0.09797137
## 6   0.001800751 0.04945509        0.04945509   -0.09512944    0.09873095
## 7   0.001800751 0.04984092        0.04984092   -0.09588566    0.09948716
## 8   0.001800751 0.05022508        0.05022508   -0.09663860    0.10024010
## 9   0.001800751 0.05060761        0.05060761   -0.09738834    0.10098985
## 10  0.001800751 0.05098855        0.05098855   -0.09813498    0.10173648
```

Creating a univariate GARCH specification object prior to fitting using ugarchspecmethod: It will first specify the single ARMA and ARCH/GARCH model & then fit that model on the given data.

1. With GARCH(1,1)

```
library(rugarch)
model<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                  mean.model = list(armaOrder = c(2, 2), include.mean = FALSE),
                  distribution.model = "norm")
m.11_3<-ugarchfit(spec=model,data=r.bhp)
```

And output of the model is:

```
m.11_3
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(2,0,2)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## ar1      0.71137    0.001439  494.1906 0.000000
## ar2     -0.99883    0.001195 -835.6406 0.000000
## ma1     -0.70753    0.001652 -428.1602 0.000000
## ma2      0.99585    0.000278 3582.2174 0.000000
## omega    0.33349    0.102688    3.2476 0.001164
## alpha1   0.14127    0.018397    7.6790 0.000000
## beta1    0.85773    0.018199   47.1300 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## ar1      0.71137    0.002044  348.02099 0.000000
## ar2     -0.99883    0.001790 -557.93807 0.000000
## ma1     -0.70753    0.002028 -348.86644 0.000000
## ma2      0.99585    0.000411 2424.29701 0.000000
## omega    0.33349    0.355450    0.93821 0.348138
## alpha1   0.14127    0.050639    2.78975 0.005275
## beta1    0.85773    0.060038   14.28655 0.000000
##
## LogLikelihood : -4847.813
##
## Information Criteria
## ------------------------------------
##
## Akaike       5.4826
## Bayes        5.5042
## Shibata      5.4825
## Hannan-Quinn 5.4906
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic   p-value
## Lag[1]                      6.773 9.257e-03
## Lag[2*(p+q)+(p+q)-1][11]   20.229 0.000e+00
## Lag[4*(p+q)+(p+q)-1][19]   28.923 2.175e-08
```

```
## d.o.f=4
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.7574  0.3841
## Lag[2*(p+q)+(p+q)-1][5]    1.4852  0.7435
## Lag[4*(p+q)+(p+q)-1][9]    2.2112  0.8783
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1151 0.500 2.000  0.7344
## ARCH Lag[5]    1.3369 1.440 1.667  0.6362
## ARCH Lag[7]    1.5865 2.315 1.543  0.8039
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.2876
## Individual Statistics:
## ar1    0.07239
## ar2    0.01344
## ma1    0.11315
## ma2    0.07438
## omega  0.25842
## alpha1 0.07192
## beta1  0.15488
##
## 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           0.9517 0.3414
## Negative Sign Bias  0.9617 0.3363
## Positive Sign Bias  0.7561 0.4497
## Joint Effect        1.5561 0.6694
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     265.5    2.226e-45
## 2    30     294.2    1.143e-45
## 3    40     300.2    4.957e-42
```

```
## 4    50     314.9    1.701e-40
##
##
## Elapsed time : 1.77136
```

2. With GARCH(2,1)

```
library(rugarch)
model<-ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(2,1)),
                  mean.model = list(armaOrder = c(2, 2), include.mean = FALSE),
                  distribution.model = "norm")
m.21_3<-ugarchfit(spec=model,data=r.bhp)
```

And output of the model is:

```
m.21_3
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(2,0,2)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1      0.72625    0.025914  28.0255 0.000000
## ar2      0.21038    0.029905   7.0348 0.000000
## ma1     -0.68020    0.011550 -58.8924 0.000000
## ma2     -0.23701    0.010598 -22.3629 0.000000
## omega    0.35234    0.137761   2.5576 0.010539
## alpha1   0.14280    0.023653   6.0372 0.000000
## alpha2   0.00000    0.040507   0.0000 1.000000
## beta1    0.85620    0.030852  27.7515 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## ar1      0.72625    0.027199  26.70095  0.00000
## ar2      0.21038    0.028253   7.44607  0.00000
## ma1     -0.68020    0.011003 -61.81691  0.00000
## ma2     -0.23701    0.009604 -24.67865  0.00000
## omega    0.35234    0.562284   0.62662  0.53091
## alpha1   0.14280    0.037820   3.77567  0.00016
## alpha2   0.00000    0.131763   0.00000  1.00000
## beta1    0.85620    0.126116   6.78900  0.00000
##
## LogLikelihood : -4854.958
##
## Information Criteria
## ------------------------------------
##
## Akaike       5.4918
## Bayes        5.5165
## Shibata      5.4917
## Hannan-Quinn 5.5009
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic   p-value
## Lag[1]                      1.644 1.997e-01
```

```
## Lag[2*(p+q)+(p+q)-1][11]    10.806 3.489e-11
## Lag[4*(p+q)+(p+q)-1][19]    17.343 4.424e-03
## d.o.f=4
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                       1.105  0.2932
## Lag[2*(p+q)+(p+q)-1][8]      2.541  0.7687
## Lag[4*(p+q)+(p+q)-1][14]     5.076  0.7600
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]     1.716 0.500 2.000  0.1902
## ARCH Lag[6]     1.840 1.461 1.711  0.5270
## ARCH Lag[8]     2.215 2.368 1.583  0.6980
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.5337
## Individual Statistics:
## ar1    0.02825
## ar2    0.03847
## ma1    0.02237
## ma2    0.03423
## omega  0.25008
## alpha1 0.07176
## alpha2 0.08361
## beta1  0.14821
##
## 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          1.13749 0.2555
## Negative Sign Bias 0.10863 0.9135
## Positive Sign Bias 0.05399 0.9569
## Joint Effect       2.21500 0.5290
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
```

```
## 1    20     311.9    7.309e-55
## 2    30     329.7    1.027e-52
## 3    40     346.8    5.299e-51
## 4    50     368.3    1.660e-50
##
##
## Elapsed time : 1.602647
```

We see that associated p values are all significant for the model. The table also shows the Robust standard errors and the criteria.

Further, now we will try to forecast 10 future values based on chosen model.

1. Forecast and MASE calculation using GARCH(1,1)

```
forc = ugarchforecast(m.11_3, data = r.bhp, n.ahead = 10)
#forc
forc@forecast$series
```

```
##       1772-01-01
## T+1   0.25722525
## T+2  -0.04432488
## T+3  -0.28845633
## T+4  -0.16092530
## T+5   0.17364266
## T+6   0.28426122
## T+7   0.02877418
## T+8  -0.26346046
## T+9  -0.21615779
## T+10  0.10938536
```

Now we will compare the forecasted values with original values using MASE.

```
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))
}
```

Calculating MASE for forecasted data

```
rev.100 <- as.vector(forc@forecast$seriesFor/100) # divided by 100 to revert the
class(rev.100)
```

```
## [1] "numeric"
```

```
dl <- as.vector(c(11512.60,
                  11573.30,
                  10779.90,
                  9965.57,
                  9395.01,
                  9337.55,
                  8866.00,
                  9578.63,
                  9205.12,
                  9194.85))


rev.diff = diffinv(rev.100, differences = 1, xi = bhp.log[length(bhp.log)])
rev.final = exp(rev.diff)

MASE(dl,rev.final[2:11])
```

```
## $MASE
##       MASE
## 1 3.682879
```

Calculating MASE for fitted data:

```
rev.fs <- as.vector(fitted(m.11_3)/100)
rev.fs.diff <- diffinv(rev.fs, differences = 1, xi = log(head(bhp.ts,1)[1]))
rev.fs.diff.exp <- exp(rev.fs.diff)
MASE(bhp.ts, rev.fs.diff.exp)
```

```
## $MASE
##       MASE
## 1 21.73843
```

2. Forecast and MASE calculation with GARCH(2,1)

```
forc2 = ugarchforecast(m.21_3, data = r.bhp, n.ahead = 10)
forc2@forecast$series
```

```
##      1772-01-01
## T+1  0.12405657
## T+2  0.14946893
## T+3  0.13465040
## T+4  0.12923459
## T+5  0.12218389
## T+6  0.11592396
## T+7  0.10989439
## T+8  0.10419846
## T+9  0.09879332
## T+10 0.09366955
```

Now we will compare the forecasted values with original values using MASE.

Calculation of MASE for forecasted values.

```
rev.100 <- as.vector(forc2@forecast$seriesFor/100) # divided by 100 to revert the
class(rev.100)
```

```
## [1] "numeric"
```

```
dl <- as.vector(c(11512.60,
                  11573.30,
                  10779.90,
                  9965.57,
                  9395.01,
                  9337.55,
                  8866.00,
                  9578.63,
                  9205.12,
                  9194.85))


rev.diff = diffinv(rev.100, differences = 1, xi = bhp.log[length(bhp.log)])
rev.final = exp(rev.diff)

MASE(dl,rev.final[2:11])
```

```
## $MASE
##       MASE
## 1 3.860538
```

Calculating MASE for fitted data:

```
rev.fs <- as.vector(fitted(m.21_3)/100)
rev.fs.diff <- diffinv(rev.fs, differences = 1, xi = log(head(bhp.ts,1)[1]))
rev.fs.diff.exp <- exp(rev.fs.diff)
MASE(bhp.ts, rev.fs.diff.exp)
```

```
## $MASE
##       MASE
## 1 20.85628
```

Conclusion:

The best fit model which captures both the variance and the mean of the series to predict the future values with lesser error rate is GARCH(1,1) ARMA(2,2) model. And on the other side GARCH(2,1) ARMA(2,2) model is better for fitted values. But since GARCH(1,1) ARMA(2,2) is more generalized model and more accurate in predicting unknown values, So we will claim this to be the best model. MASE (Mean Absolute Scaled Error) for these two models is as follows (both for fitted and forecast data).

(ARMA(2,2) GARCH(1,1)

Over fitted values —-> 21.73

Over forecasts —-> 3.68

ARMA(2,2) GARCH(2,1)

Over fitted values —-> 20.86

Over forecasts —-> 3.86

Model ARMA(2,2) GARCH(1,1) is best for forecasting.

Summary:

In this task we dealt with the dataset concerning Bitcoin Historical price data. On visualizing the data we observed that there is trend and changing variance as the main features of the dataset. So we took log and differencing of the data to attain the stationarity. And then we applied various ARIMA models on the data to capture the Auto Regressive and Moving Average behaviour of the series. The best model came to be ARIMA(2,1,2). Though the ACF and PACF plot for the residuals of the model reflected white noise but on using McLeod-Li test and plotting ACF and PACF of square and absolute values of the residuals it was clear that there is ARCH effect in the series. So using EACF of squared and absolute data values of the residuals we specifi models to capture ARCH/GARCH effect. The best model came out to be GARCH(1,1). Further using ugarch method we combined ARIMA and GARCH model as one model and used it to forecast 10 future values of Bitcoin prices. We compared these forecasted values with the actual values and calculated MASE. MASE was also calculated for fitted values. The results are included in the conclusion.

References:

https://www.kaggle.com/tejalnarkar/time-series-analysis-of-bitcoin
(https://www.kaggle.com/tejalnarkar/time-series-analysis-of-bitcoin)

https://stackoverflow.com/ (https://stackoverflow.com/)