Study of VIX Index Forecasts

Minyang Xu

1 Abstract

Objective: Since the VIX Index acts as an important role in the measurement of market for investor. Some investors failed to make right decisions because of misunderstanding of VIX Index. Therefore, construct an accurate forecasting model for VIX Index can save many investors from failing of investment. The study aims to evaluate an appropriate and effective model to predict the future VIX Index. Methods: The forecasts processes will be based on the data of VIX Index from 1990-02-01 to 2021-12-13. The forecast model will be constructed by the Box Jenkins method. Data Preprocessing: Training dataset is the VIX Index from 1990-12-01 to 2018-11-01 which is the 90% of the initial dataset. Test dataset is the VIX Index from 2018-12-01 to 2021-12-13. Model Specification: The analysis will have time plot, mean-level plot, ACF plot, PACF plot, ADF and KPSS test to find an appropriate ARMA model. Then we use the model to forecasts the tail of initial dataset, and compared to the test dataset. Finally predict the 1-step ahead. Results: It is useful to transform the series by logarithm transform and one step difference. The final model from Box Jenkings method is ARIMA(1, 1, 2). Conclusions: We can use Box Jenkings method to forecasts the value of VIX Index, but the unexpected events can largely influence the results.

2 Introduction

The study of VIX Index Forecasts aims to build an appropriate time series model to forecast the VIX Index in 2022 based on the data of the VIX Index from 1990 to 2021. VIX Index is an essential concept of the stock market's expectation of volatility based on S&P 500 index options. For investors, VIX Index provides a market risk, volatility, and strategy measurement[6]. (1989, Brener) Generally speaking, a VIX Index higher than 40 will be considered high volatility, usually accompanied by market fear. The VIX Index lower than 15 will be considered low volatility. Forecasting the VIX Index can help the investor study the stock market and provide an apparent measurement of the market, and the investor can make different decisions based on the forecasts.

It is impossible to forecast the monthly VIX Index based on the historical VIX Index. The VIX Index is not an independent variable, and the VIX Index is influenced by various factors, such as wars, strikes, crises, and other unexpected events. Based on the arrangements of each segment, once an event happens, the VIX Index will not only increase but also fluctuate sharply. For example, segment 4 indicates the VIX Index from 2003-01 to 2007-09. The whole segment four only fluctuated at a steady level. Imagine 2007-09 now, and scientists aim to forecast the VIX Index of 2007-10, 2007-11, and 2007-12. The scientists fit the model by following the time series analysis steps and getting the three values of the VIX Index. However, a crisis happened unexpectedly. The market fear overgrows, and the VIX Index increases rapidly. The example tells us that the VIX Index has uncontrollable decisive factors. Another example is the covid segment which is happening right now; no one can forecast what time the next infection wave will come. Therefore, the monthly VIX Index will become even more unpredictable.[2] (2019, Ytableau)

To simplify the analysis, we have to omit the uncontrollable factors. Based on the dataset, we have two ways to analyze the dataset. The first one is to consider the VIX Index from 1990 to 2021; the second is to consider the VIX Index as six different segments. (2021, Prof Masoud) In the following study, we will investigate which one is more appropriate to forecast the VIX Index in the future. However, if we omit the additional factors, then it means that the analysis may be somewhat remote from reality. The analysis will

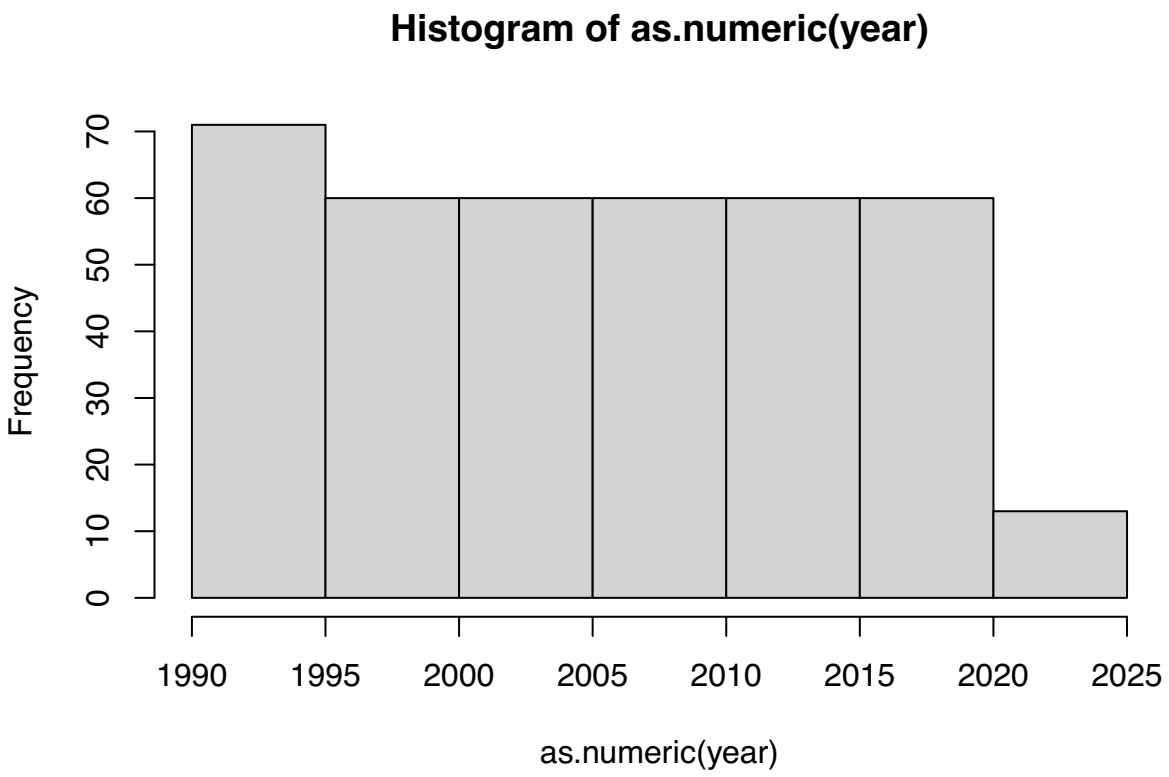
be idealistic, which is not a good idea to be a measurement of investment. In the following study, we will process the Box Jenkins Method for the historical VIX Index and the historical VIX Index in 6 segments, respectively.

```
Data = read.csv("./Data.csv")
attach(Data)
```

Variable Analysis

Date

```
Data$Date = as.Date(Data$Date, format="%Y-%m-%d") # formalize the date
year = format(Data[, 'Date'], "%Y")
month = format(Data[, 'Date'], "%m")
day = format(Data[, 'Date'], "%d")
hist(as.numeric(year))
```




```
summary(Data$Date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "1990-02-01" "1998-01-24" "2006-01-16" "2006-01-15" "2014-01-08" "2021-12-13"
```

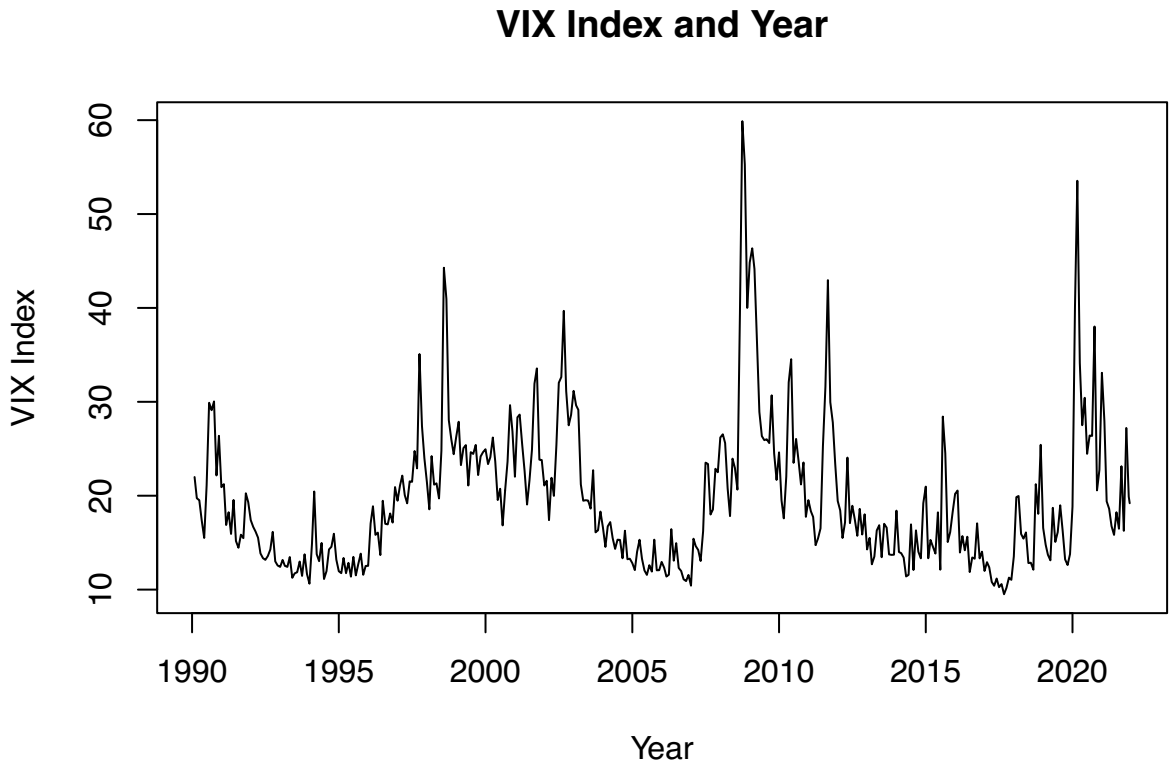
The data of VIX Index is from 1990-02-01 to 2021-12-13. Every year almost have equal number of data. There are 12 data for each year, from January to December respectively.

VIX Index

```
summary(Data$VIX)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.51   13.77   17.75   19.57   23.50   59.89
```

```r
plot(Data$Date, Data$VIX, main = "VIX Index and Year",
     xlab = "Year",
     ylab = "VIX Index",
     type = "l")
```



The minimum VIX Index is 9.51 and the maximum is 59.89 which is in absolutely market fear.

3 Box Jenkings Method

Data Preprocessing

The VIX Index data has two variables. The first variable is date, and the second variable is VIX. We should formalize the date variable first. The data variable is from 1990-02 to 2021-12-13. The VIX index is collected once a month. The VIX variable is a numeric variable. It directly shows the VIX Index in each month. The next step of data preprocessing is split the dataset into two parts, training dataset and test dataset. The training dataset is from 1990-02 to 2018-11. The test dataset is from 2018-12 to 2021-12-13.

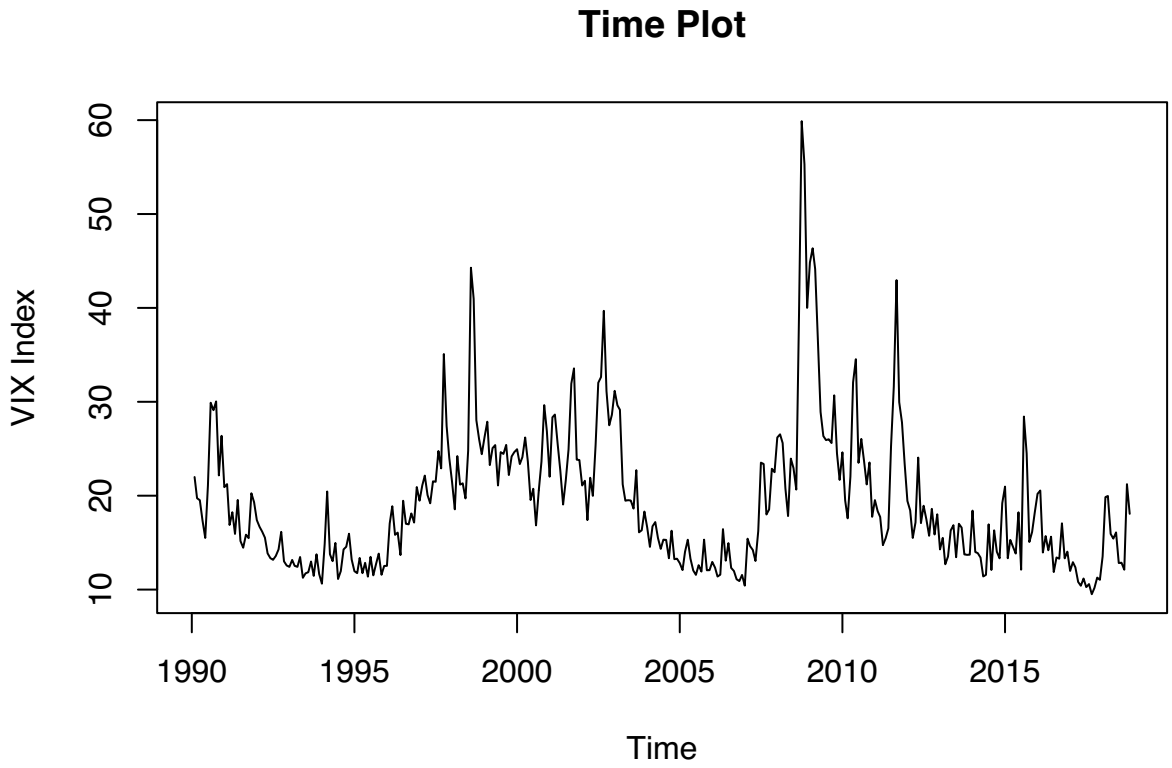
```r
# Split the dataset
# 90% training data
training_Date = Data$Date[1:round(length(Data$Date)*0.9, 0)]
training_Vix = Data$VIX[1:round(length(Data$Date)*0.9, 0)]
training = data.frame(date = training_Date, VIX = training_Vix) # Select the training dataset
x = training$VIX
test = Data[-c(1:round(length(Data$Date)*0.9)),]
```

Model Specification

Based on the time series plot, acf plot, pacf plot. The time series does not have any trend. The mean level plot does not show constant mean level. The next step is unit root / non-stationary, stationary test. The non-stationary test has p-value > 0.05, so we fail to reject the non-stationary null hypothesis. The level

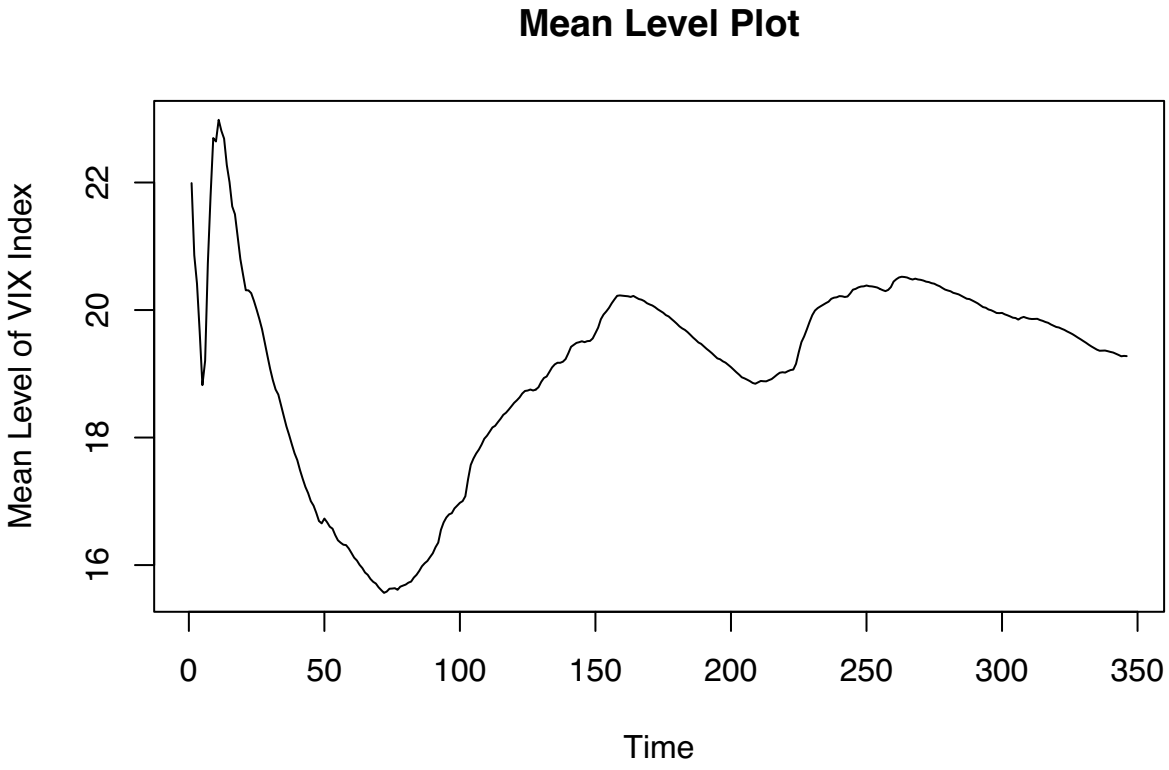
stationary test is also concluded. Although we reject the null hypothesis of trend stationary test, we still derive a contradiction. To sum up, we conclude that the time series is non-stationary.

```
# time plot
plot(training, type = "l", main = "Time Plot", xlab = "Time", ylab = "VIX Index")
```



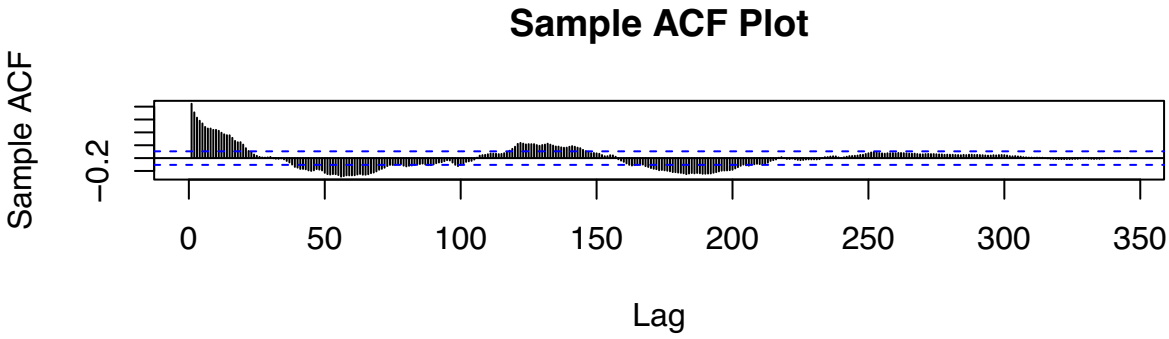
The time series plot does not have trend.

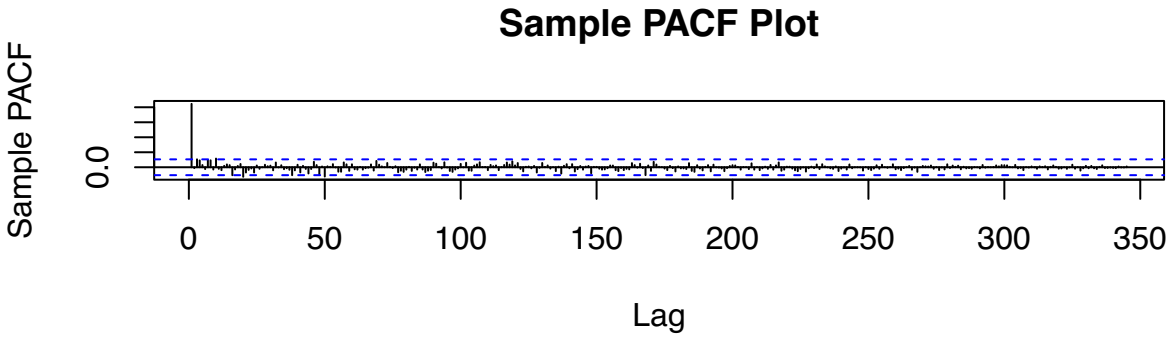
```
cummeanx = cumsum(x) / seq_along(x)
plot(cummeanx, type = "l", xlab = "Time", ylab = "Mean Level of VIX Index", main = "Mean Level Plot") #
```



The mean level plot has more than one mean levels. We have to defference the model.

```
par(mfrow=c(2,1))
acf(x, xlab = "Lag", ylab = "Sample ACF", main = "Sample ACF Plot", lag.max = length(Data$VIX)) # Sampl
acf(x, type = "partial", xlab = "Lag", ylab = "Sample PACF", main = "Sample PACF Plot", lag.max = lengt
```





The ACF plot shows that lags before 20 are all significant, and keep declining until the end of the series. For

PACF plot, only lag 1 is significant. The rest are all insignificant.

```
adf.test(x, alternative = "stationary") # adf test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  x
## Dickey-Fuller = -2.981, Lag order = 7, p-value = 0.1629
## alternative hypothesis: stationary
```

```
kpss.test(x, null = "Level")
```

```
## Warning in kpss.test(x, null = "Level"): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  x
## KPSS Level = 0.33347, Truncation lag parameter = 5, p-value = 0.1
```

```
kpss.test(x, null = "Trend") # kpss stationary test
```

```
## Warning in kpss.test(x, null = "Trend"): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  x
## KPSS Trend = 0.32861, Truncation lag parameter = 5, p-value = 0.01
```

The non stationary test (adf test) has p-value = 0.1629, which derive the result that fail to reject the null hypothesis. We conclude that the series is non-stationary. The level, trend stationary test (KPSS test) has p-value = 0.1, 0.01 respectively. We have a contradicting result. Once we find the time series is non stationary. We have to proceed some transformations.

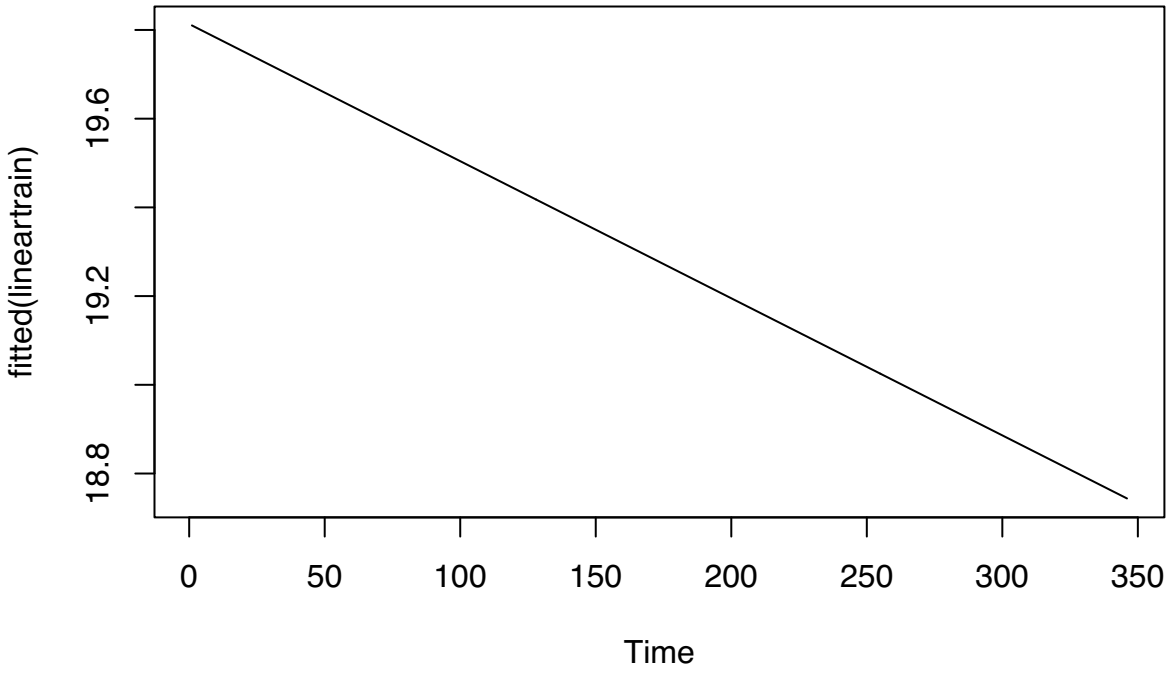
```
lineartrain = lm(training$VIX~time(training$VIX), na.action = NULL)
summary(lineartrain)
```

Linear regression

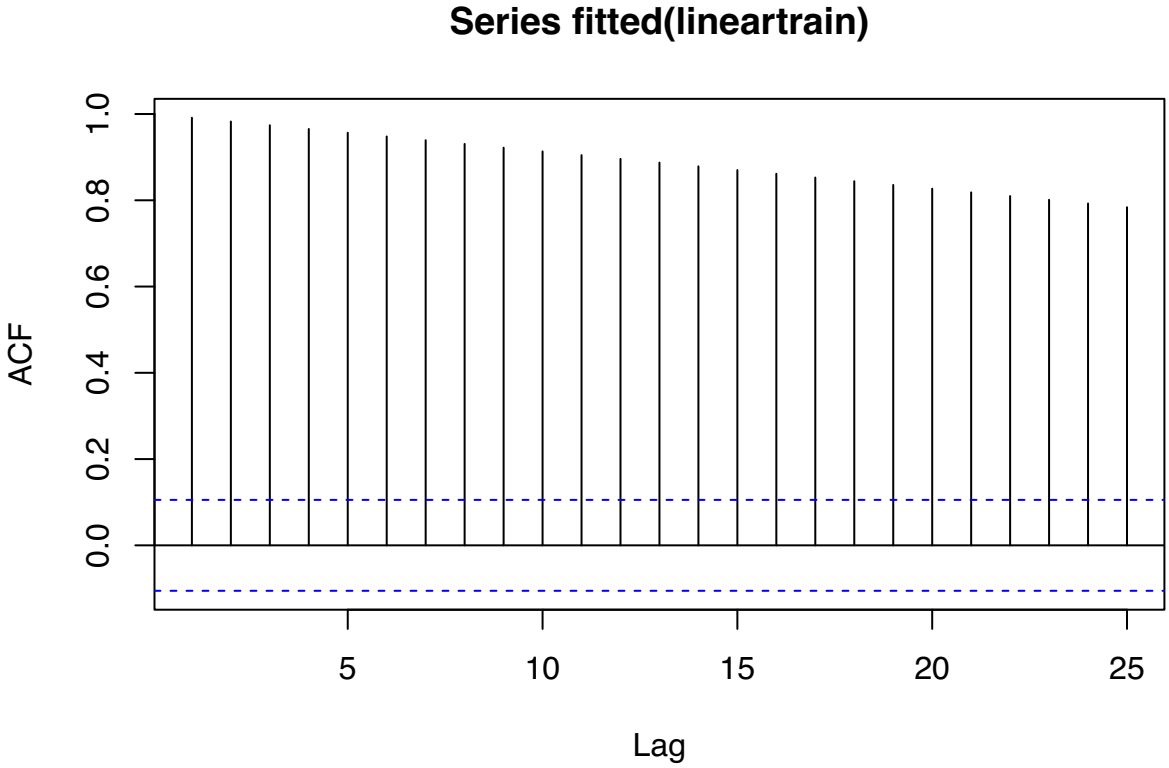
```
##
## Call:
## lm(formula = training$VIX ~ time(training$VIX), na.action = NULL)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.277 -5.501 -1.870  3.901 40.772
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         19.813296   0.802907  24.677   <2e-16 ***
## time(training$VIX) -0.003091   0.004011  -0.771    0.441
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.451 on 344 degrees of freedom
```

```
## Multiple R-squared:  0.001724,   Adjusted R-squared:  -0.001178
## F-statistic: 0.594 on 1 and 344 DF,  p-value: 0.4414
```

```
ts.plot(fitted(lineartrain))
```




```
acf(fitted(lineartrain))
```




```
adf.test(fitted(lineartrain))
```

```
## Warning in adf.test(fitted(lineartrain)): p-value greater than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  fitted(lineartrain)
## Dickey-Fuller = 0.050173, Lag order = 7, p-value = 0.99
## alternative hypothesis: stationary
```

```
kpss.test(fitted(lineartrain), null = 'Level')
```

```
## Warning in kpss.test(fitted(lineartrain), null = "Level"): p-value smaller than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  fitted(lineartrain)
## KPSS Level = 5.8656, Truncation lag parameter = 5, p-value = 0.01
```

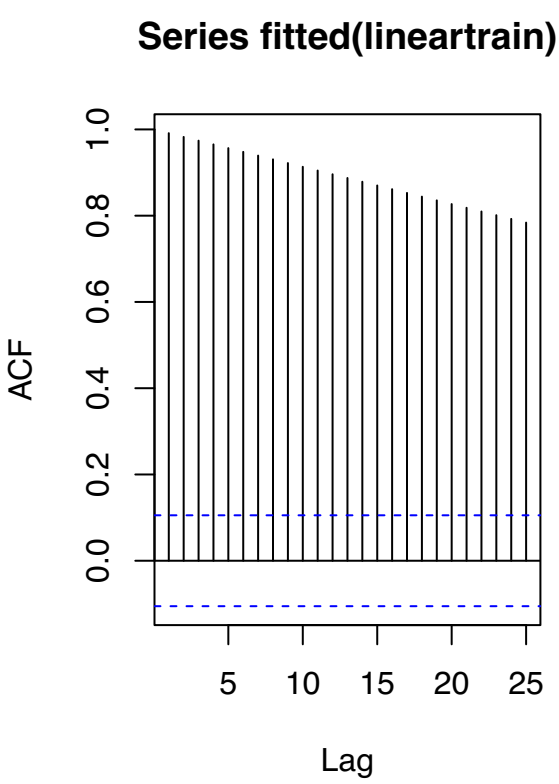
```
kpss.test(fitted(lineartrain), null = 'Trend')
```

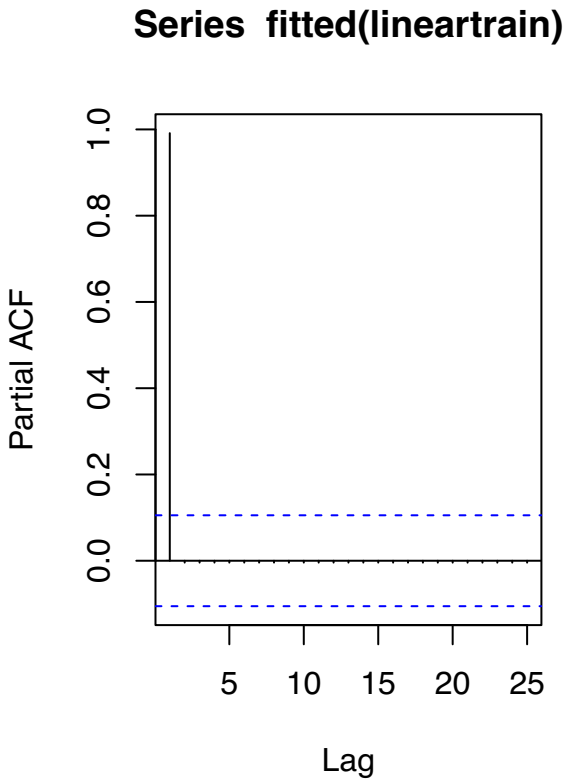
```
## Warning in kpss.test(fitted(lineartrain), null = "Trend"): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  fitted(lineartrain)
## KPSS Trend = 0.08066, Truncation lag parameter = 5, p-value = 0.1
```

The R-square is very small, which implies that the VIX Index is not explained well in the linear regression model. The p-value is $0.441 > 0.05$, so it is not significant.

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






```
adf.test(fitted(lineartrain))
```

```
## Warning in adf.test(fitted(lineartrain)): p-value greater than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  fitted(lineartrain)
## Dickey-Fuller = 0.050173, Lag order = 7, p-value = 0.99
## alternative hypothesis: stationary
```

```
kpss.test(fitted(lineartrain), null = 'Level')
```

```
## Warning in kpss.test(fitted(lineartrain), null = "Level"): p-value smaller than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  fitted(lineartrain)
## KPSS Level = 5.8656, Truncation lag parameter = 5, p-value = 0.01
```

```
kpss.test(fitted(lineartrain), null = 'Trend')
```

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

```
##
##  KPSS Test for Trend Stationarity
##
## data:  fitted(lineartrain)
## KPSS Trend = 0.08066, Truncation lag parameter = 5, p-value = 0.1
```

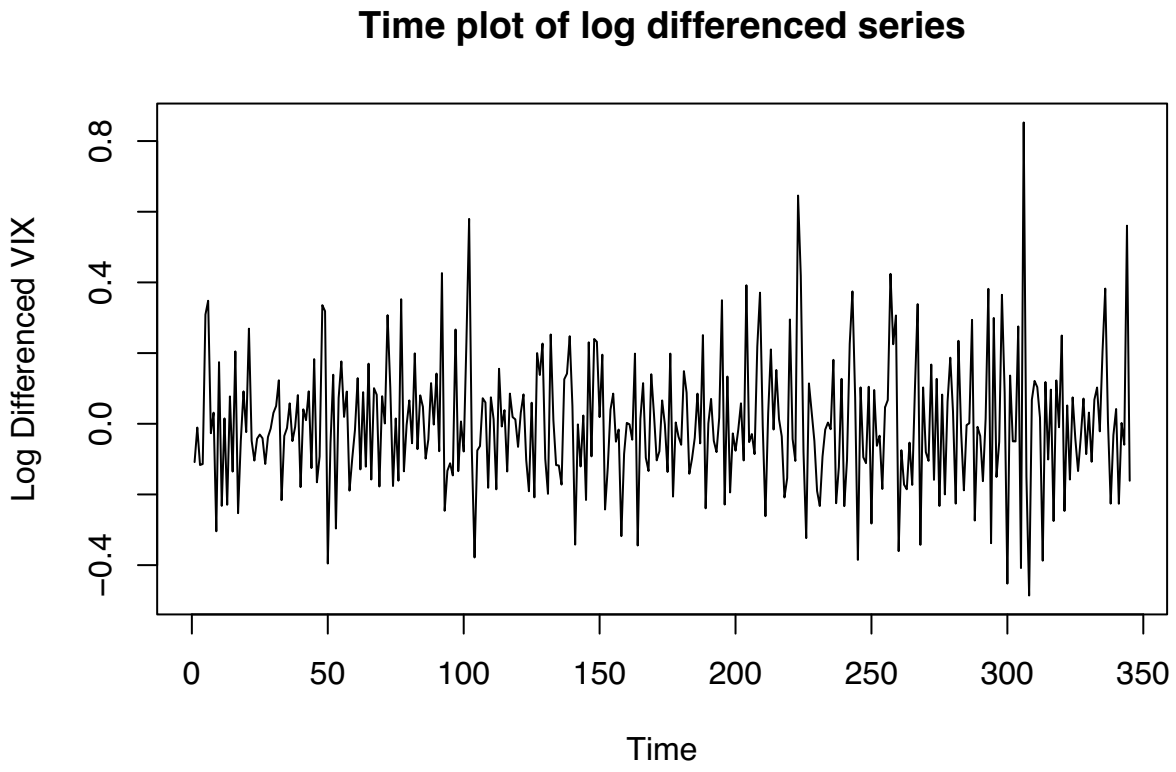
The result of non-stationary test has p-value = 0.99, which implies that the model is not stationary.

Difference and Logarithm Transformation4. Based on the original time series analysis. We have already concluded that the time series is non-stationary. Therefore, we should provide transformations to make the time series stationary. Therefore, we provide a difference transformation after logarithm transformation. The time series plot has more randomization patterns. The mean level plot shows a relatively constant mean. The unit root / non-stationary test has a p-value = 0.01, so we reject the null hypothesis that the time series is stationary. The level stationary and trend stationary tests have p-value = 0.1, so we conclude the null hypothesis that the time series is level stationery and trend stationery. The Shapiro test shows a p-value < 0.05, so we reject the null hypothesis and conclude that the time does not have normality. The next step of the model specification is to derive the candidate models. The 5 5 grid shows the feasible ARMA model for the stationary time series. The smallest AIC and BIC values are primarily considered appropriate models since the PACF plot has already indicated that more MA terms than AR terms will appear. Based on the standards above, the candidate models are ARMA(2, 3), ARMA(2, 4), ARMA(4, 5), ARMA(2, 2), ARMA(1, 4), ARMA(2, 3), ARMA(1, 5), and ARMA(1, 2).

```
y = diff(log(x)) # Since it is not stationary, so we apply transformations
```

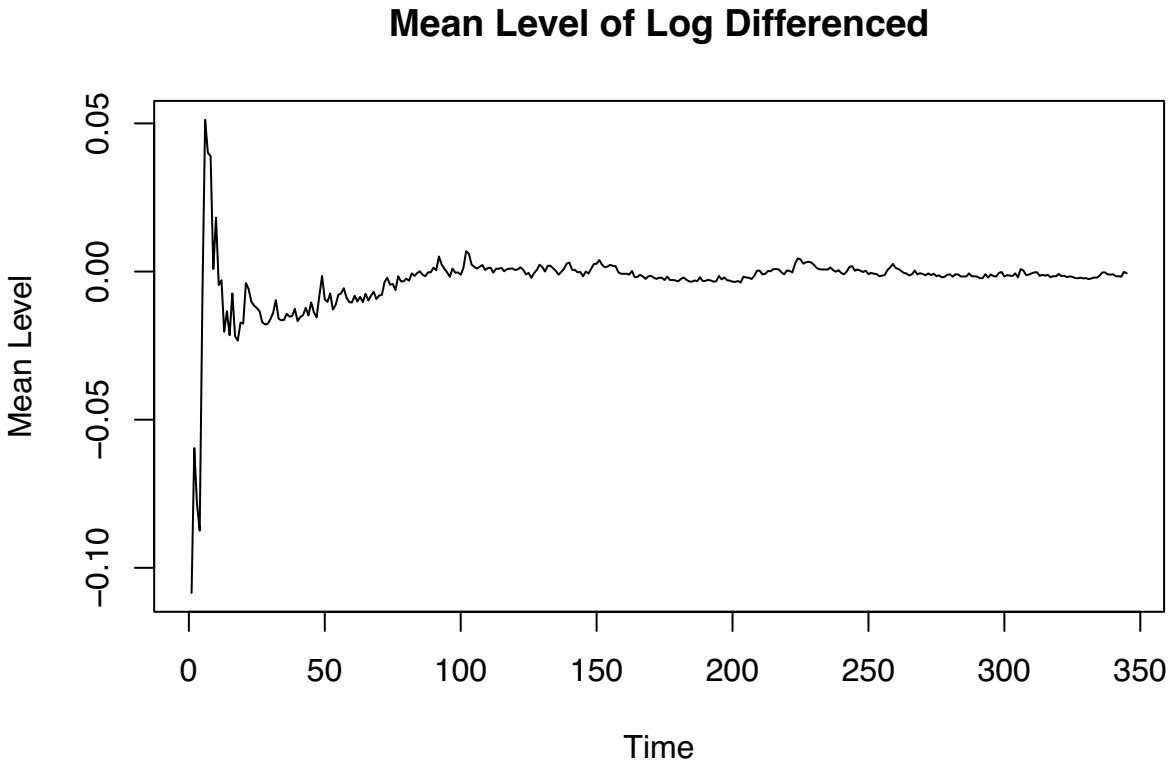
We use logarithm transformation first, then one step differenced time series to transform.

```
plot(y, type = "l", xlab = "Time", ylab = "Log Differenced VIX", main = "Time plot of log differenced s
```



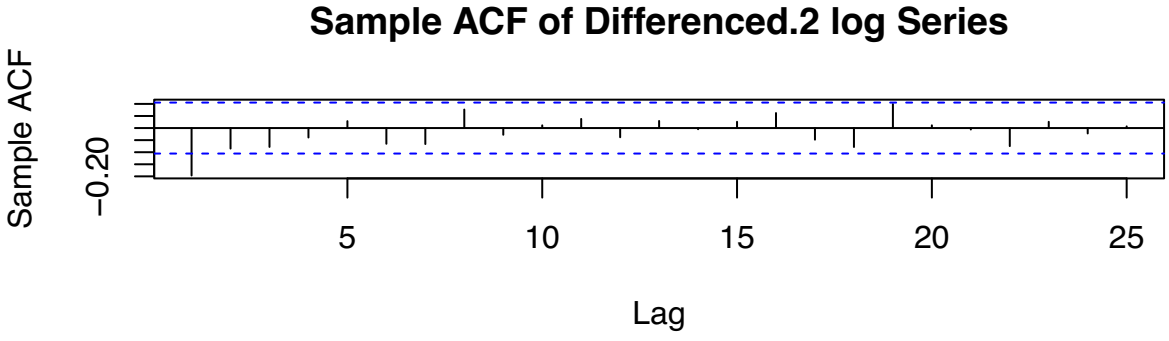
The time series plot has more randomness compared to the original time series, and fluctuate around 0. To prove the time series is valid, we should proceed the same steps as above.

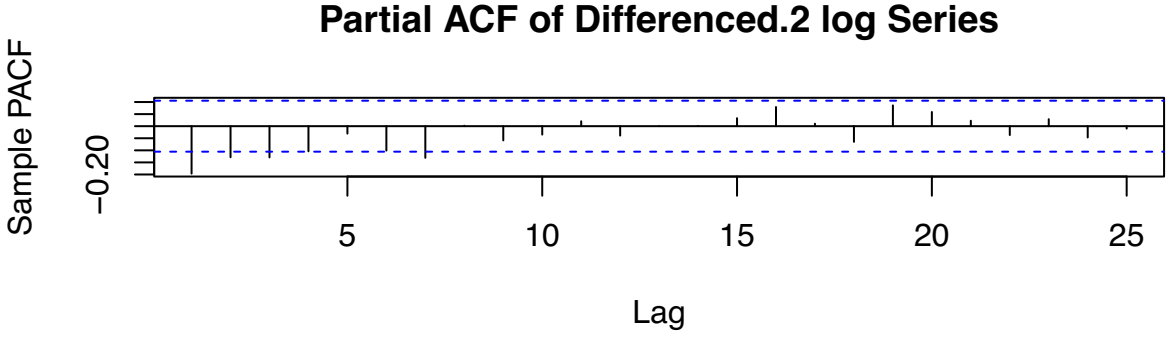
```
cummeany = cumsum(y) / seq_along(y)
plot(cummeany, type = "l", xlab = "Time", ylab = "Mean Level", main = "Mean Level of Log Differenced")
```



The mean level plot has only one constant mean. We still need more evidence to show the time series is stationary.

```
par(mfrow=c(2,1))
acf(y, xlab = "Lag", ylab = "Sample ACF", main = "Sample ACF of Differenced.2 log Series")
acf(y, xlab = "Lag", ylab = "Sample PACF", type = "partial", main = "Partial ACF of Differenced.2 log Se
```





The ACF plot shows only one significant at lag 1. The rest of lags are all insignificant. The PACF plot shows significant at lags 1, 2, 3, 7. More significant lags in PACF plot means more MA models will be have in the final model.

```
adf.test(y, alternative = "stationary") # adf test
```

```
## Warning in adf.test(y, alternative = "stationary"): p-value smaller than printed
## p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y
## Dickey-Fuller = -8.7397, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

The non-stationary test has p-value = 0.01 which means we should reject the null hypothesis. We conclude that the time series is not non-stationary.

```
kpss.test(y, null = "Level")
```

```
## Warning in kpss.test(y, null = "Level"): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  y
## KPSS Level = 0.023119, Truncation lag parameter = 5, p-value = 0.1
```

```
kpss.test(y, null = "Trend") # kpss stationary test
```

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

```
##
##  KPSS Test for Trend Stationarity
##
## data:  y
## KPSS Trend = 0.023672, Truncation lag parameter = 5, p-value = 0.1
```

The level and stationary test both have p-value = 0.1, which means we fail to reject the null hypothesis. We conclude that the time series is stationary.

```
shapiro.test(as.vector(y))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  as.vector(y)
## W = 0.97826, p-value = 4.501e-05
```

The normality test has p-value = 3.944e-05, which is smaller than 0.05. We fail to reject the null hypothesis. We conclude that the time series is not normal. We have already successfully transformed the original non-stationary time series to stationary time series. We are able to move on to the next stage: computing EACF and deciding possible ARMA models.

```
eacf(y, ar.max = 5, ma.max = 5) # compute the EACF
```

```
## AR/MA
##   0 1 2 3 4 5
## 0 x o o o o o
## 1 x o o o o o
```

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

By following the steps of Box Jenkings method, we should compute the AIC and BIC in increasing order and choose the most possible models. For this study, we will be choosing 8 possible models.

```
aic = matrix(0, 5, 5)
bic = matrix(0, 5, 5)
for (i in 0:4) for (j in 0:4){
  fit = arima(y, order = c(i, 0, j), method = "ML", include.mean = TRUE)
  aic[i+1, j+1] = fit$aic
  bic[i+1, j+1] = BIC(fit)
}
```

```
## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

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

```
aic_vec = sort(unmatrix(aic, byrow = FALSE))[1:13]
bic_vec = sort(unmatrix(bic, byrow = FALSE))[1:13]
aic_vec # lowest one is 4, 4
```

```
##     r2:c2     r4:c3     r5:c4     r4:c5     r3:c2     r2:c3     r4:c4     r1:c4
## -216.5675 -215.1461 -214.8761 -214.8759 -214.5688 -214.5687 -213.9485 -213.7762
##     r5:c5     r4:c2     r2:c4     r1:c5     r3:c3
## -213.7586 -212.9453 -212.9446 -212.9373 -212.5926
```

```
bic_vec # lowest one is 2, 2
```

```
##     r2:c2     r3:c2     r2:c3     r1:c4     r1:c3     r1:c2     r4:c2     r2:c4
## -199.1933 -193.3511 -193.3509 -192.5584 -192.5384 -189.0963 -187.8841 -187.8834
##     r1:c5     r3:c3     r4:c3     r2:c1     r4:c1
## -187.8760 -187.5313 -186.2413 -183.9368 -183.9179
```

In the previous analysis, the PACF plot has already declared that there will be more MA terms than AR terms. Therefore, the models that have more AR terms than MA terms will be canceled. Based on the AIC and BIC lists, choose the models from left to right. The results show that ARMA(2, 3), ARMA(2, 4), ARMA(4, 5), ARMA(2, 2), ARMA(1, 4), ARMA(2, 3), ARMA(1, 5), and ARMA(1, 2) are possible models for the study. In the following analysis, we will be fitting the possible models until finding the most appropriate and accurate model.

4 Fitting and Diagnostics

We will fit the candidate models to the transformed time series first. To find the most appropriate and accurate model, we should show the residuals plot, Ljung-Box test, qq plot, and normality test since the residuals of all models do not have a normal distribution. To measure the normality better, we can use the Jarque-Bera test to test the normality.[3] (2020, juliastats) The test designed to those properties: t is large if and only in skewness s or kurtosis, or both, are large. (2014, Willi) However, the result of the Jarque-Bera test is the same as the Shapiro test. The confidence interval of the parameter estimation is under the parameter estimation. The analysis below shows that all candidate models have uncorrelated residual terms

but do not have a normal distribution. Therefore, we cannot remove any possible models with residuals that do not follow a white noise. We can select the most appropriate one by observing the value of AIC, BIC, and harmonic mean. Based on the value of AIC, BIC, and harmonic mean, model 8 has the smallest AIC and BIC, and the harmonic mean is not high. Generally, the most appropriate model is ARIMA(1, 1, 2).

ARIMA(2, 0, 3)

```
fit_1 = arima(y, order = c(2, 0, 3), method = "ML", include.mean = TRUE) # fit the model
fit_1
```

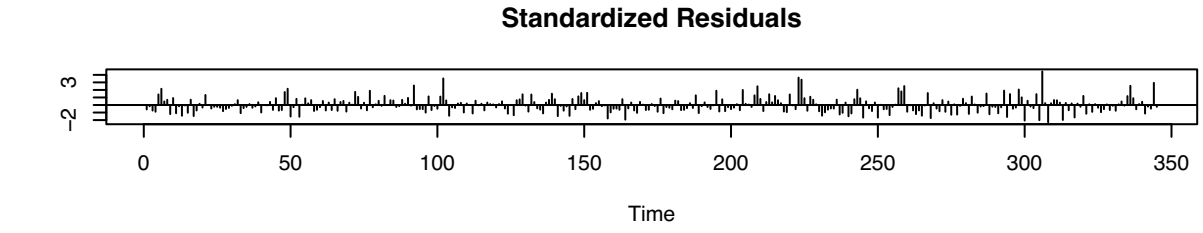
```
##
## Call:
## arima(x = y, order = c(2, 0, 3), include.mean = TRUE, method = "ML")
##
## Coefficients:
##          ar1     ar2      ma1      ma2      ma3  intercept
##       0.4006  -0.010  -0.6748  -0.0134  -0.0611    -0.0008
## s.e.  0.9543   0.502   0.9530   0.7487   0.1016     0.0039
##
## sigma^2 estimated as 0.03066:  log likelihood = 111.47,  aic = -210.95
```

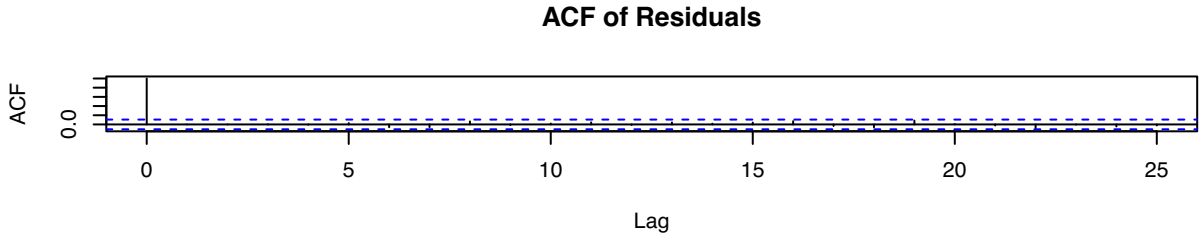
```
confint(fit_1)
```

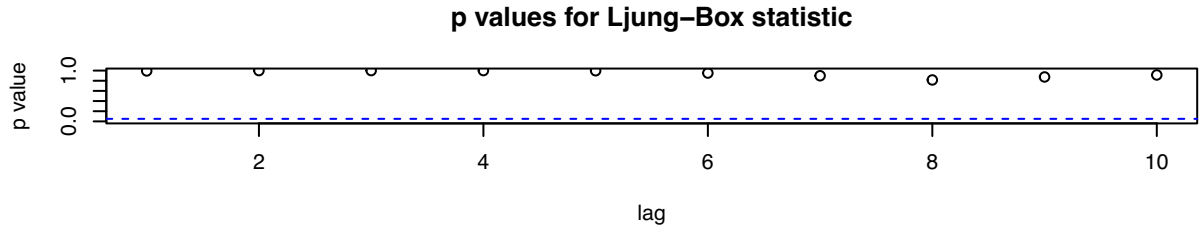
```
##                  2.5 %      97.5 %
## ar1       -1.469806233 2.271081228
## ar2       -0.993797340 0.973831083
## ma1       -2.542499116 1.192999033
## ma2       -1.480877704 1.454101005
## ma3       -0.260357323 0.138083868
## intercept -0.008445159 0.006894374
```

Residual analysis:

```
tsdiag(fit_1)
```





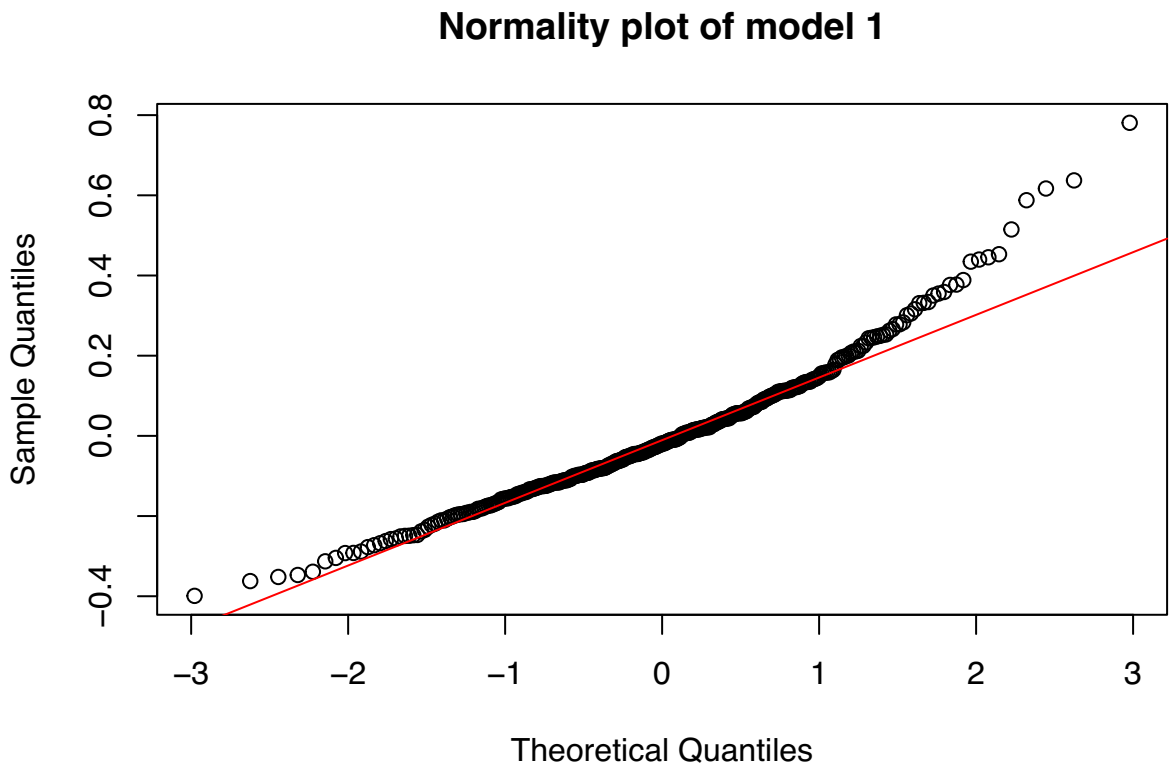



```
Box.test(fit_1$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_1$residuals
## X-squared = 6.8555e-05, df = 1, p-value = 0.9934
```

The residuals plot and Ljung-Box test show that residuals are mostly in randomness. The p-values for Ljung-Box Test are all insignificant where we fail to reject the null hypothesis. The result conclude that residuals are independently residuals for all lags. Normality:

```
qqnorm(fit_1$residuals, main = "Normality plot of model 1")
qqline(fit_1$residuals, col = "red")
```




```
shapiro.test(fit_1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_1$residuals
## W = 0.95996, p-value = 4.183e-08
```

```
jarque.bera.test(resid(fit_1))
```

```
##
##  Jarque Bera Test
##
## data:  resid(fit_1)
## X-squared = 89.135, df = 2, p-value < 2.2e-16
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(2, 0, 4)

```
fit_2 = arima(y, order = c(2, 0, 4), method = "ML", include.mean = TRUE) # fit the model
```

```
## Warning in log(s2): NaNs produced
```

```
## Warning in log(s2): NaNs produced
```

```
## Warning in log(s2): NaNs produced
```

```
fit_2
```

```
##
## Call:
```

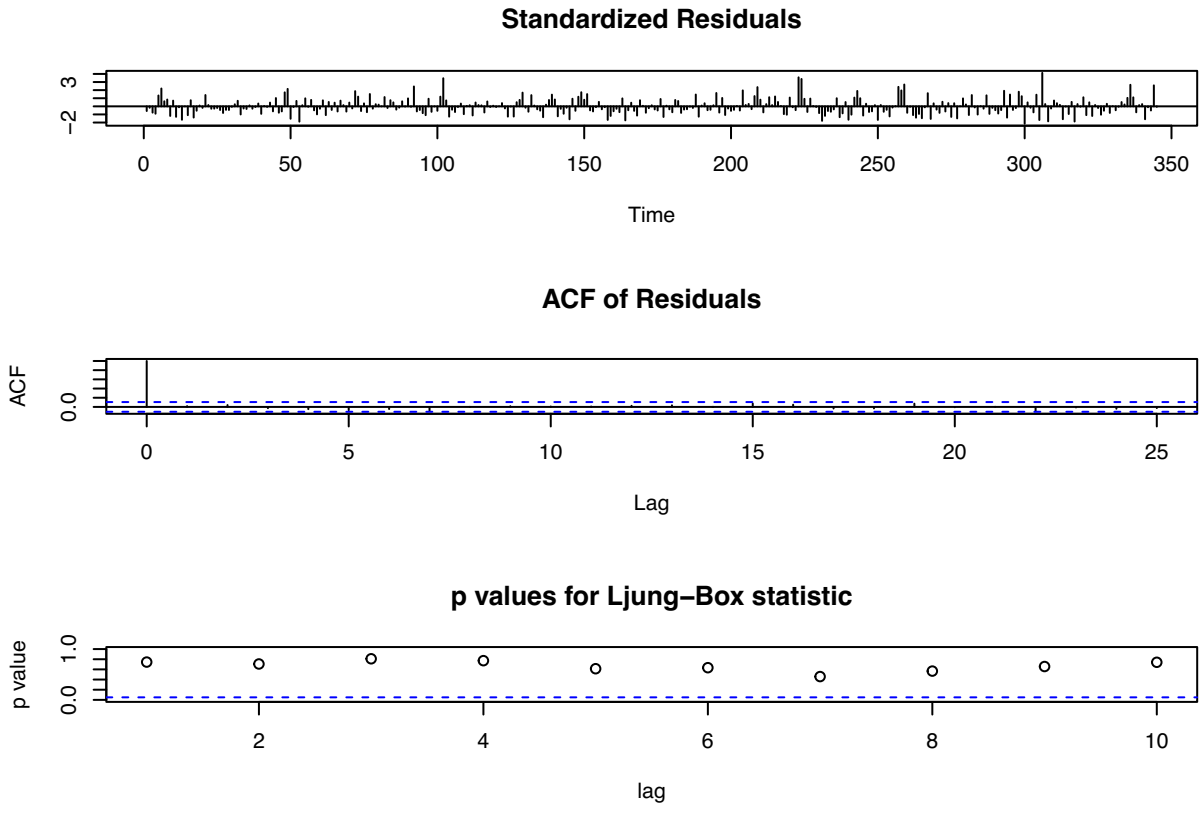
```
## arima(x = y, order = c(2, 0, 4), include.mean = TRUE, method = "ML")
##
## Coefficients:
##           ar1      ar2     ma1     ma2      ma3      ma4  intercept
##       -1.0205  -0.8865  0.7460  0.4407  -0.4959  -0.2248    -0.0006
## s.e.   0.0743   0.0607  0.0863  0.0939   0.0734   0.0615     0.0048
##
## sigma^2 estimated as 0.03045:  log likelihood = 112.4,  aic = -210.8
```

```
confint(fit_2)
```

```
##                  2.5 %       97.5 %
## ar1       -1.166043561 -0.874966997
## ar2       -1.005509945 -0.767514410
## ma1        0.576944456  0.915102725
## ma2        0.256749971  0.624664092
## ma3       -0.639764714 -0.351959962
## ma4       -0.345306591 -0.104342256
## intercept -0.009977999  0.008678761
```

Residual analysis:

```
tsdiag(fit_2)
```



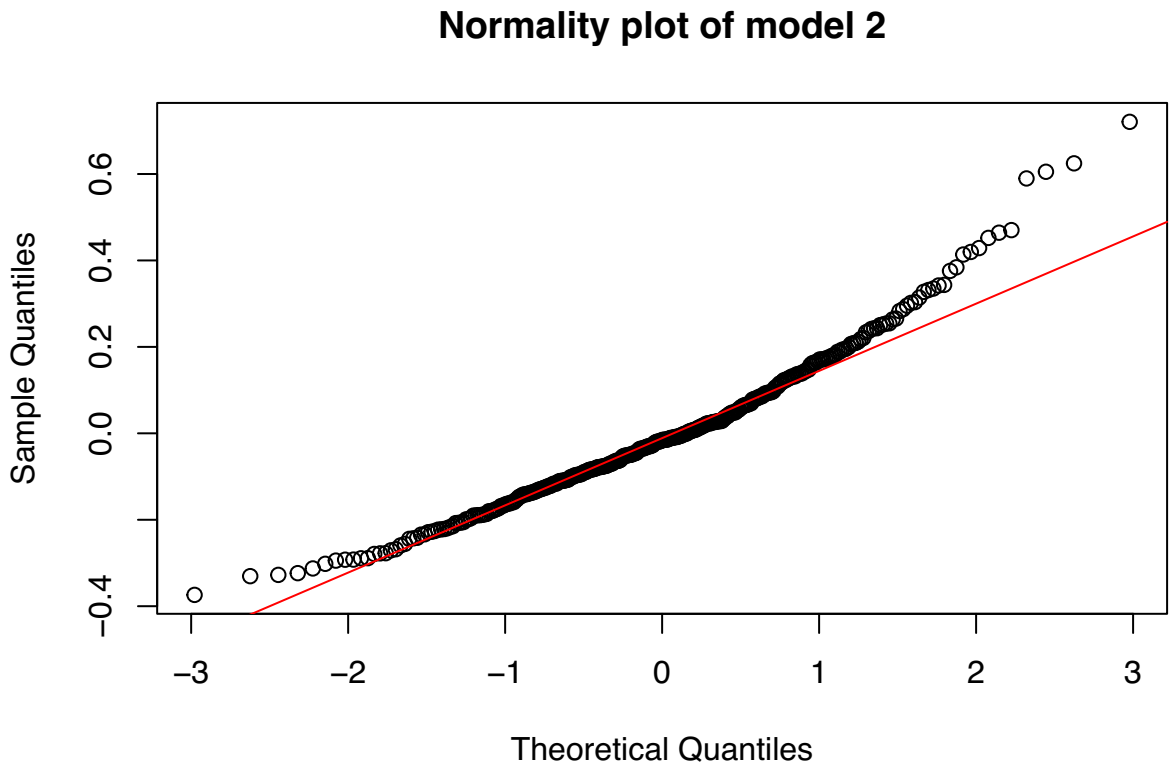

```
Box.test(fit_2$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_2$residuals
## X-squared = 0.10659, df = 1, p-value = 0.7441
```

The residuals plots show the residuals are mostly in randomness. The p-values of each time lags are all insignificant, which imply failure of rejection of null hypothesis.

Normality:

```
qqnorm(fit_2$residuals, main = "Normality plot of model 2")
qqline(fit_2$residuals, col = "red")
```




```
shapiro.test(resid(fit_2))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(fit_2)
## W = 0.96422, p-value = 1.758e-07
```

```
jarque.bera.test(resid(fit_2))
```

```
##
##  Jarque Bera Test
##
## data:  resid(fit_2)
## X-squared = 63.174, df = 2, p-value = 1.91e-14
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(4, 0, 5)

```
fit_3 = arima(y, order = c(4, 0, 5), method = "ML", include.mean = TRUE) # fit the model
fit_3
```

```
##
```

```
## Call:
## arima(x = y, order = c(4, 0, 5), include.mean = TRUE, method = "ML")
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##           ar1      ar2     ar3      ar4      ma1     ma2    ma3     ma4
##       -0.2081  -0.3382  0.5647  -0.1536  -0.0486  0.1648  -0.84  0.2155
## s.e.      NaN      NaN     NaN      NaN      NaN  0.0972    NaN     NaN
##           ma5  intercept
##       -0.0168    -0.0008
## s.e.   0.0845     0.0039
##
## sigma^2 estimated as 0.02963:  log likelihood = 115.46,  aic = -210.92
```

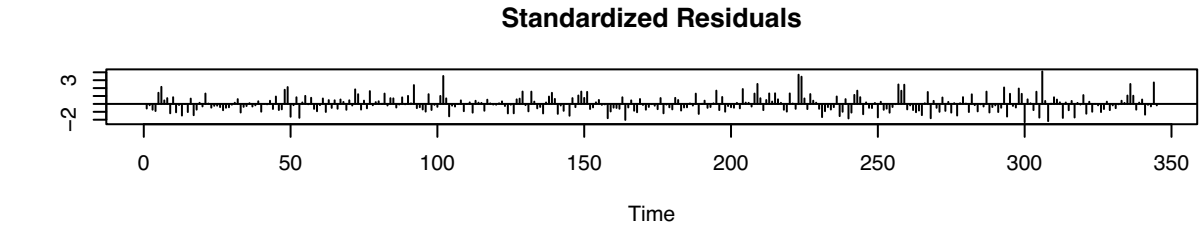
```
confint(fit_3)
```

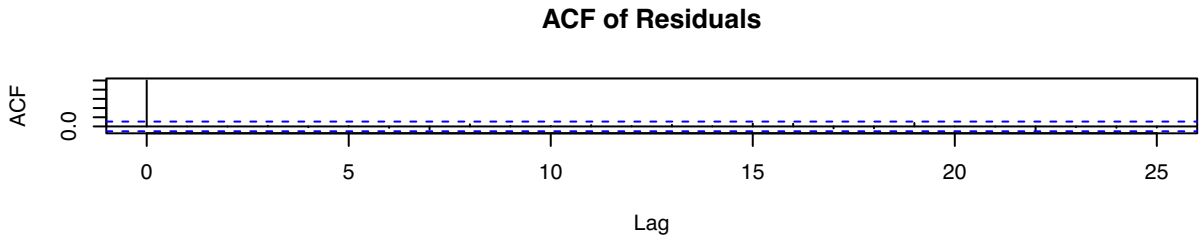
```
## Warning in sqrt(diag(vcov(object))): NaNs produced
```

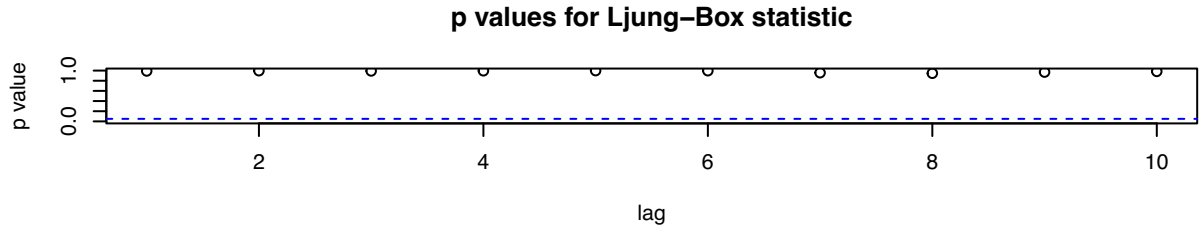
```
##                   2.5 %      97.5 %
## ar1                 NaN         NaN
## ar2                 NaN         NaN
## ar3                 NaN         NaN
## ar4                 NaN         NaN
## ma1                 NaN         NaN
## ma2       -0.025627738 0.355225412
## ma3                 NaN         NaN
## ma4                 NaN         NaN
## ma5       -0.182423194 0.148851591
## intercept -0.008455319 0.006878274
```

Residual analysis:

```
tsdiag(fit_3)
```



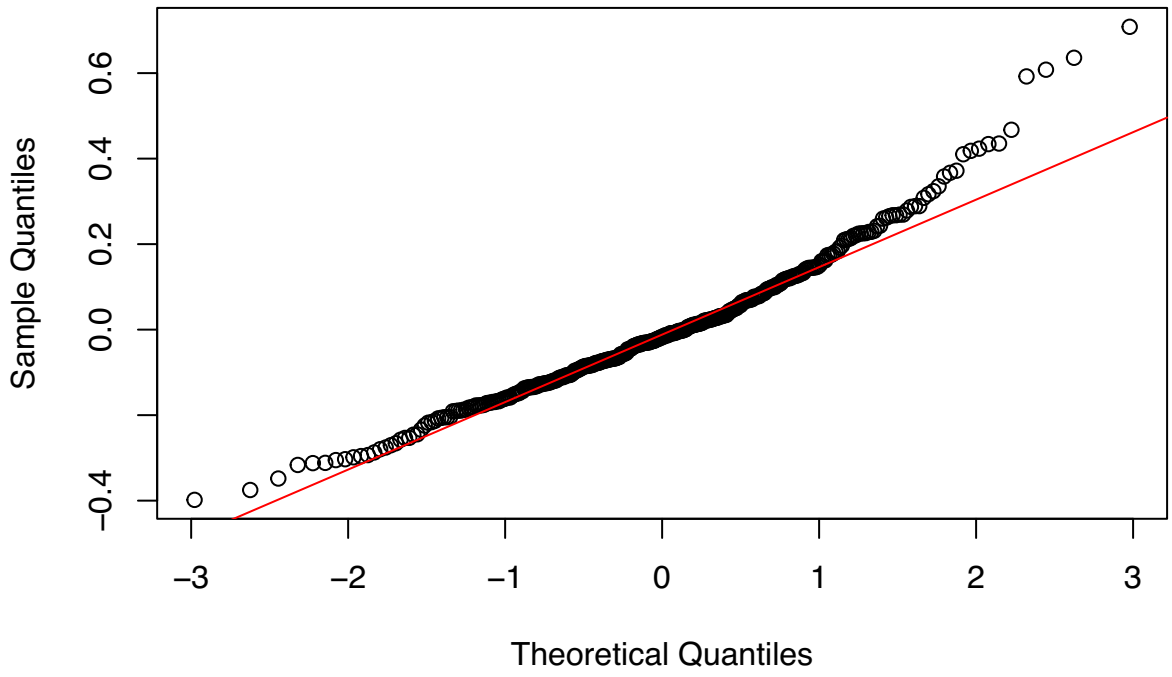




The residuals plots show the residuals are mostly in randomness. The p-values of each time lags are all insignificant, which imply failure of rejection of null hypothesis.

Normality:

```
qqnorm(fit_3$residuals, main = "Normality plot of model 3")
qqline(fit_3$residuals, col = "red")
```




```
shapiro.test(fit_3$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_3$residuals
## W = 0.96532, p-value = 2.587e-07
```

```
jarque.bera.test(resid(fit_3))
```

```
##
##  Jarque Bera Test
##
## data:  resid(fit_3)
## X-squared = 66.725, df = 2, p-value = 3.22e-15
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(2, 0, 2)

```
fit_4 = arima(y, order = c(2, 0, 2), method = "ML", include.mean = TRUE) # fit the model
fit_4
```

```
##
## Call:
## arima(x = y, order = c(2, 0, 2), include.mean = TRUE, method = "ML")
##
## Coefficients:
##           ar1     ar2      ma1      ma2  intercept
##       -0.0123  0.3049  -0.2657  -0.4479    -0.0008
## s.e.   3.0863  1.6368   3.0945   2.5045     0.0039
```

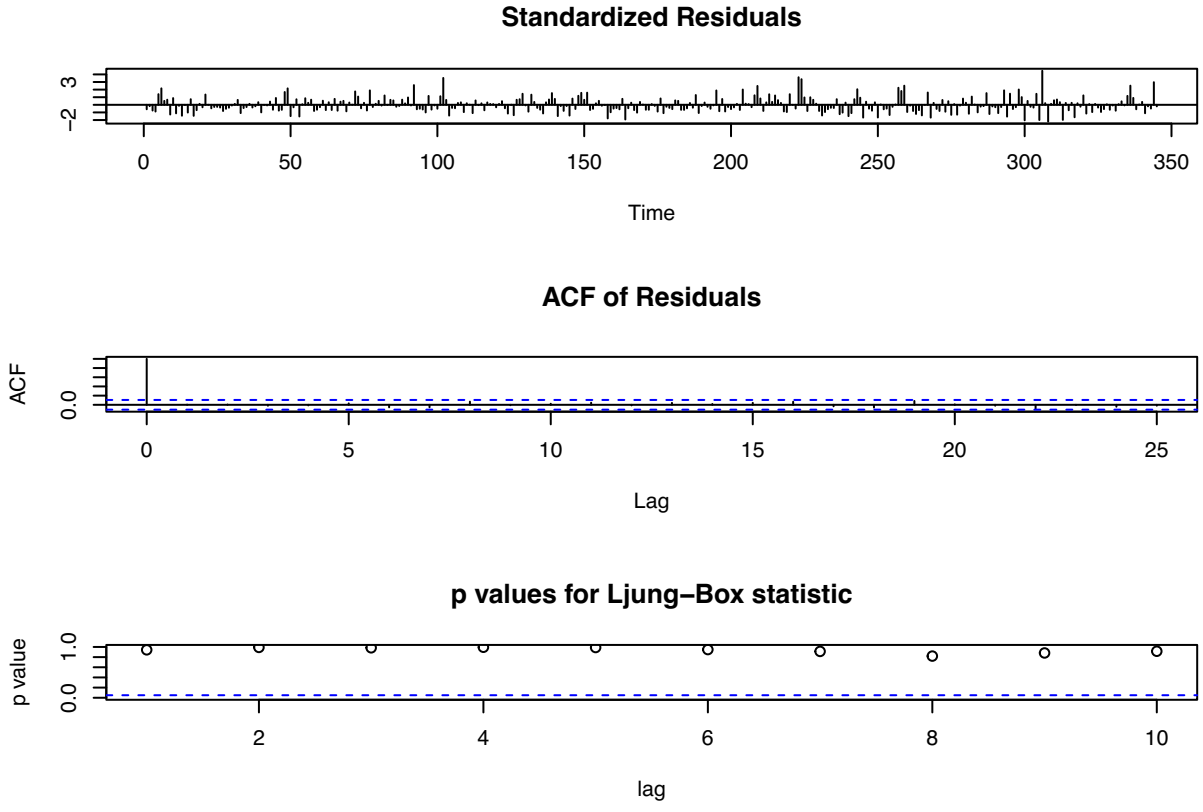
```
##
## sigma^2 estimated as 0.03069:  log likelihood = 111.3,  aic = -212.59
```

```
confint(fit_4)
```

```
##                 2.5 %      97.5 %
## ar1       -6.06147162 6.036784424
## ar2       -2.90314014 3.512853095
## ma1       -6.33085146 5.799398684
## ma2       -5.35659519 4.460773054
## intercept -0.00834649 0.006764229
```

Residual analysis:

```
tsdiag(fit_4)
```



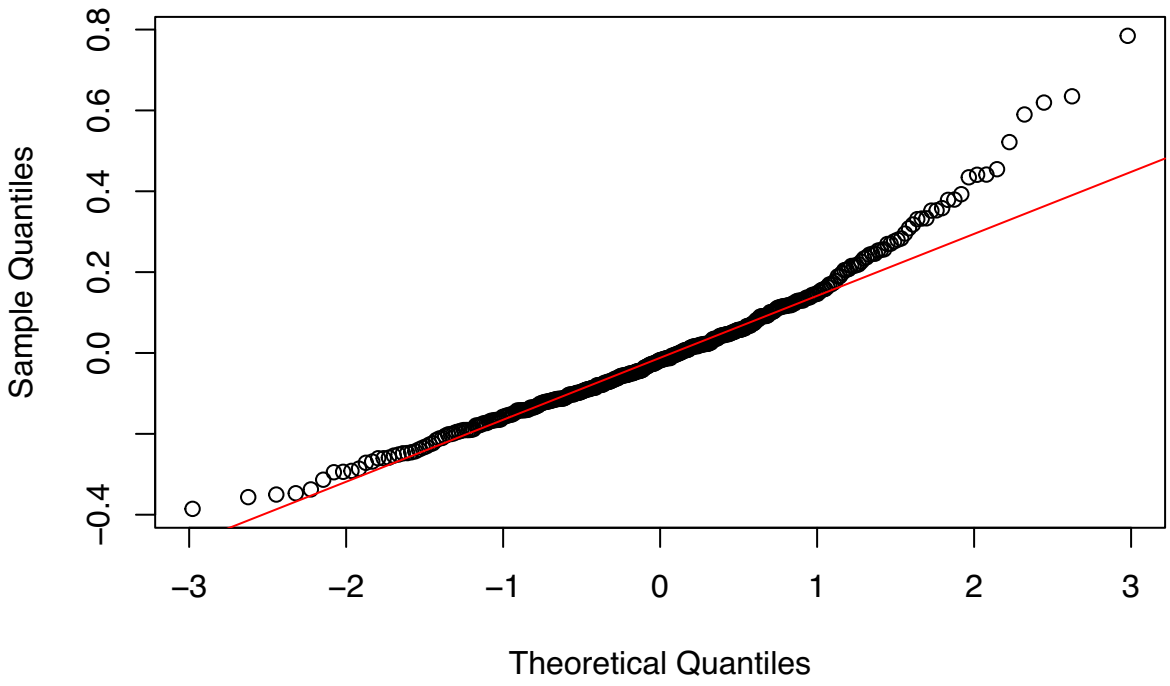

```
Box.test(fit_4$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_4$residuals
## X-squared = 0.0050623, df = 1, p-value = 0.9433
```

The residuals plots show the residuals are mostly in randomness. The p-values of each time lags are all insignificant, which imply failure of rejection of null hypothesis.

Normality:

```
qqnorm(fit_4$residuals, main = "Normality plot of model 4")
qqline(fit_4$residuals, col = "red")
```




```
shapiro.test(resid(fit_4))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(fit_4)
## W = 0.95849, p-value = 2.601e-08
```

```
jarque.bera.test(resid(fit_4))
```

```
##
##  Jarque Bera Test
##
## data:  resid(fit_4)
## X-squared = 91.833, df = 2, p-value < 2.2e-16
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(1, 0, 4)

```
fit_5 = arima(y, order = c(1, 0, 4), method = "ML", include.mean = TRUE) # fit the model
fit_5
```

```
##
## Call:
## arima(x = y, order = c(1, 0, 4), include.mean = TRUE, method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2      ma3      ma4  intercept
##       -0.3959  0.1238  -0.2447  -0.1793  -0.1074     -8e-04
## s.e.   0.5380  0.5370   0.1544   0.1002   0.0829      4e-03
```

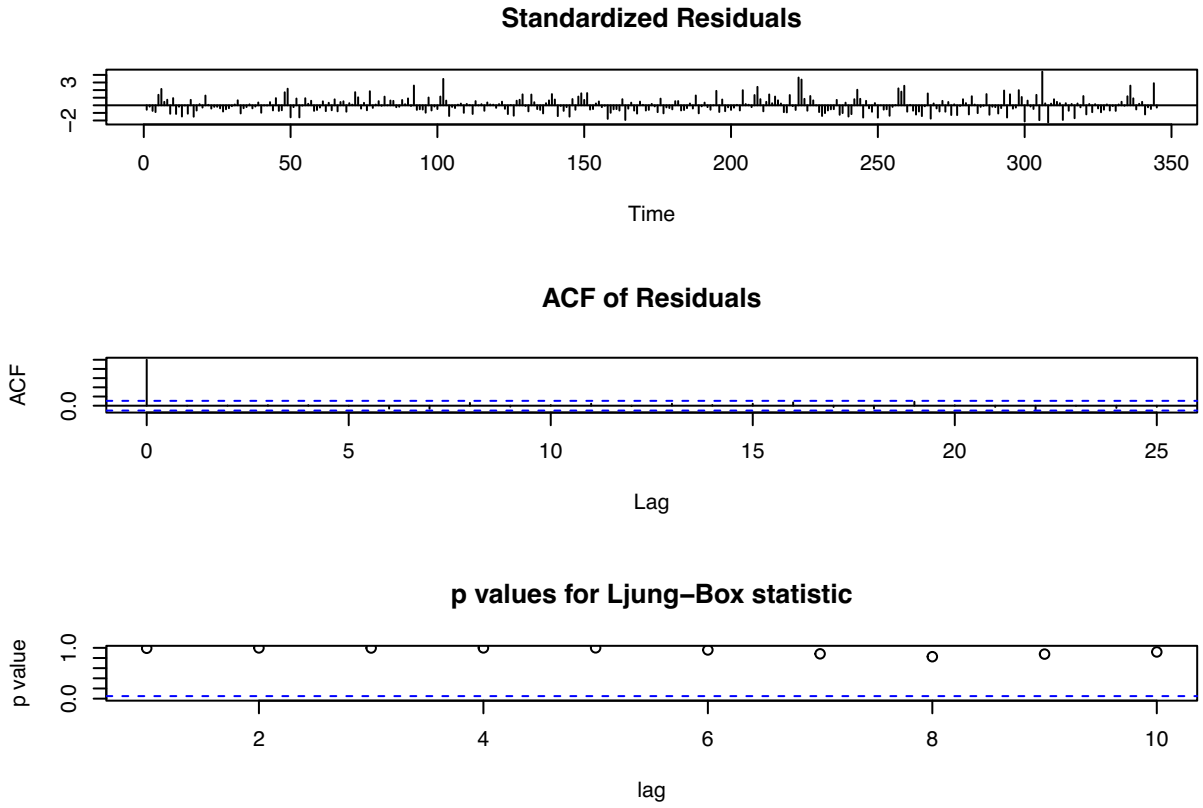
```
##
## sigma^2 estimated as 0.03064:  log likelihood = 111.56,  aic = -211.13
```

```
confint(fit_5)
```

```
##                 2.5 %      97.5 %
## ar1       -1.45023416 0.658524163
## ma1       -0.92867865 1.176330240
## ma2       -0.54737853 0.057949355
## ma3       -0.37570981 0.017036186
## ma4       -0.26984499 0.055091516
## intercept -0.00865095 0.007147506
```

Residual analysis:

```
tsdiag(fit_5)
```



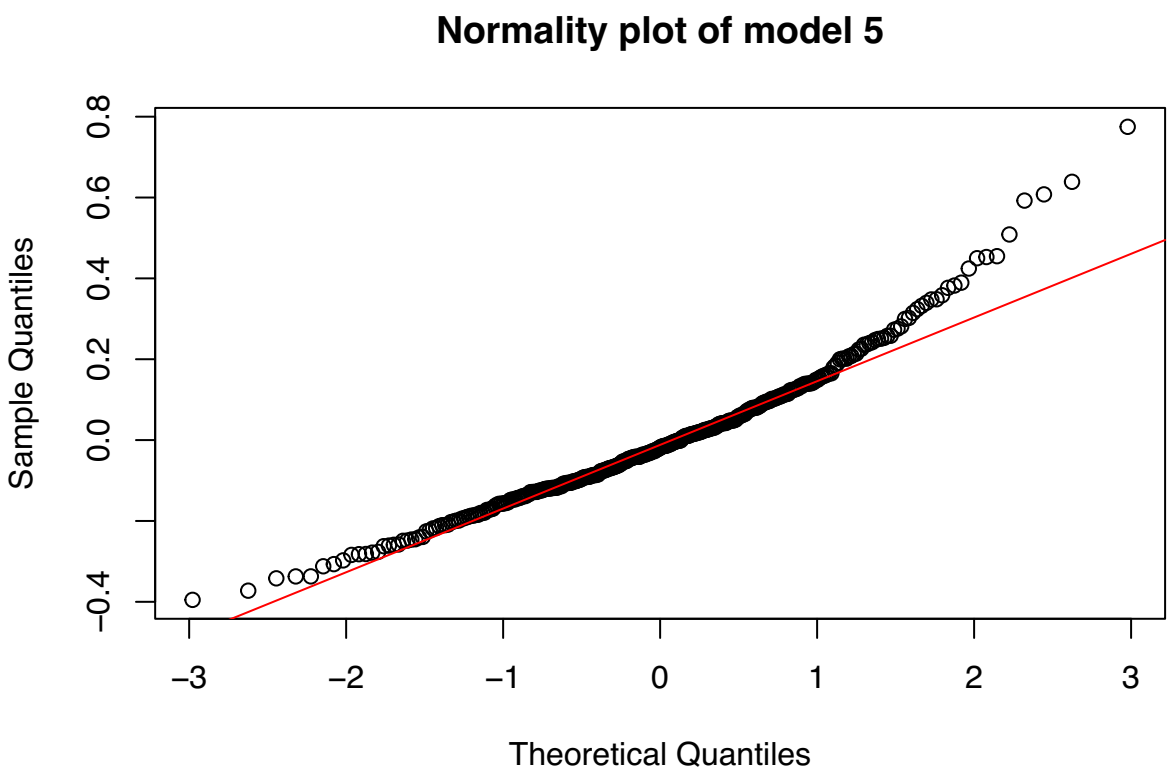

```
Box.test(fit_5$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_5$residuals
## X-squared = 0.00011068, df = 1, p-value = 0.9916
```

The residuals plots show the residuals are mostly in randomness. The p-values of each time lags are all insignificant, which imply failure of rejection of null hypothesis.

Normality:

```
qqnorm(fit_5$residuals, main = "Normality plot of model 5")
qqline(fit_5$residuals, col = "red")
```




```
shapiro.test(resid(fit_5))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(fit_5)
## W = 0.96083, p-value = 5.561e-08
```

```
jarque.bera.test(resid(fit_5))
```

```
##
##  Jarque Bera Test
##
## data:  resid(fit_5)
## X-squared = 85.888, df = 2, p-value < 2.2e-16
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(2, 0, 3)

```
fit_6 = arima(y, order = c(2, 0, 3), method = "ML", include.mean = TRUE) # fit the model
fit_6
```

```
##
## Call:
## arima(x = y, order = c(2, 0, 3), include.mean = TRUE, method = "ML")
##
## Coefficients:
##          ar1     ar2      ma1      ma2      ma3  intercept
##       0.4006  -0.010  -0.6748  -0.0134  -0.0611    -0.0008
## s.e.  0.9543   0.502   0.9530   0.7487   0.1016     0.0039
```

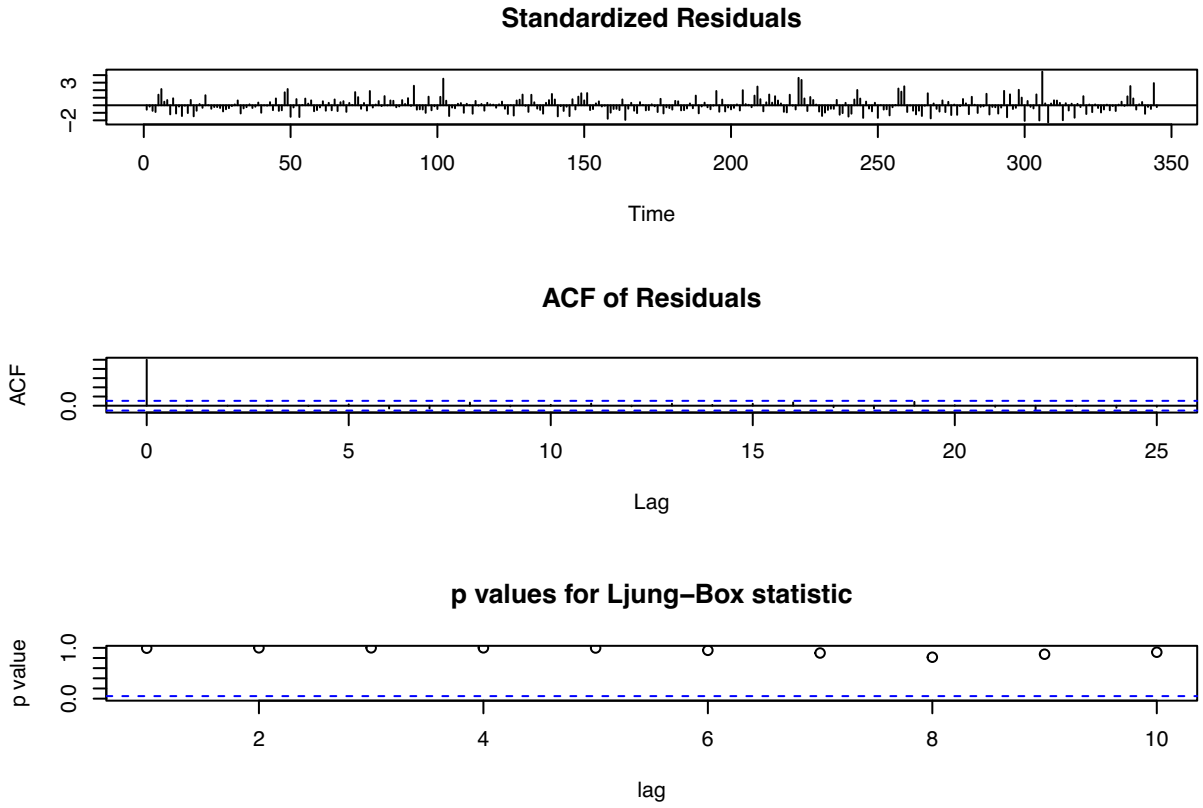
```
##
## sigma^2 estimated as 0.03066:  log likelihood = 111.47,  aic = -210.95
```

```
confint(fit_6)
```

```
##                  2.5 %      97.5 %
## ar1       -1.469806233 2.271081228
## ar2       -0.993797340 0.973831083
## ma1       -2.542499116 1.192999033
## ma2       -1.480877704 1.454101005
## ma3       -0.260357323 0.138083868
## intercept -0.008445159 0.006894374
```

Residual analysis:

```
tsdiag(fit_6)
```



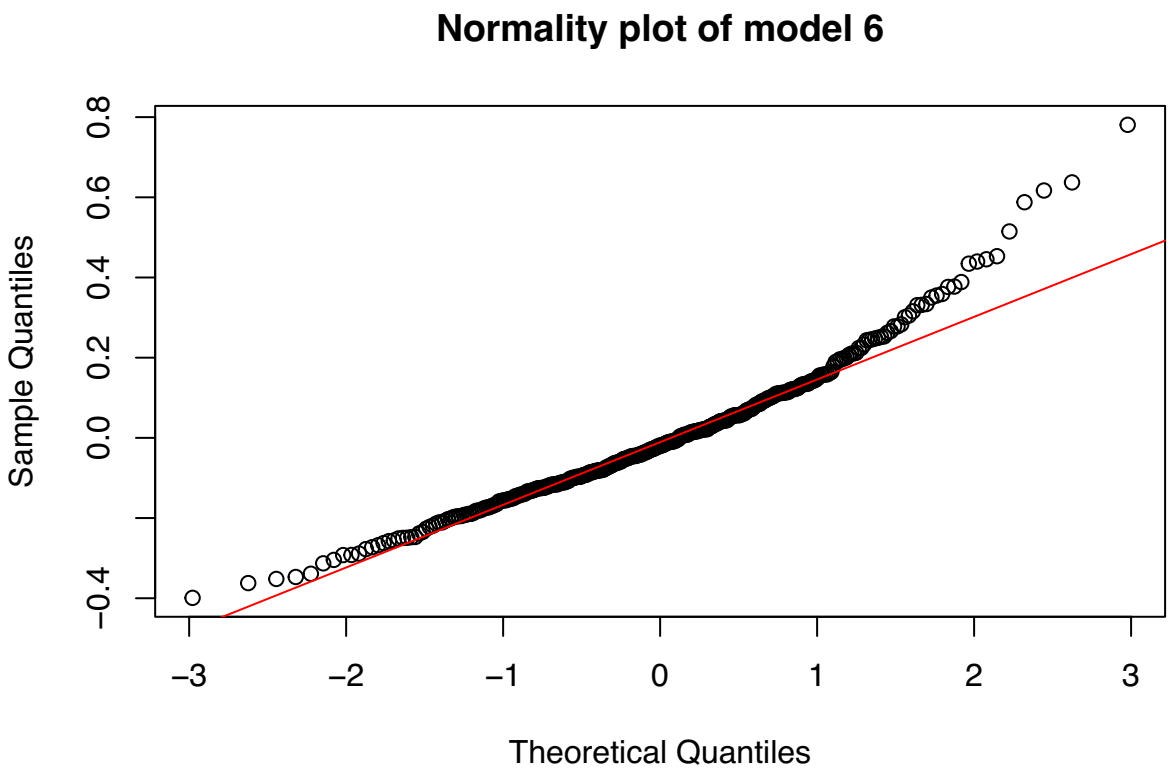

```
Box.test(fit_6$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_6$residuals
## X-squared = 6.8555e-05, df = 1, p-value = 0.9934
```

The residuals plots show the residuals are mostly in randomness. The p-values of each time lags are all insignificant, which imply failure of rejection of null hypothesis.

Normality:

```
qqnorm(fit_6$residuals, main = "Normality plot of model 6")
qqline(fit_6$residuals, col = "red")
```




```
shapiro.test(resid(fit_6))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(fit_6)
## W = 0.95996, p-value = 4.183e-08
```

```
jarque.bera.test(resid(fit_6))
```

```
##
##  Jarque Bera Test
##
## data:  resid(fit_6)
## X-squared = 89.135, df = 2, p-value < 2.2e-16
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(1, 0, 5)

```
fit_7 = arima(y, order = c(1, 0, 5), method = "ML", include.mean = TRUE) # fit the model
fit_7
```

```
##
## Call:
## arima(x = y, order = c(1, 0, 5), include.mean = TRUE, method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2      ma3      ma4     ma5  intercept
##       -0.3543  0.0823  -0.2337  -0.1746  -0.1023  0.0042     -7e-04
## s.e.   0.6771  0.6761   0.1879   0.1084   0.0976  0.0659      4e-03
```

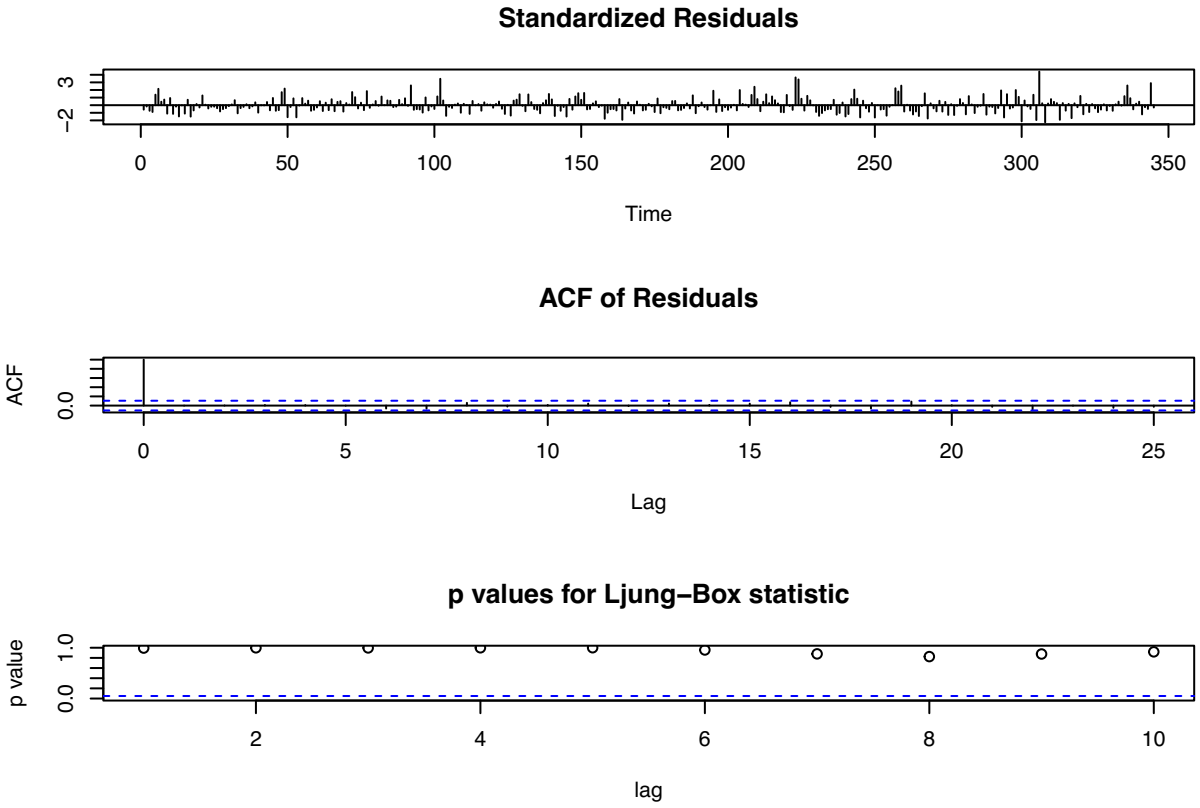
```
##
## sigma^2 estimated as 0.03064:  log likelihood = 111.56,  aic = -209.13
```

```
confint(fit_7)
```

```
##                 2.5 %      97.5 %
## ar1       -1.681356572 0.972786488
## ma1       -1.242865266 1.407458065
## ma2       -0.602044348 0.134580246
## ma3       -0.386988370 0.037762363
## ma4       -0.293558673 0.088959840
## ma5       -0.124838047 0.133311733
## intercept -0.008664176 0.007164836
```

Residual analysis:

```
tsdiag(fit_7)
```



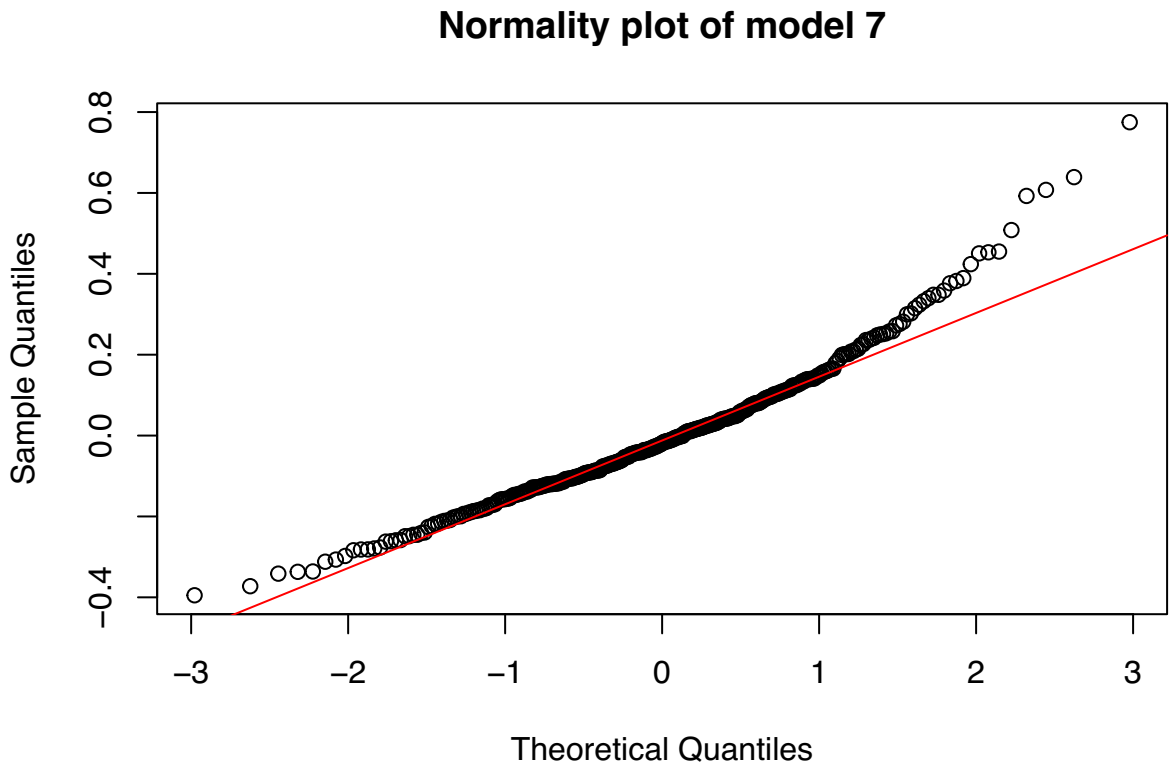

```
Box.test(fit_7$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_7$residuals
## X-squared = 6.9743e-05, df = 1, p-value = 0.9933
```

The residuals plots show the residuals are mostly in randomness. The p-values of each time lags are all insignificant, which imply failure of rejection of null hypothesis.

Normality:

```
qqnorm(fit_7$residuals, main = "Normality plot of model 7")
qqline(fit_7$residuals, col = "red")
```




```
shapiro.test(resid(fit_7))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(fit_7)
## W = 0.96081, p-value = 5.524e-08
```

```
jarque.bera.test(resid(fit_7))
```

```
##
##  Jarque Bera Test
##
## data:  resid(fit_7)
## X-squared = 85.875, df = 2, p-value < 2.2e-16
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(1, 0, 2)

```
fit_8 = arima(y, order = c(1, 0, 2), method = "ML", include.mean = TRUE) # fit the model
fit_8
```

```
##
## Call:
## arima(x = y, order = c(1, 0, 2), include.mean = TRUE, method = "ML")
##
## Coefficients:
```

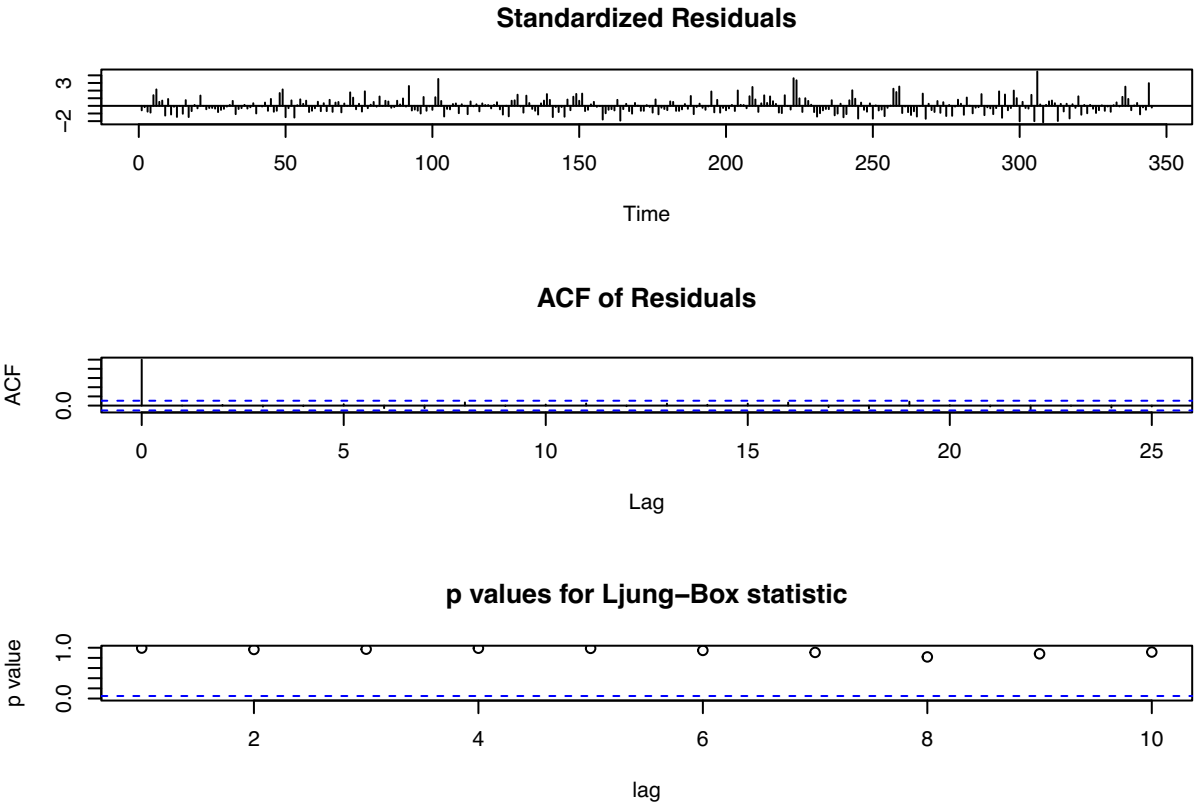
```
##          ar1      ma1      ma2  intercept
##       0.5334  -0.8074  -0.0031    -0.0008
## s.e.  0.1470   0.1538   0.0893     0.0039
##
## sigma^2 estimated as 0.03069:  log likelihood = 111.28,  aic = -214.57
```

```
confint(fit_8)
```

```
##                  2.5 %       97.5 %
## ar1        0.245350679  0.821462691
## ma1       -1.108794745 -0.506053311
## ma2       -0.178145415  0.171918982
## intercept -0.008369441  0.006787417
```

Residual analysis:

```
tsdiag(fit_8)
```



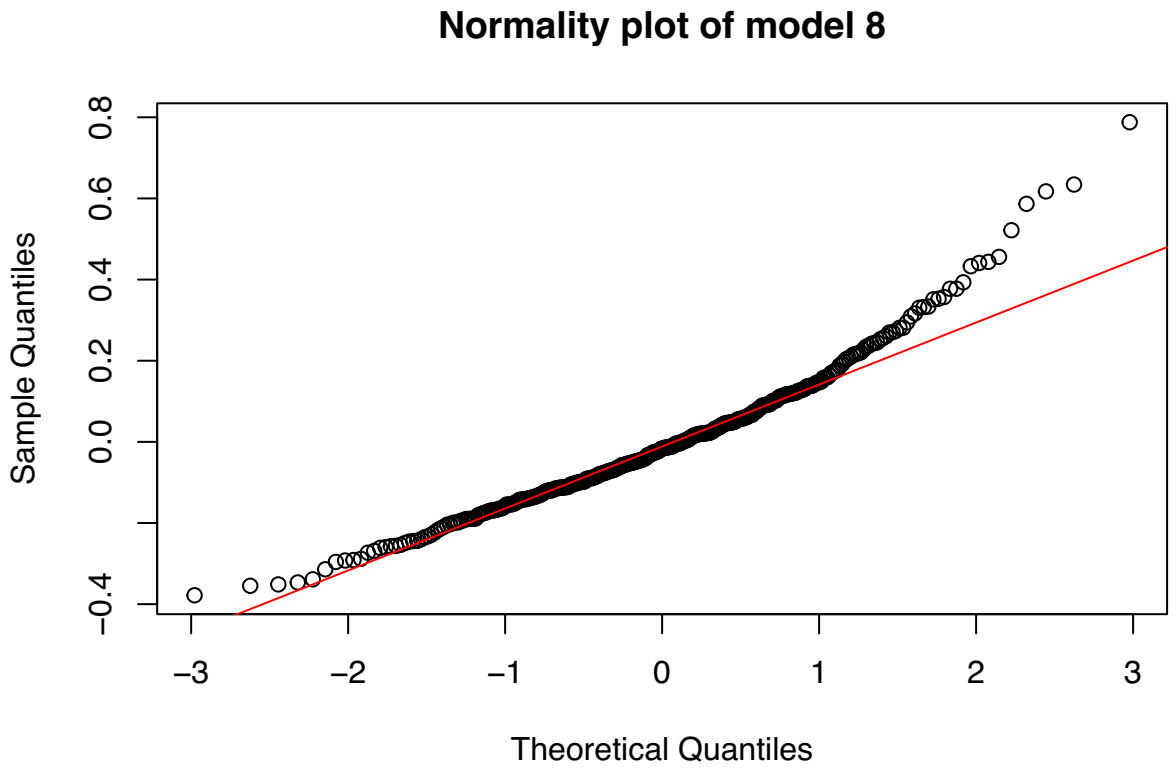

```
Box.test(fit_8$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_8$residuals
## X-squared = 6.7807e-05, df = 1, p-value = 0.9934
```

The residuals plots show the residuals are mostly in randomness. The p-values of each time lags are all insignificant, which imply failure of rejection of null hypothesis.

Normality:

```r
qqnorm(fit_8$residuals, main = "Normality plot of model 8")
qqline(fit_8$residuals, col = "red")
```




```r
shapiro.test(resid(fit_8))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(fit_8)
## W = 0.95854, p-value = 2.643e-08
```

```r
jarque.bera.test(resid(fit_8))
```

```
##
##  Jarque Bera Test
##
## data:  resid(fit_8)
## X-squared = 91.703, df = 2, p-value < 2.2e-16
```

All of the 8 models' residuals are non-normally distributed, uncorrelated based on Shapiro test and normal plot. All models have p-values of Box-Ljung test larger than 0.05, which imply insignificant. For the analysis above, we have a contradiction. The analysis can neither reject any possible models nor finding the most appropriate model. So, we have to apply further comparison, whcih is the magnitude of AIC, BIC and harmonic mean.

RMSE

```r
RMSE = function(z){
  fval = sqrt(sum(z^2) / length(z))
  return(fval)
```

```
}
print(RMSE(as.numeric(fit_1$residuals)))
```

```
## [1] 0.1750902
```

```
print(RMSE(as.numeric(fit_2$residuals)))
```

```
## [1] 0.1745082
```

```
print(RMSE(as.numeric(fit_3$residuals)))
```

```
## [1] 0.1721272
```

```
print(RMSE(as.numeric(fit_4$residuals)))
```

```
## [1] 0.1751821
```

```
print(RMSE(as.numeric(fit_5$residuals)))
```

```
## [1] 0.1750417
```

```
print(RMSE(as.numeric(fit_6$residuals)))
```

```
## [1] 0.1750902
```

```
print(RMSE(as.numeric(fit_7$residuals)))
```

```
## [1] 0.1750415
```

```
print(RMSE(as.numeric(fit_8$residuals)))
```

```
## [1] 0.1751879
```

Harmonic mean

```
harmonic_mean = function(z){
  fval = length(z) / sum(1 / abs(z))
  return(fval)
}
print(harmonic_mean(as.numeric(fit_1$residuals)))
```

```
## [1] 0.04360516
```

```
print(harmonic_mean(as.numeric(fit_2$residuals)))
```

```
## [1] 0.02837465
```

```
print(harmonic_mean(as.numeric(fit_3$residuals)))
```

```
## [1] 0.009605899
```

```
print(harmonic_mean(as.numeric(fit_4$residuals)))
```

```
## [1] 0.02454442
```

```
print(harmonic_mean(as.numeric(fit_5$residuals)))
```

```
## [1] 0.03311293
```

```
print(harmonic_mean(as.numeric(fit_6$residuals)))
```

```
## [1] 0.04360516
```

```
print(harmonic_mean(as.numeric(fit_7$residuals)))
```

```
## [1] 0.03422845
```

```
print(harmonic_mean(as.numeric(fit_8$residuals)))
```

```
## [1] 0.0186544
```

The smallest harmonic mean is model 8: ARMA(1, 2)

AIC

```
AIC_list = c(
AIC(fit_1),
AIC(fit_2),
AIC(fit_3),
AIC(fit_4),
AIC(fit_5),
AIC(fit_6),
AIC(fit_7),
AIC(fit_8))
AIC_list
```

```
## [1] -208.9455 -208.8011 -208.9197 -210.5926 -209.1273 -208.9455 -207.1276
## [8] -212.5687
```

```
sort(AIC_list)
```

```
## [1] -212.5687 -210.5926 -209.1273 -208.9455 -208.9455 -208.9197 -208.8011
## [8] -207.1276
```

The smallest AIC is model 8: ARMA(1, 2

BIC

```
BIC_list = c(
BIC(fit_1),
BIC(fit_2),
BIC(fit_3),
BIC(fit_4),
BIC(fit_5),
BIC(fit_6),
BIC(fit_7),
BIC(fit_8))
BIC_list
```

```
## [1] -182.0407 -178.0528 -166.6407 -187.5313 -182.2225 -182.0407 -176.3792
## [8] -193.3509
```

```
sort(BIC_list)
```

```
## [1] -193.3509 -187.5313 -182.2225 -182.0407 -182.0407 -178.0528 -176.3792
## [8] -166.6407
```

The smallest BIC is model 8: ARMA(1, 2)

```
fit_8
```

```
##
## Call:
## arima(x = y, order = c(1, 0, 2), include.mean = TRUE, method = "ML")
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.5334  -0.8074  -0.0031    -0.0008
## s.e.  0.1470   0.1538   0.0893     0.0039
##
## sigma^2 estimated as 0.03069:  log likelihood = 111.28,  aic = -214.57
```

The model estimation is: Xt = 0.5334Xt+Wt-0.8074Xt-1-0.0031Wt-1-0.0008 To sum up, model 8 has the smallest BIC, AIC and harmonic mean, so we choose model 8: ARMA(1, 2) to be the final model.

4 Forecasting

In practice, data analytics can explore how data fluctuate over time and find an appropriate model to fit the time series. Forecasting can provide a reference for things in the future; if it is accurate enough, then forecasting can help people predict the future. That is the reason why forecasting is essential. In the study, if the forecasting is accurate, the investor can know the VIX Index in the future. When an investor trades the VIX Index, the investor can measure how the VIX Index goes on and make a correct decision. We use model 8 from the last step to fit the original time series.[5] (2008, Casella) The RMSE and harmonic mean are close to 0, so the model is good. Therefore, the number of prediction values should be equal to the length of the test dataset. The plot shows the forecast values combined with the original dataset. The red lines show the prediction interval. Finally, we use the model to shows the predicted dataset after one year, whcih can help investor to know the market fear and volatility.

```
fitttt = arima(x, order = c(1, 1, 2), seasonal = list(order=c(0, 0, 1)))
pre = predict(arima(x, order = c(1, 1, 2), seasonal = list(order=c(0, 0, 1))), n.ahead = length(test$VI
pre$pred
```

```
## Time Series:
## Start = 347
## End = 384
## Frequency = 1
##  [1] 16.71490 16.28047 16.06710 15.95425 15.89456 15.86299 15.84629 15.83746
##  [9] 15.83279 15.83032 15.82901 15.82832 15.82795 15.82776 15.82766 15.82760
## [17] 15.82757 15.82756 15.82755 15.82755 15.82754 15.82754 15.82754 15.82754
## [25] 15.82754 15.82754 15.82754 15.82754 15.82754 15.82754 15.82754 15.82754
## [33] 15.82754 15.82754 15.82754 15.82754 15.82754 15.82754
```

```
pre$se
```

```
## Time Series:
## Start = 347
## End = 384
## Frequency = 1
##  [1]  3.964836  5.225329  5.804497  6.151355  6.402797  6.609325  6.792469
##  [8]  6.962266  7.123690  7.279318  7.430544  7.578162  7.722652  7.864330
## [15]  8.003417  8.140085  8.274473  8.406702  8.536877  8.665094  8.791439
## [22]  8.915993  9.038830  9.160020  9.279628  9.397713  9.514332  9.629540
## [29]  9.743385  9.855915  9.967175 10.077207 10.186050 10.293742 10.400319
## [36] 10.505815 10.610263 10.713692
```

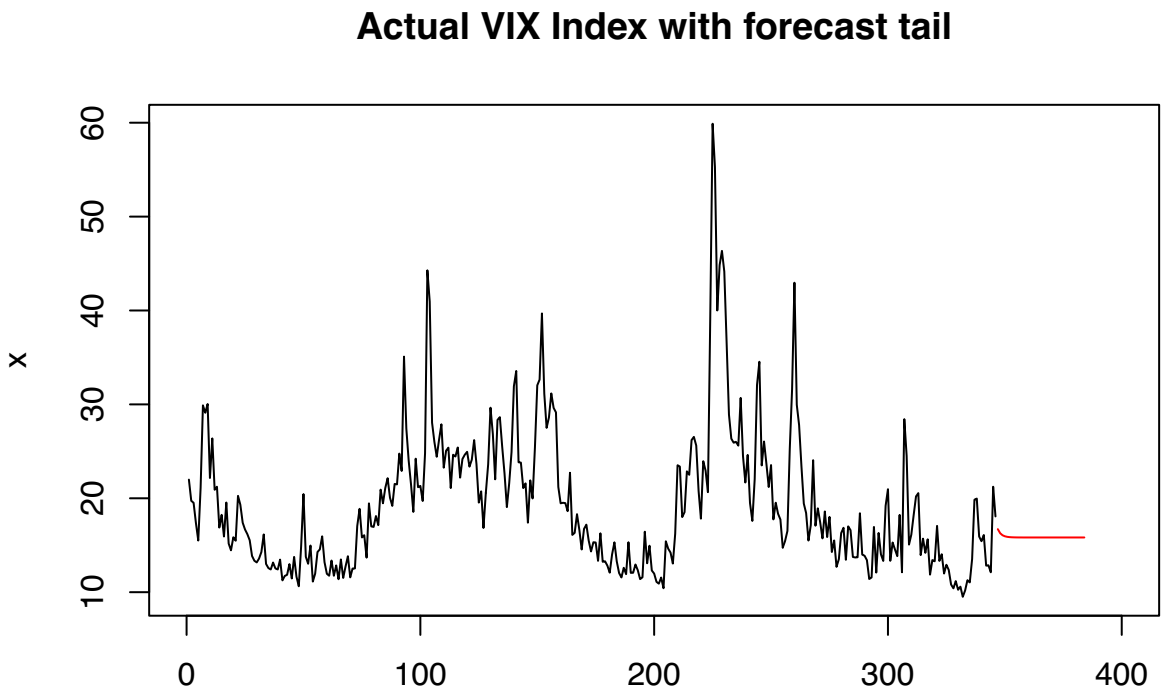
```
RMSE(fitttt$resid)
```

```
## [1] 3.959103
```

```
harmonic_mean(fitttt$resid)
```

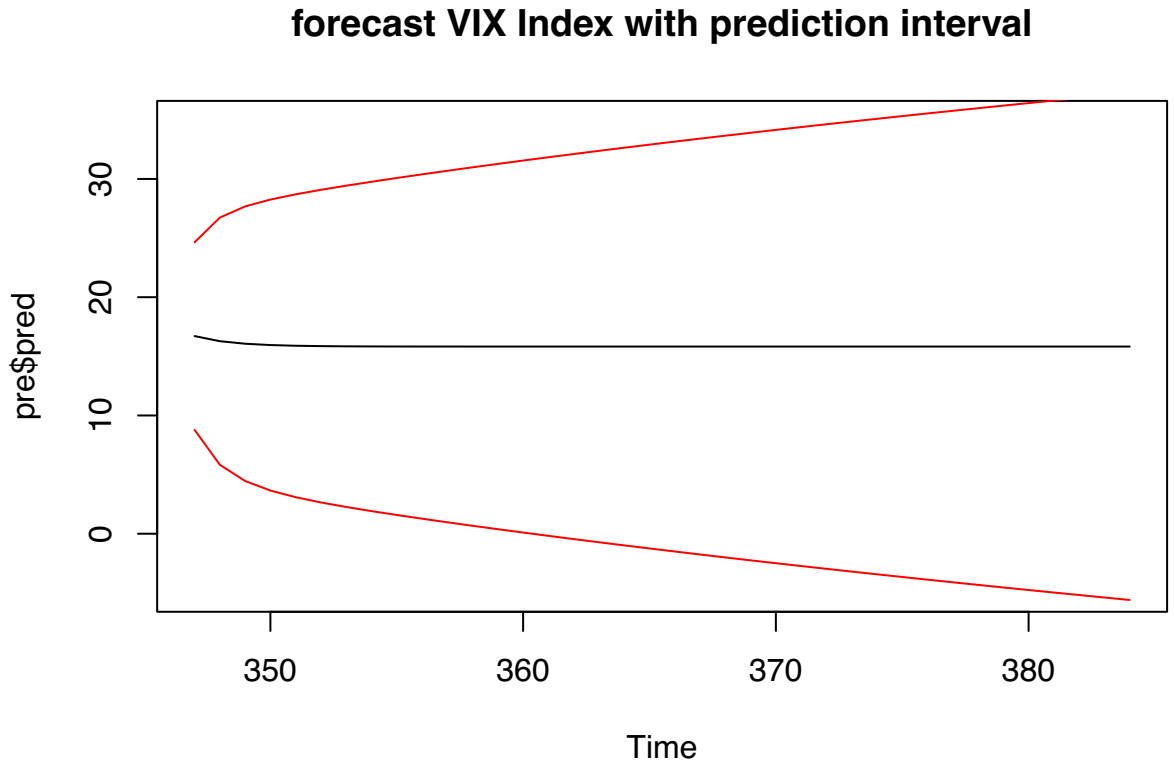
```
## [1] 0.3216785
```

```
plot(x, type = "l", xlim = c(0, 400), main = "Actual VIX Index with forecast tail")
lines(pre$pred, col = 'red')
```



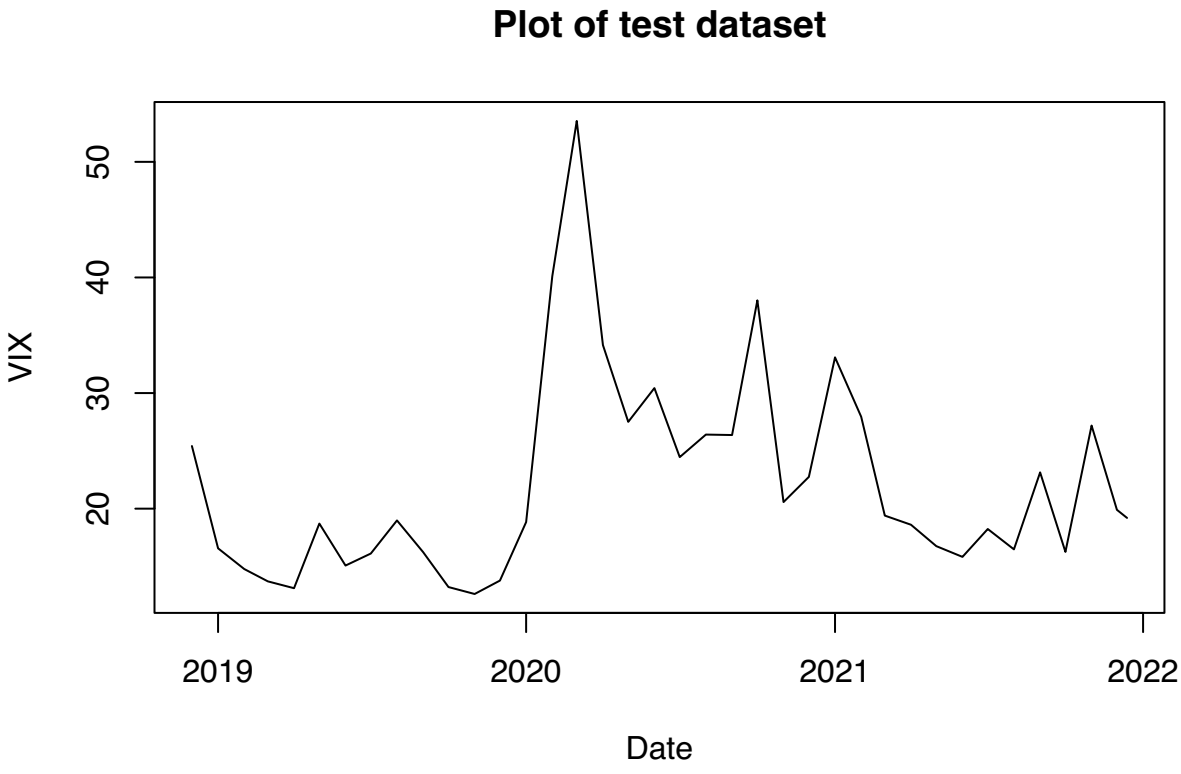
The red lines show the prediction by using the model ARIMA(1, 0, 2).

```
upperbound = (pre$pred) + (pre$se)*2
lowerbound = (pre$pred) - (pre$se)*2
ts.plot(pre$pred, ylim = c(-5, 35), main = "forecast VIX Index with prediction interval")
lines(lowerbound, col = 'red')
lines(upperbound, col = 'red')
```



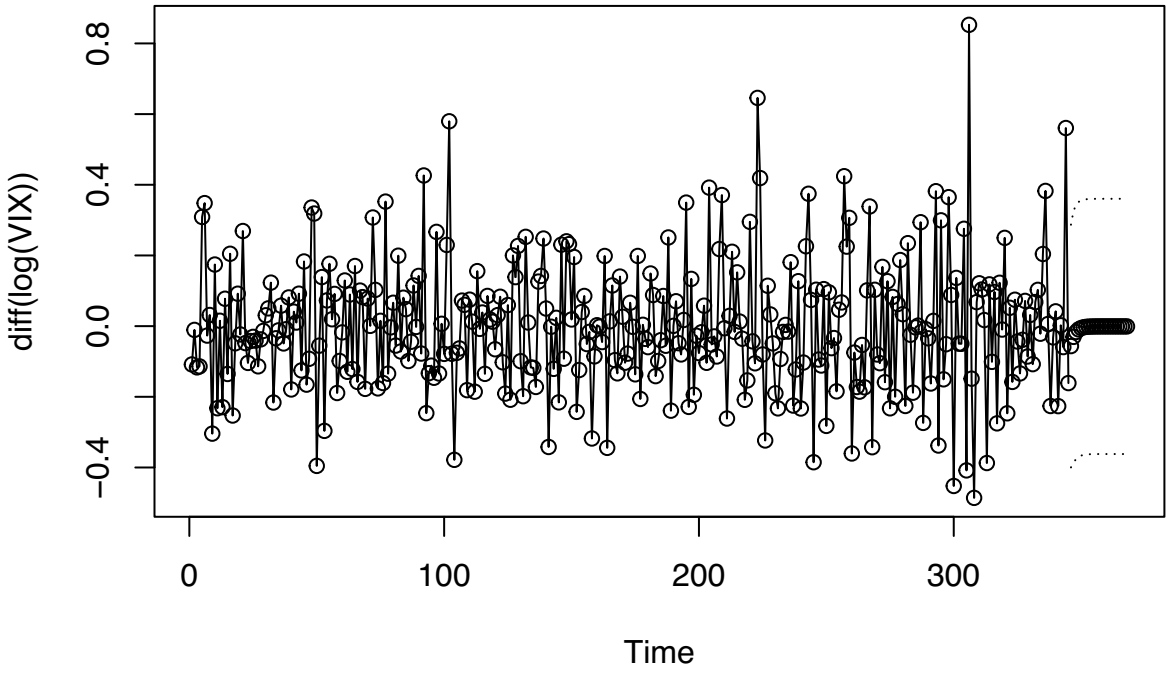
The black line shows the prediction values and the two red lines are the prediction interval.

```
plot(test, type = "l", main = "Plot of test dataset")
```



This is the forecast from model 8 for diff(log(x))

```
plot(fit_8, n.ahead = 23, ylab = "diff(log(VIX))", xlab = "Time")
```

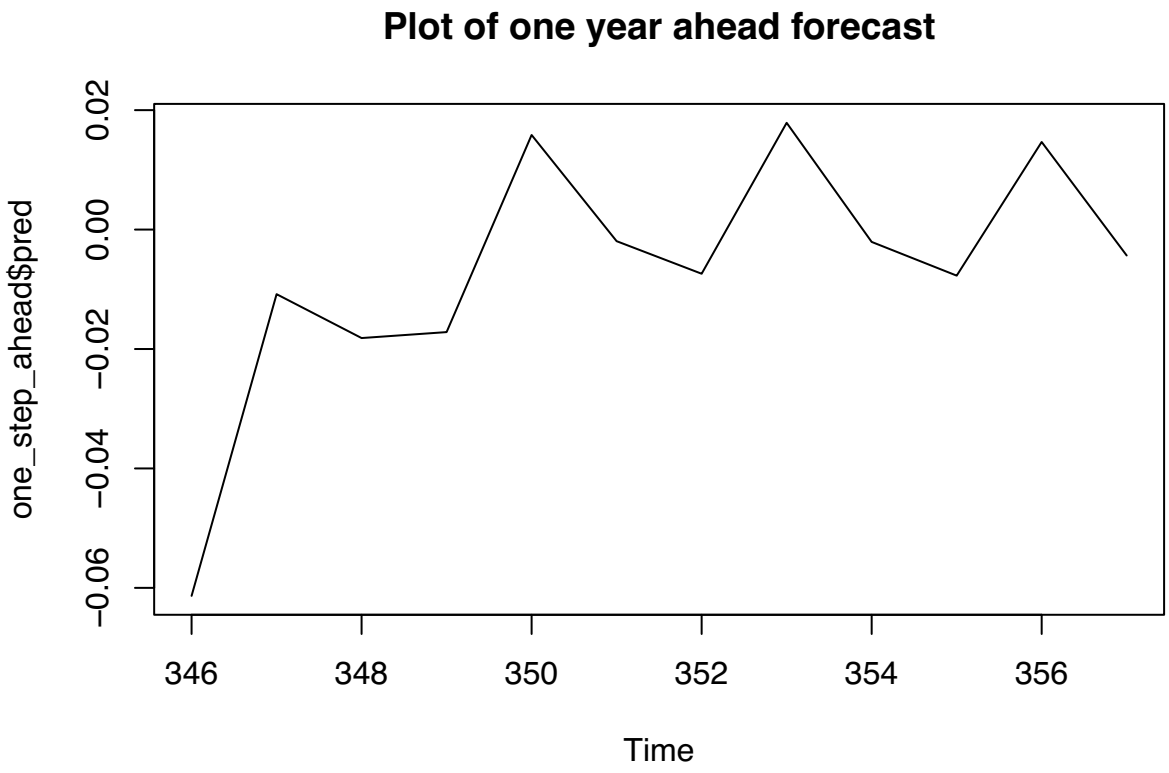


One year ahead

```
one_step_ahead = predict(fit, n.ahead = 12)
one_step_ahead$pred
```

```
## Time Series:
## Start = 346
## End = 357
## Frequency = 1
##  [1] -0.061336616 -0.010819159 -0.018165834 -0.017170894  0.015839184
##  [6] -0.001940681 -0.007397678  0.017875636 -0.002068428 -0.007713802
## [11]  0.014665511 -0.004341251
```

```
ts.plot(one_step_ahead$pred, main = "Plot of one year ahead forecast")
```



The one step ahead forecast is 16.70382.

4 Analysis of segments

Based on the plots from professor Masoud, the VIX Index from 1990 to 2019 can be divided into 6 segments. The first segment is from 1990-01 to 1993-06. The second segment is from 1993-07 to 1998-06. The third segment is from 1998-07 to 2002-12. The fourth segment is from 2003-01 to 2007-09. The fifth segment is from 2007-10 to 2009-12. The sixth segment is from 2010-01 to 2019-12. The rest of the data is the seventh segment which is happening now. The sample size is too small, so we ignore it from the study.

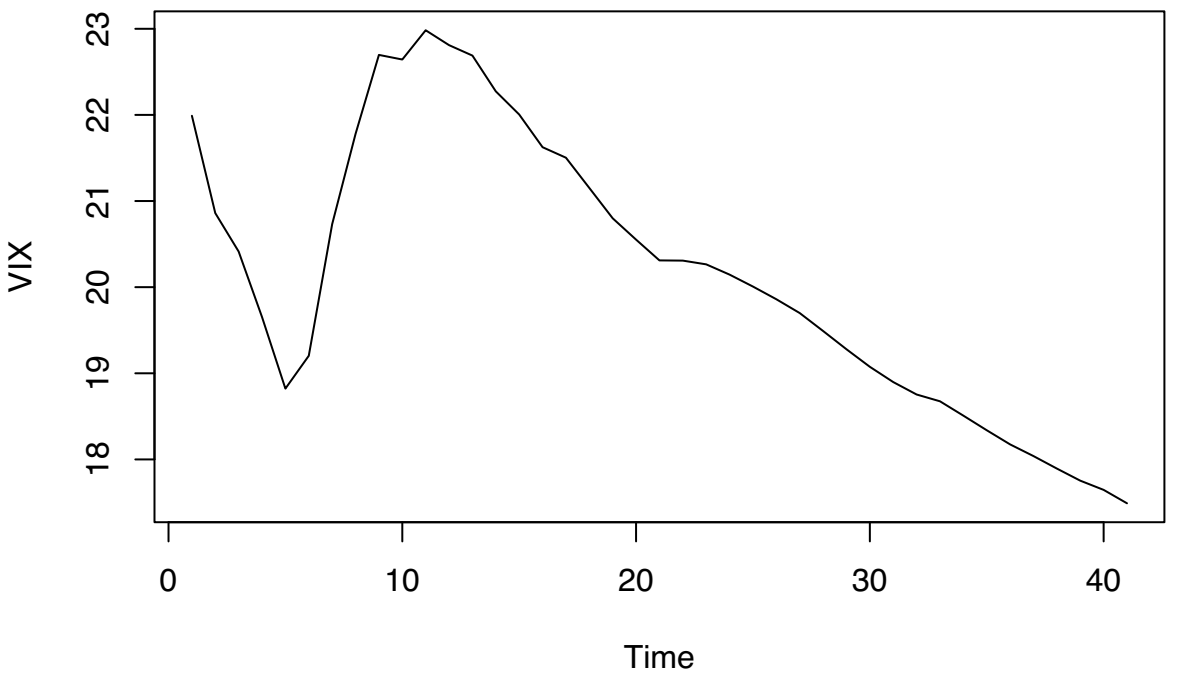
```
df = Data # rename it
df$Date = as.Date(df$Date)
VIX = ts(Data$VIX, start = c(1990, 2), end = c(2021,12), frequency = 12)
seg1 = window(VIX, start = c(1990, 2), end = c(1993, 6))
seg2 = window(VIX, start = c(1993, 7), end = c(1998, 6))
seg3 = window(VIX, start = c(1998, 7), end = c(2002, 12))
seg4 = window(VIX, start = c(2003, 1), end = c(2007, 9))
seg5 = window(VIX, start = c(2007, 10), end = c(2009, 12))
seg6 = window(VIX, start = c(2010, 1), end = c(2019, 12))
```

Since the seventh segment is still happening, so we temporarily combined the beginning of the seventh segment to the previous one.

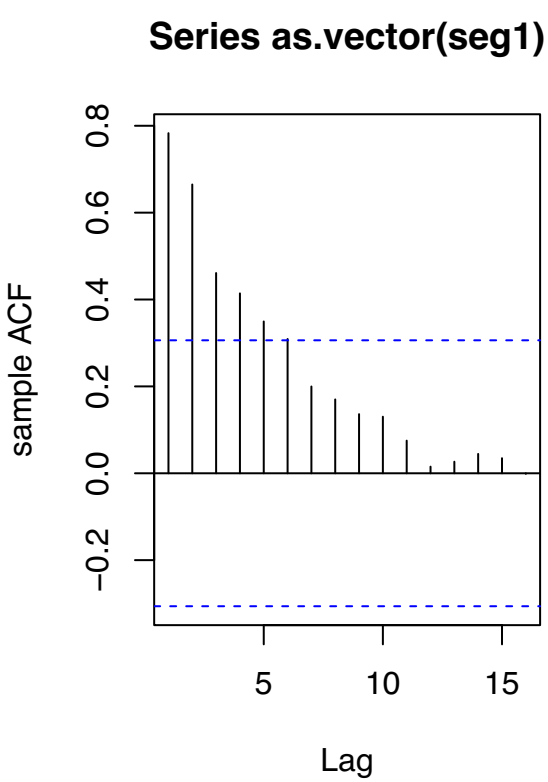
Segment 1

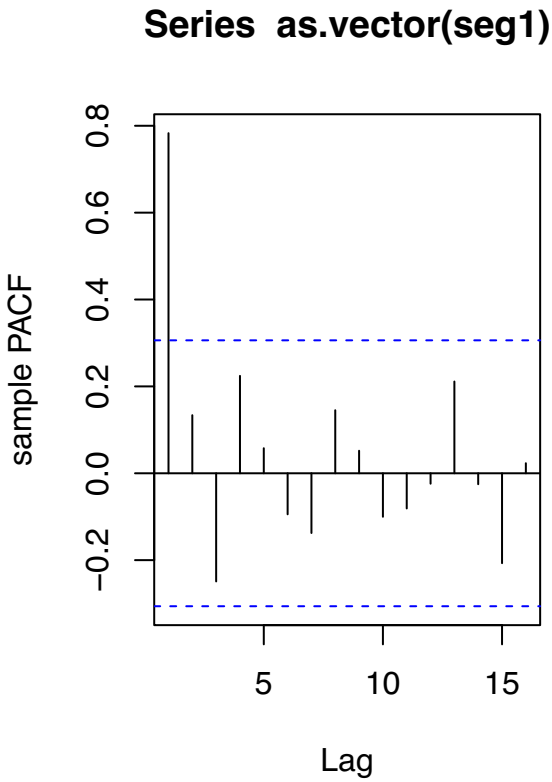
```
ts.plot(seg1)
```


```
cummean_seg1 = cumsum(as.vector(seg1)) / seq_along(as.vector(seg1))
plot(cummean_seg1, type = "l", xlab = "Time", ylab = "VIX", main = "Plot of segment 1")
```




```
par(mfrow=c(1, 2))
acf(as.vector(seg1), xlab = "Lag", ylab = "sample ACF")
pacf(as.vector(seg1), xlab = "Lag", ylab = "sample PACF")
```






```
adf.test(as.vector(seg1))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  as.vector(seg1)
## Dickey-Fuller = -3.2888, Lag order = 3, p-value = 0.08739
## alternative hypothesis: stationary
```

```
kpss.test(as.vector(seg1), null = "Trend")
```

```
## Warning in kpss.test(as.vector(seg1), null = "Trend"): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  as.vector(seg1)
## KPSS Trend = 0.057246, Truncation lag parameter = 3, p-value = 0.1
```

```
kpss.test(as.vector(seg1), null = "Level")
```

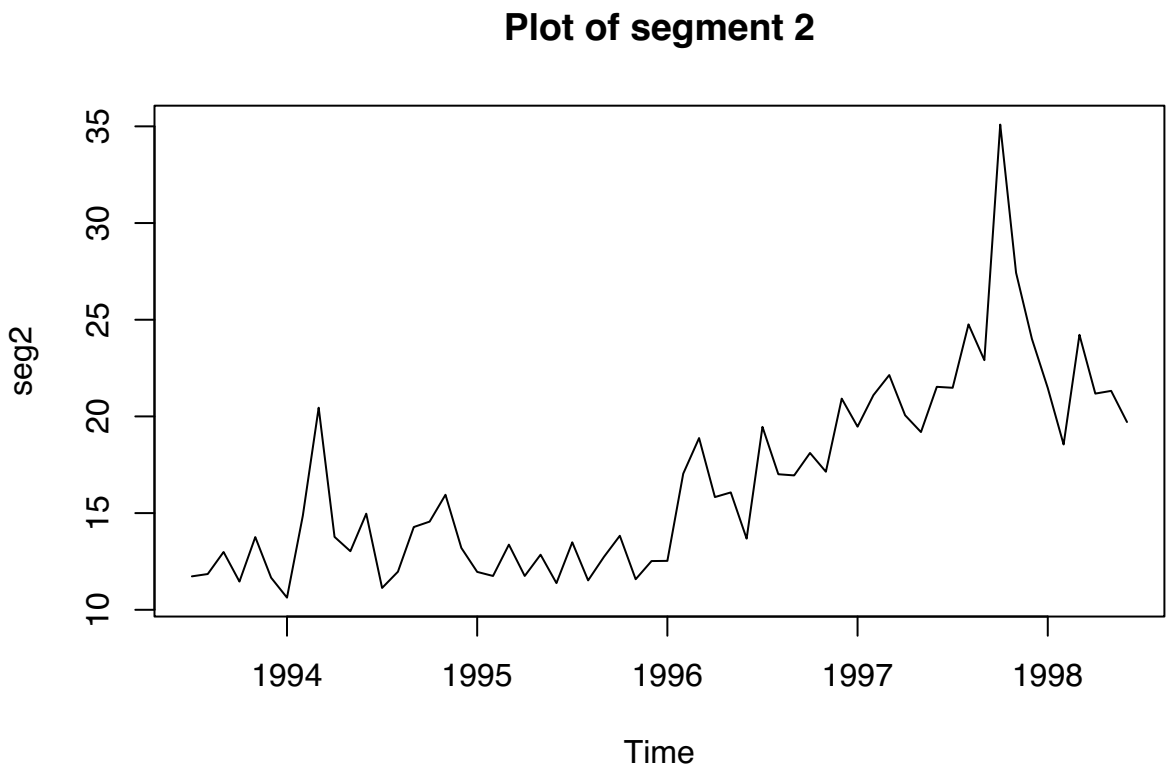
```
## Warning in kpss.test(as.vector(seg1), null = "Level"): p-value smaller than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  as.vector(seg1)
## KPSS Level = 0.78741, Truncation lag parameter = 3, p-value = 0.01
```

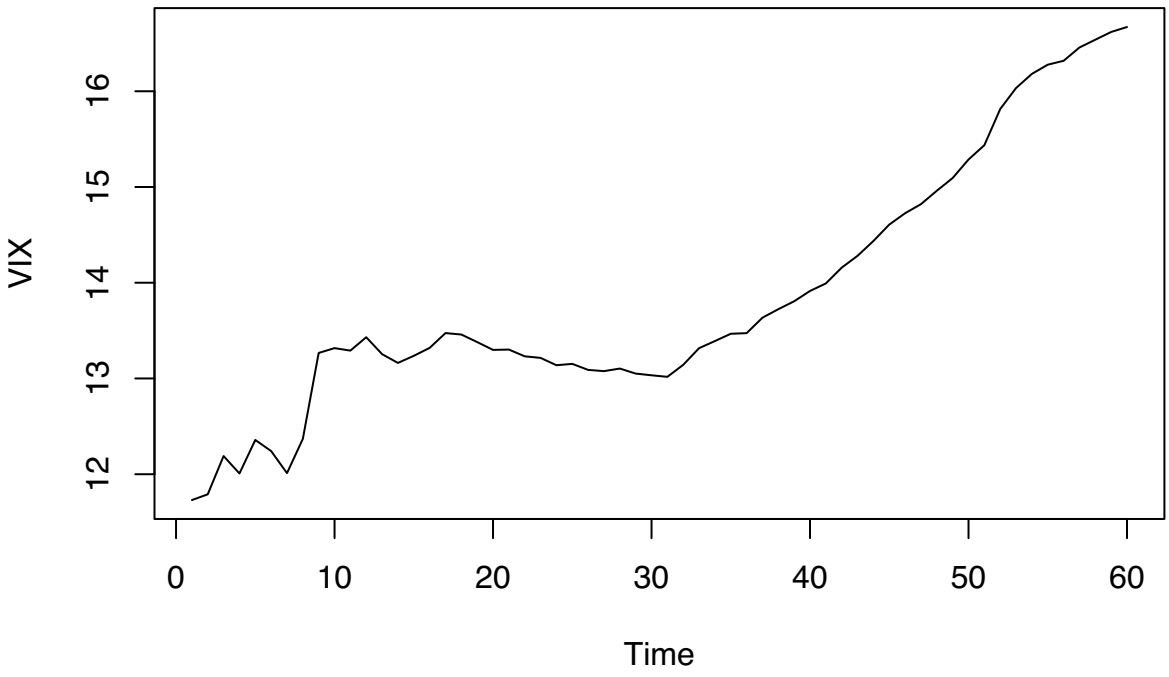
The segment 1 doesn't show stationary.

Segment 2

```
ts.plot(seg2, main = "Plot of segment 2")
```

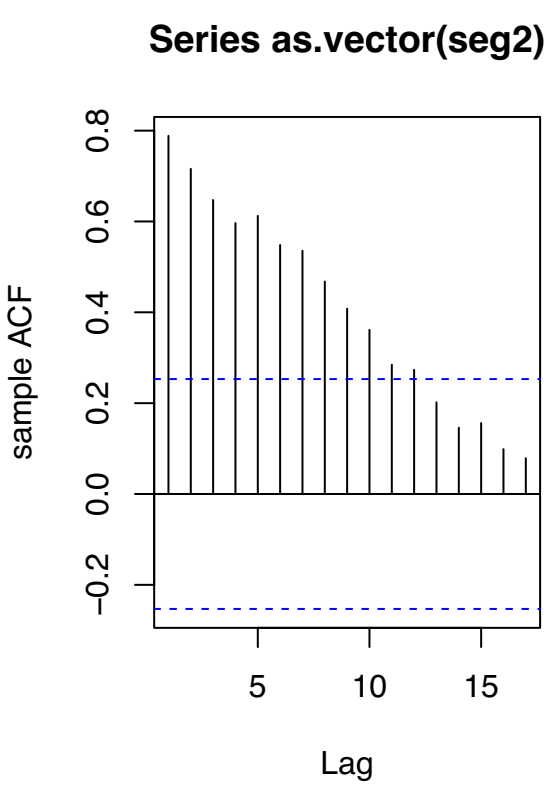


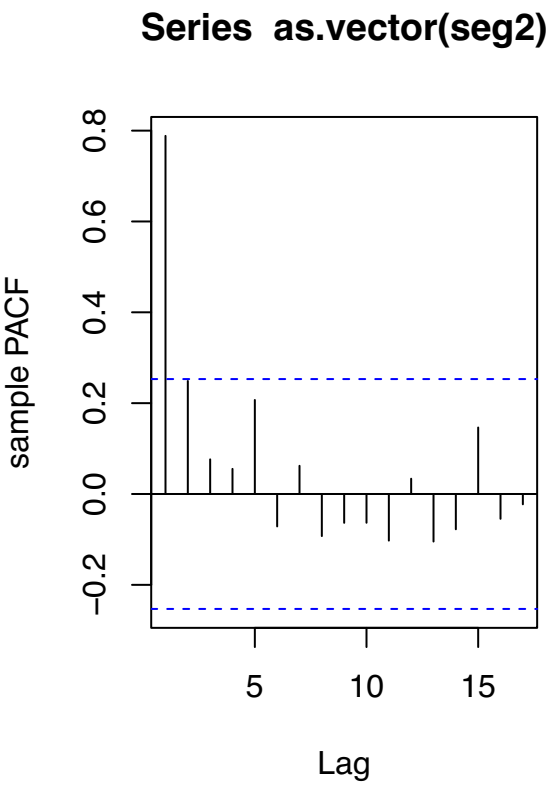

```
cummean_seg2 = cumsum(as.vector(seg2)) / seq_along(as.vector(seg2))
plot(cummean_seg2, type = "l", xlab = "Time", ylab = "VIX")
```




```
par(mfrow=c(1, 2))
acf(as.vector(seg2), xlab = "Lag", ylab = "sample ACF")
```

```
pacf(as.vector(seg2), xlab = "Lag", ylab = "sample PACF")
```






```
adf.test(as.vector(seg2))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  as.vector(seg2)
## Dickey-Fuller = -2.4773, Lag order = 3, p-value = 0.3819
## alternative hypothesis: stationary
```

```
kpss.test(as.vector(seg2), null = "Trend")
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  as.vector(seg2)
## KPSS Trend = 0.2022, Truncation lag parameter = 3, p-value = 0.01517
```

```
kpss.test(as.vector(seg2), null = "Level")
```

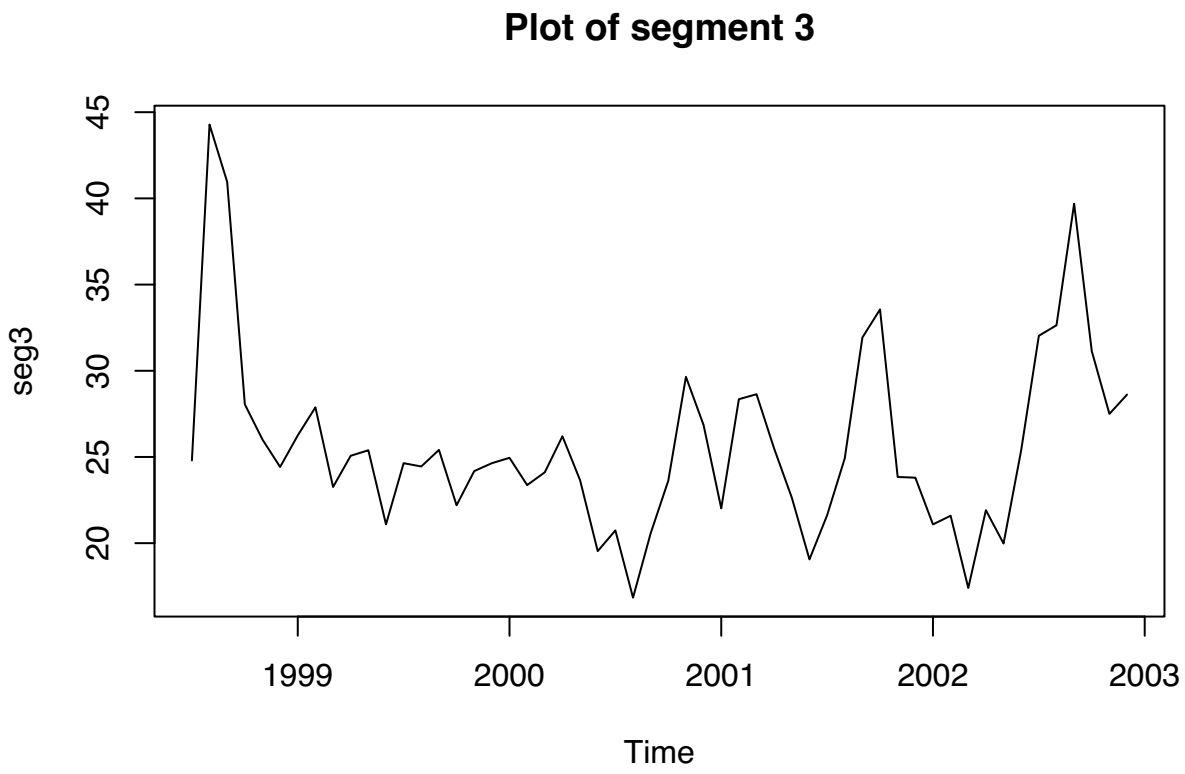
```
## Warning in kpss.test(as.vector(seg2), null = "Level"): p-value smaller than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  as.vector(seg2)
## KPSS Level = 1.2138, Truncation lag parameter = 3, p-value = 0.01
```

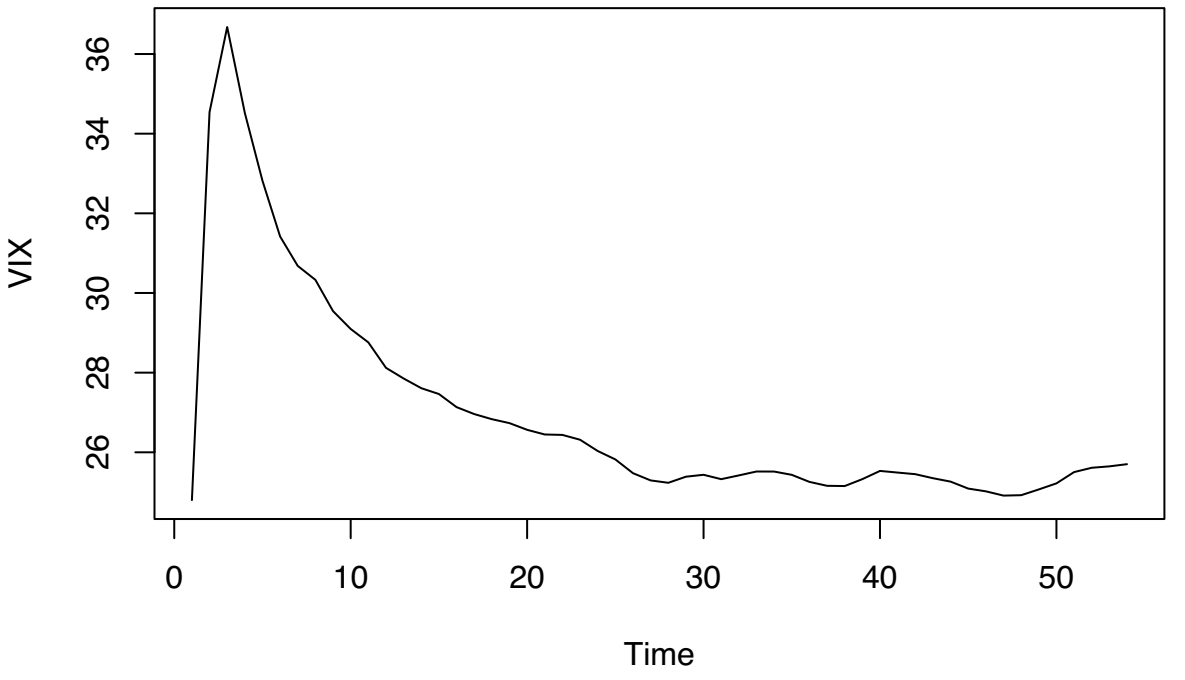
The segment 2 doesn't show stationary because the fail to reject the non-stationary test, but we can reject the stationary test.

Segment 3

```
ts.plot(seg3, main = "Plot of segment 3")
```

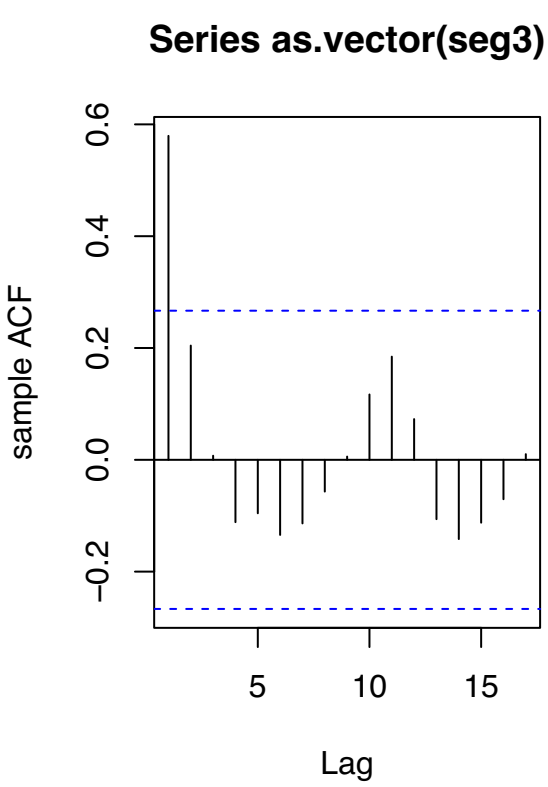


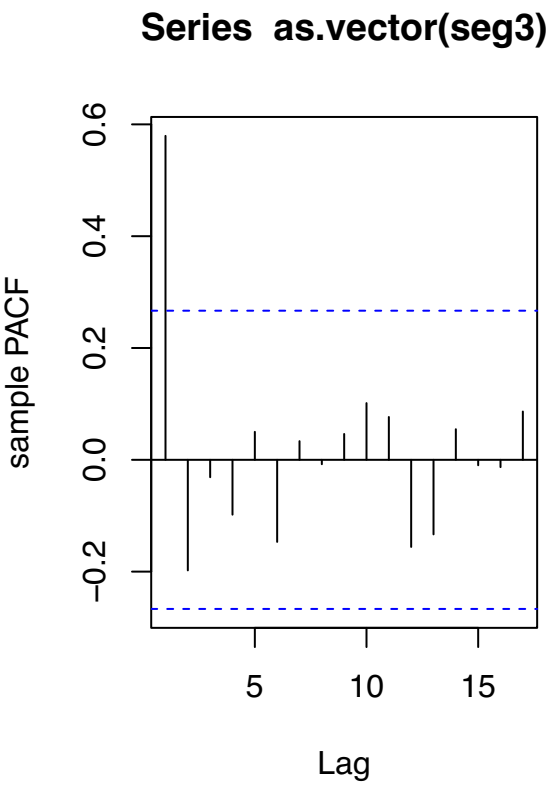

```
cummean_seg3 = cumsum(as.vector(seg3)) / seq_along(as.vector(seg3))
plot(cummean_seg3, type = "l", xlab = "Time", ylab = "VIX")
```




```
par(mfrow=c(1, 2))
acf(as.vector(seg3), xlab = "Lag", ylab = "sample ACF")
```

```
pacf(as.vector(seg3), xlab = "Lag", ylab = "sample PACF")
```






```
adf.test(as.vector(seg3))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  as.vector(seg3)
## Dickey-Fuller = -3.3716, Lag order = 3, p-value = 0.06967
## alternative hypothesis: stationary
```

```
kpss.test(as.vector(seg3), null = "Trend")
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  as.vector(seg3)
## KPSS Trend = 0.15193, Truncation lag parameter = 3, p-value = 0.04506
```

```
kpss.test(as.vector(seg3), null = "Level")
```

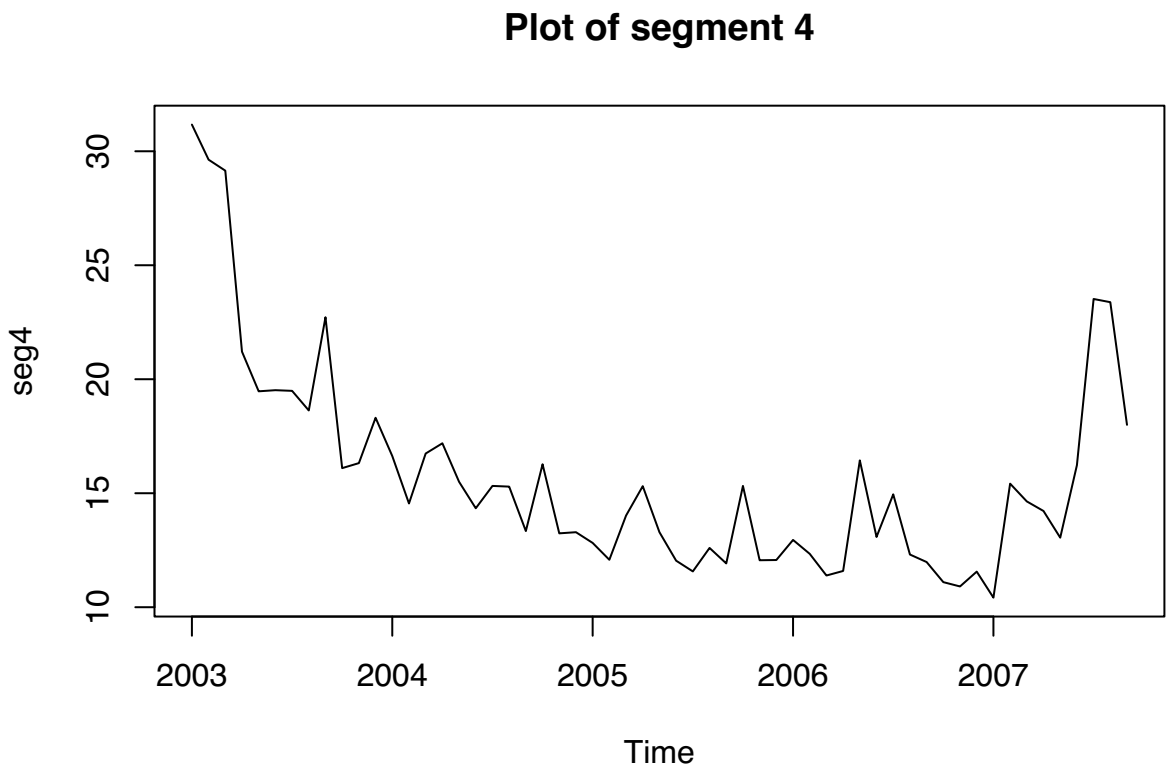
```
## Warning in kpss.test(as.vector(seg3), null = "Level"): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  as.vector(seg3)
## KPSS Level = 0.15336, Truncation lag parameter = 3, p-value = 0.1
```

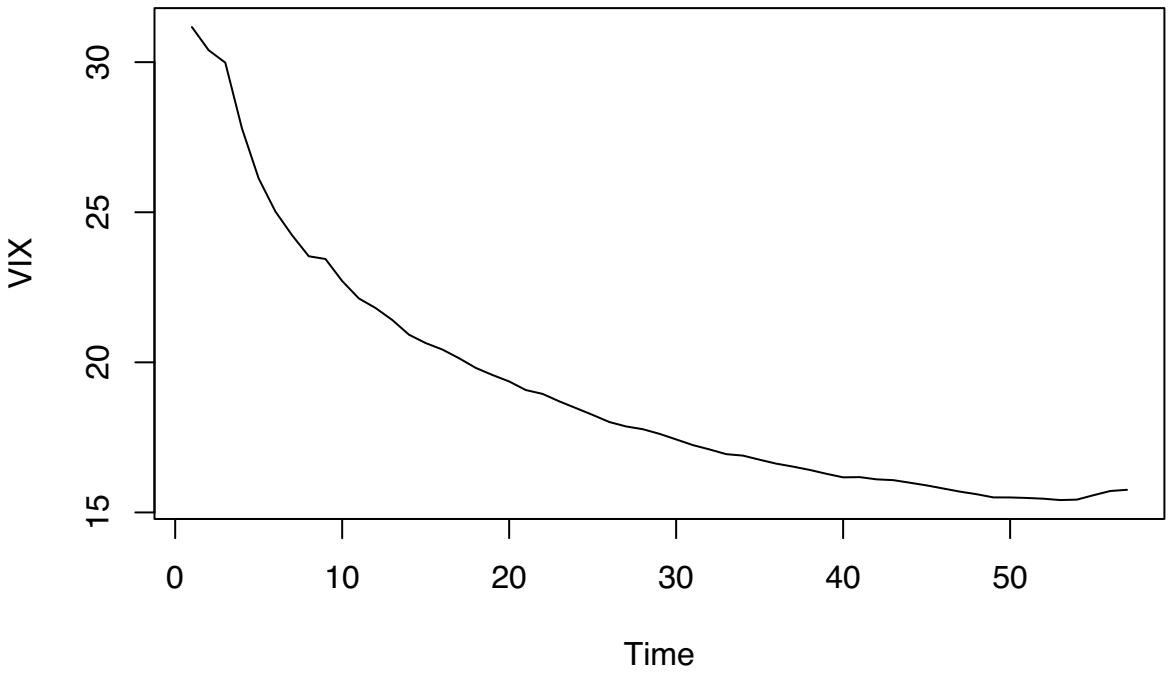
The segment 3 doesn't show stationary because we fail to reject the non-stationary test and stationary test, which derive a contradiction.

Segment 4

```
ts.plot(seg4, main = "Plot of segment 4")
```

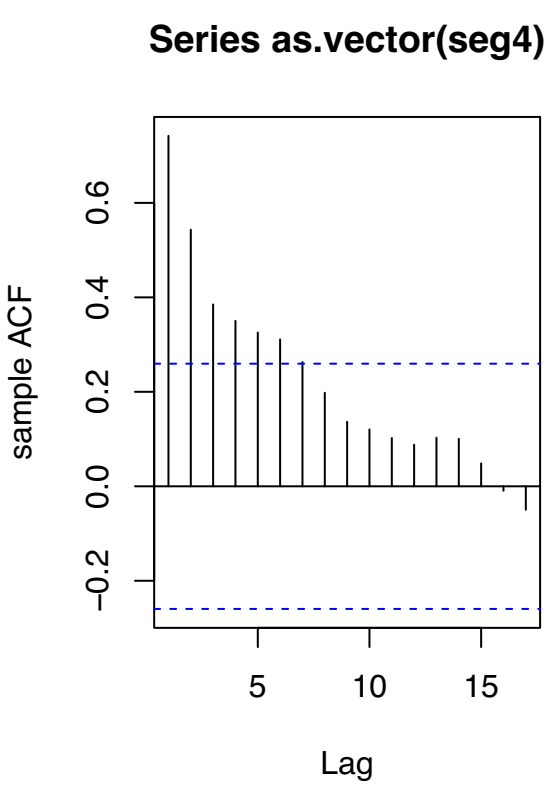


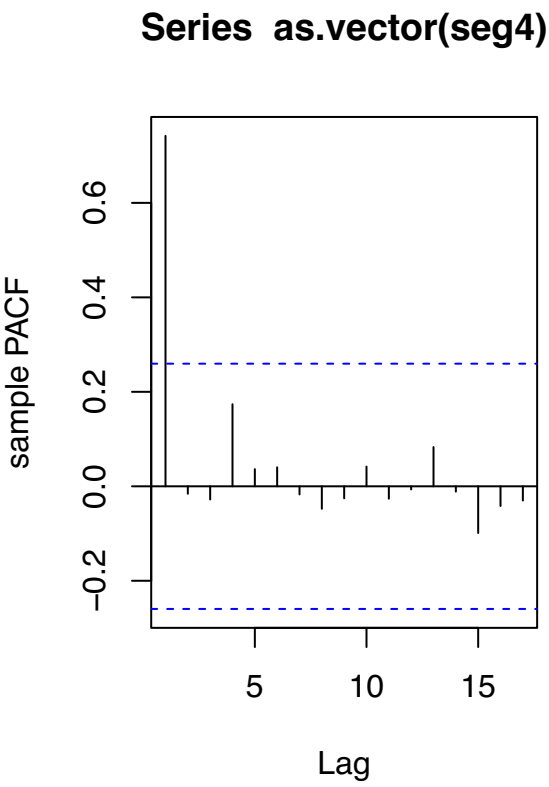

```
cummean_seg4 = cumsum(as.vector(seg4)) / seq_along(as.vector(seg4))
plot(cummean_seg4, type = "l", xlab = "Time", ylab = "VIX")
```




```
par(mfrow=c(1, 2))
acf(as.vector(seg4), xlab = "Lag", ylab = "sample ACF")
```

```
pacf(as.vector(seg4), xlab = "Lag", ylab = "sample PACF")
```






```
adf.test(as.vector(seg4))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  as.vector(seg4)
## Dickey-Fuller = -1.5539, Lag order = 3, p-value = 0.7546
## alternative hypothesis: stationary
```

```
kpss.test(as.vector(seg4), null = "Trend")
```

```
## Warning in kpss.test(as.vector(seg4), null = "Trend"): p-value smaller than
## printed p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  as.vector(seg4)
## KPSS Trend = 0.29202, Truncation lag parameter = 3, p-value = 0.01
```

```
kpss.test(as.vector(seg4), null = "Level")
```

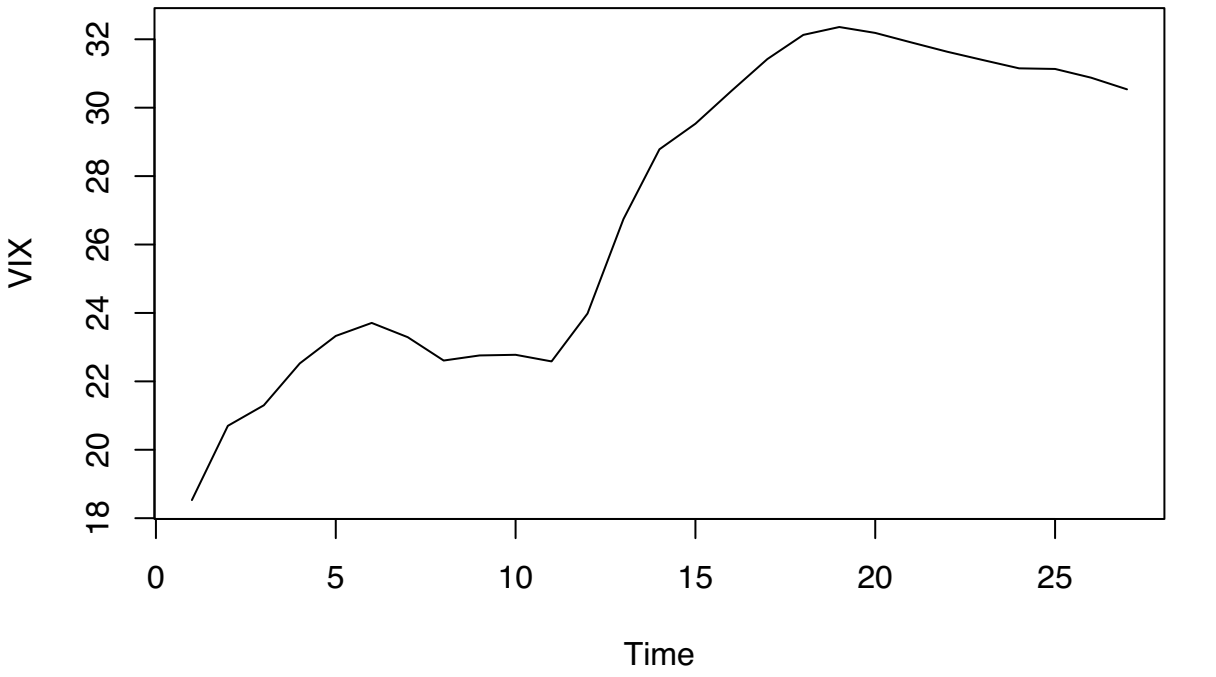
```
##
##  KPSS Test for Level Stationarity
##
## data:  as.vector(seg4)
## KPSS Level = 0.68093, Truncation lag parameter = 3, p-value = 0.01528
```

The segment 4 doesn't show stationary. We fail to reject the stationary test non-stationary test, and reject the stationary test.

Segment 5

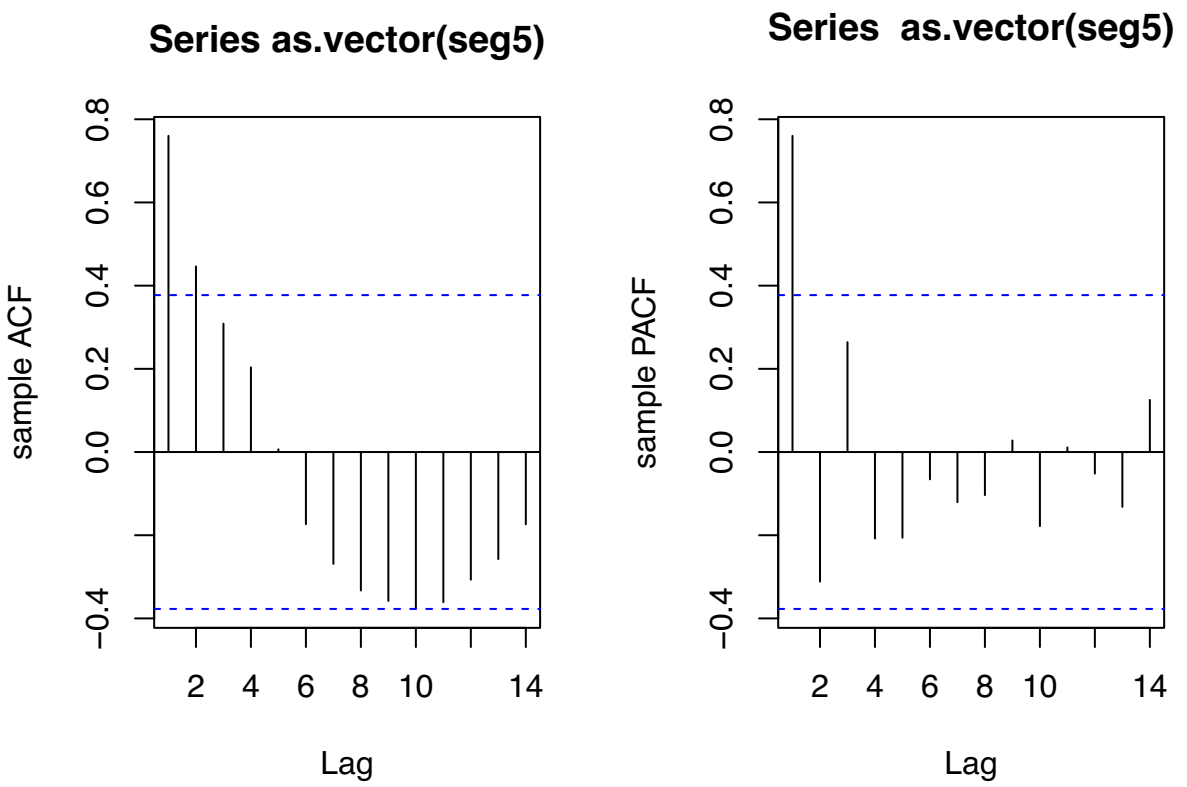
```
ts.plot(seg5, main = "Plot of segment 5")
```


```
cummean_seg5 = cumsum(as.vector(seg5)) / seq_along(as.vector(seg5))
plot(cummean_seg5, type = "l", xlab = "Time", ylab = "VIX")
```




```
par(mfrow=c(1, 2))
acf(as.vector(seg5), xlab = "Lag", ylab = "sample ACF")
```

```
pacf(as.vector(seg5), xlab = "Lag", ylab = "sample PACF")
```




```
adf.test(as.vector(seg5))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  as.vector(seg5)
## Dickey-Fuller = -1.2007, Lag order = 2, p-value = 0.8776
## alternative hypothesis: stationary
```

```
kpss.test(as.vector(seg5), null = "Trend")
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  as.vector(seg5)
## KPSS Trend = 0.17354, Truncation lag parameter = 2, p-value = 0.02705
```

```
kpss.test(as.vector(seg5), null = "Level")
```

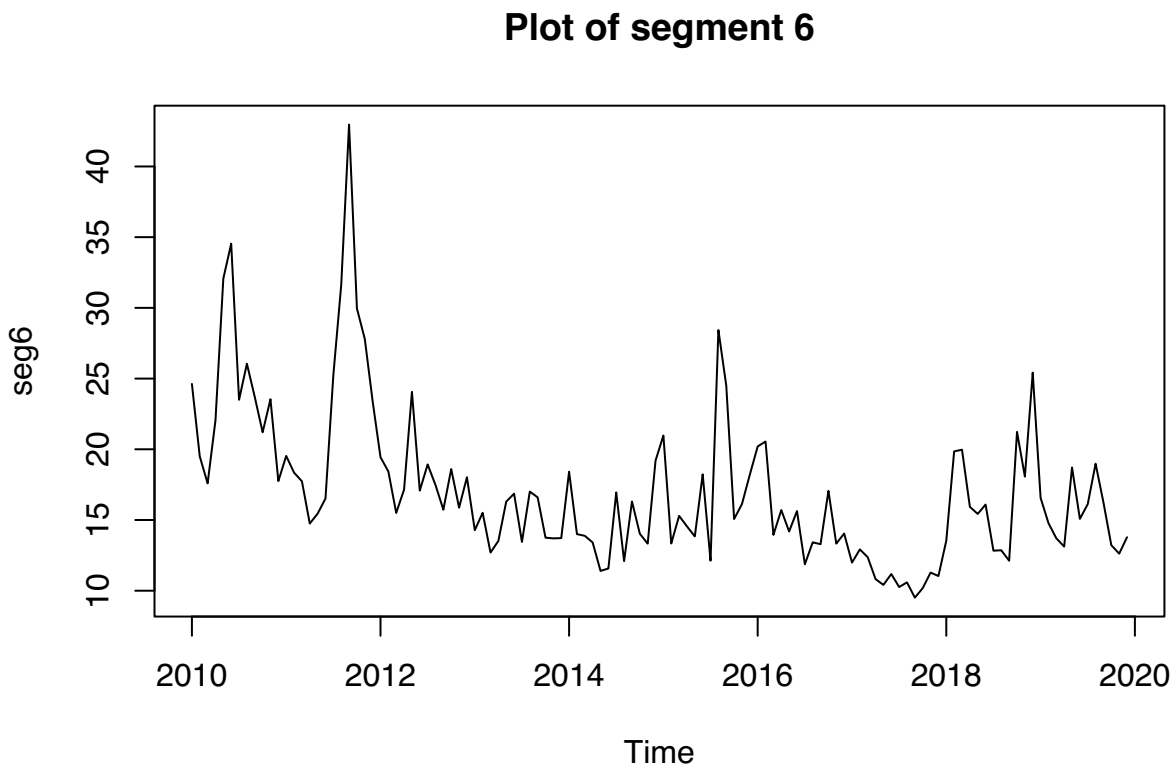
```
## Warning in kpss.test(as.vector(seg5), null = "Level"): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  as.vector(seg5)
## KPSS Level = 0.22133, Truncation lag parameter = 2, p-value = 0.1
```

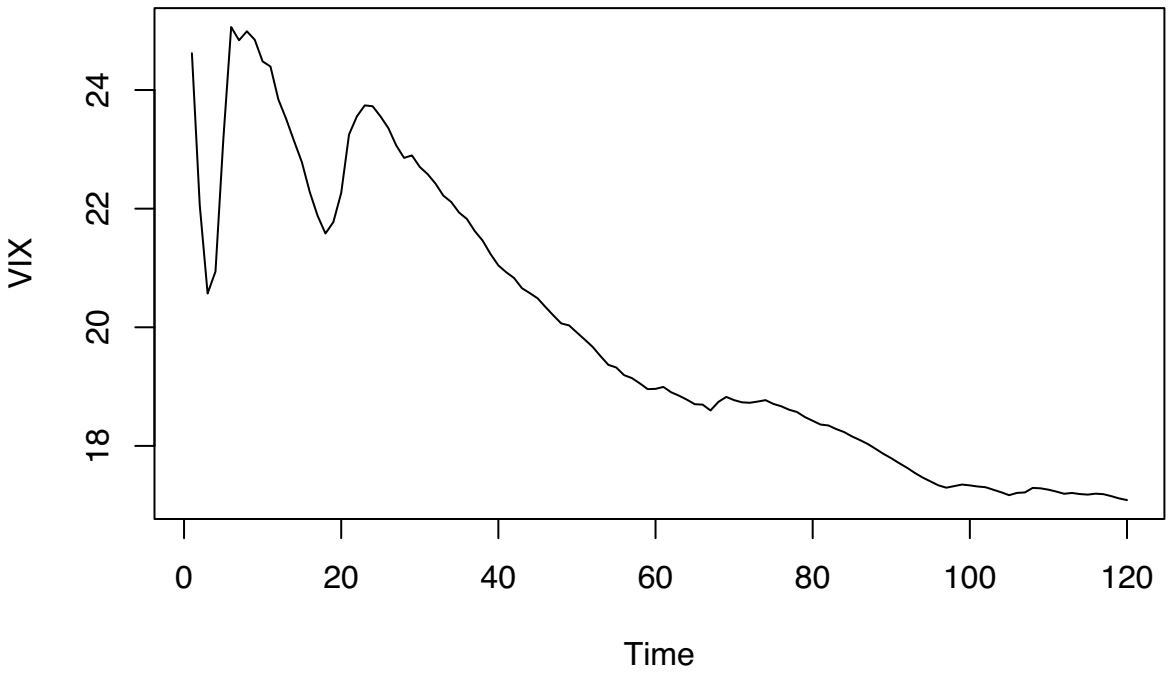
The segment 5 doesn't show stationary. We fail to reject the non-stationary test, and we reject the trend stationary test, but fail to reject the level stationary test. Such that we derive a contradiction.

Segment 6

```
ts.plot(seg6, main = "Plot of segment 6")
```

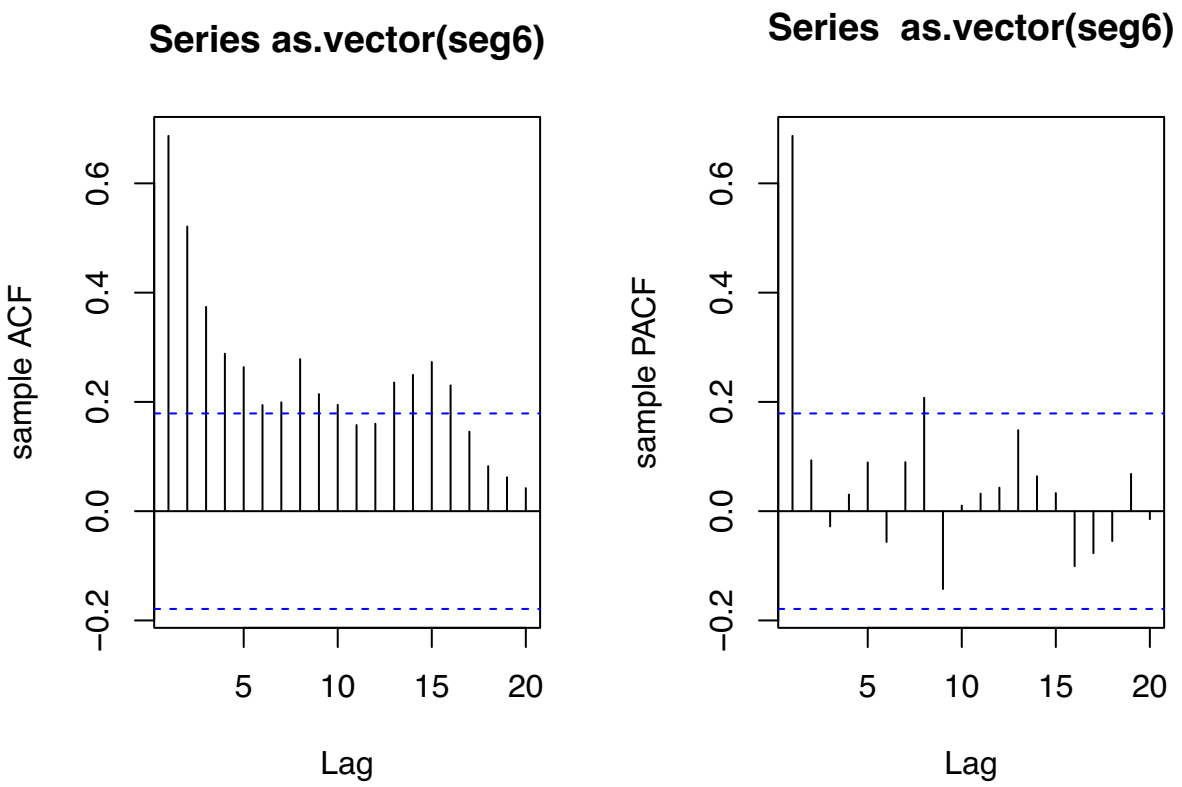



```
cummean_seg6 = cumsum(as.vector(seg6)) / seq_along(as.vector(seg6))
plot(cummean_seg6, type = "l", xlab = "Time", ylab = "VIX")
```




```
par(mfrow=c(1, 2))
acf(as.vector(seg6), xlab = "Lag", ylab = "sample ACF")
```

```
pacf(as.vector(seg6), xlab = "Lag", ylab = "sample PACF")
```




```
adf.test(as.vector(seg6))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  as.vector(seg6)
## Dickey-Fuller = -3.7023, Lag order = 4, p-value = 0.02704
## alternative hypothesis: stationary
```

```
kpss.test(as.vector(seg6), null = "Trend")
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  as.vector(seg6)
## KPSS Trend = 0.18331, Truncation lag parameter = 4, p-value = 0.02226
```

```
kpss.test(as.vector(seg6), null = "Level")
```

```
## Warning in kpss.test(as.vector(seg6), null = "Level"): p-value smaller than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  as.vector(seg6)
## KPSS Level = 1.0142, Truncation lag parameter = 4, p-value = 0.01
```

The segment 6 doesn't show stationary. The non-stationary test and kpss test show contradicting result.

Since we are forecasting the future value of VIX Index, so we only need to consider the last segment.

Segment 6 Analysis

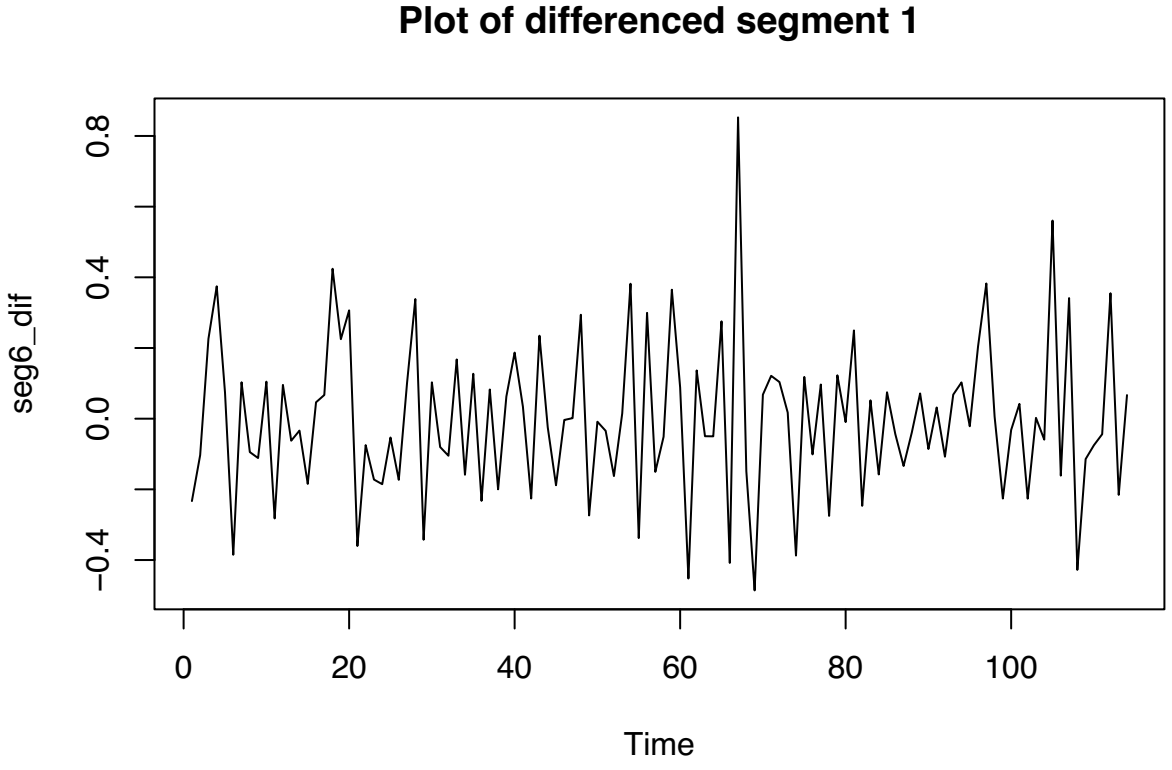
We have already shown the original segment 6 time series is non-stationary, so we should transform the dataset. We should split the dataset first. The time series of transformed segment 6 have more randomness patterns. The candidate models are ARIMA(2, 0, 2), ARIMA(2, 0, 3), ARIMA(3, 0, 2), ARIMA(1, 0, 4) and ARIMA(1, 0, 1). The same situation happens again. We should compare AIC, BIC and harmonic mean. Finally, model 1 has rhe smallest AIC and BIC. Such that the most appropriate model is ARIMA(2, 1, 2). We predict the length of the test dataset steps. The last plot is the forecast of one year later, which can be a measurement of market fear for investors.

```
seg6_training = seg6[1:round(length(seg6)*0.96, 0)]
seg6_test = seg6[-c(1:round(length(seg6)*0.96, 0))]
```

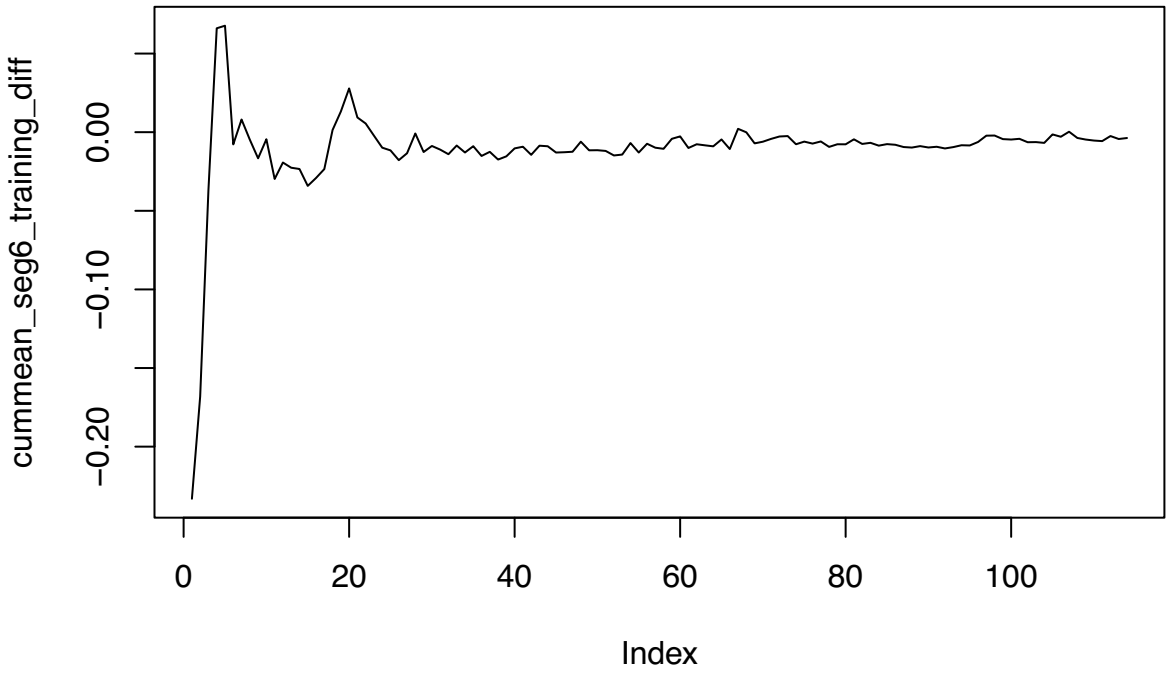
```
seg6
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 2010 24.62 19.50 17.59 22.05 32.07 34.54 23.50 26.05 23.70 21.20 23.54 17.75
## 2011 19.53 18.35 17.74 14.75 15.45 16.52 25.25 31.62 42.96 29.96 27.80 23.40
## 2012 19.44 18.43 15.50 17.15 24.06 17.08 18.93 17.47 15.73 18.60 15.87 18.02
## 2013 14.28 15.51 12.70 13.52 16.30 16.86 13.45 17.01 16.60 13.75 13.70 13.72
## 2014 18.41 14.00 13.88 13.41 11.40 11.57 16.95 12.09 16.31 14.03 13.33 19.20
## 2015 20.97 13.34 15.29 14.55 13.84 18.23 12.12 28.43 24.50 15.07 16.13 18.21
## 2016 20.20 20.55 13.95 15.70 14.19 15.63 11.87 13.42 13.29 17.06 13.33 14.04
## 2017 11.99 12.92 12.37 10.82 10.41 11.18 10.26 10.59  9.51 10.18 11.28 11.04
## 2018 13.54 19.85 19.97 15.93 15.43 16.09 12.83 12.86 12.12 21.23 18.07 25.42
## 2019 16.57 14.78 13.71 13.12 18.71 15.08 16.12 18.98 16.24 13.22 12.62 13.78
```

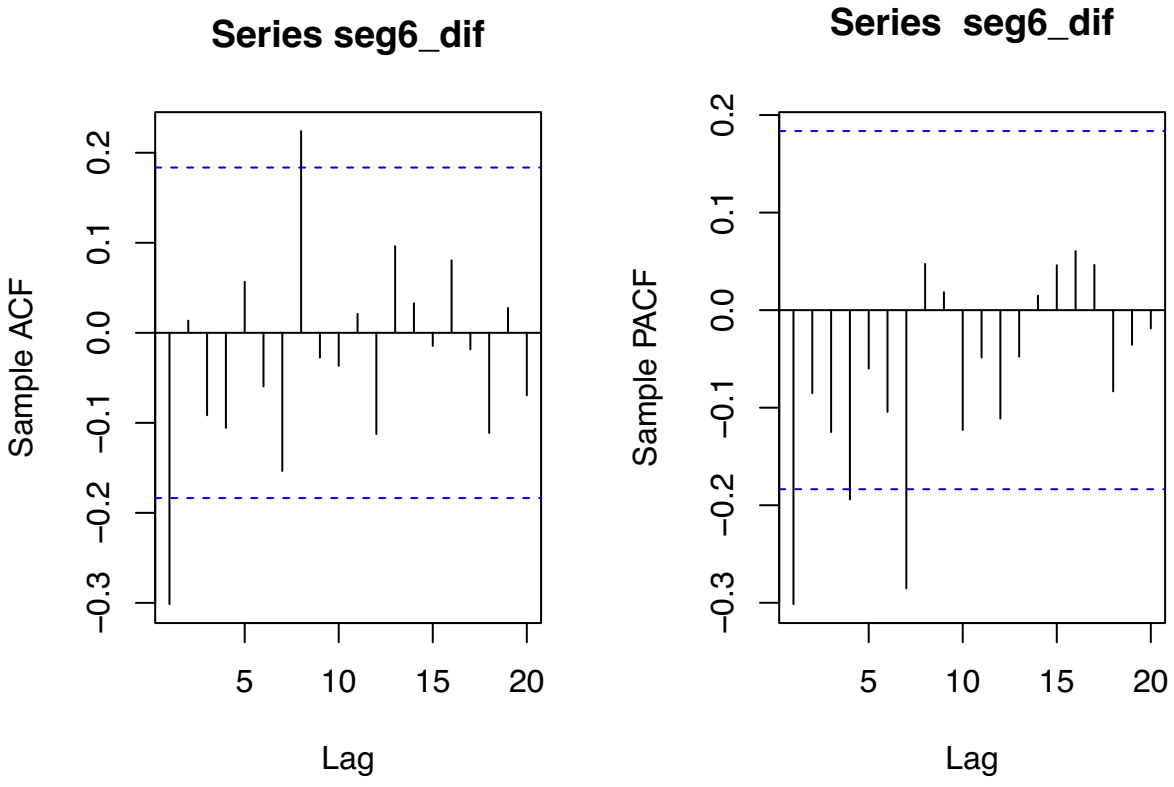
```
seg6_dif = diff(log(seg6_training))
ts.plot(seg6_dif, main = "Plot of differenced segment 1")
```



```
cummean_seg6_training_diff = cumsum(seg6_dif) / seq_along(seg6_dif)
plot(cummean_seg6_training_diff, type = 'l')
```




```
par(mfrow=c(1, 2))
acf(seg6_dif, xlab = "Lag", ylab = "Sample ACF")
pacf(seg6_dif, xlab = "Lag", ylab = "Sample PACF")
```



```r
adf.test(seg6_dif)
```

```
## Warning in adf.test(seg6_dif): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  seg6_dif
## Dickey-Fuller = -6.5032, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```r
kpss.test(seg6_dif, null = "Level")
```

```
## Warning in kpss.test(seg6_dif, null = "Level"): p-value greater than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  seg6_dif
## KPSS Level = 0.041526, Truncation lag parameter = 4, p-value = 0.1
```

```r
kpss.test(seg6_dif, null = "Trend")
```

```
## Warning in kpss.test(seg6_dif, null = "Trend"): p-value greater than printed p-
## value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  seg6_dif
## KPSS Trend = 0.022531, Truncation lag parameter = 4, p-value = 0.1
```

```r
shapiro.test(seg6_dif)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  seg6_dif
## W = 0.97715, p-value = 0.04808
```

Based on the result of the non-stationary test and stationary test. The series is stationary.

We repeat the same steps above.

```r
eacf(seg6_dif, ar.max = 5, ma.max = 5) # compute the EACF
```

```
## AR/MA
##   0 1 2 3 4 5
## 0 x o o o o o
## 1 x o o o o o
## 2 x o o o o o
## 3 x o x o o o
## 4 x x x o o o
## 5 x o x o o o
```

```r
aic_seg6 = matrix(0, 5, 5)
bic_seg6 = matrix(0, 5, 5)
for (i in 0:4) for (j in 0:4){
```

```
  fit_seg = arima(seg6_dif, order = c(i, 0, j), method = "ML", include.mean = TRUE)
  aic_seg6[i+1, j+1] = fit_seg$aic
  bic_seg6[i+1, j+1] = BIC(fit_seg)
}
```

```
## Warning in log(s2): NaNs produced
```

```
aic_vec_seg6 = sort(unmatrix(aic_seg6, byrow = FALSE))[1:13]
bic_vec_seg6 = sort(unmatrix(bic_seg6, byrow = FALSE))[1:13]
aic_vec_seg6 # lowest one is 2, 2
```

```
##     r2:c2     r3:c2     r4:c4     r2:c3     r1:c4     r2:c4     r4:c2     r3:c3
## -36.66052 -36.64897 -36.47171 -36.31303 -35.06263 -34.99239 -34.96978 -34.86920
##     r4:c5     r5:c4     r1:c5     r3:c4     r3:c5
## -34.54426 -34.52127 -34.23399 -33.08102 -33.06671
```

```
bic_vec_seg6 # lowest one is 2, 2
```

```
##     r2:c2     r3:c2     r2:c3     r1:c2     r1:c4     r1:c3     r2:c1     r2:c4
## -23.71573 -20.96798 -20.63204 -19.60253 -19.38163 -17.69475 -17.00148 -16.57520
##     r4:c2     r3:c3     r1:c5     r3:c1     r4:c4
## -16.55259 -16.45201 -15.81680 -13.13063 -12.58212
```

Candidate Models

ARIMA(2, 0, 2)

```
fit_s1 = arima(seg6_dif, order = c(2, 0, 2), method = "ML", include.mean = TRUE) # fit the model
fit_s1
```

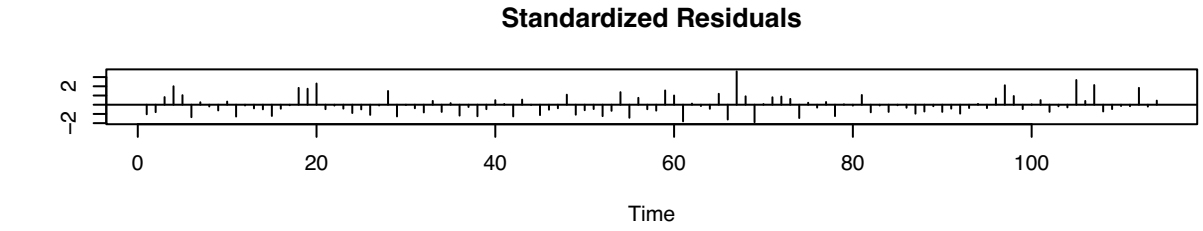
```
##
## Call:
## arima(x = seg6_dif, order = c(2, 0, 2), include.mean = TRUE, method = "ML")
##
## Coefficients:
##          ar1     ar2      ma1      ma2  intercept
##       0.2841  0.2773  -0.7585  -0.2414    -0.0043
## s.e.  0.4505  0.2696   0.4598   0.4585     0.0015
##
## sigma^2 estimated as 0.03841:  log likelihood = 22.43,  aic = -34.87
```

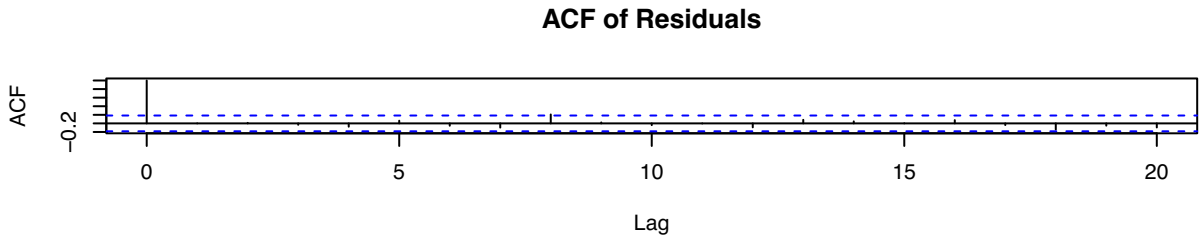
```
confint(fit_s1)
```

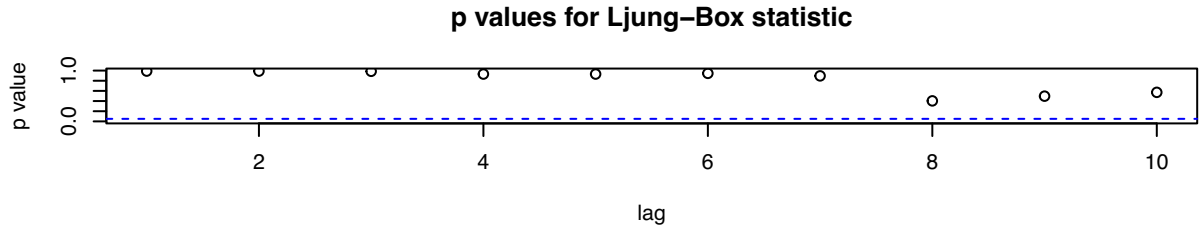
```
##                   2.5 %       97.5 %
## ar1       -0.598885022  1.167042382
## ar2       -0.251118664  0.805634642
## ma1       -1.659631443  0.142677073
## ma2       -1.139983413  0.657198960
## intercept -0.007221888 -0.001419212
```

Residual analysis:

```
tsdiag(fit_s1)
```





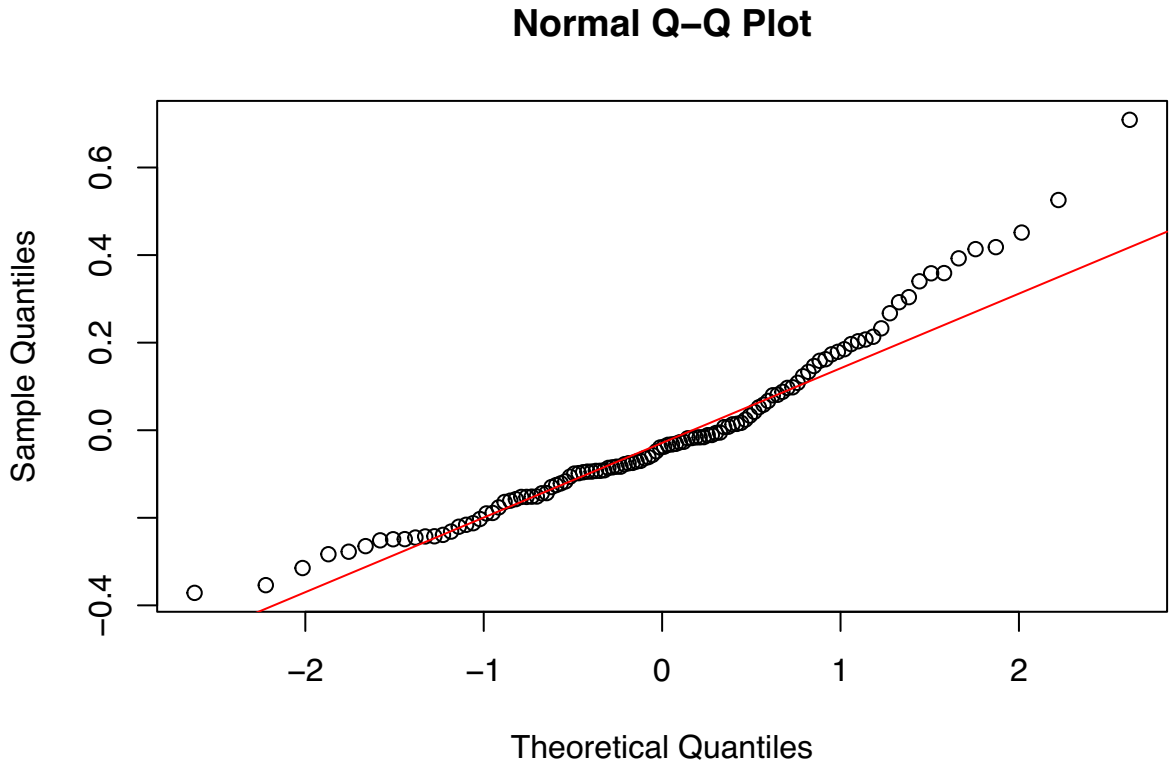



```
Box.test(fit_s1$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_s1$residuals
## X-squared = 0.00015144, df = 1, p-value = 0.9902
```

The residuals plot and Ljung-Box test show that residuals are mostly in randomness. The p-values for Ljung-Box Test are all insignificant where we fail to reject the null hypothesis. The result conclude that residuals are independently residuals for all lags. Normality:

```
qqnorm(fit_s1$residuals)
qqline(fit_s1$residuals, col = "red")
```




```
shapiro.test(fit_s1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_s1$residuals
## W = 0.9455, p-value = 0.0001555
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(2, 0, 3)

```
fit_s2 = arima(seg6_dif, order = c(2, 0, 3), method = "ML", include.mean = TRUE) # fit the model
fit_s2
```

```
##
## Call:
## arima(x = seg6_dif, order = c(2, 0, 3), include.mean = TRUE, method = "ML")
##
## Coefficients:
##          ar1     ar2      ma1      ma2      ma3  intercept
##       0.4230  0.1263  -0.8919  -0.0081  -0.1000    -0.0044
## s.e.  0.5077  0.3722   0.5053   0.5892   0.1931     0.0014
##
## sigma^2 estimated as 0.0383:  log likelihood = 22.54,  aic = -33.08
```

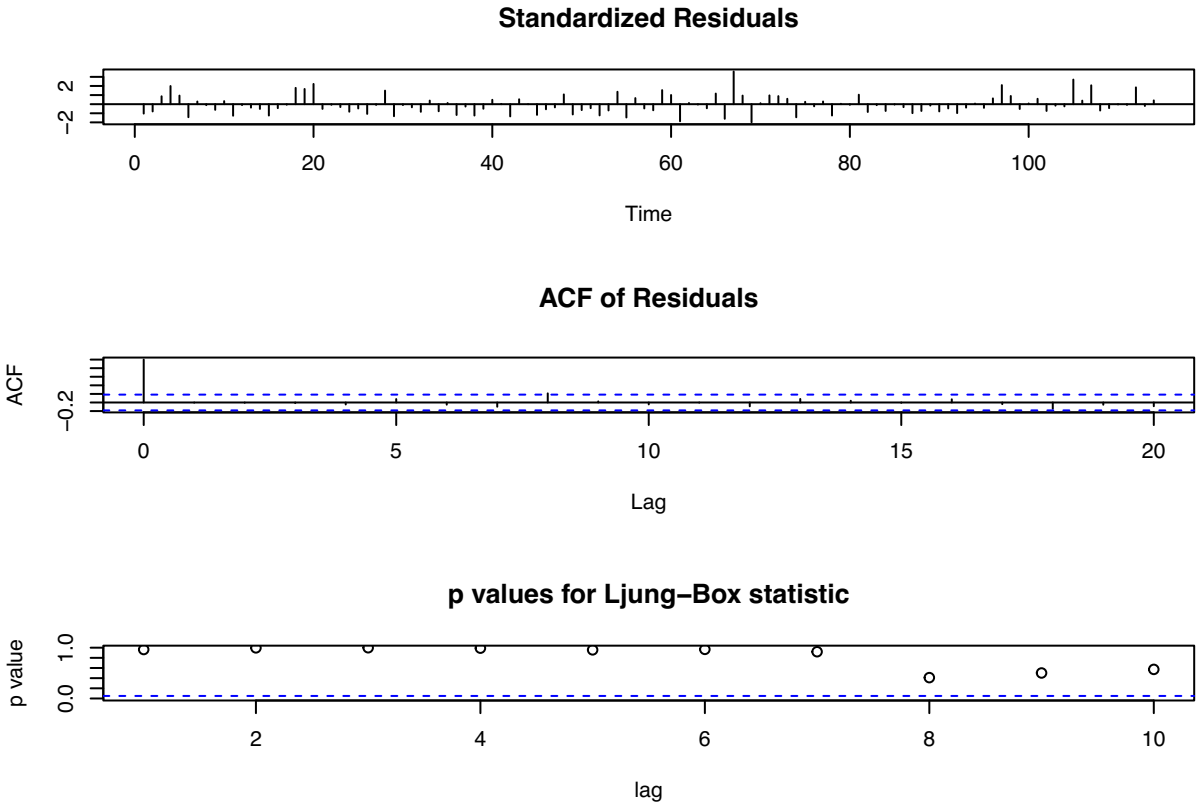
```
confint(fit_s2)
```

```
##                  2.5 %       97.5 %
## ar1       -0.572045359  1.417970580
## ar2       -0.603216466  0.855764402
```

```
## ma1       -1.882308823  0.098523548
## ma2       -1.163004367  1.146775028
## ma3       -0.478533968  0.278566565
## intercept -0.007138039 -0.001589548
```

Residual analysis:

```
tsdiag(fit_s2)
```

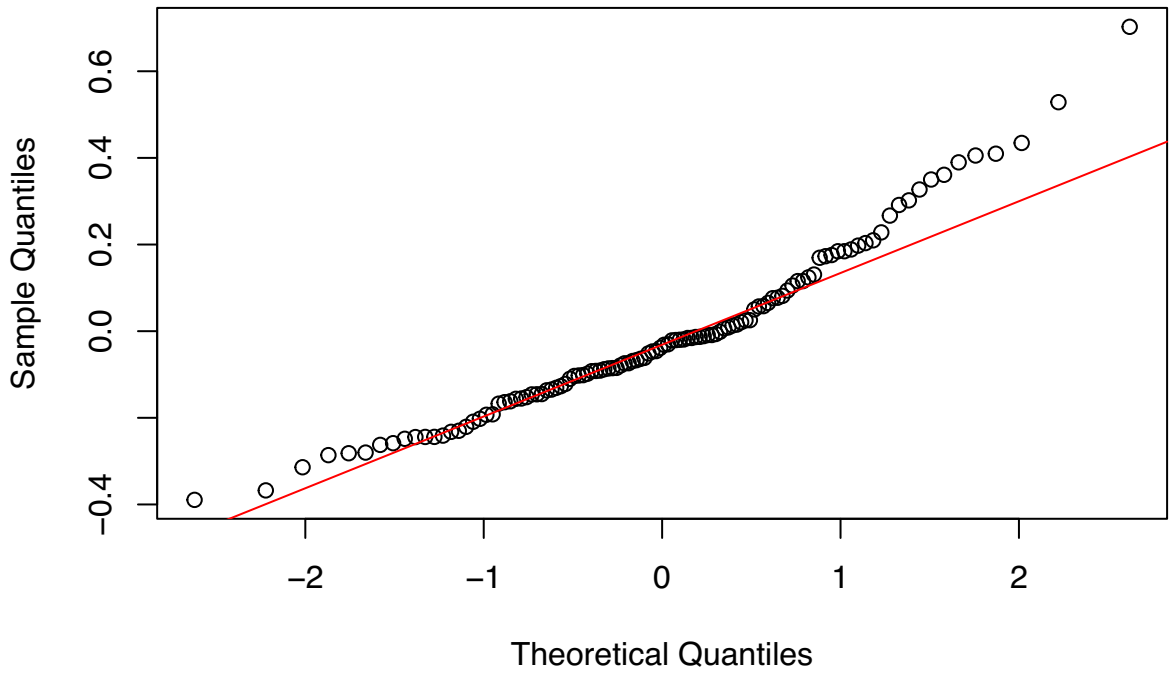



```
Box.test(fit_s2$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_s2$residuals
## X-squared = 0.0020864, df = 1, p-value = 0.9636
```

The residuals plot and Ljung-Box test show that residuals are mostly in randomness. The p-values for Ljung-Box Test are all insignificant where we fail to reject the null hypothesis. The result conclude that residuals are independently residuals for all lags. Normality:

```
qqnorm(fit_s2$residuals)
qqline(fit_s2$residuals, col = "red")
```




```
shapiro.test(fit_s2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_s2$residuals
## W = 0.95213, p-value = 0.0004569
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(3, 0, 2)

```
fit_s3 = arima(seg6_dif, order = c(3, 0, 2), method = "ML", include.mean = TRUE) # fit the model
```

```
## Warning in log(s2): NaNs produced
```

```
fit_s3
```

```
##
## Call:
## arima(x = seg6_dif, order = c(3, 0, 2), include.mean = TRUE, method = "ML")
##
## Coefficients:
##          ar1     ar2      ar3      ma1     ma2  intercept
##       0.5442  0.1512  -0.0556  -1.0151  0.0151    -0.0043
## s.e.  0.8566  0.4595   0.1444   0.8564  0.8557     0.0014
##
## sigma^2 estimated as 0.03835:  log likelihood = 22.48,  aic = -32.97
```

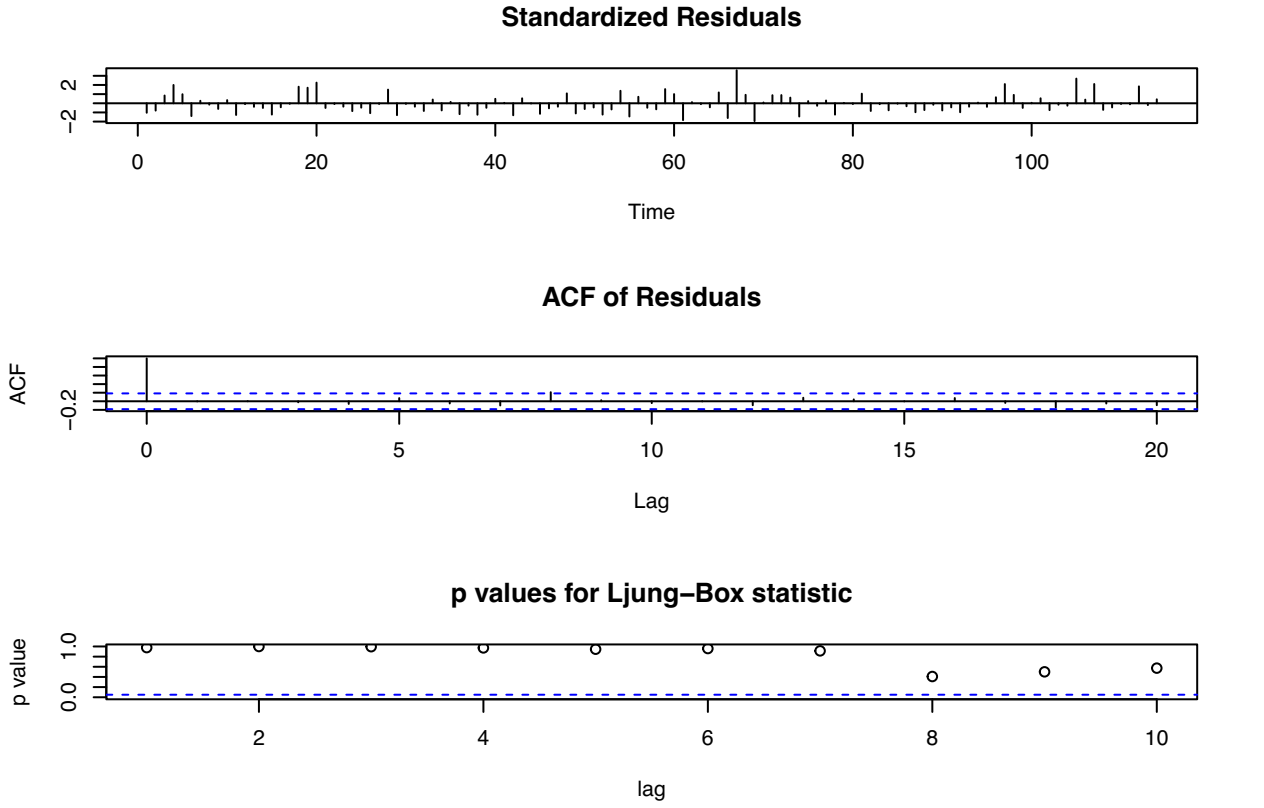
```
confint(fit_s3)
```

```
##                  2.5 %       97.5 %
```

```
## ar1       -1.134836369  2.223165324
## ar2       -0.749408630  1.051719840
## ar3       -0.338557755  0.227359014
## ma1       -2.693677192  0.663502417
## ma2       -1.661985676  1.692174817
## intercept -0.007166379 -0.001523445
```

Residual analysis:

```
tsdiag(fit_s3)
```

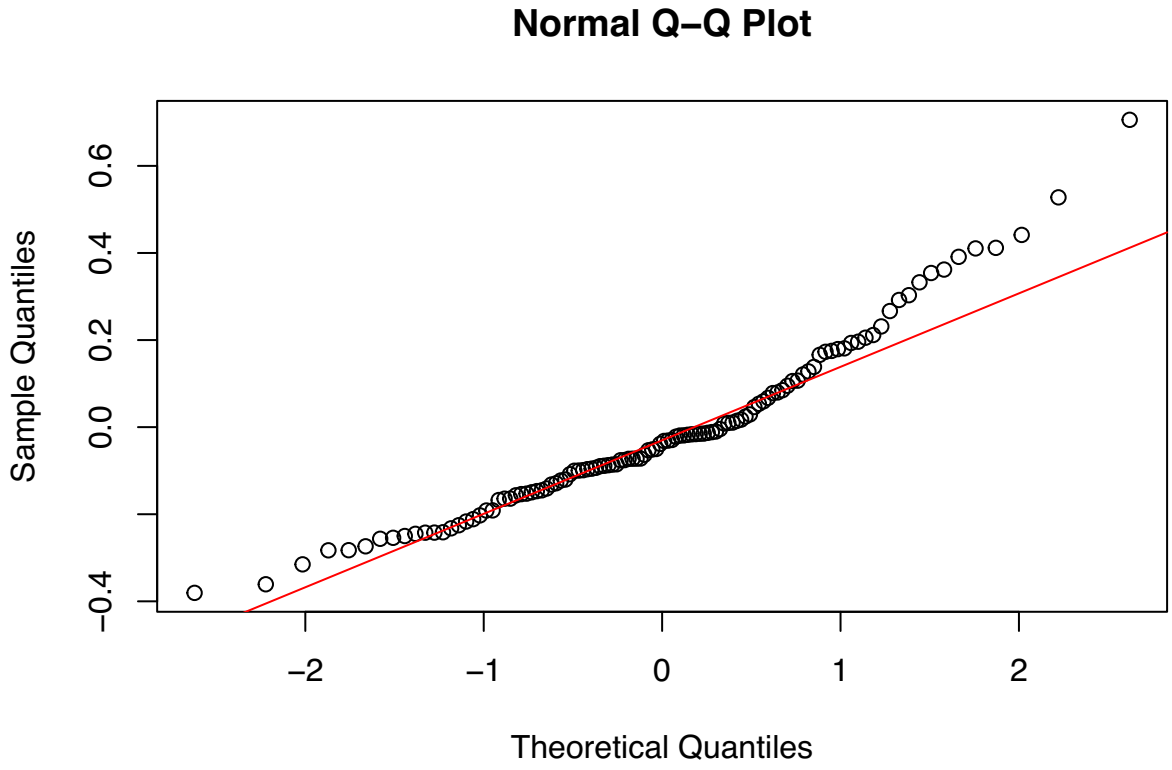



```
Box.test(fit_s3$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_s3$residuals
## X-squared = 0.00080808, df = 1, p-value = 0.9773
```

The residuals plot and Ljung-Box test show that residuals are mostly in randomness. The p-values for Ljung-Box Test are all insignificant where we fail to reject the null hypothesis. The result conclude that residuals are independently residuals for all lags. Normality:

```
qqnorm(fit_s3$residuals)
qqline(fit_s3$residuals, col = "red")
```




```
shapiro.test(fit_s3$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_s3$residuals
## W = 0.9489, p-value = 0.000268
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(1, 0, 4)

```
fit_s4 = arima(seg6_dif, order = c(1, 0, 4), method = "ML", include.mean = TRUE) # fit the model
fit_s4
```

```
##
## Call:
## arima(x = seg6_dif, order = c(1, 0, 4), include.mean = TRUE, method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2      ma3      ma4  intercept
##       -0.4597  0.0383  -0.2638  -0.2864  -0.2336    -0.0041
## s.e.   0.4128  0.4122   0.1833   0.1167   0.1445     0.0036
##
## sigma^2 estimated as 0.03921:  log likelihood = 22.35,  aic = -32.71
```

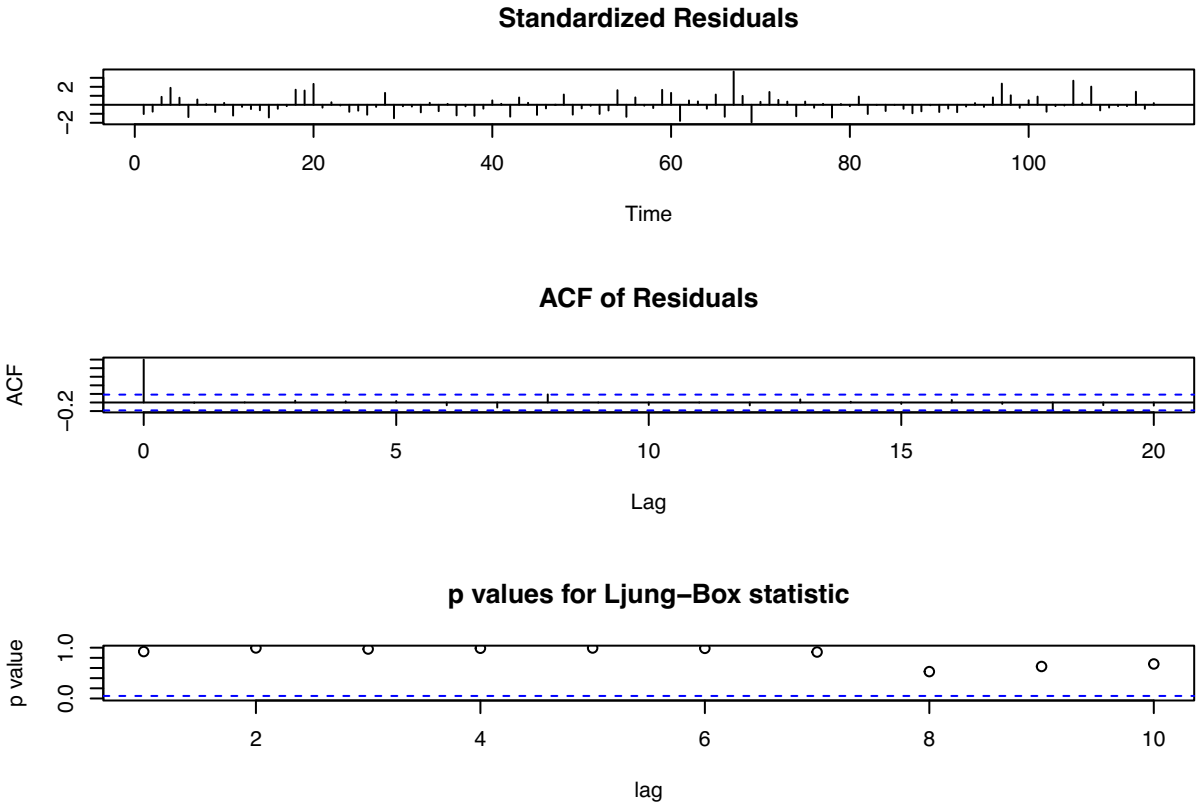
```
confint(fit_s4)
```

```
##                 2.5 %       97.5 %
## ar1       -1.26872532  0.349403013
## ma1       -0.76963891  0.846150456
```

```
## ma2       -0.62307040  0.095495105
## ma3       -0.51507311 -0.057667608
## ma4       -0.51679439  0.049691404
## intercept -0.01103235  0.002925913
```

Residual analysis:

```
tsdiag(fit_s4)
```

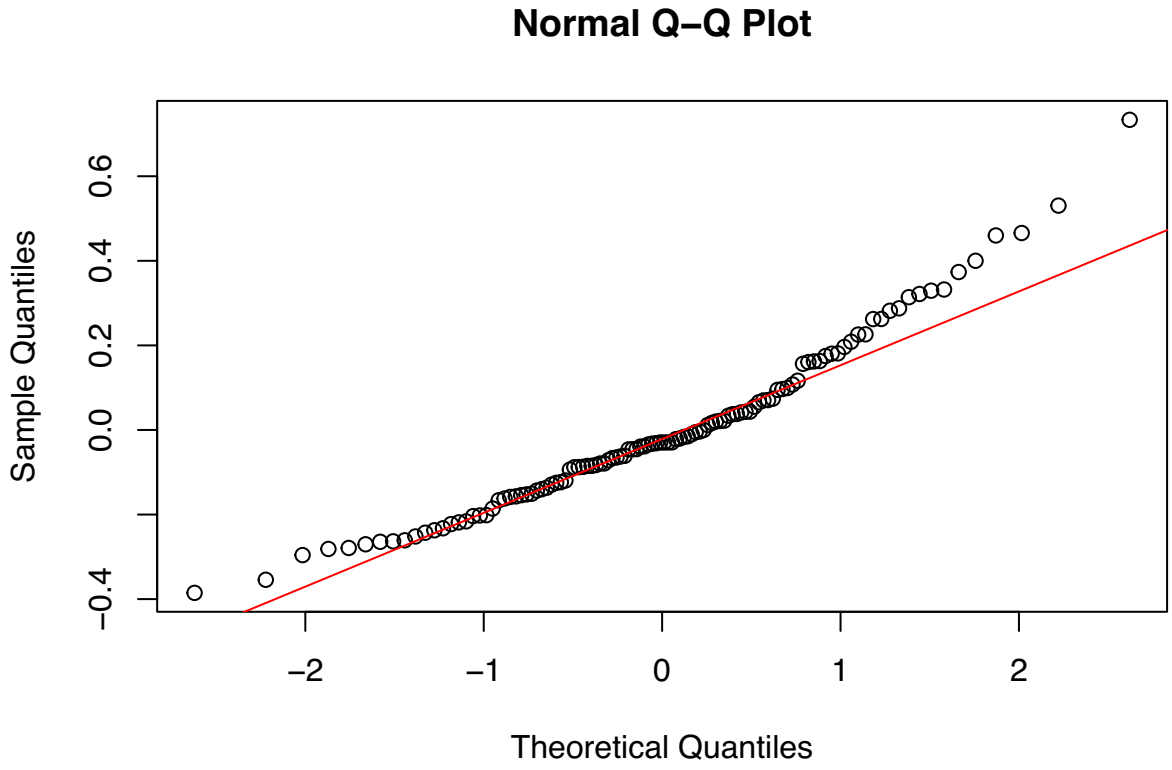



```
Box.test(fit_s4$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_s4$residuals
## X-squared = 0.0092743, df = 1, p-value = 0.9233
```

The residuals plot and Ljung-Box test show that residuals are mostly in randomness. The p-values for Ljung-Box Test are all insignificant where we fail to reject the null hypothesis. The result conclude that residuals are independently residuals for all lags. Normality:

```
qqnorm(fit_s4$residuals)
qqline(fit_s4$residuals, col = "red")
```




```
shapiro.test(fit_s4$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_s4$residuals
## W = 0.95668, p-value = 0.0009907
```

The shapiro test for residuals implies that non-normally distributed.

ARIMA(1, 0, 1)

```
fit_s5 = arima(seg6_dif, order = c(1, 0, 1), method = "ML", include.mean = TRUE) # fit the model
fit_s5
```

```
##
## Call:
## arima(x = seg6_dif, order = c(1, 0, 1), include.mean = TRUE, method = "ML")
##
## Coefficients:
##          ar1      ma1  intercept
##       0.5991  -0.9999    -0.0043
## s.e.  0.0768   0.0430     0.0013
##
## sigma^2 estimated as 0.0391:  log likelihood = 21.33,  aic = -36.66
```

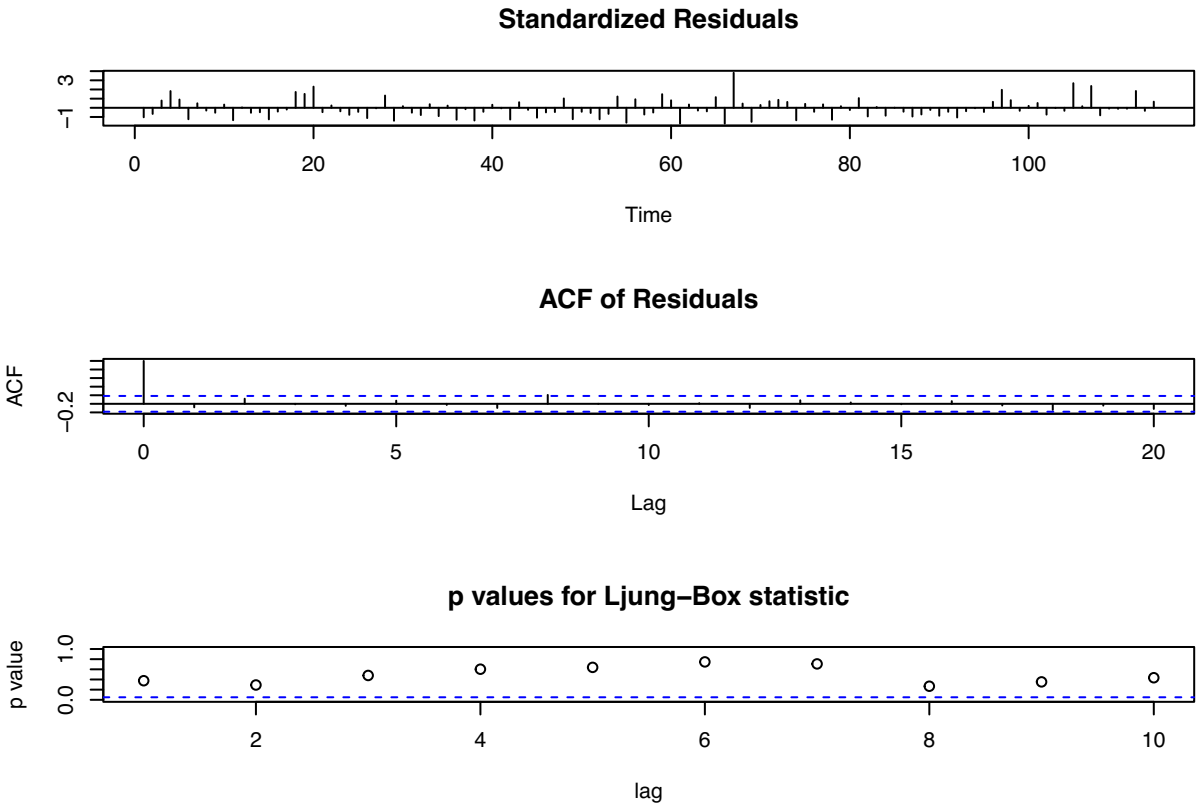
```
confint(fit_s5)
```

```
##                  2.5 %       97.5 %
## ar1        0.448623238  0.749602995
## ma1       -1.084092479 -0.915683864
```

```
## intercept -0.006965553 -0.001725829
```

Residual analysis:

```
tsdiag(fit_s5)
```

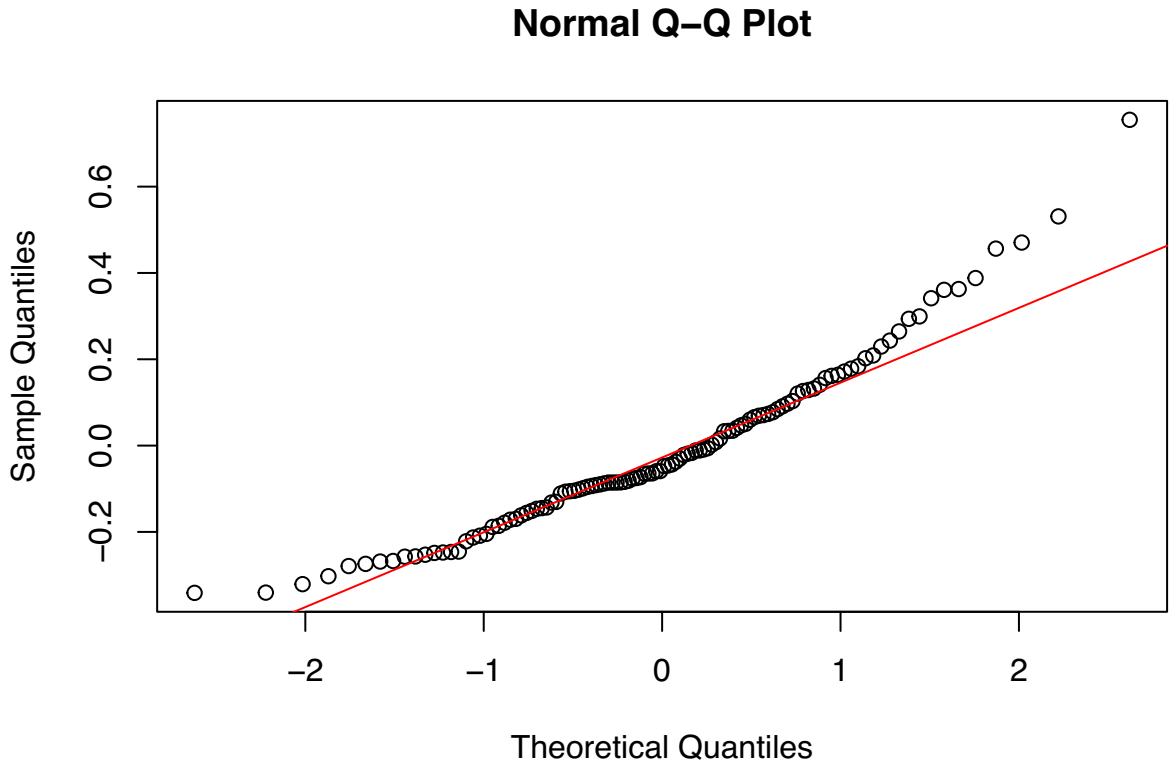



```
Box.test(fit_s5$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit_s5$residuals
## X-squared = 0.78453, df = 1, p-value = 0.3758
```

The residuals plot and Ljung-Box test show that residuals are mostly in randomness. The p-values for Ljung-Box Test are all insignificant where we fail to reject the null hypothesis. The result conclude that residuals are independently residuals for all lags. Normality:

```
qqnorm(fit_s5$residuals)
qqline(fit_s5$residuals, col = "red")
```




```
shapiro.test(fit_s5$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_s5$residuals
## W = 0.94681, p-value = 0.0001916
```

The shapiro test for residuals implies that non-normally distributed.

Since the models above are all non-normally distributed. To decide the final models, we will the same method as above: smallest value of AIC, BIC, Harmonic mean.

```
harmonic_mean = function(e){
  fval = length(e) / sum(1 / abs(e))
  return(fval)
}
print(harmonic_mean(as.numeric(fit_s1$residuals)))
```

```
## [1] 0.04972581
```

```
print(harmonic_mean(as.numeric(fit_s2$residuals)))
```

```
## [1] 0.03802945
```

```
print(harmonic_mean(as.numeric(fit_s3$residuals)))
```

```
## [1] 0.04937411
```

```
print(harmonic_mean(as.numeric(fit_s4$residuals)))
```

```
## [1] 0.007115335
```

```
print(harmonic_mean(as.numeric(fit_s5$residuals)))
```

```
## [1] 0.05060359
```

The smallest harmonic mean is model 3.

```
AIC_list = c(
AIC(fit_1),
AIC(fit_2),
AIC(fit_3),
AIC(fit_4),
AIC(fit_5),
AIC(fit_6),
AIC(fit_7),
AIC(fit_8))

BIC_list = c(
BIC(fit_1),
BIC(fit_2),
BIC(fit_3),
BIC(fit_4),
BIC(fit_5),
BIC(fit_6),
BIC(fit_7),
BIC(fit_8))

AIC_list
```

```
## [1] -208.9455 -208.8011 -208.9197 -210.5926 -209.1273 -208.9455 -207.1276
## [8] -212.5687
```

```
BIC_list
```

```
## [1] -182.0407 -178.0528 -166.6407 -187.5313 -182.2225 -182.0407 -176.3792
## [8] -193.3509
```

```
sort(AIC_list)
```

```
## [1] -212.5687 -210.5926 -209.1273 -208.9455 -208.9455 -208.9197 -208.8011
## [8] -207.1276
```

```
sort(BIC_list)
```

```
## [1] -193.3509 -187.5313 -182.2225 -182.0407 -182.0407 -178.0528 -176.3792
## [8] -166.6407
```

The smallest AIC is model 1. The smallest BIC is model 1.

Based on the result above, the final model we choose is ARIMA(2, 1, 2)

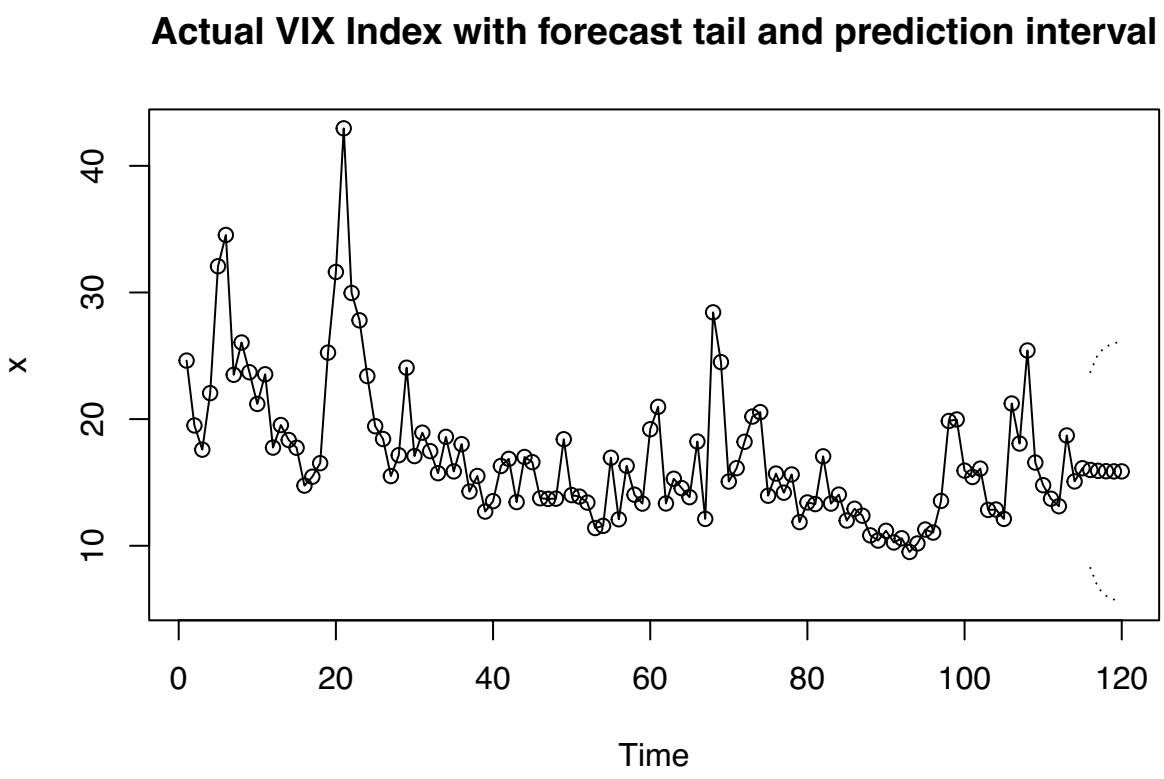
```
fit_seg_final = arima(seg6_training, order = c(2, 1, 2))
pre_seg6 = predict(fit_seg_final, n.ahead = length(seg6_test))
pre_seg6$pred
```

```
## Time Series:
## Start = 116
## End = 120
## Frequency = 1
## [1] 15.98811 15.91689 15.88350 15.87206 15.87221
```

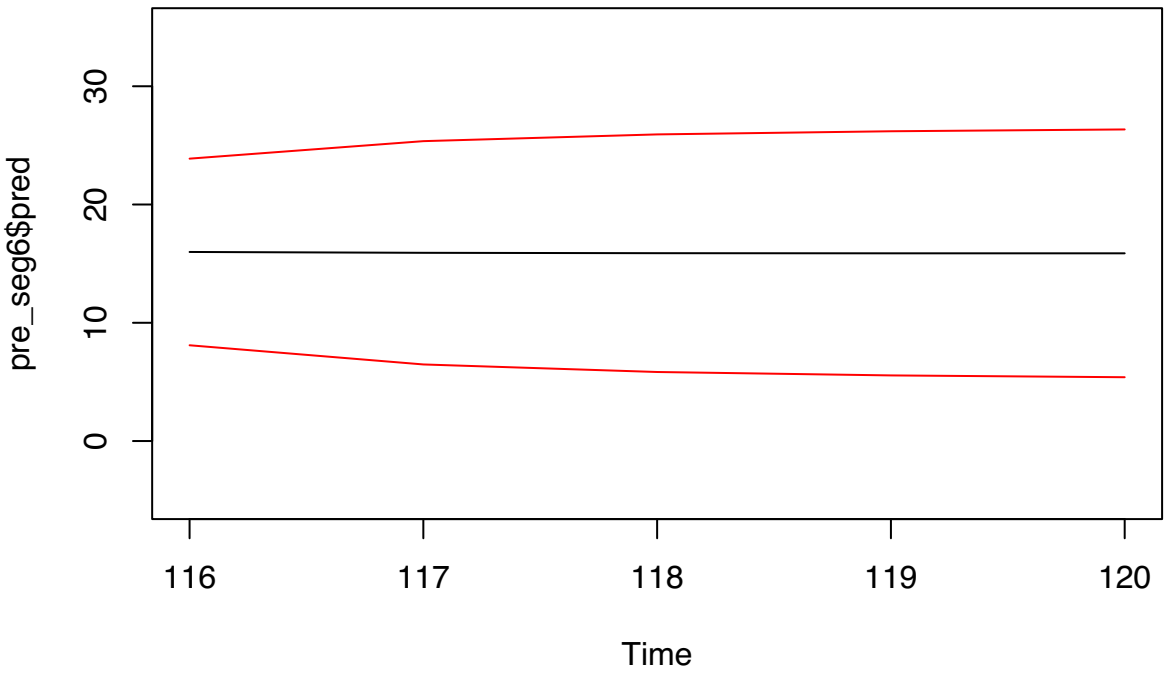
```
pre_seg6$se
```

```
## Time Series:
## Start = 116
## End = 120
## Frequency = 1
## [1] 3.945399 4.720365 5.023089 5.161649 5.238065
```

```
plot(fit_seg_final, n.ahead = length(seg6_test), main = "Actual VIX Index with forecast tail and predict
```




```
upperbound = (pre_seg6$pred) + (pre_seg6$se)*2
lowerbound = (pre_seg6$pred) - (pre_seg6$se)*2
ts.plot(pre_seg6$pred, ylim = c(-5, 35))
lines(lowerbound, col = 'red')
lines(upperbound, col = 'red')
```



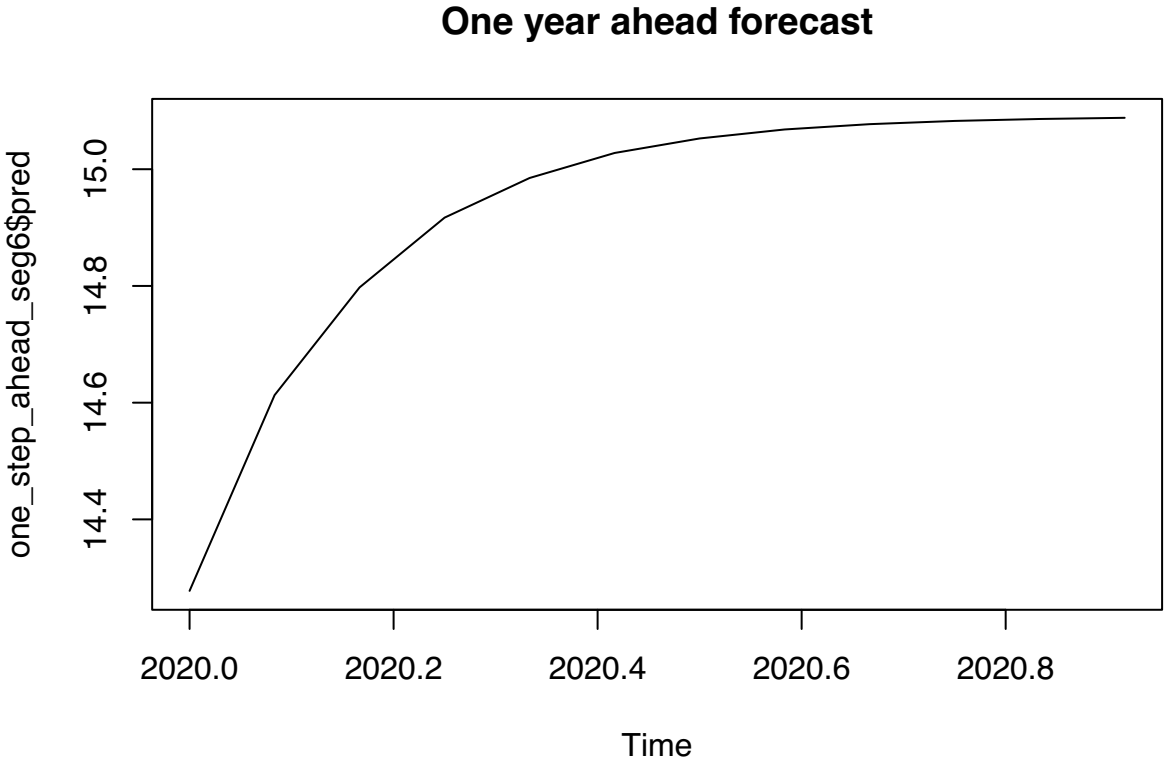
one year ahead forecast

We use the model above to estimate the one step ahead.

```
fit_seg6 = arima(seg6, order = c(2, 1, 2), method = "ML", include.mean = TRUE)
one_step_ahead_seg6 = predict(fit_seg6, n.ahead = 12)
one_step_ahead_seg6$pred
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2020 14.27754 14.61293 14.79749 14.91707 14.98497 15.02785 15.05270 15.06814
##           Sep      Oct      Nov      Dec
## 2020 15.07720 15.08277 15.08607 15.08808
```

```
plot(one_step_ahead_seg6$pred, main = "One year ahead forecast")
```



5 Discussion

VIX Index as a whole

Generally speaking, we can theoretically predict the VIX Index by logarithm transformation and differencing the original dataset if we consider the dataset as a whole. However, if we combine the predicted VIX Index with the actual time series, the predicted VIX Index shows a difference. One possible reason is the small sample size. We only have 384 historical VIX Index to study, so the monthly VIX Index might be too narrow to forecast. Another possible reason for the uncontrollable factor. As mentioned in the introduction, some unexpected events happened, and the VIX Index's fluctuation will be significant. In the "VIX Index as a whole" part of the study, the predicted VIX Index in the plot looks like a line, but the prediction interval has a widespread upperbound and lowerbound. We ignore the uncontrollable factors as long as we consider the historical VIX Index without segments. Therefore, we have a "VIX Index in segments" study to show a better forecast.

In the study of the "VIX Index in segments", the six segments can somewhat eliminate the effect of uncontrollable factors. The partition of segments is based on historical events. From the plot of each segment, we can see the fluctuation of each segment is different, so we only use the last segment to forecast and eliminate the effect of other segments. However, the insufficiency of the method is also the small sample size, even worse than the previous method because we only have 120 data.

Generally speaking, an unexpected event does not happen regularly. Therefore, we have no way to find a perfect model based on the pure VIX Index without any other additional information about unexpected events.

6 Bibliography

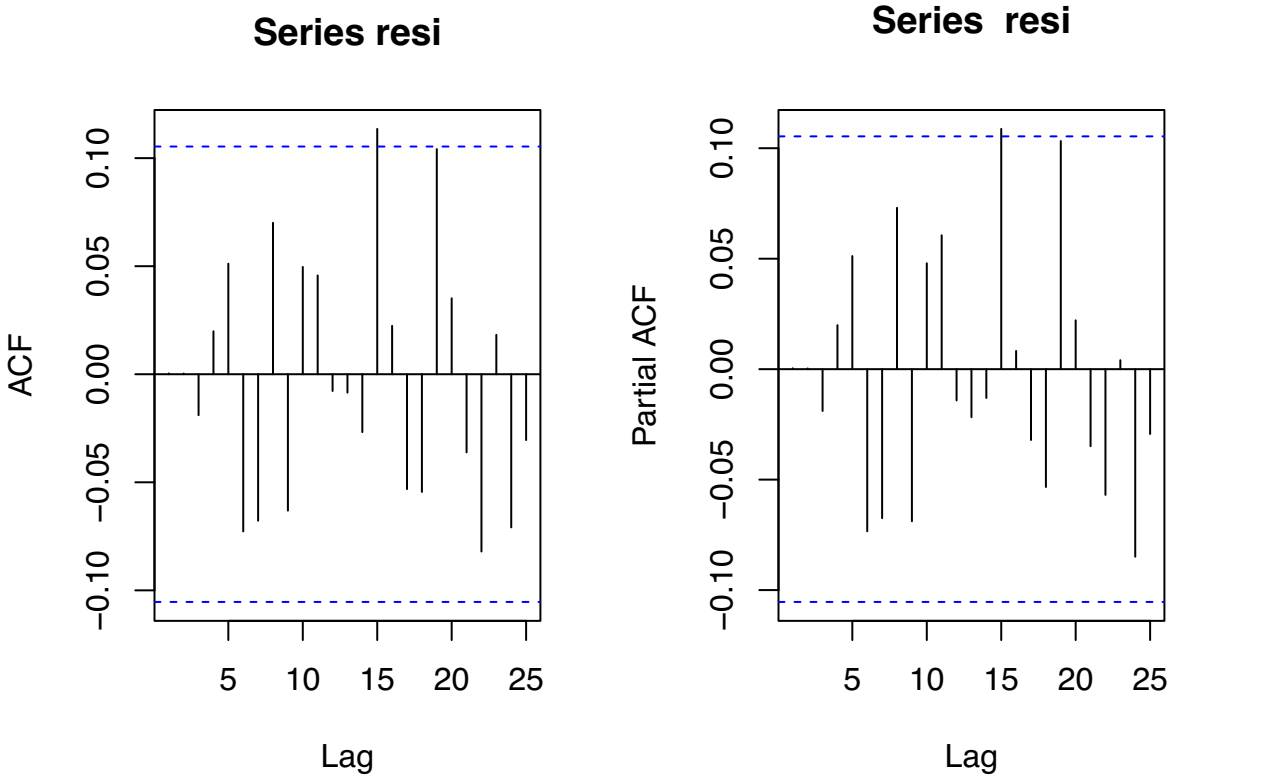
1. https://www.researchgate.net/publication/228856268_Forecasting_VIX
2. https://www.tableau.com/learn/articles/time-series-forecasting#:~:text=Analysts%20can%20tell%20the%20difference

3. https://juliastats.org/HypothesisTests.jl/latest/time_series/
4. https://search.r-project.org/CRAN/refmans/tsoutliers/html/jarque-bera-test.html
5. Casella, Fienberg, S., & Okin, I. (2008). Time Series Analysis: With Applications in R. In Time series analysis: with applications in R (pp. xiii–xiii). Springer New York.
6. Brenner, Menachem; Galai, Dan (July–August 1989). "New Financial Instruments for Hedging Changes in Volatility" (PDF). Financial Analysts Journal. 45 (4): 61–65. doi:10.2469/faj.v45.n4.61

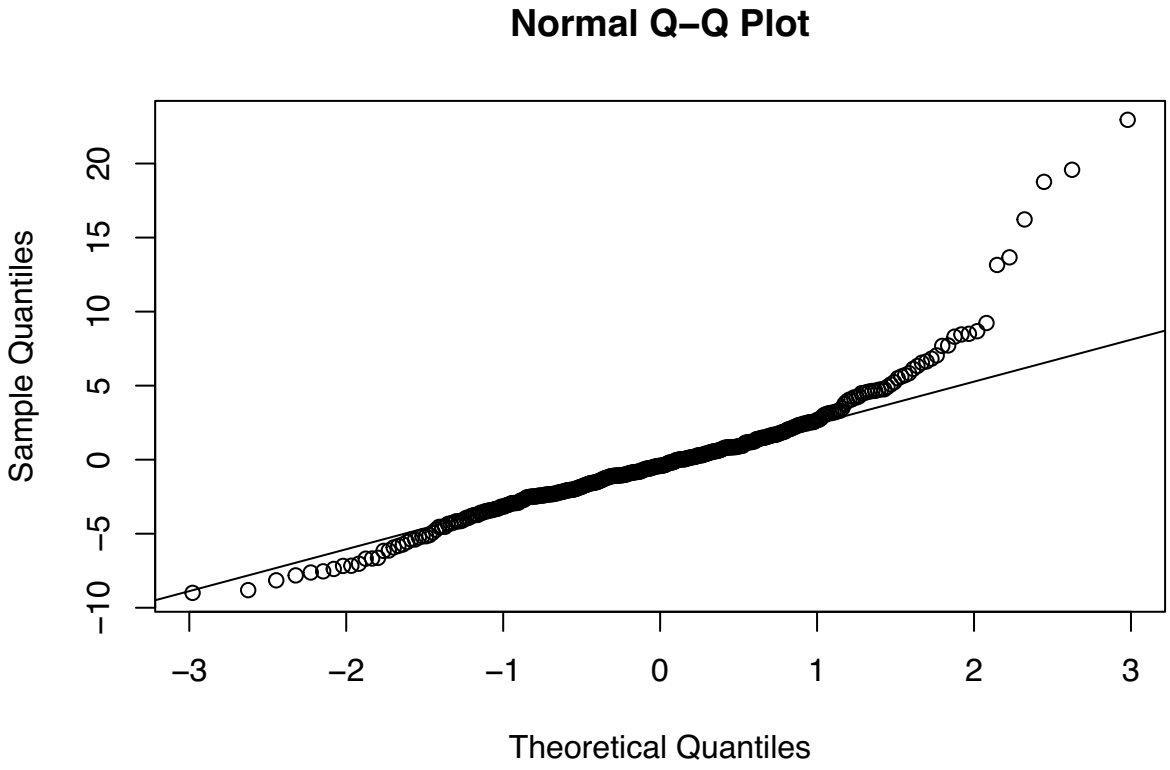
7 Appendix

Model check

```
fit = arima(x, order = c(1, 1, 2))
resi = residuals(fit)
par(mfrow=c(1,2))
acf(resi)
pacf(resi)
```




```
qqnorm(residuals(fit))
qqline(residuals(fit))
```



The residuals show normality based on the graph.

```
RMSE(fit$residuals)
```

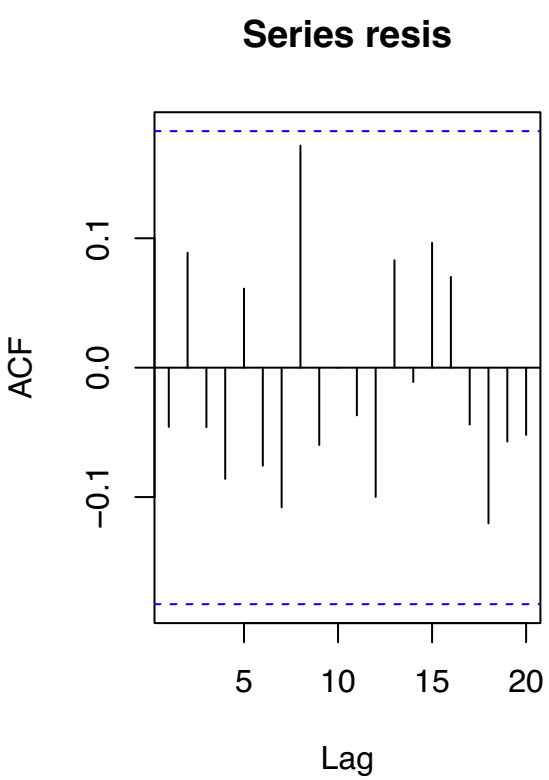
```
## [1] 3.959236
```

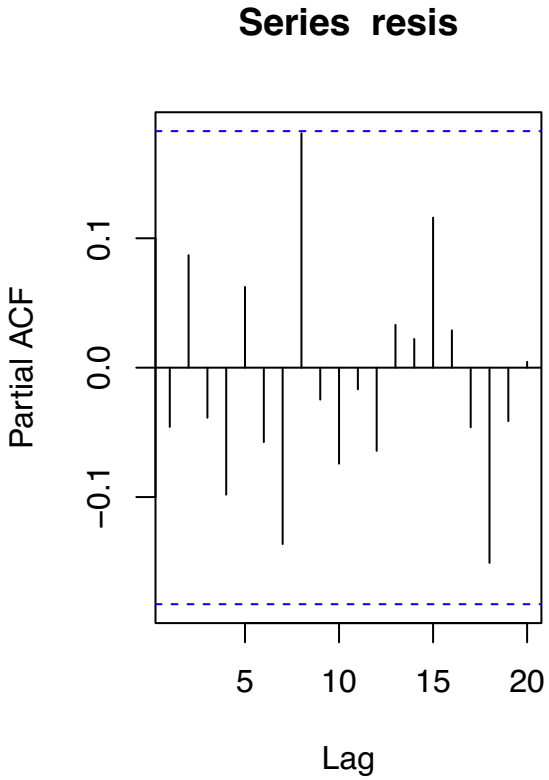
```
harmonic_mean(fit$residuals)
```

```
## [1] 0.1019815
```

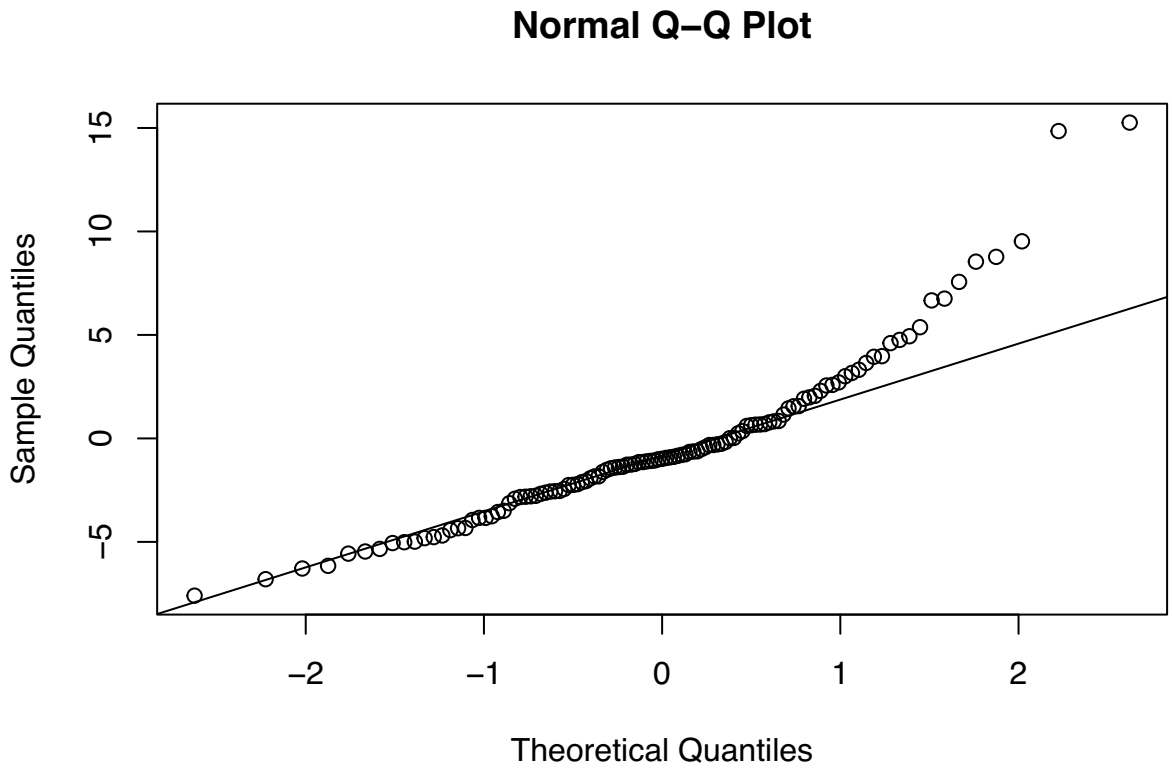
The RMSE of the fit model is relatively small, and the harmonic mean of the fit model is very close to 0.

```
fit_seg_final = arima(seg6_training, order = c(2, 1, 2))
resis = residuals(fit_seg_final)
par(mfrow=c(1,2))
acf(resis)
pacf(resis)
```






```
qqnorm(residuals(fit_seg_final))
qqline(residuals(fit_seg_final))
```



The residuals show normality based on the graph.

```
RMSE(fit_seg_final$residuals)
```

```
## [1] 3.928208
```

```
harmonic_mean(fit_seg_final$residuals)
```

```
## [1] 0.4720845
```

The RMSE of the fit model is relatively small, and the harmonic mean of the fit model is very close to 0.