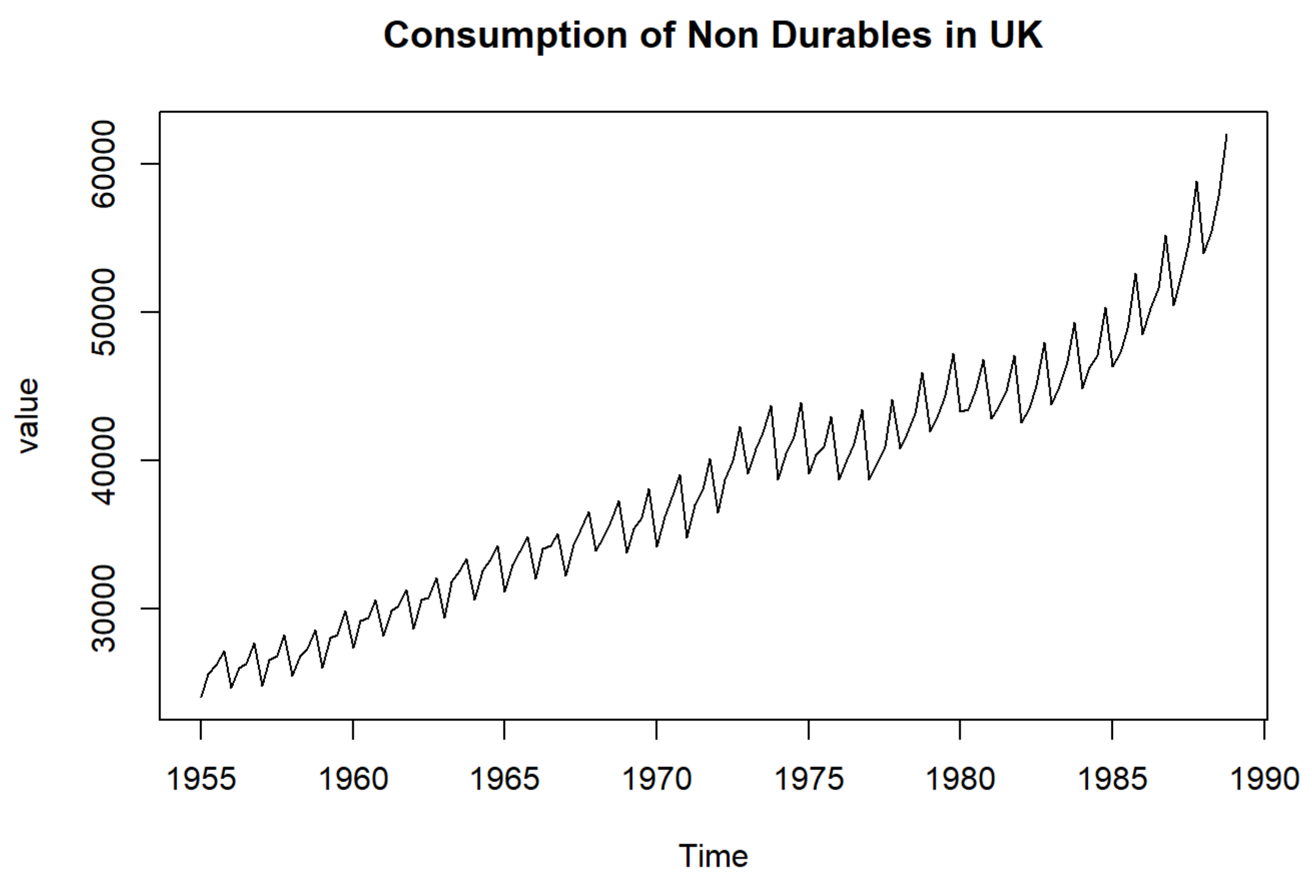
UK Non Durables Seasonal

From the basic ARIMA forecasting of the consumpion of non durables in UK, we saw that there were various problems with the best model chosen. This was particularly due to the assumption that there is no seasonality in the data. So we are going to address that here. As usual due to the wide variance of values in the data we are going to work on the log transformed data.

```
library(tseries)
library(forecast)
library(dplyr)
library(moments)
data = read.table('UKNonDurables.csv',sep=',',header=T,stringsAsFactors = F)
data = data[, c('time','value')]

#Ordering data based on time
data = data[order(data$time),]

#Check for missing data
sum(is.na(data))
```

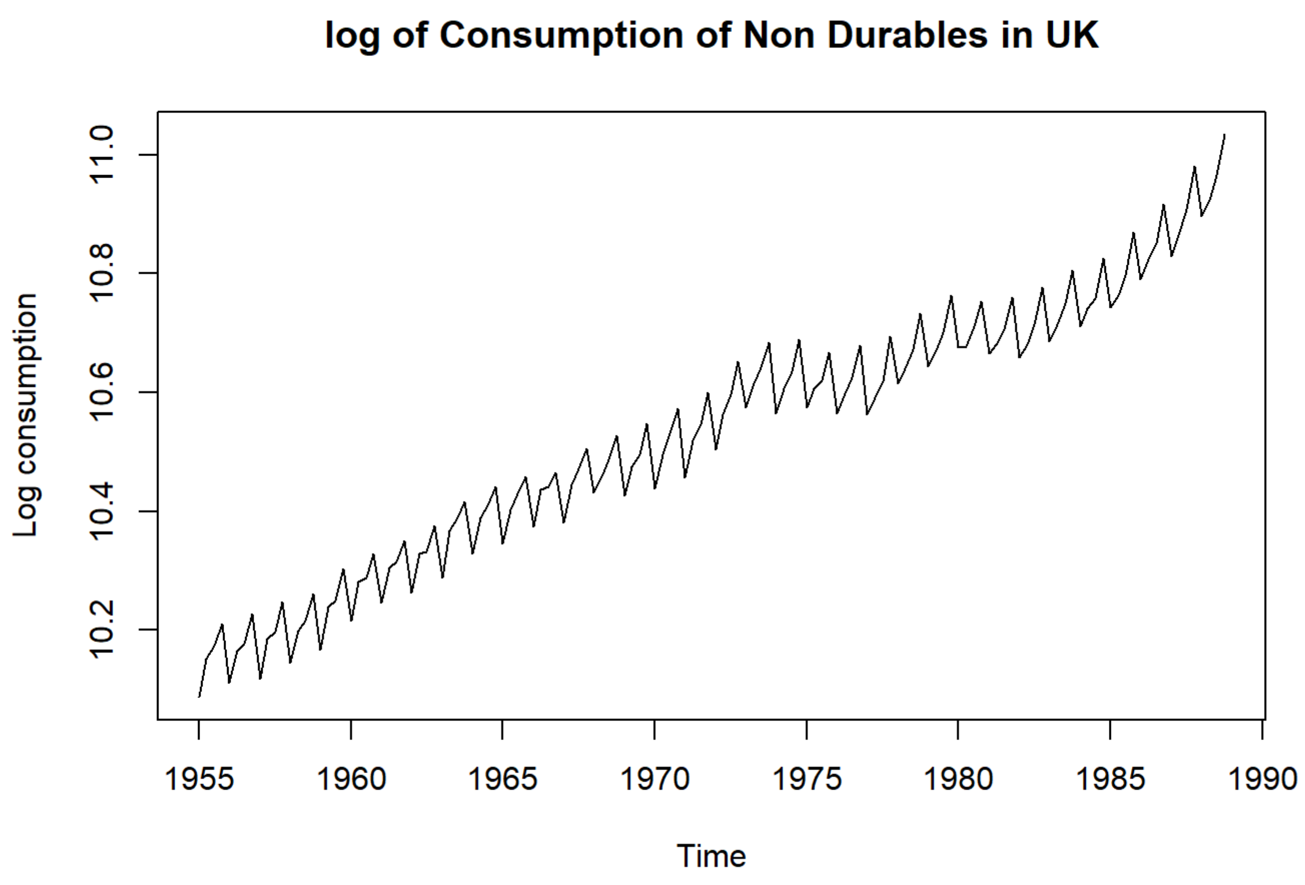
```
## [1] 0
```

```
#So there are no missing values in the dataset

#Cleaning data
start_date = data$time[1]
value = ts(data$value, start = start_date,freq =4)
value = tsclean(value)
data$value = value


#Time series plot of the whole data
ts.plot(value,main = "Consumption of Non Durables in UK")
```



Working with the log values instead:

```
data['log_value'] = log(data$value)
ts.plot(data$log_value,main = "log of Consumption of Non Durables in UK",
        ylab = 'Log consumption')
```



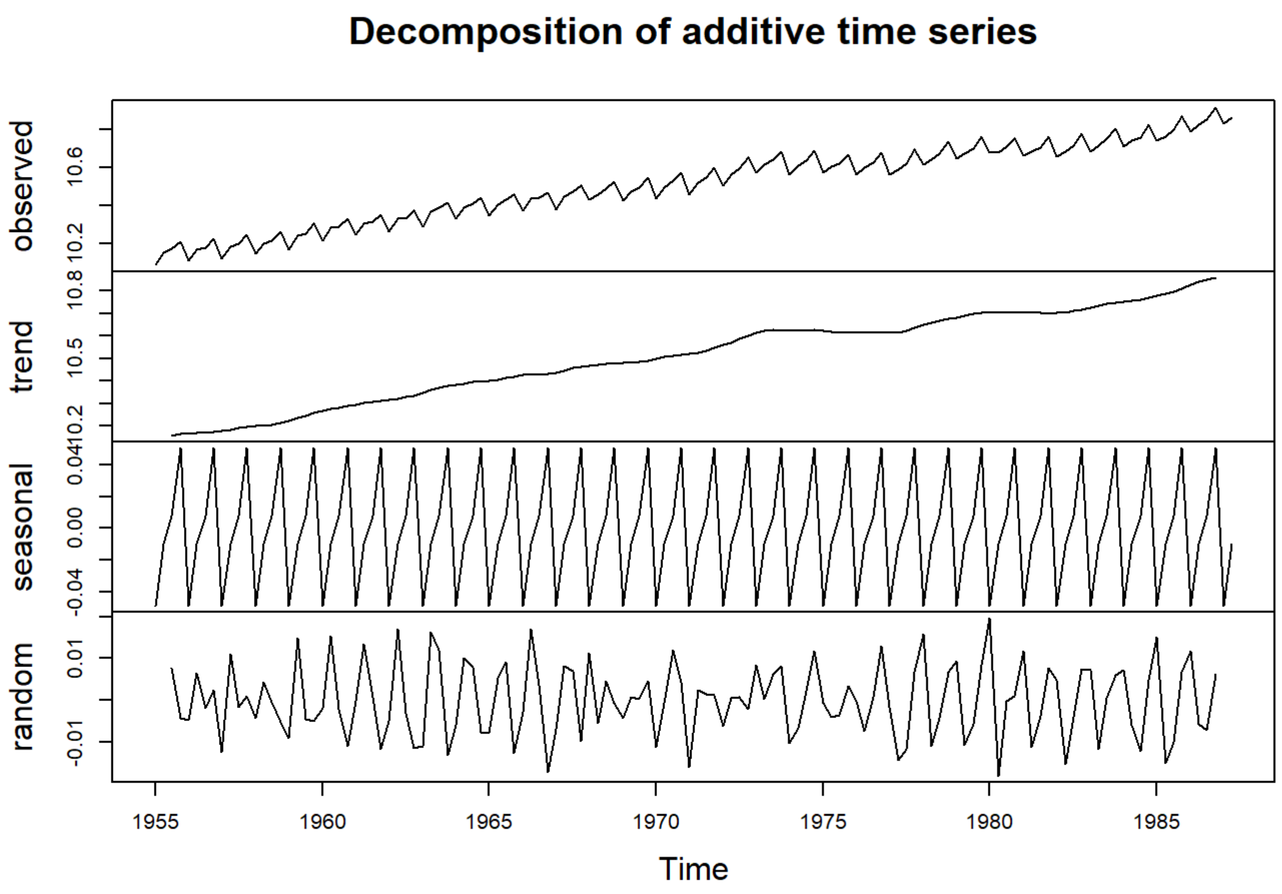
Setting apart a test set of last 6 quarters.

```
tm = data$time

period<- 6
n = dim(data)[1]
train_lim = n - period


#Split data into train data and test data with train data = 95% of the data
train_data = data[1:train_lim,]
test_data = data[(train_lim+1):n,]
```

*** Part 1: Decomposing data into seasonality and trend ***

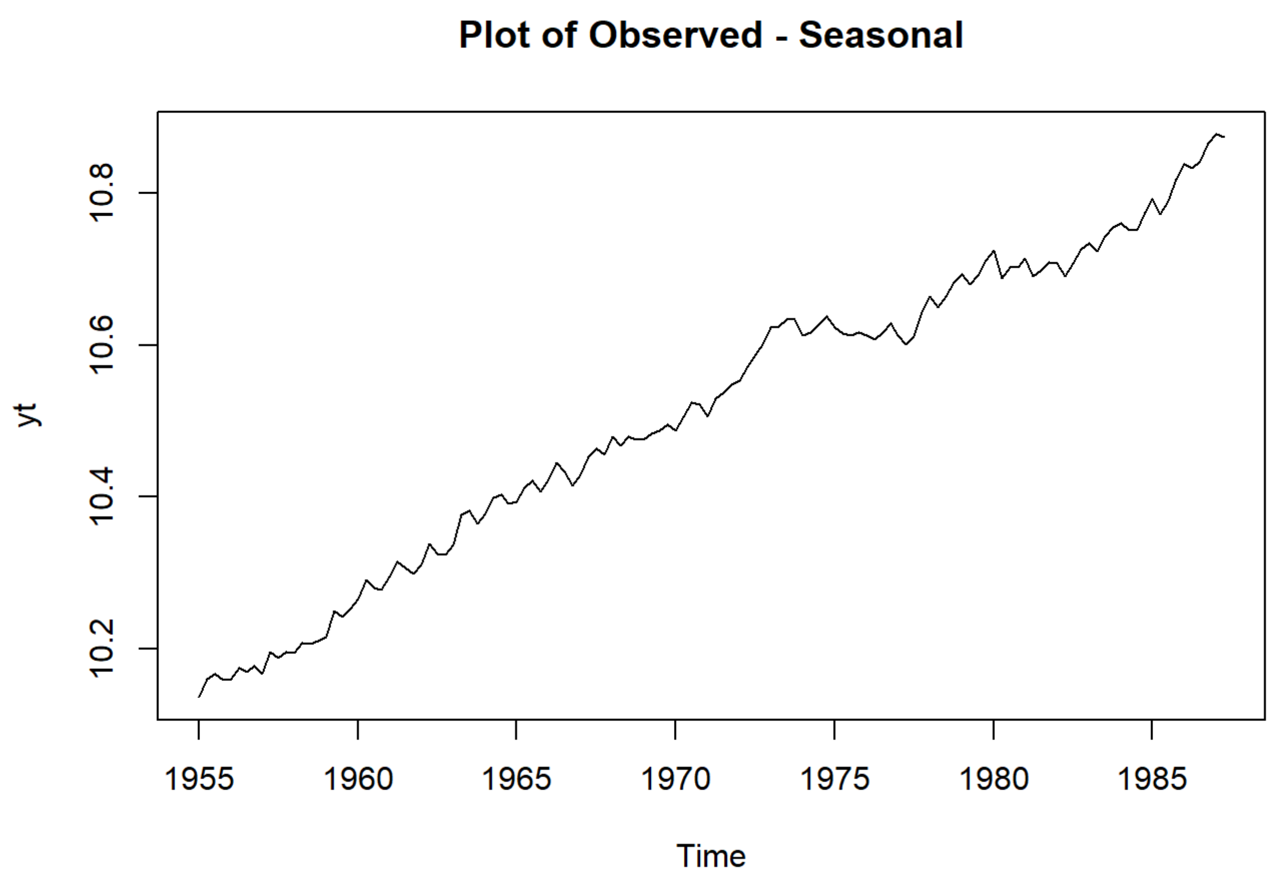
From the time series plot, there seems to be an additive seasonality. So, we decompose the original data into trend, seasonality and the random components. We subtract the seasonality from the data to get a transformed time series Y'. We build our ARIMA model on Y' and after we forecast the values we add back the seasonality components for each quarter.

```
value<- ts(train_data$log_value, start = start_date, frequency = 4)
decomposed<- decompose(value, type = 'additive')
plot(decomposed)
```



```
seasonal<- decomposed$seasonal
trend<- decomposed$trend
rdm<- decomposed$random
```

```
yt<- value - seasonal

ts.plot(yt, main ='Plot of Observed - Seasonal')
```



Checking Stationarity of Y'

```
adf.test(yt)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  yt
## Dickey-Fuller = -2.7866, Lag order = 5, p-value = 0.2495
## alternative hypothesis: stationary
```

```
kpss.test(yt)
```

```
## Warning in kpss.test(yt): p-value smaller than printed p-value
```

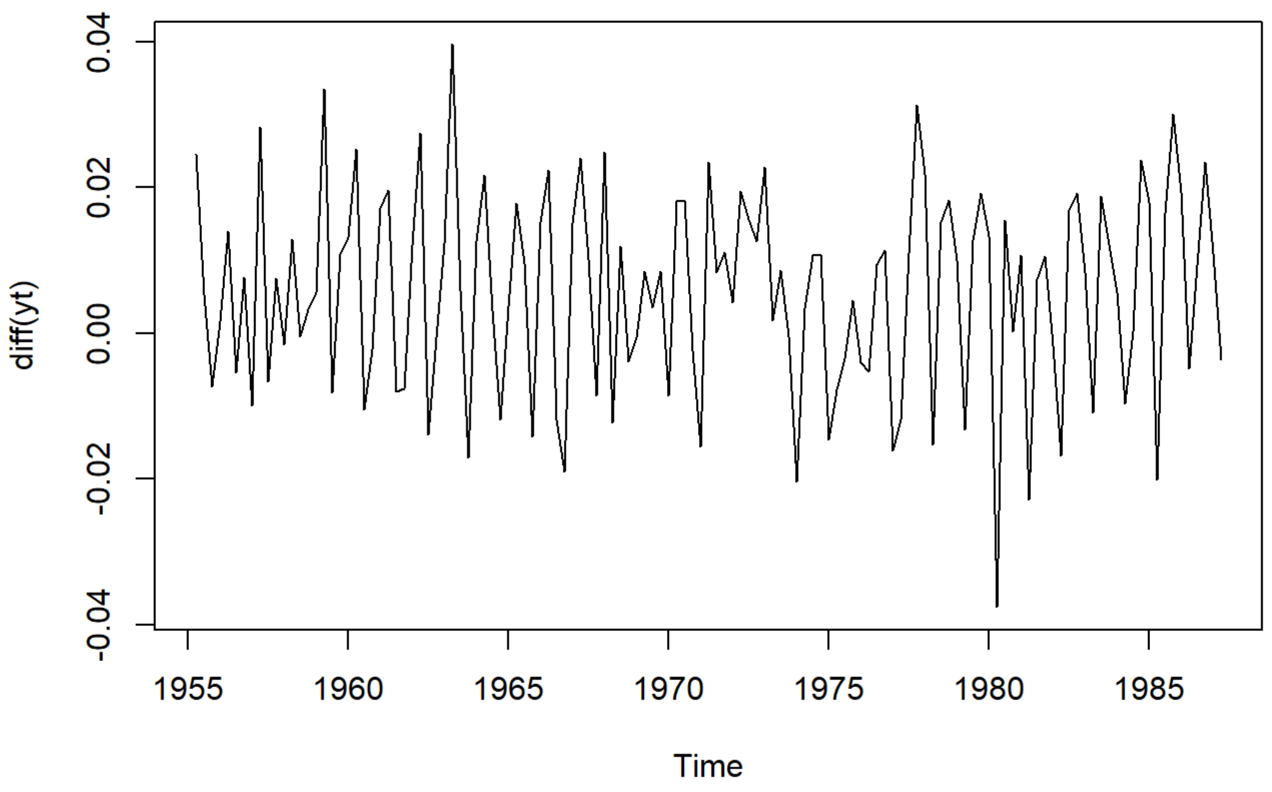
```
##
##  KPSS Test for Level Stationarity
##
## data:  yt
## KPSS Level = 2.6547, Truncation lag parameter = 4, p-value = 0.01
```

```
pp.test(yt)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  yt
## Dickey-Fuller Z(alpha) = -13.714, Truncation lag parameter = 4, p-value
## = 0.3242
## alternative hypothesis: stationary
```

So, the series is non stationary. We are going to try to check the stationarity after differencing.

```
ts.plot(diff(yt))
```



The first order differenced data looks stationary. We are going to test for the same.

```
adf.test(diff(yt))
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(yt)
## Dickey-Fuller = -4.2145, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(diff(yt))
```

```
## Warning in kpss.test(diff(yt)): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(yt)
## KPSS Level = 0.076093, Truncation lag parameter = 4, p-value = 0.1
```

```
pp.test(diff(yt))
```

```
## Warning in pp.test(diff(yt)): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff(yt)
## Dickey-Fuller Z(alpha) = -117.03, Truncation lag parameter = 4, p-value
## = 0.01
## alternative hypothesis: stationary
```

So, the first ordered differenced data is stationary. We are going to work with a differencing order of 1. Now we find the appropriate ARIMA model with lowest AICc value.

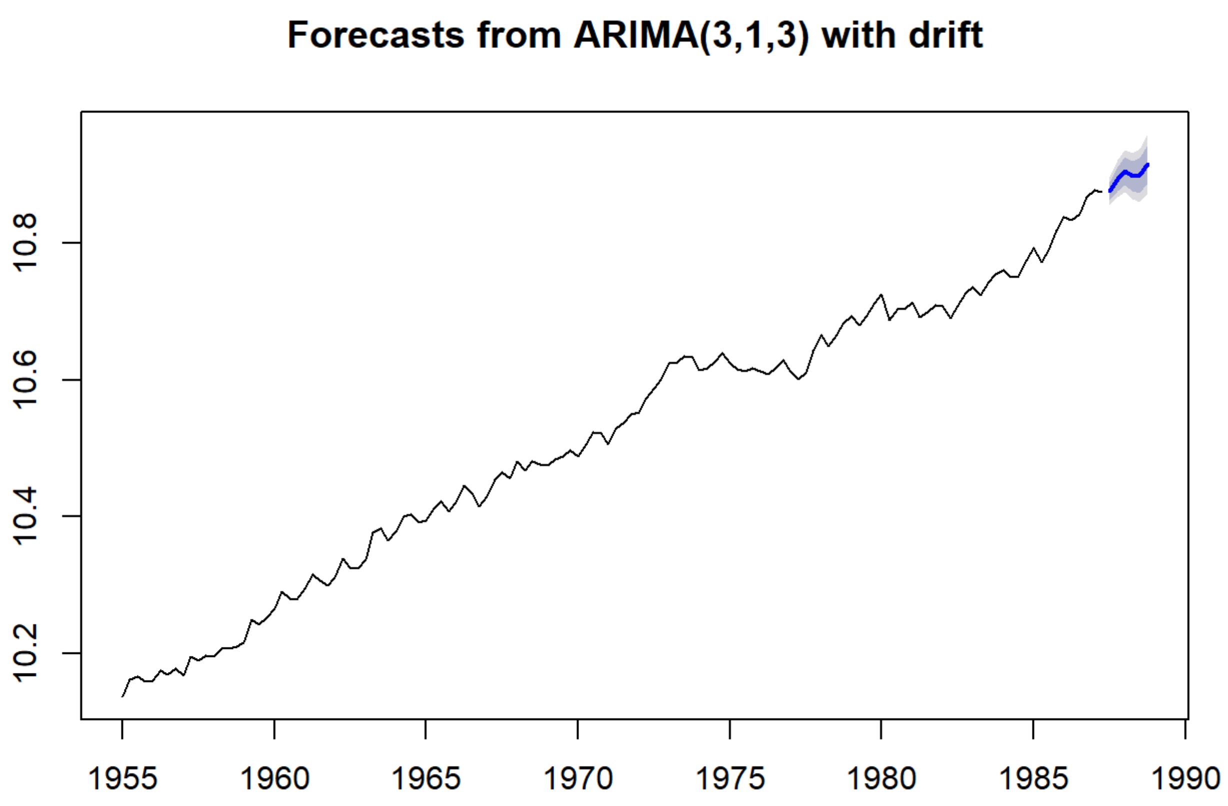
```
d<- ndiffs(yt)

model = auto.arima(yt,seasonal = F,d=d,max.p = 5, max.q = 5, ic = "aicc")
summary(model)
```

```
## Series: yt
## ARIMA(3,1,3) with drift
##
## Coefficients:
##           ar1      ar2      ar3     ma1     ma2     ma3   drift
##       -0.8902  -0.9287  -0.9074  0.7415  0.6964  0.6024  0.0057
## s.e.   0.0607   0.0341   0.0585  0.1111  0.0840  0.1037  0.0007
##
## sigma^2 estimated as 0.0001081:  log likelihood=408.86
## AIC=-801.72   AICc=-800.52   BIC=-778.84
##
## Training set error measures:
##                        ME       RMSE         MAE         MPE       MAPE
## Training set 0.0001196259 0.01007311 0.007995833 0.001139374 0.07604132
##                   MASE       ACF1
## Training set 0.3247051 0.06879735
```

So, ARIMA (3,1,3) with drift is chosen as the best model of $Y^{'}$.

We see the forecast of $Y^{'}$ for last 6 quarters as follows:

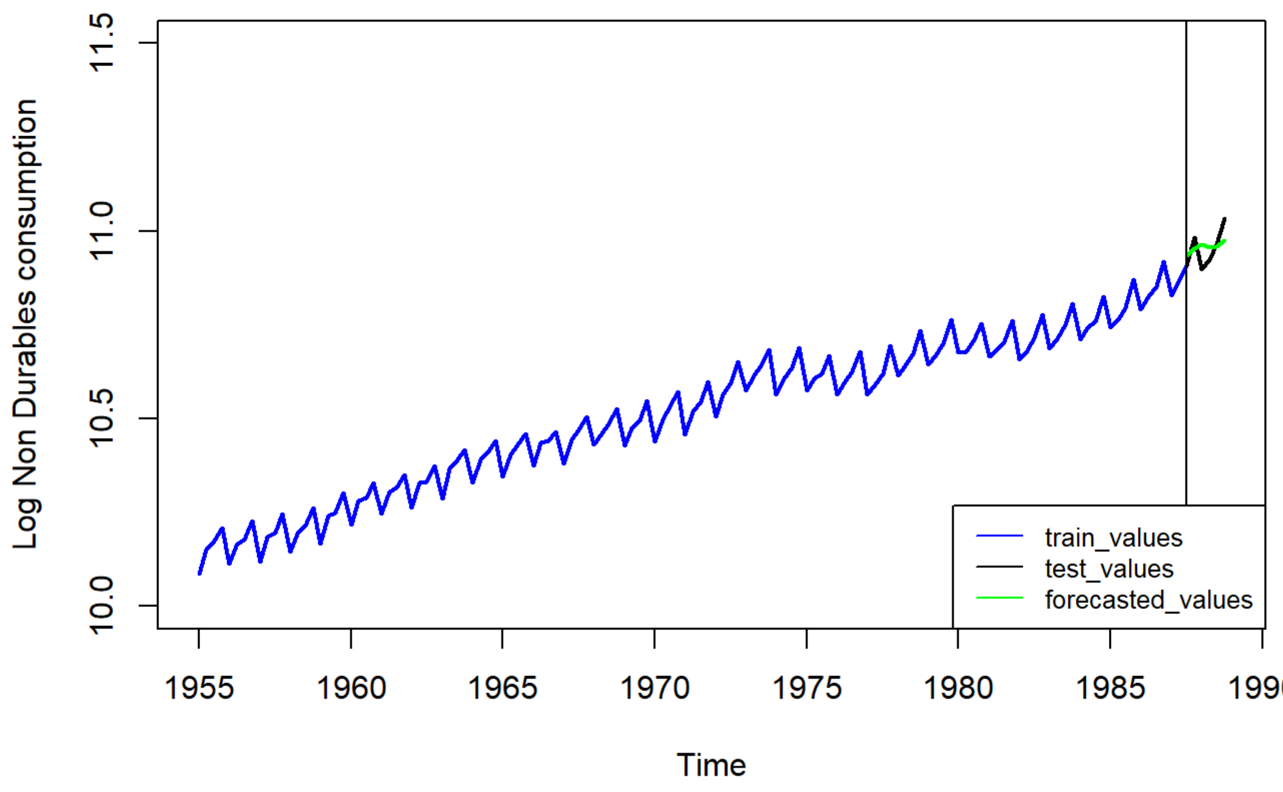
```
plot(forecast(model,h=period),type='l')
```



Now, we add the seasonal components to the trend forecast to get our actual forecast.

```
get_quarter<- function(date)
{
  return (((date - floor(date))*4)+1)
}

arima_forecast<- forecast(model, h = period)
trend_pred<- arima_forecast$mean

pred <- trend_pred
for (i in 1:period)
{
  date = test_data$time[i]
  qtr = get_quarter(date)

  seasonal_comp <- seasonal[qtr]
  pred<- pred + seasonal_comp
}
```

```
plot(NULL,xlim = c(tm[1],tm[n]),ylim = c(10,11.5),ylab = 'Log Non Durables consumption',xlab=
'Time')
points(data$time, data$log_value,type='l',lwd=2,col='blue')
points(test_data$time, test_data$log_value,type='l',lwd=2,col='black')
points(test_data$time, pred,type='l',lwd=2,col='green')
abline(v= test_data$time[1], col = 'black')
legend("bottomright",
       legend =  c("train_values","test_values","forecasted_values"),
       col = c("blue","black","green"),lty = 1,
       cex = 0.8)
```

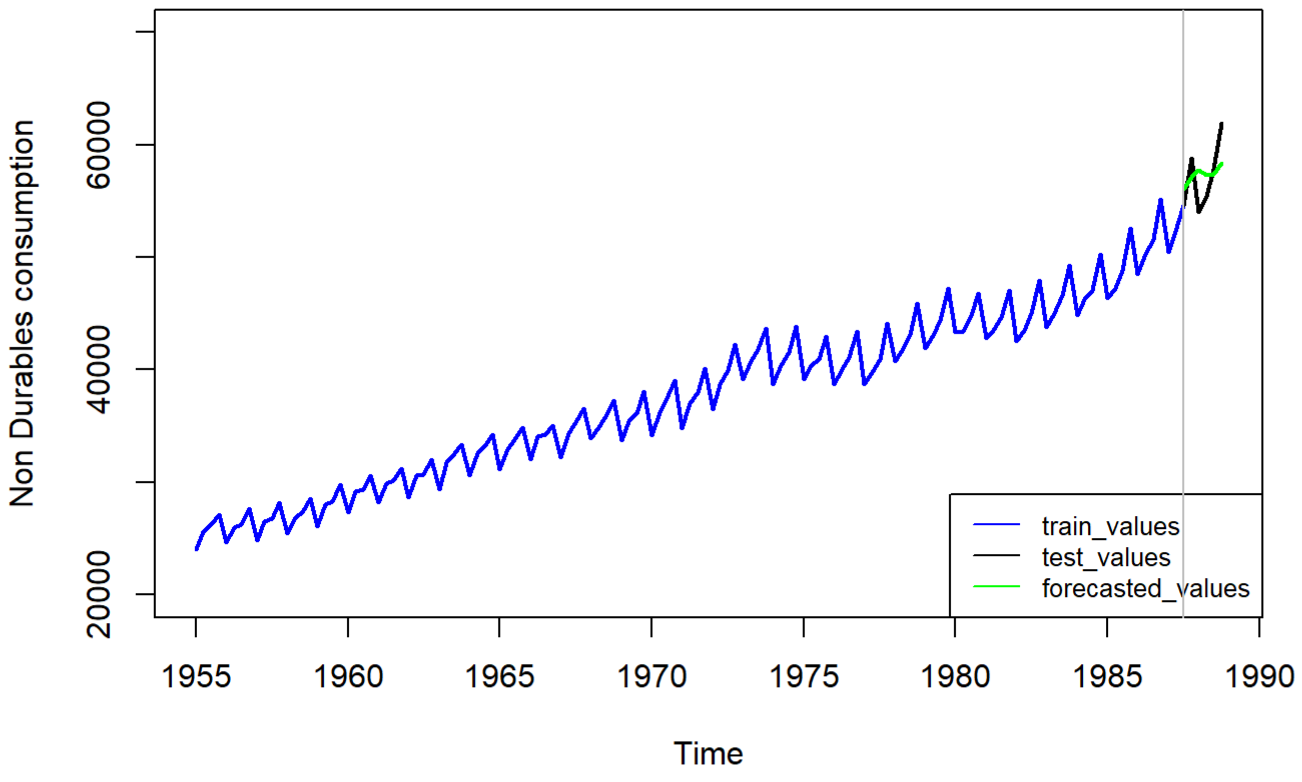


We also check the mean squared log error.

```
log_mse<- sum((test_data$log_value - pred)^2)
log_mse
```

```
## [1] 0.01080178
```

```
pred_exp<- exp(pred)
plot(NULL,xlim = c(tm[1],tm[n]),ylim = c(20000,70000),ylab = 'Non Durables consumption',xlab=
'Time')
points(data$time, data$value,type='l',lwd=2,col='blue')
points(test_data$time, test_data$value,type='l',lwd=2,col='black')
points(test_data$time, pred_exp,type='l',lwd=2,col='green')
legend("bottomright",
       legend =  c("train_values","test_values","forecasted_values"),
       col = c("blue","black","green"),lty = 1,
              cex = 0.8)

abline(v=tm[(train_lim+1)],col='grey')
```



```
rmse<- sqrt(sum((test_data$value - pred_exp)^2))
rmse
```

```
## [1] 5976.036
```

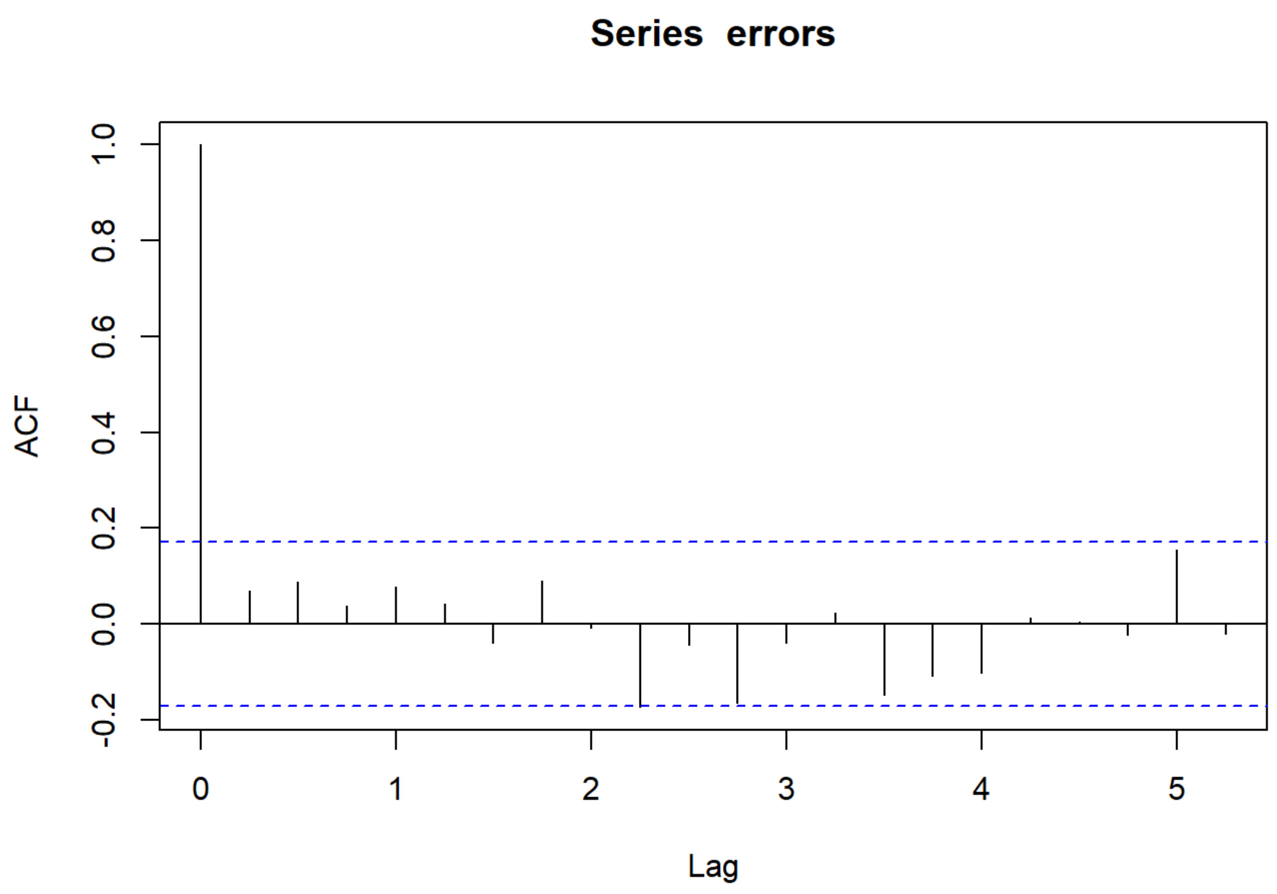
```
cbind(test_data$value, pred_exp)
```

```
##         test_data$value pred_exp
## 1987 Q3           54633 56103.28
## 1987 Q4           58802 57132.10
## 1988 Q1           53990 57764.12
## 1988 Q2           55477 57349.52
## 1988 Q3           57850 57407.25
## 1988 Q4           61978 58398.00
```

We see that the model was close with respect to the first two quarters forecast but then it fell off.

However, let’s check if the model satisfies the assumptions of ARIMA modelling which failed when we did not consider seasonality.

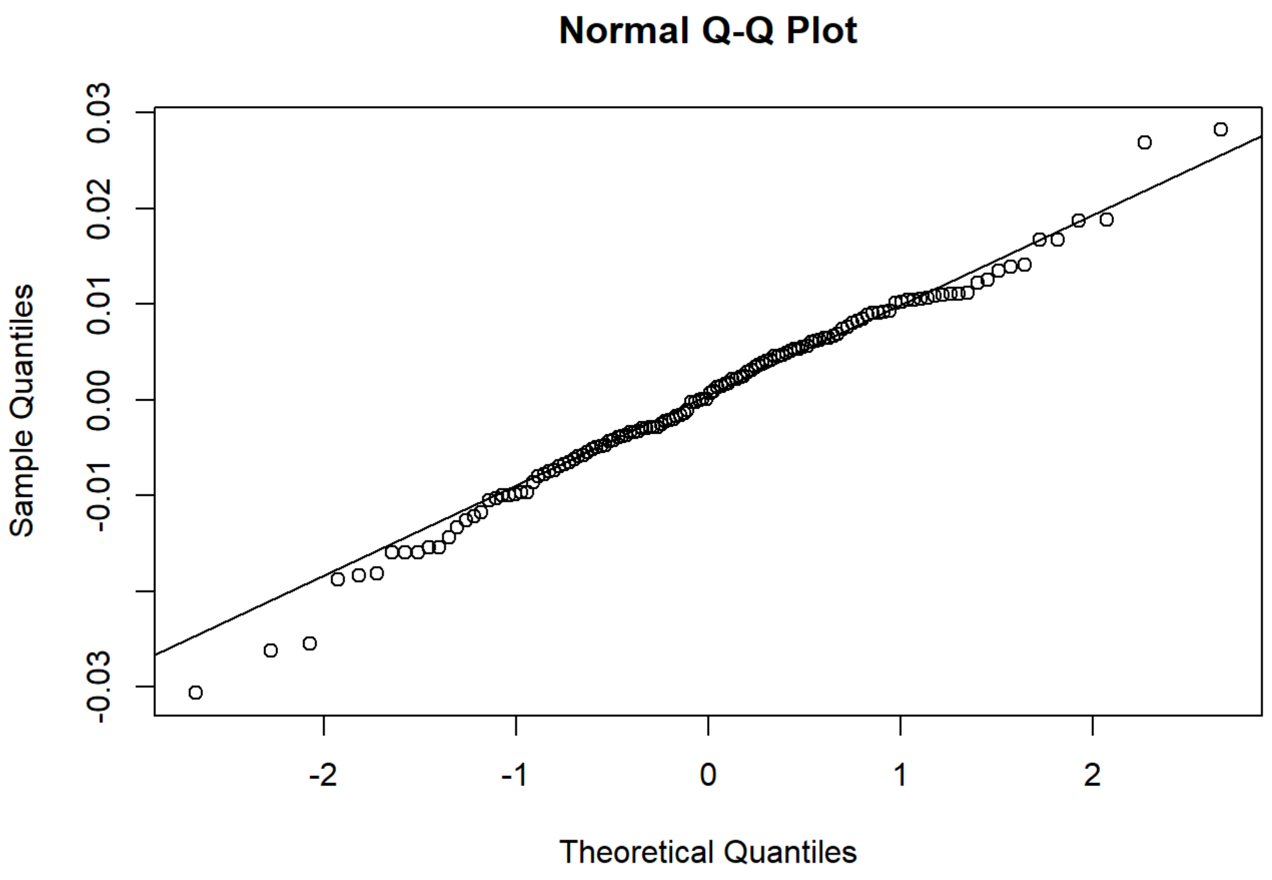
```
#Assumptions
errors<- model$residuals

acf(errors)
```



```
Box.test(errors, lag = log(train_lim), type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  errors
## X-squared = 2.6566, df = 4.8675, p-value = 0.737
```

```
#Asumption of independece valid

qqnorm(errors)
qqline(errors)
```



```
shapiro.test(errors)
```

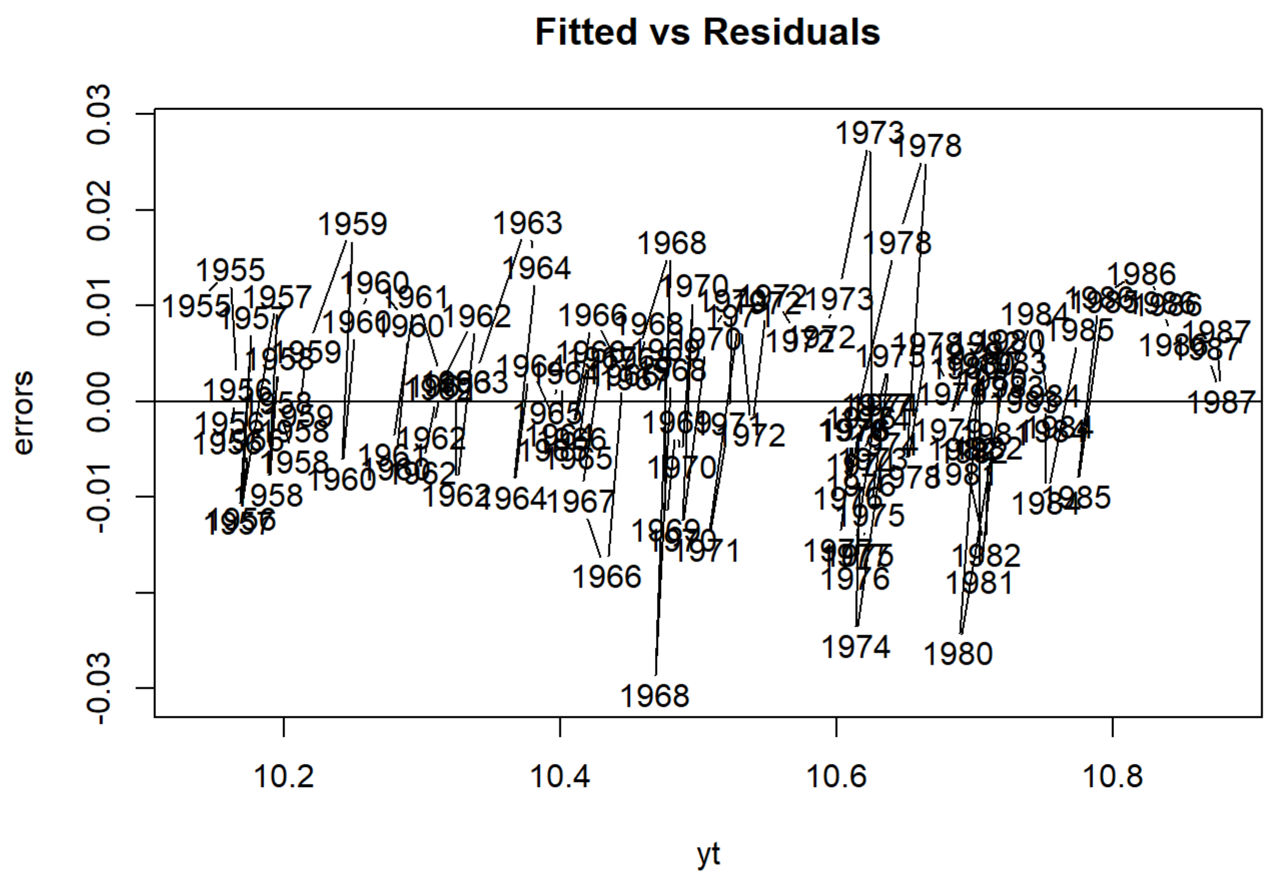
```
##
##  Shapiro-Wilk normality test
##
## data:  errors
## W = 0.9896, p-value = 0.4389
```

```
ks.test(errors, 'pnorm', 0, sqrt(model$sigma2))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  errors
## D = 0.057757, p-value = 0.7787
## alternative hypothesis: two-sided
```

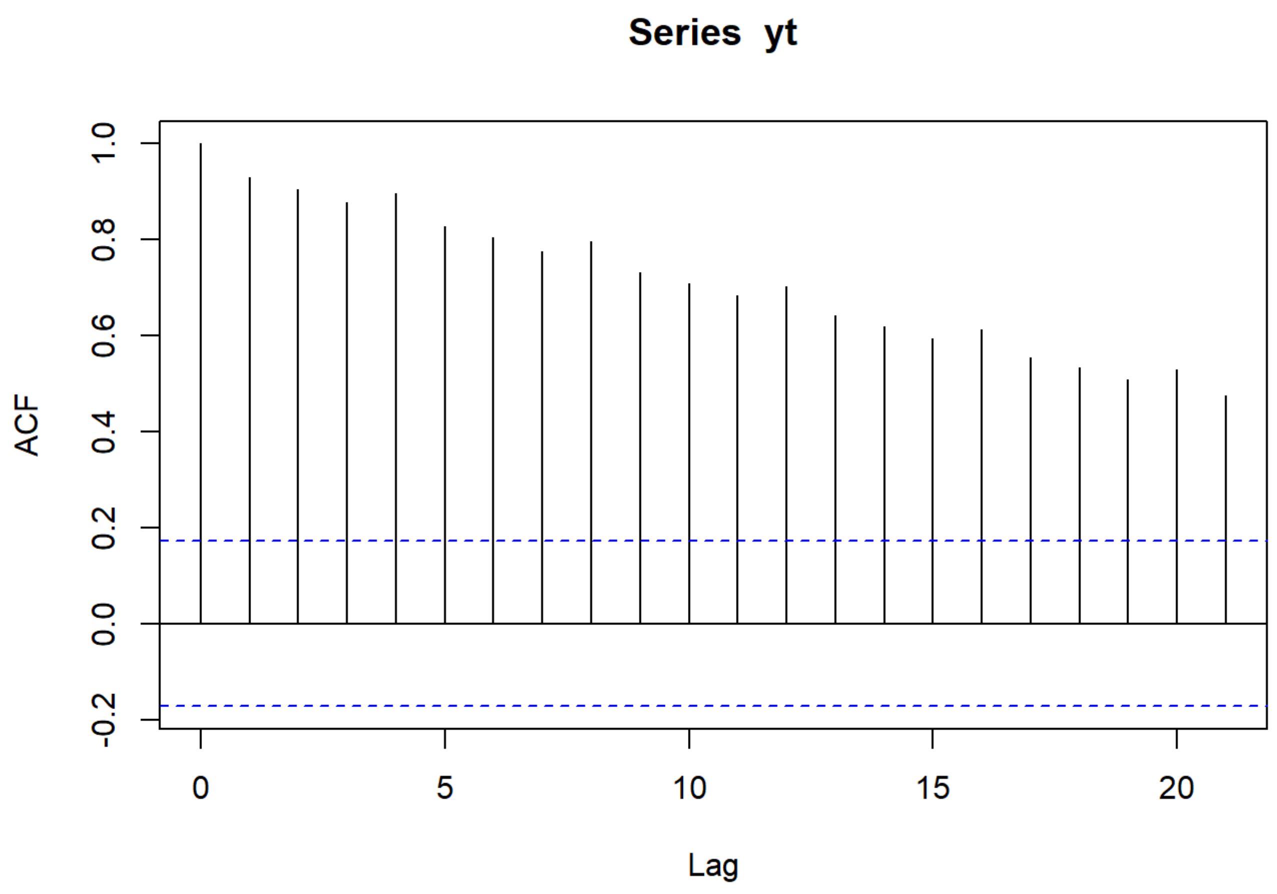
From the ACF plot and Box-Ljung Test, assumption of no autocorrelation of residuals is satisfied. From the QQPlot and Shapiro Wilk and Kolmogorov-Smirnov test, assumption of normality of residuals is valid.

```
plot(yt, errors, main = 'Fitted vs Residuals')
abline(h=0)
```

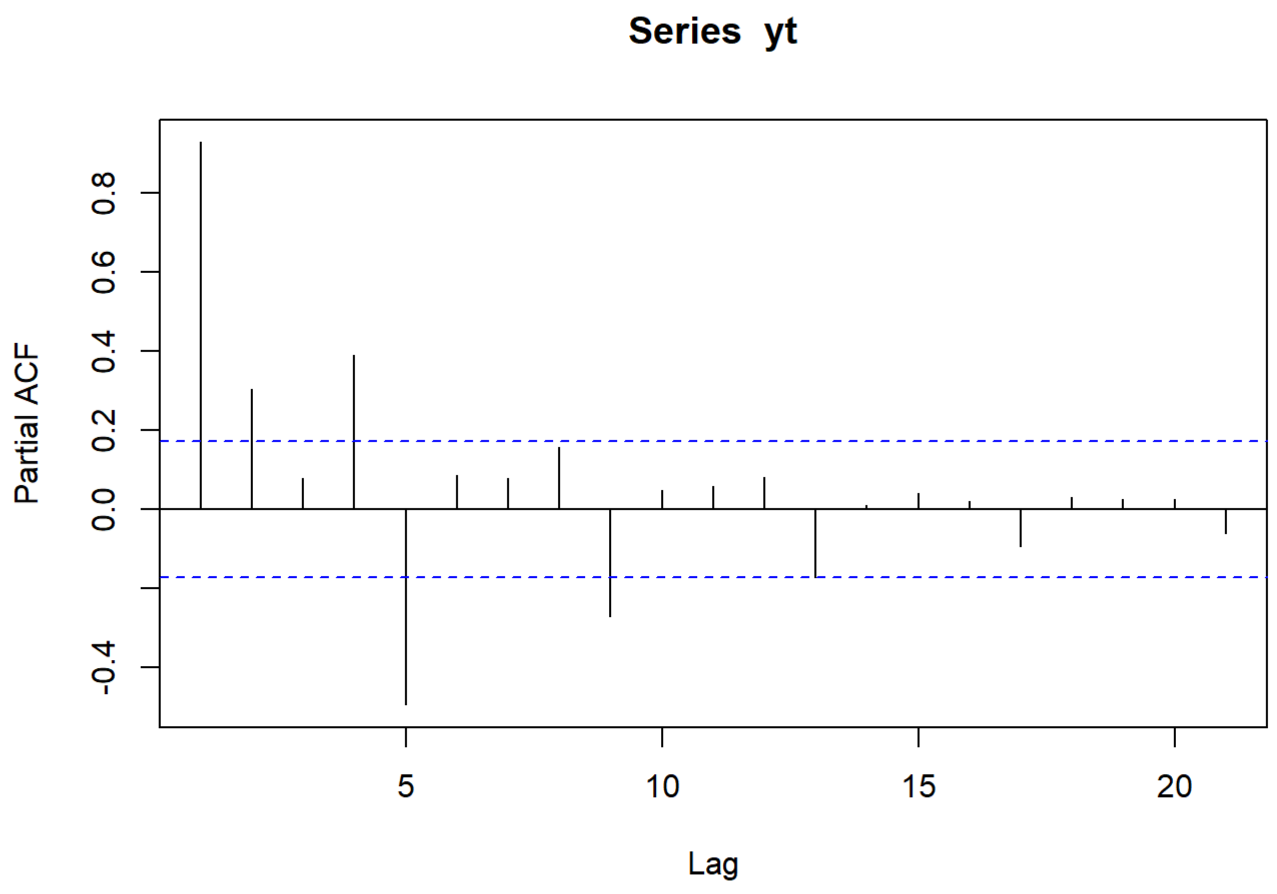


*** Building Seasonal ARIMA model ***

```
yt<- train_data$log_value
acf(yt)
```


Series yt
ACF
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Lag

```
pacf(yt)
```


Series yt
Partial ACF
-0.4
0.0
0.2
0.4
0.6
0.8
5
10
15
20
Lag

From the acf and pacf plots of the data, it looks like seasonal lag of 1 is necessary for AR part.

```
sarima_model<- arima(yt, order = c(3,1,3), seasonal = list(order = c(1,1,0), period = 4))
sarima_model
```

```
##
## Call:
## arima(x = yt, order = c(3, 1, 3), seasonal = list(order = c(1, 1, 0), period = 4))
##
## Coefficients:
```

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

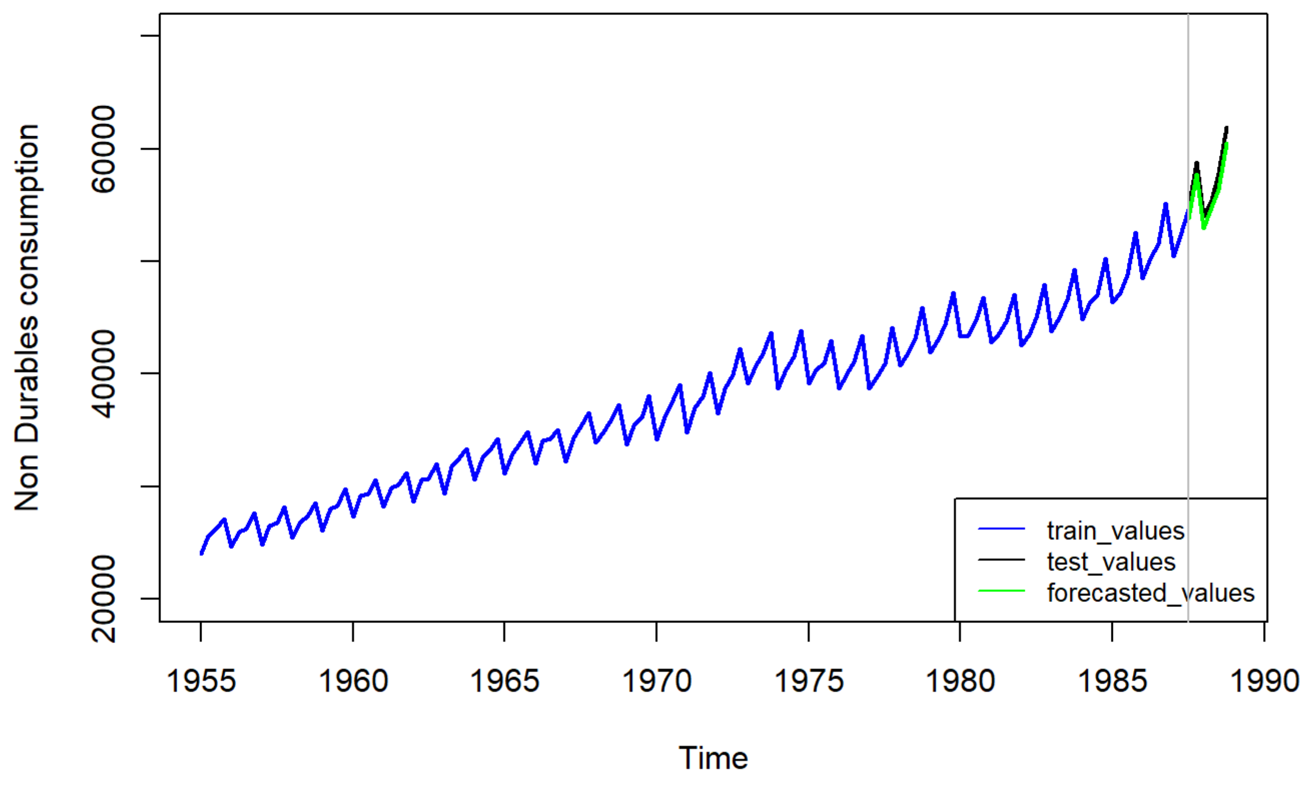
```
##          ar1      ar2    ar3      ma1     ma2      ma3     sar1
##       0.0714  -0.7904  0.231  -0.1184  0.9507  -0.2883  -0.3491
## s.e.     NaN      NaN    NaN      NaN     NaN      NaN   0.0881
##
## sigma^2 estimated as 0.0001122:  log likelihood = 388.81,  aic = -761.62
```

```
sarima_forecast<- forecast(sarima_model, h= period)
sarima_pred<- sarima_forecast$mean

sarima_pred
```

```
## Time Series:
## Start = 131
## End = 136
## Frequency = 1
## [1] 10.89383 10.96387 10.87737 10.91202 10.94076 11.01022
```

```
log_mse_sarima<- sum((sarima_pred - test_data$log_value)^2)
```

```
sarima_pred_exp<- exp(sarima_pred)

plot(NULL,xlim = c(tm[1],tm[n]),ylim = c(20000,70000),ylab = 'Non Durables consumption',xlab=
'Time')
points(data$time, data$value,type='l',lwd=2,col='blue')
points(test_data$time, test_data$value,type='l',lwd=2,col='black')
points(test_data$time, sarima_pred_exp,type='l',lwd=2,col='green')
legend("bottomright",
       legend =  c("train_values","test_values","forecasted_values"),
       col = c("blue","black","green"),lty = 1,
       cex = 0.8)

abline(v=tm[train_lim+1],col='grey')
```



```
rmse<- sqrt(sum((test_data$value - sarima_pred_exp)^2))
rmse
```

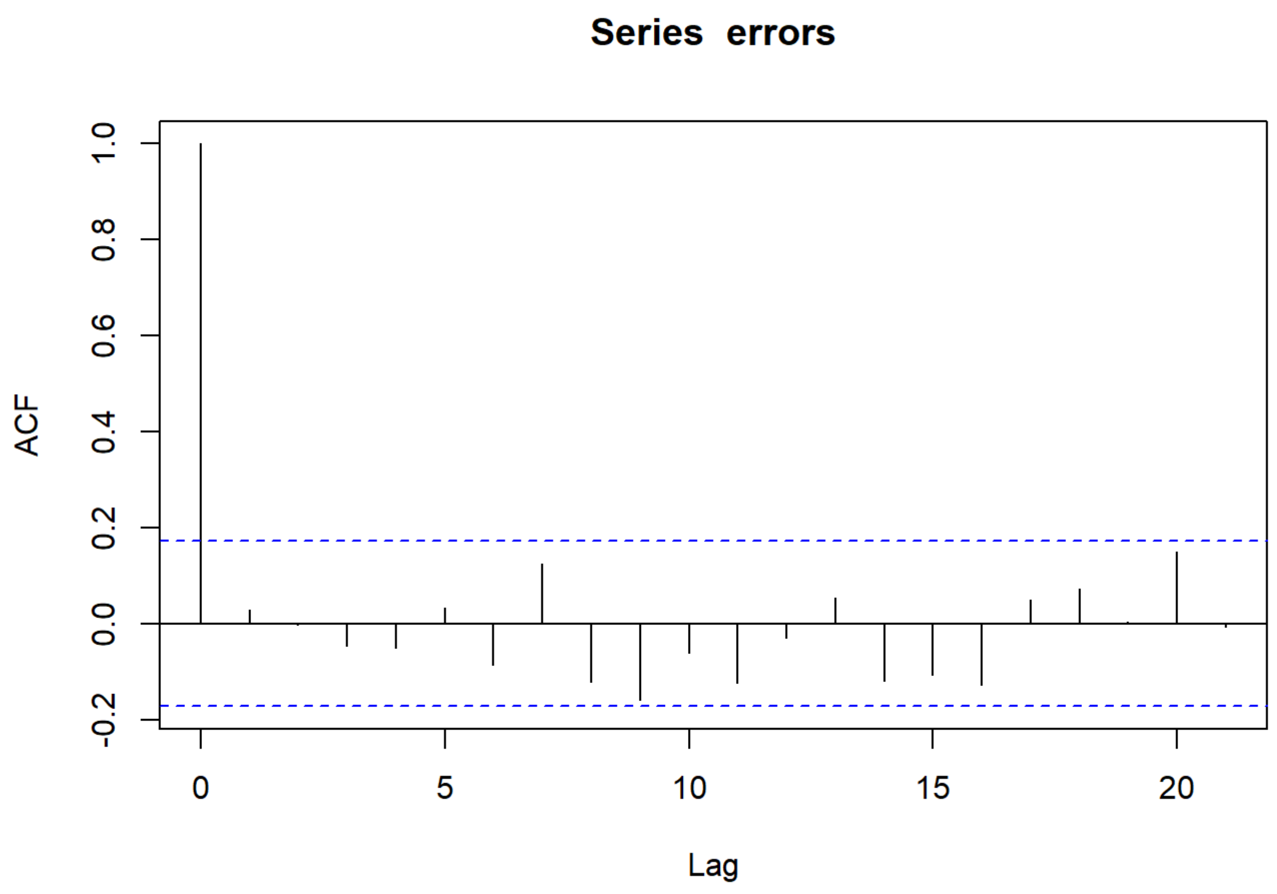
```
## [1] 2726.478
```

```
cbind(test_data$value, pred_exp)
```

```
##         test_data$value pred_exp
## 1987 Q3           54633 56103.28
## 1987 Q4           58802 57132.10
## 1988 Q1           53990 57764.12
## 1988 Q2           55477 57349.52
## 1988 Q3           57850 57407.25
## 1988 Q4           61978 58398.00
```

We see that the SARIMA model is performing way better the models before. We perform the test for the assumptions of the model to see if it violates assumptions.

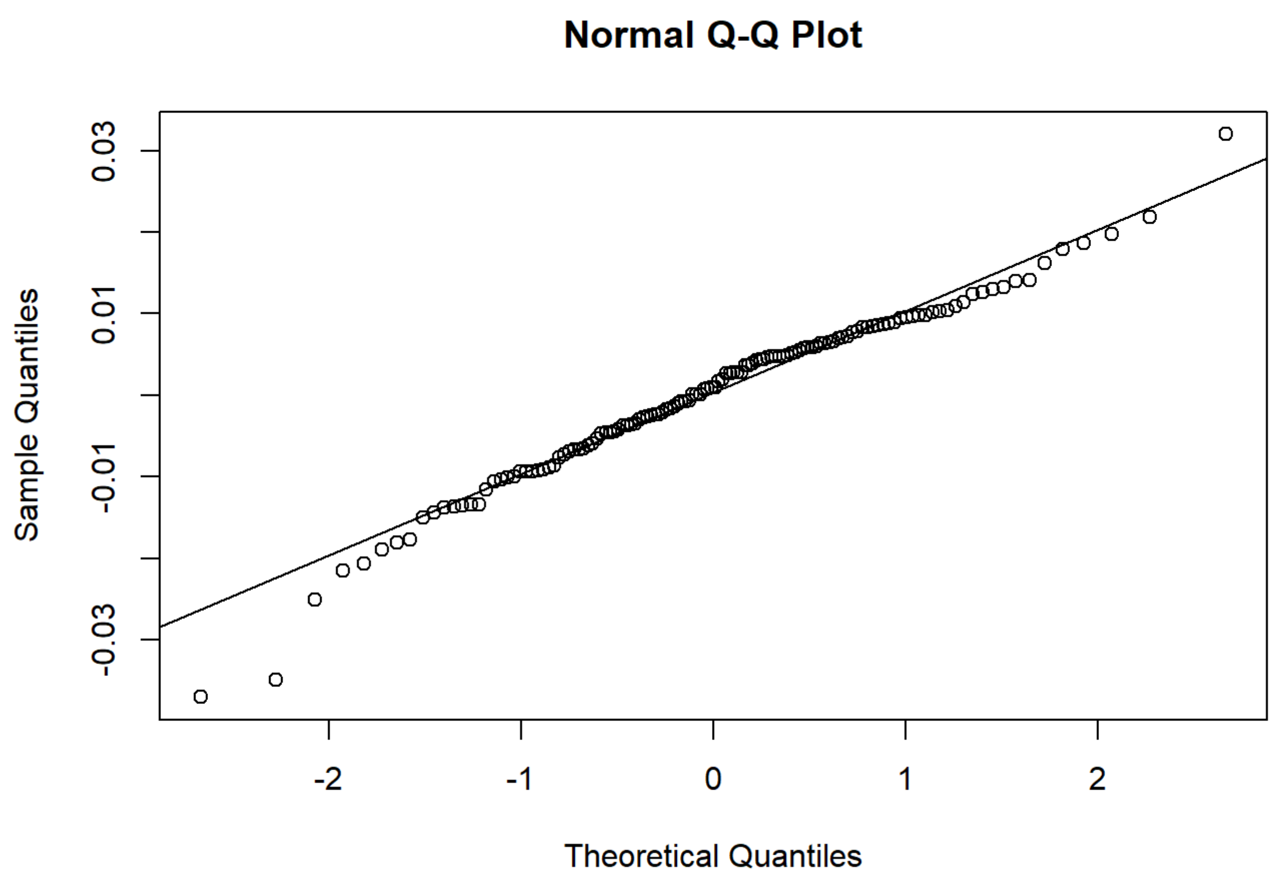
```
#Assumptions
errors<- sarima_model$residuals

acf(errors)
```



```
Box.test(errors, lag = log(train_lim), type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  errors
## X-squared = 0.74849, df = 4.8675, p-value = 0.9772
```

```
#Asumption of independece valid

qqnorm(errors)
qqline(errors)
```



```
shapiro.test(errors)
```

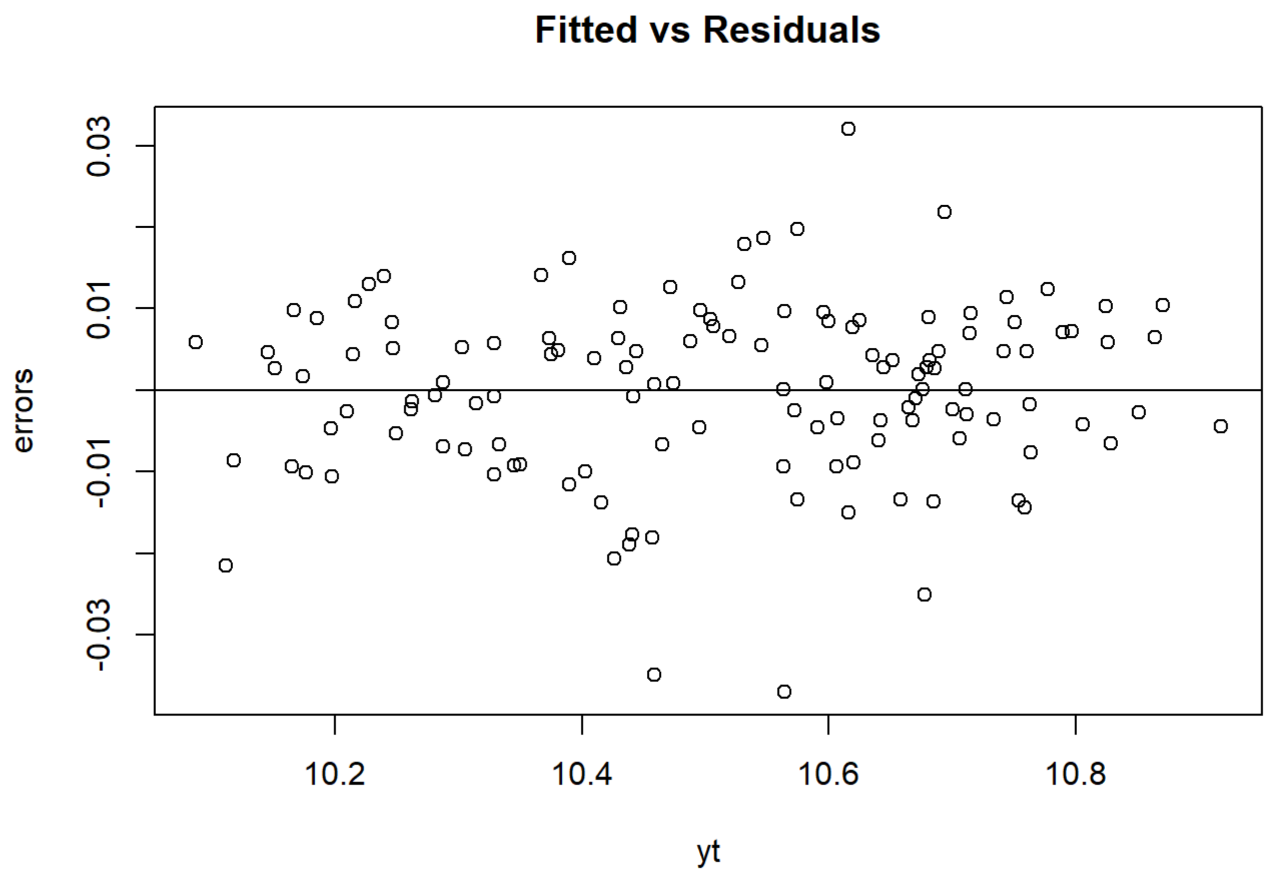
```
##
##  Shapiro-Wilk normality test
##
## data:  errors
## W = 0.97396, p-value = 0.01322
```

```
ks.test(errors, 'pnorm', 0, sqrt(sarima_model$sigma2))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  errors
## D = 0.075519, p-value = 0.4487
## alternative hypothesis: two-sided
```

We see that for Shapiro Wilk, the assumption of normality fails whereas for Kolmogorv Smirnov test, the assumption of normality cannot be rejected.

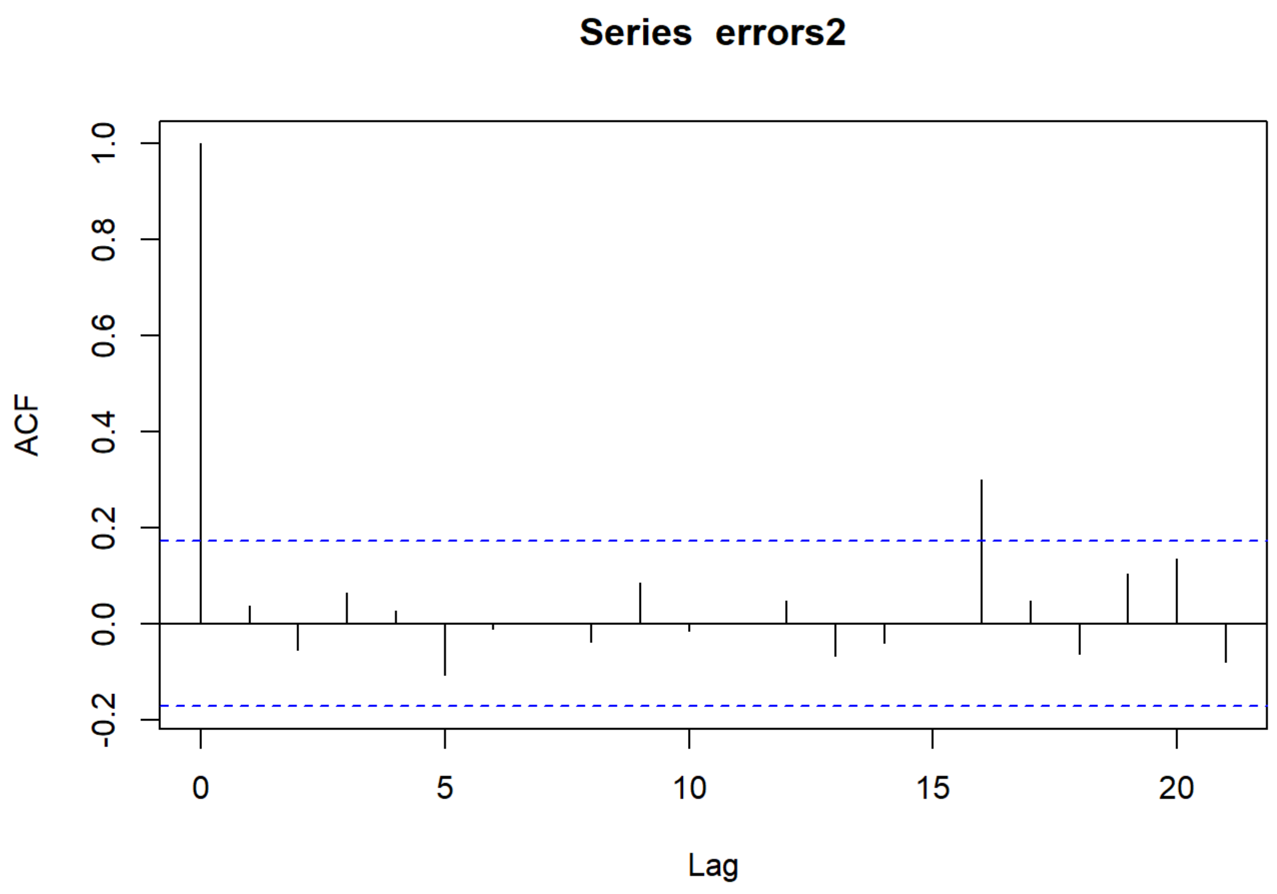
```
plot(yt, errors, main = 'Fitted vs Residuals')
abline(h=0)
```



Other than the debatable normality of errors assumption this model is outperforming every other model so far.

Let’s look at the acf of squared residuals to check if it is fine or ARCH/GARCH model is needed.

```
errors2<- errors^2
acf(errors2)
```



```
Box.test(errors2, lag =log(train_lim), type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  errors2
## X-squared = 1.252, df = 4.8675, p-value = 0.933
```

We see that the model is fine and has produced best results seen so far.