INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Department of Chemical Engineering

CH5350 Applied Time-Series Analysis (Jul – Nov 2018)

Honor Code

I Abhijeet Mavi with Roll No. BE14B001, have worked out the project in all honesty without consulting any other person or referring to other resources for solutions.

Signature of the student

Date: 03/12/2018

Project Report: CH5350

Abhijeet Mavi

TAR Modelling

For a 2-regime TAR model, we have

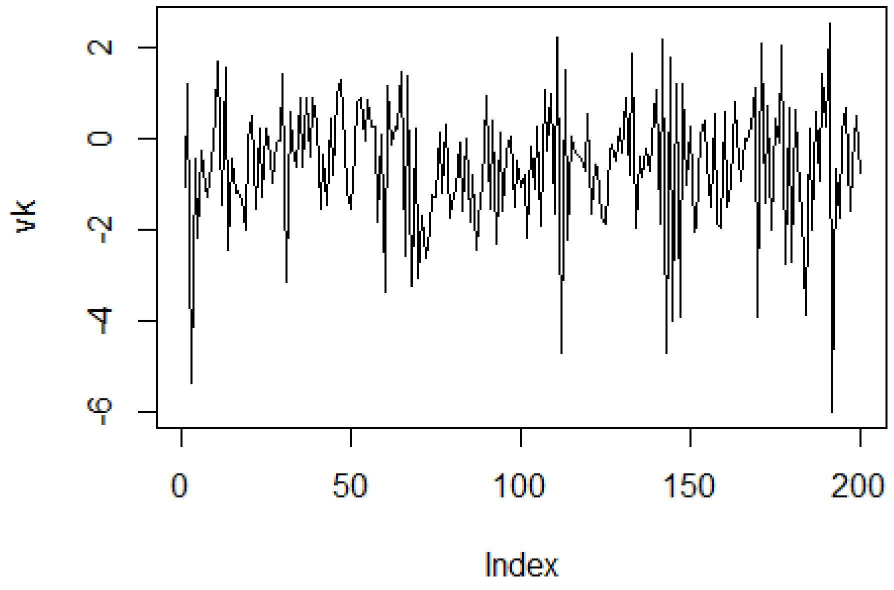
$$v[k] = -d_0^{(1)} - d_1^{(1)} v[k-1] - d_2^{(1)} v[k-2] + e_1[k]$$

$$v[k] = -d_0^{(2)} - d_1^{(2)} v[k-1] - d_2^{(2)} v[k-2] + e_2[k]$$

where, $v[k-1] <= \gamma$ for first series and $v[k-1] > \gamma$ for the second series. Here, the error in question have different standard deviations as well.

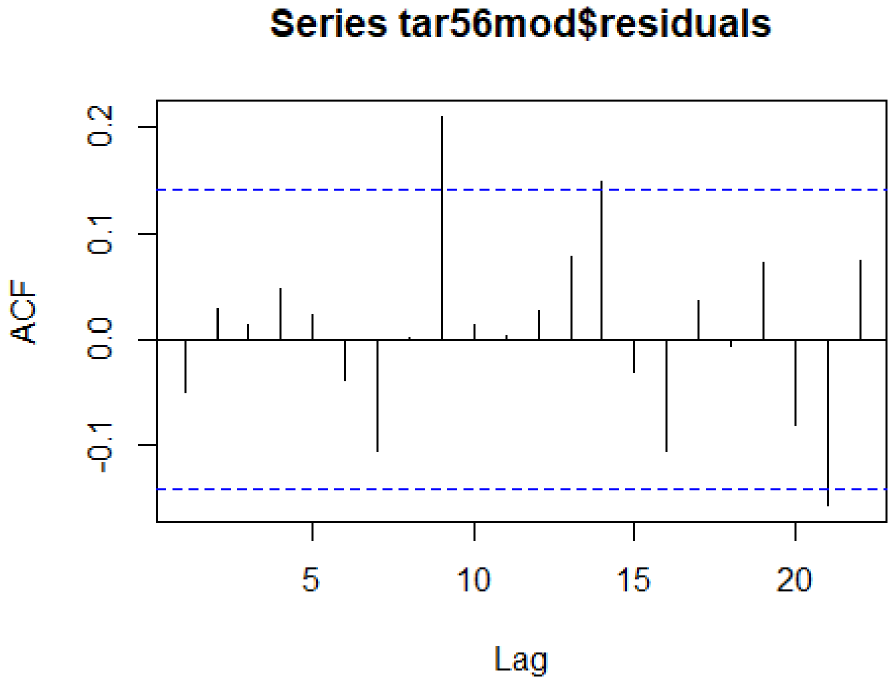
Now, using **tar()** we will fit a TAR model here. Let's try for a higher model.

```
load('projq1a.Rdata')
library(TSA)

##
## Attaching package: 'TSA'

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

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

plot(vk,type='l')
```

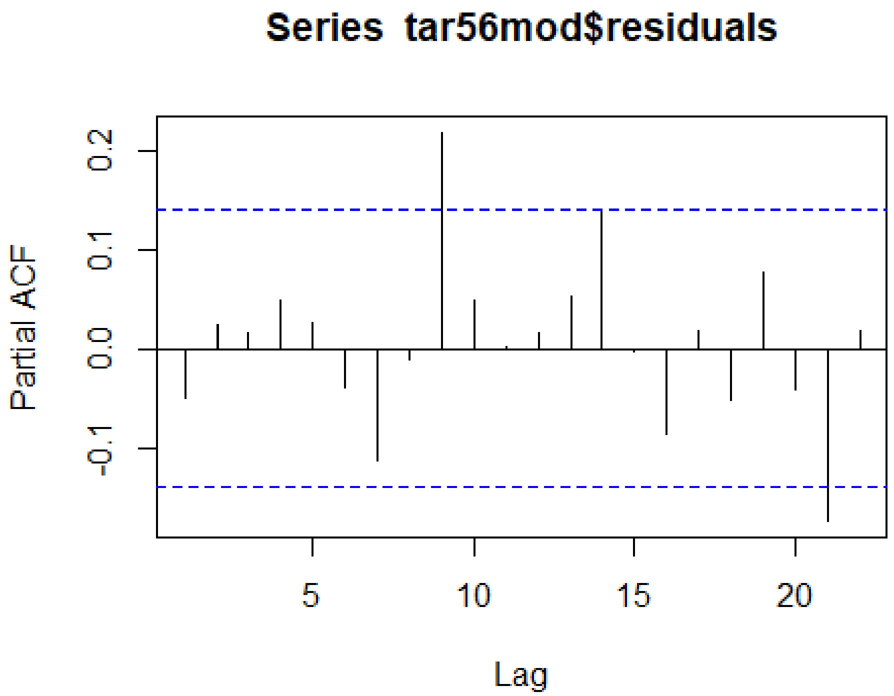


Clearly, the series does not tell anything straight forward. We will now check for a high order TAR model.

```r
tar56mod=tar(vk,5,6,5)
acf(tar56mod$residuals)
```

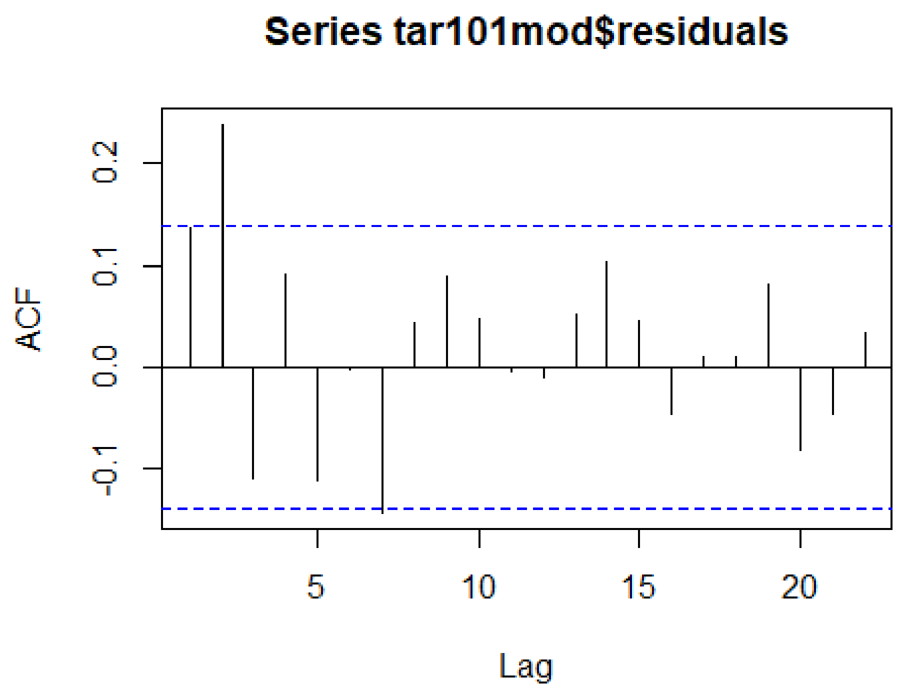


```
pacf(tar56mod$residuals)
```

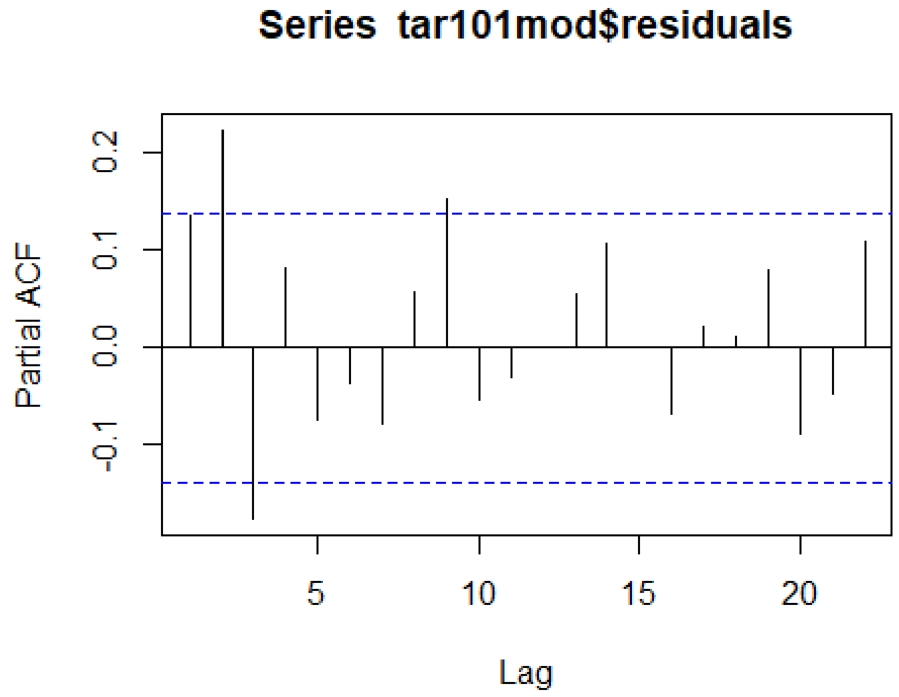


Here, I observe that the model is properly underfit, but has a lot of parameters making it an expensive computational process. We, instead fit a lower order TAR model as follows:

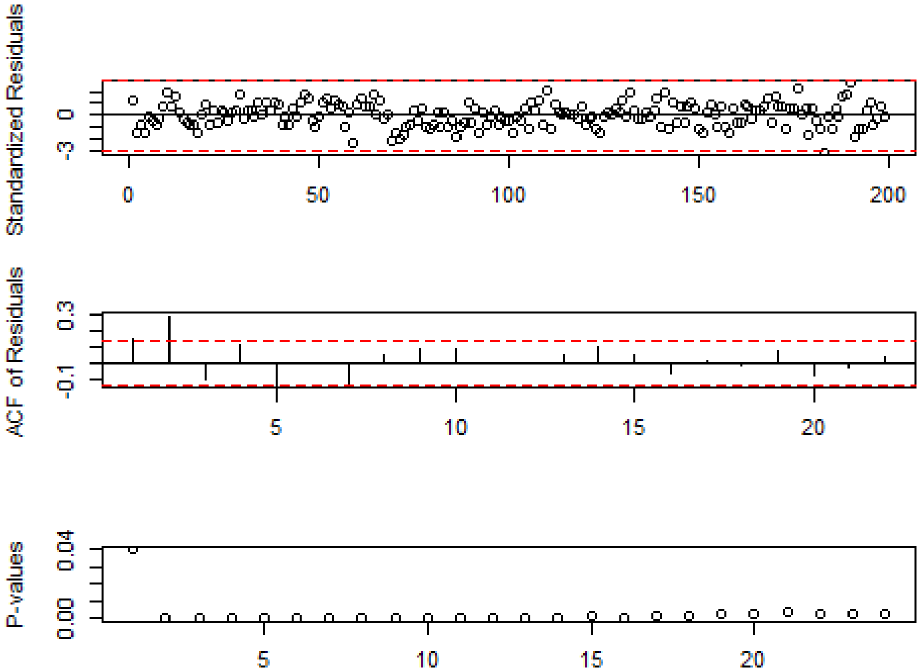
```
tar101mod=tar(vk,1,0,1)
acf(tar101mod$residuals)
```



```
pacf(tar101mod$residuals)
```

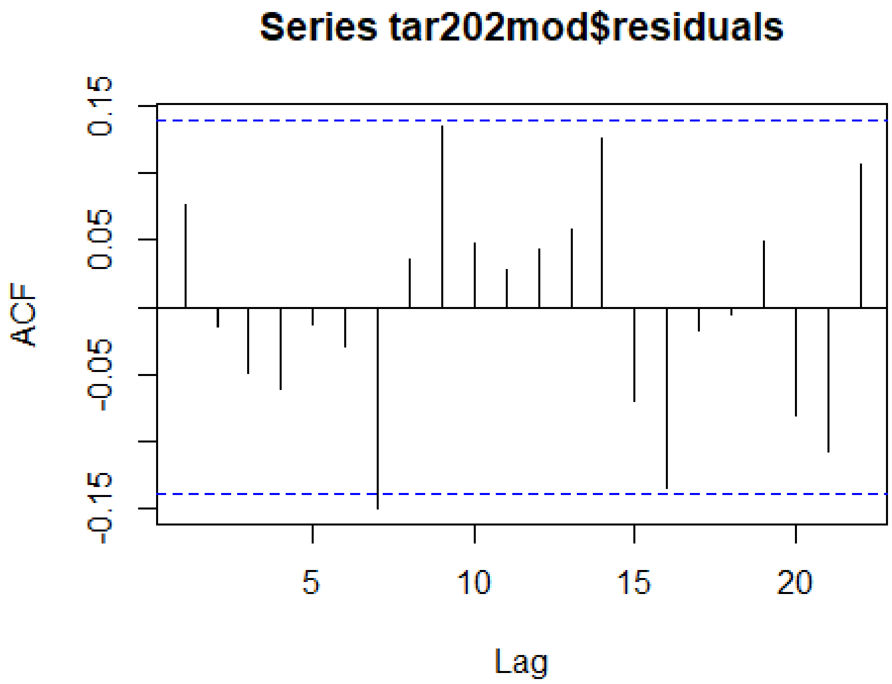


```
tsdiag(tar101mod)
```

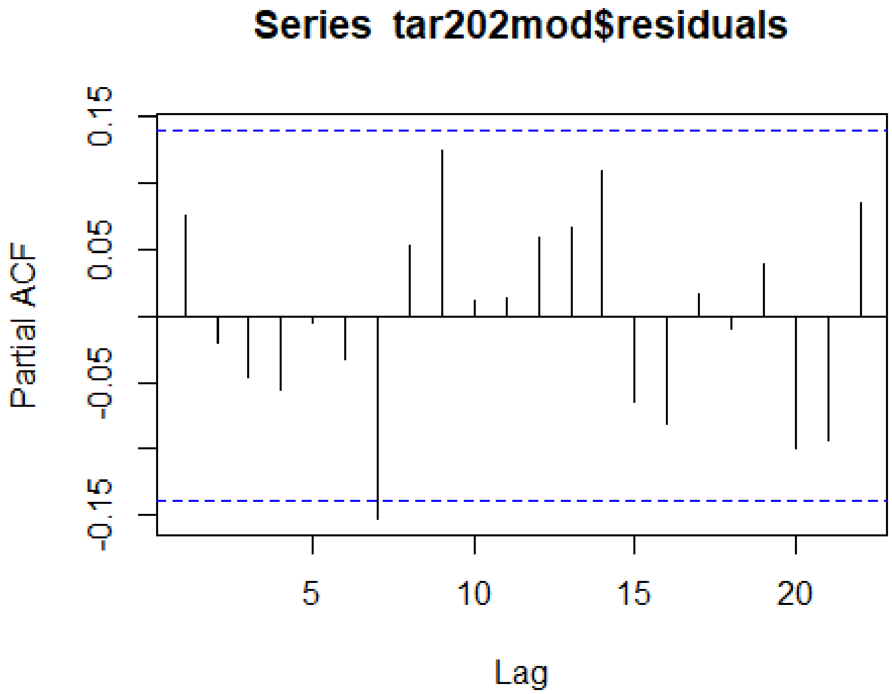


It has some values coming out for white noise test. Let's see what we get for TAR(2,0,2).

```
tar202mod=tar(vk,2,0,2)
acf(tar202mod$residuals)
```



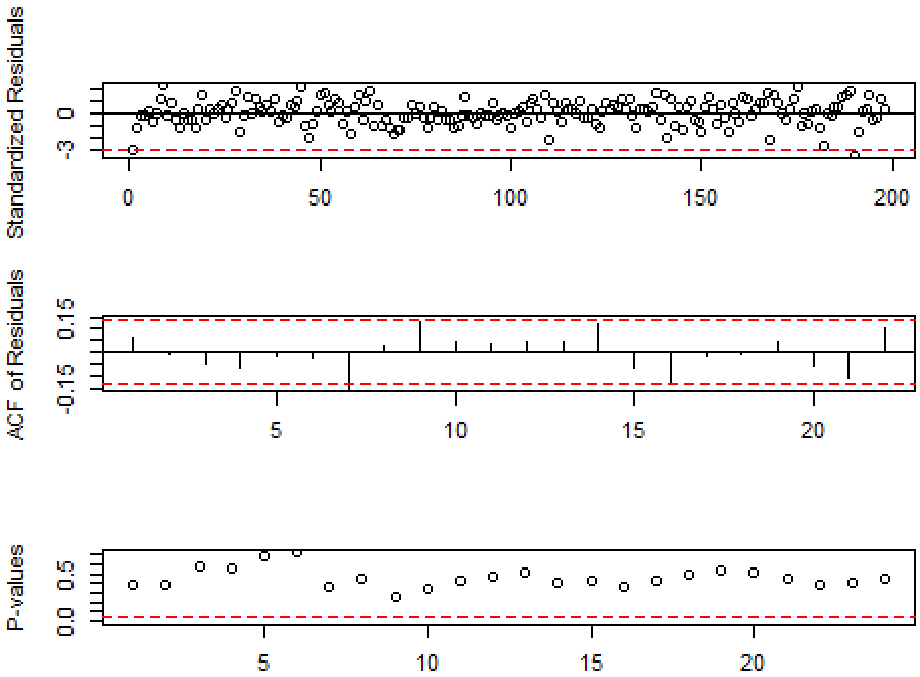
```r
pacf(tar202mod$residuals)
```



This model fits perfectly for our given dataset. Hence, we conclude that the given dataset follows a 2-regime TAR(2,0,2) model. PLease note that, I tried various permutations of model orders and this one came to be with least number of parameters.

Now, we check the parmaters for our proven TAR model.

```
tar202mod$qr1$coefficients

## intercept-vk       lag1-vk      lag2-vk
##   -0.7312821   -0.3731717    0.2474873

tar202mod$qr2$coefficients

## intercept-vk
##    0.8008254

tar202mod$thd

##
## 0.877825

tsdiag(tar202mod)
```



Hence, our 2-regime TAR model is as follows.

$$v[k] = -0.73 - 0.37v[k-1] + 0.25v[k-2] + e_1[k]$$

Here, $\sigma_1^2 = 1.53$

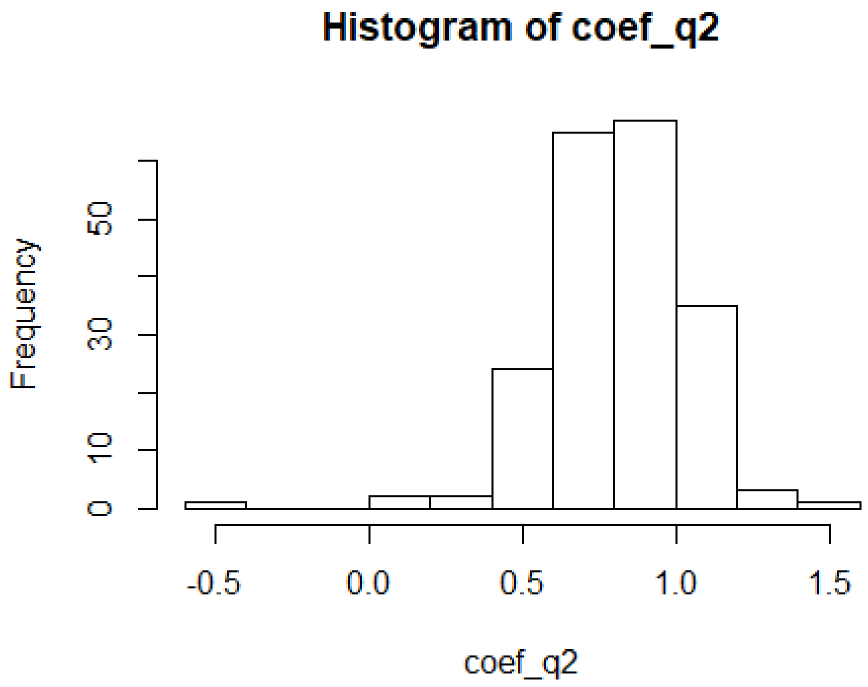
$$v[k] = 0.80 + e_2[k]$$

And, $\sigma_2^2 = 0.97$

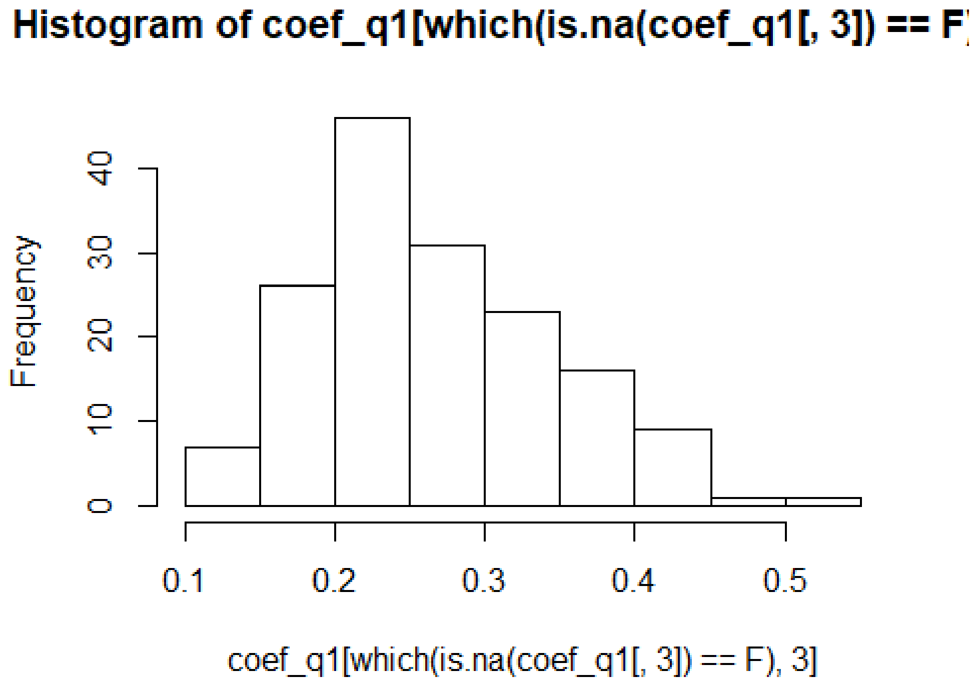
Here, the threshold $\gamma = 0.88$

We now look at **bootstrapping**. I first extracted the residuals from the two series and then sampled them to add to the original series.

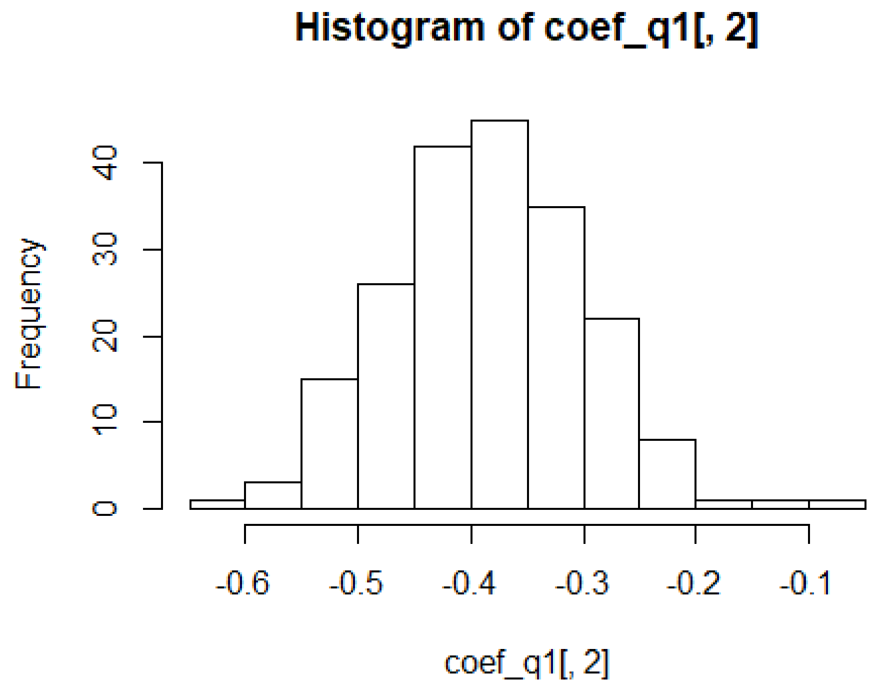
```r
coef_q1=matrix(NA,nrow=200,ncol=3)
coef_q2={}

for (i in 1:200){
s1=sample(tar202mod$residuals, length(tar202mod$residuals),replace = T)


vk_new=tar.sim(n=198,tar202mod,e=s1,ntransient = 0)

tar_new=tar(vk_new$y[1:196],2,0,2)

coef_q1[i,1]=tar_new$qr1$coefficients[1]
coef_q1[i,2]=tar_new$qr1$coefficients[2]
coef_q1[i,3]=tar_new$qr1$coefficients[3]

coef_q2[i]=tar_new$qr2$coefficients
}

hist(coef_q2)
```



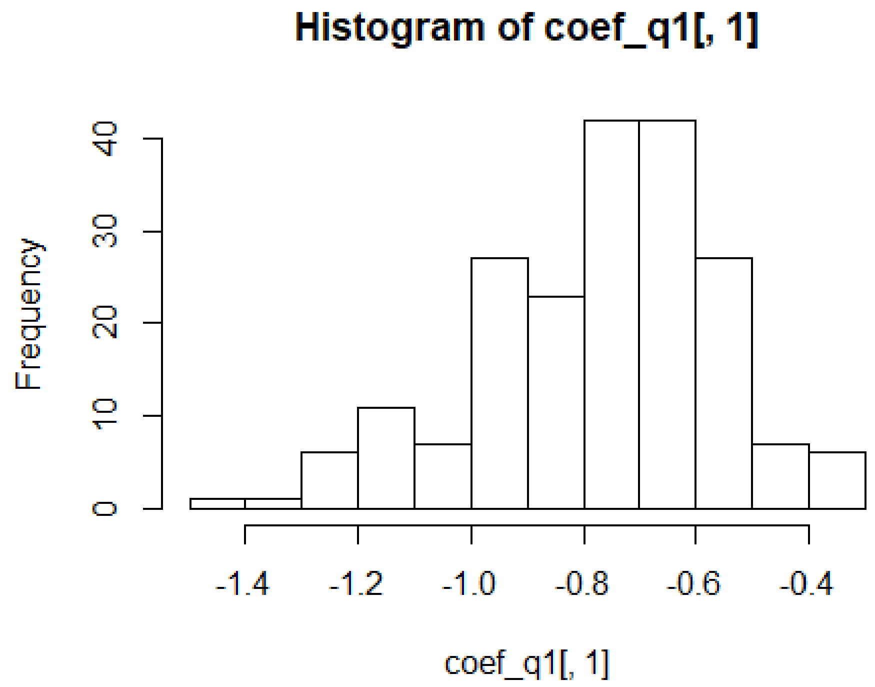
```
hist(coef_q1[which(is.na(coef_q1[,3])==F),3])
```



```
hist(coef_q1[,2])
```



```r
hist(coef_q1[,1])
```



```r
mean(coef_q1[,1])
```

```
## [1] -0.7705045

var(coef_q1[,1])

## [1] 0.04546536

mean(coef_q1[,2])

## [1] -0.3830073

var(coef_q1[,2])

## [1] 0.007464849

mean(coef_q1[which(is.na(coef_q1[,3])==F),3])

## [1] 0.2664015

var(coef_q1[which(is.na(coef_q1[,3])==F),3])

## [1] 0.006176923

mean(coef_q2)

## [1] 0.8060616

var(coef_q2)

## [1] 0.05269835
```

In the bootstrapping method, I used **tar.sim()** to simulate the TAR model for 198 time points. We did so, so that we can have 2 NA values for the **xstart** attribute of the tar.sim. I took the new residues through **sample()** function in R and then tar.sim() was used to simulate the time series data.

The bootstrapping results are clearly shown above. I have tabulated the same as shown below.

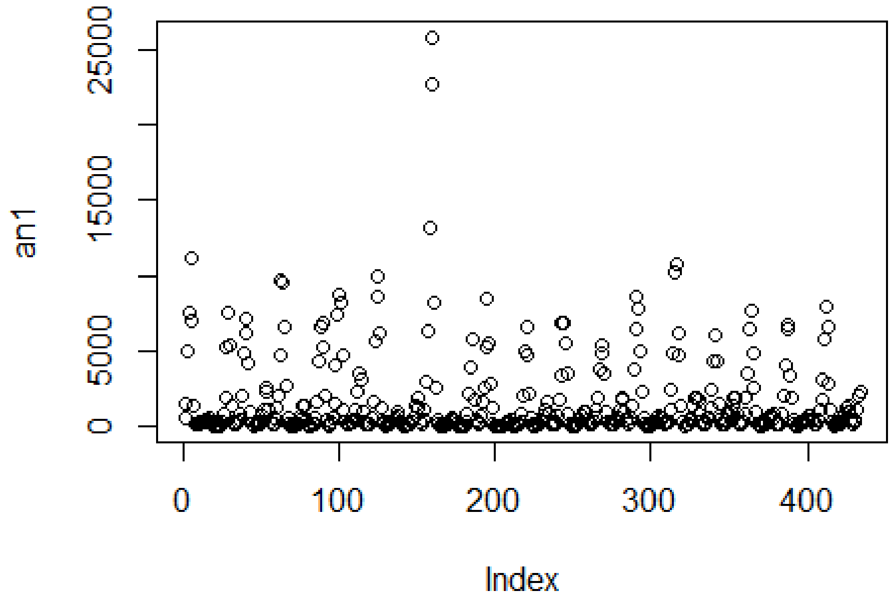
```
a1=c(sqrt(var(coef_q1[,1])),sqrt(var(coef_q1[,2])),sqrt(var(coef_q1[which(is.na(coef_q1[,3])==F),3])),sqrt(var(coef_q2)))

a2=c(mean(coef_q1[,1]),mean(coef_q1[,2]),
mean(coef_q1[which(is.na(coef_q1[,3])==F),3]),mean(coef_q2))

final_table=as.matrix(cbind(a2,a1))
rownames(final_table)=c('Regime1_d0','Regime1_d1','Regime1_d2','Regime2_d0')
colnames(final_table)=c('Mean','Standard Deviation')

final_table

##                  Mean Standard Deviation
## Regime1_d0 -0.7705045         0.21322607
## Regime1_d1 -0.3830073         0.08639936
## Regime1_d2  0.2664015         0.07859340
## Regime2_d0  0.8060616         0.22956122
```

As we can see, the mean and error are close to the real value in the 99% confidence interval for all paramters.

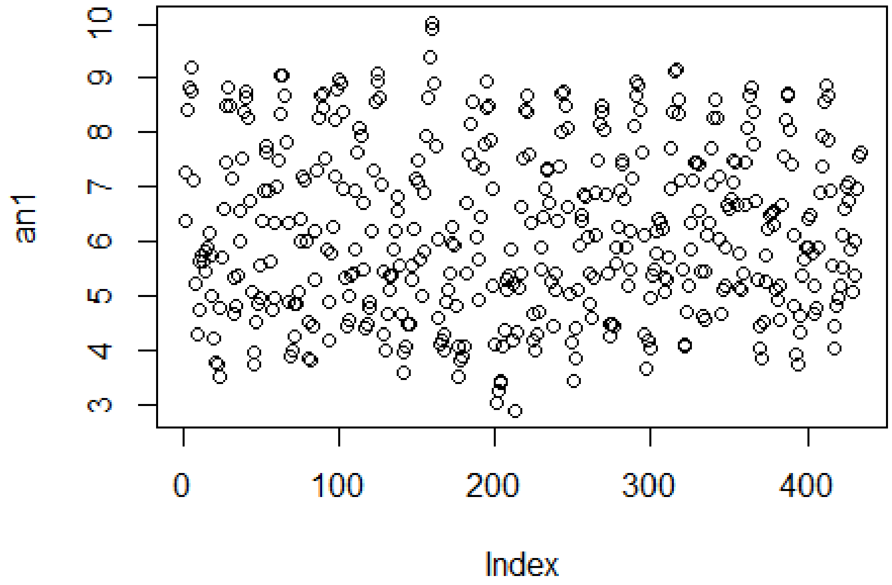
Measles Data: New York City (1928:1972)

We first plot the series here as follows and check if it needs differencing. Also, I have taken 434/534 data points as test set and the rest as the validation set. But, I presumed that the dta shows non-stationarity in variance as can be vaguely conveyed through the plot of the model.

```
library(readr)

##
## Attaching package: 'readr'

## The following object is masked from 'package:TSA':
##
##     spec

monthly_reported_number_of_cases <- read_csv("monthly-reported-number-of-
cases.csv")

## Parsed with column specification:
## cols(
##   Month = col_character(),
##   `Monthly reported number of cases of measles, New York city, 1928-1972`
= col_character()
## )

## Warning: 1 parsing failure.
## row col  expected    actual                                   file
## 535  -- 2 columns 3 columns 'monthly-reported-number-of-cases.csv'

an=monthly_reported_number_of_cases$`Monthly reported number of cases of
measles, New York city, 1928-1972`

an=as.numeric(an)

## Warning: NAs introduced by coercion

an1=as.numeric(an[1:434])
library(forecast)
plot(an1)
```



```
an1=BoxCox(as.numeric(an1),lambda = 'auto')

plot(an1)
```

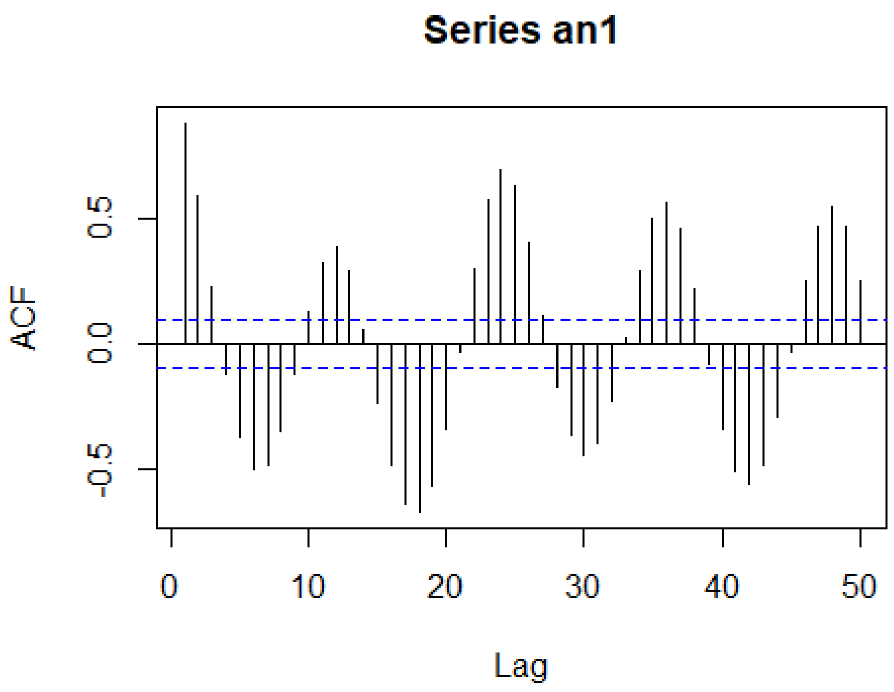


```
library(aTSA)

##
## Attaching package: 'aTSA'

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

## The following object is masked from 'package:graphics':
##
##     identify

adf.test(an1)

## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -1.17  0.2620
## [2,]   1 -3.01  0.0100
## [3,]   2 -2.14  0.0329
## [4,]   3 -1.68  0.0906
## [5,]   4 -1.40  0.1779
## [6,]   5 -1.16  0.2645
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0  -5.24    0.01
## [2,]   1 -15.46    0.01
## [3,]   2 -12.49    0.01
## [4,]   3 -11.14    0.01
## [5,]   4  -9.79    0.01
## [6,]   5  -8.94    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0  -5.24    0.01
## [2,]   1 -15.45    0.01
## [3,]   2 -12.48    0.01
## [4,]   3 -11.14    0.01
## [5,]   4  -9.80    0.01
## [6,]   5  -8.95    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

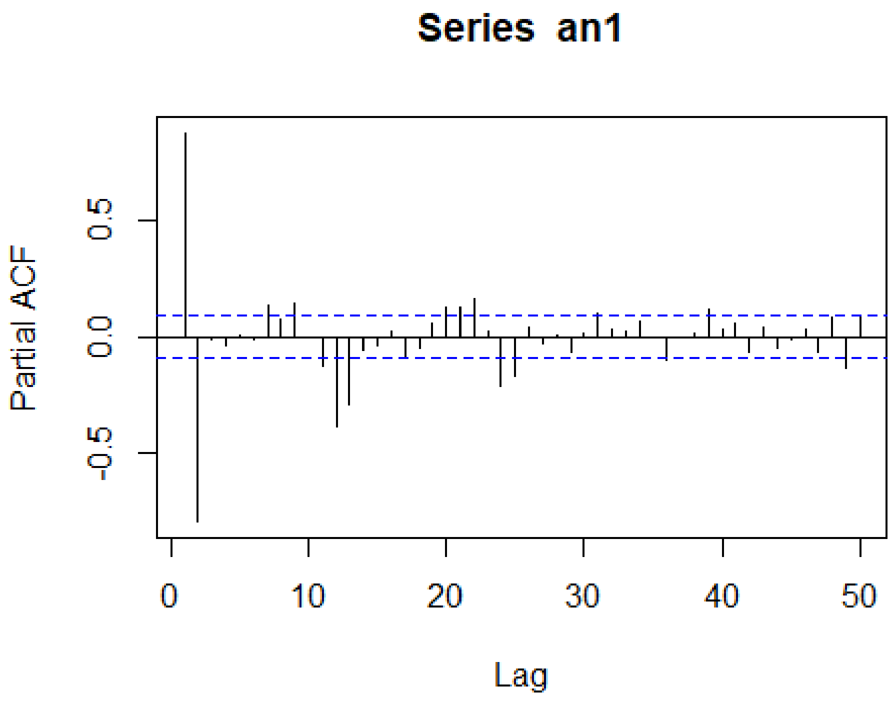
Hence, as one can see the data is much well-scattered now. **Boxcox** routine from forecast gave us $\lambda = 1.22$.

Clearly, the adf.test() shows that we do not need to difference the series. I now try to check for ACF and PACF of the series to gather some information for the true model.

```r
acf(an1,lag.max = 50)
```

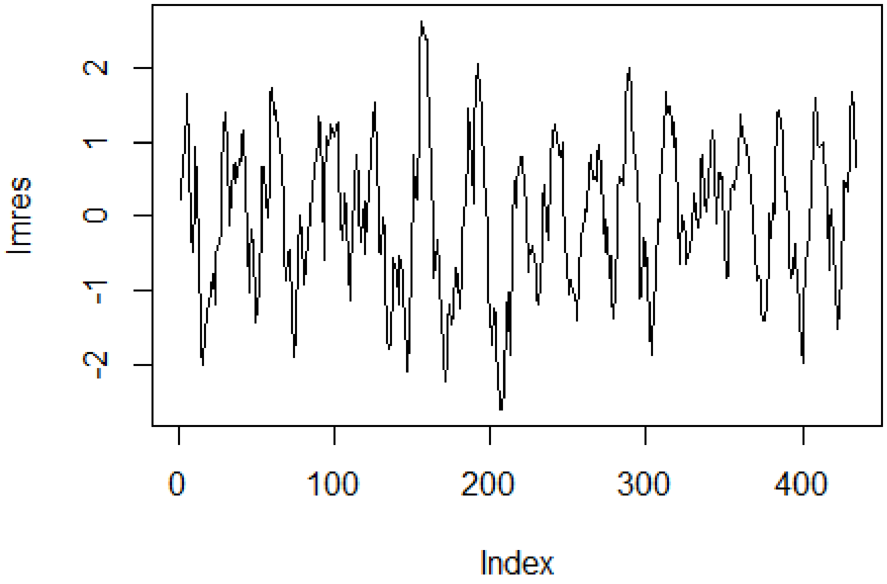


```r
pacf(an1,lag.max = 50)
```

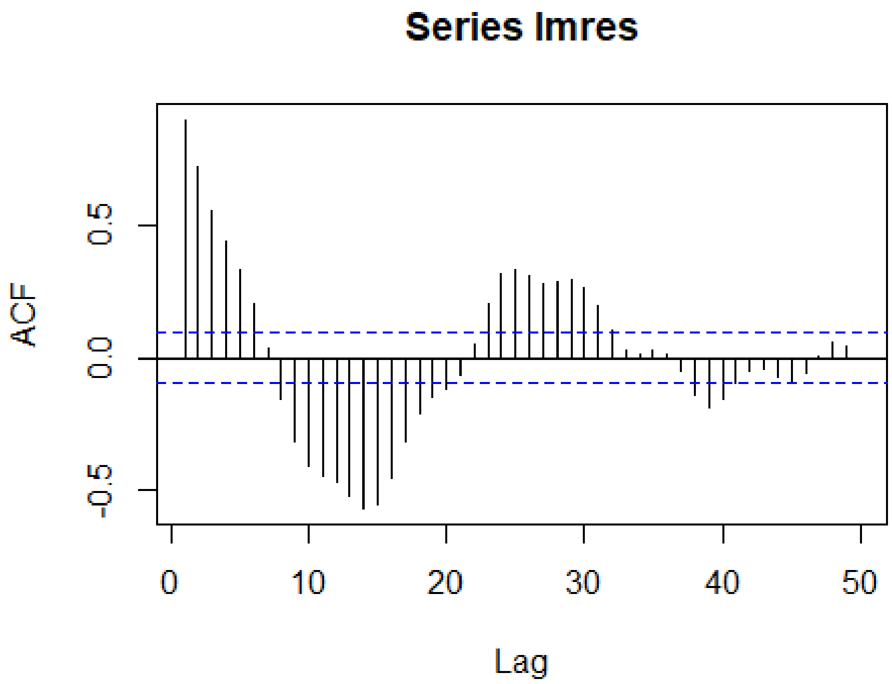


It is pretty evident from the ACF plot that we are dealing with a periodic model of period=**12**. Now, I fit the sinusoidal regression model to this series.

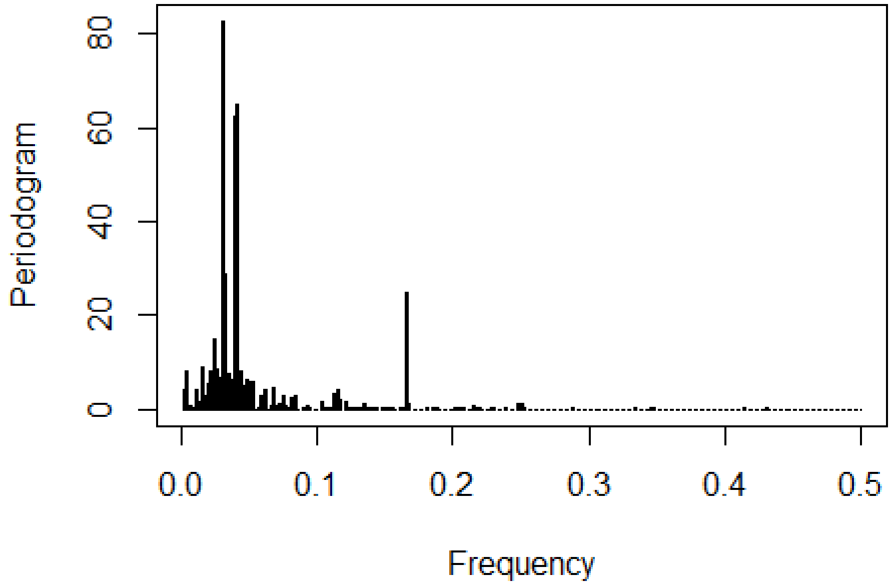
```
tvec=1:434
lmsincos=lm(an1[1:434]~I(sin(2*pi*tvec/12))+I(cos(2*pi*tvec/12)))

lmres=lmsincos$residuals
plot(lmres,type='l')
```



```
acf(lmres,lag.max = 50)
```



```
periodogram(lmres)
```

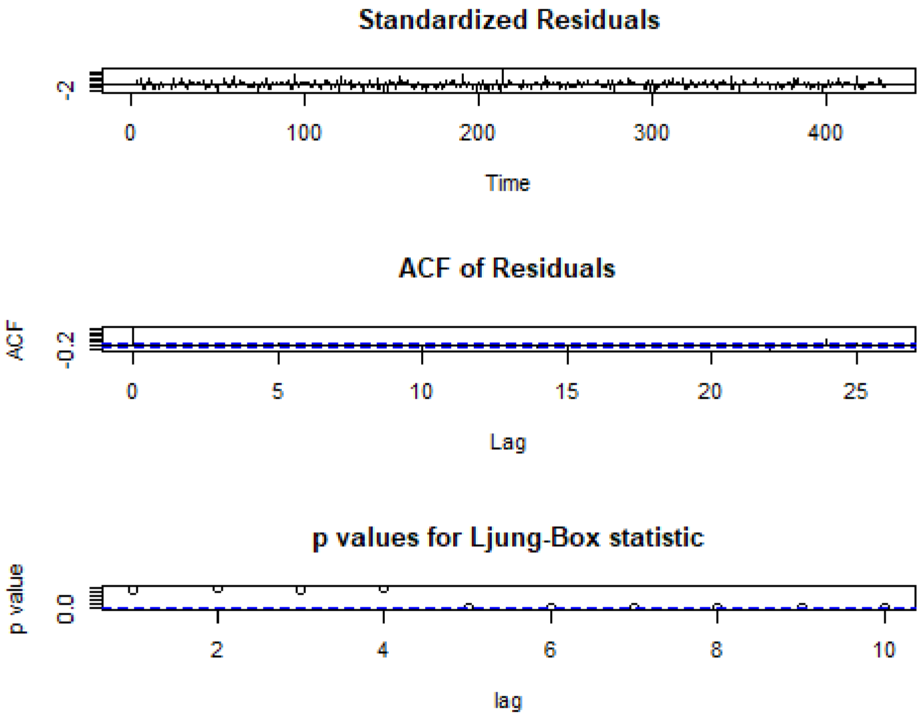


The series here clearly shows that there is decay in periodicity and hence we go for SARIMA modelling. One may wonder why do we do SARIMA. but seasonality component cannot be completely nullified by just regression modelling.

```
ar401102=arima(lmres,c(4,0,1),seasonal=list(order=c(4,0,2)))

ar401102

##
## Call:
## arima(x = lmres, order = c(4, 0, 1), seasonal = list(order = c(4, 0, 2)))
##
## Coefficients:

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

##          ar1      ar2     ar3      ar4      ma1    sar1     sar2    sar3
##       2.8487  -3.7489  2.7463  -0.8978  -0.7694  0.0447  -0.1934  0.0171
## s.e.  0.0278   0.0525  0.0525   0.0250   0.0941  0.0939   0.0704  0.0672
##          sar4     sma1    sma2  intercept
##       -0.0188  -1.0034  0.9979    -0.0051
## s.e.   0.0592      NaN  0.0001     0.0615
##
## sigma^2 estimated as 0.1089:  log likelihood = -140.2,  aic = 304.39
```

I started with a model with high AR coefficients, and I ended up with NA values. I then fit something smaller and through various permutations and combinations, I got the following model as follows:

```
ar001102=arima(lmres,c(0,0,1),seasonal=list(order=c(1,0,2)))

tsdiag(ar001102)
```



```
ar001102

##
## Call:
## arima(x = lmres, order = c(0, 0, 1), seasonal = list(order = c(1, 0, 2)))
##
## Coefficients:
##          ma1    sar1    sma1    sma2  intercept
##       0.5070  0.7115  0.1544  0.1968     0.0027
## s.e.  0.1535  0.0708  0.1166  0.1108     0.1315
##
## sigma^2 estimated as 0.1528:  log likelihood = -209.19,  aic = 428.38
```

Clearly the model satisfies the BLP test for whiteness and is parsimonious also. Fuhtermore, the model is not overfitting in this respect with 99% confidence interval. Hence, this is our final model with period 12. But this is not the right answer since parameters are overfitted. Also, there are some unexplained effects for the p-value being low at later intervals. We now turn to something called **constrained SARIMA** model. Through various trials I concluded that the following method fit the best where AR1 and SMA2 are rendered 0. We get a p-value well above 0.05, hence making our model consistent.

```
ar201212=arima(lmres,c(02,0,02),seasonal=list(order=c(0,0,2))
               ,fixed = c(0,NA,NA,NA,NA,0,NA),transform.pars = F)
ar201212
```

```
##
## Call:
## arima(x = lmres, order = c(2, 0, 2), seasonal = list(order = c(0, 0, 2)),
transform.pars = F,
##     fixed = c(0, NA, NA, NA, NA, 0, NA))
##
## Coefficients:
##       ar1     ar2     ma1     ma2    sma1  sma2  intercept
##         0  0.5412  0.5105  0.2621  0.8562     0     0.0035
## s.e.    0  0.0673  0.0522  0.0715  0.0326     0     0.1328
##
## sigma^2 estimated as 0.1512:  log likelihood = -207.01,  aic = 424.02
```

```
Box.test(ar201212$residuals)
```

```
##
##  Box-Pierce test
##
## data:  ar201212$residuals
## X-squared = 0.0022559, df = 1, p-value = 0.9621
```

$$v[k] = \left(\frac{(1+0.5429q^{-2})}{1+1.3772q^{-1}+0.4683q^{-2}}(1+0.2287q^{-24})e[k] + 2.152sin(2\pi t/12) - 1.36cos(2\pi t/12) + 6.856\right)^{1.22}$$

Now, I simulated this series for 534 data points and then checked RMSE for last 100 data points using **rmse()** function in R.

```
library(CombMSC)
```

```
##
## Attaching package: 'CombMSC'

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

```
s1=sarima.Sim(n=534,period=12,
model=list(ar=c(0,0.5421),ma=c(0.5353,0.2816)),seasonal=list(ma=c(0.8621,0)))

tvec2=1:534

s2=(2.152*sin(2*pi*tvec2/12))-(1.360*cos(2*pi*tvec2/12))+6.856

s_final=(s1+s2)^1.22

rmse=(s_final[which(is.na(s_final)==F)]-an[which(is.na(s_final)==F)])^2
rmse=mean(rmse[431:530])
rmse=sqrt(rmse)
```

Hence, the final RMSE comes out to be about 752.807.