Assignment 3: Applied Time Series Analysis

Abhijeet Mavi: BE14B001

# ARIMA Process

## 1.a)

Here, it is given that

This can be written as:

This when written in simplest form to give h[n], we get,

Hence,

**The above term is for all n>=0**.

For the given ARMA (1,1) process, we can do the same transformation, we will get after Taylor series expansion,

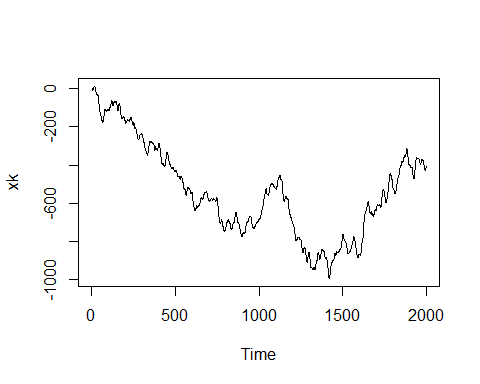
The above term is for all n>0.

We see that **h[n] is 1 for n=1** and **0 for all n<0**.

## 1.b)

Here, we first plot the data.

load("a3\_q1.Rdata")  
plot(xk)

 As one can see, the data is clearly non-stationary. So, we check using **adf.test()** whose null hypotheses states that the data has a unit root.

library(aTSA)

##   
## Attaching package: 'aTSA'

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

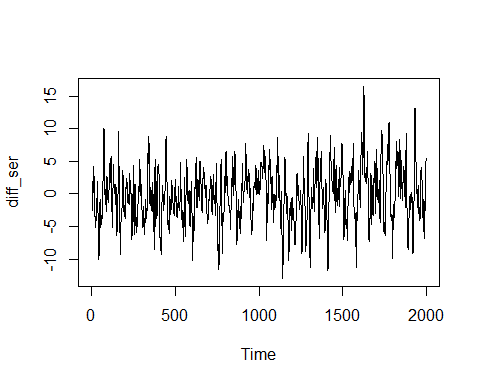
adf.test(xk)

## Augmented Dickey-Fuller Test   
## alternative: stationary   
##   
## Type 1: no drift no trend   
## lag ADF p.value  
## [1,] 0 0.599 0.816  
## [2,] 1 -0.876 0.366  
## [3,] 2 -0.184 0.591  
## [4,] 3 -0.318 0.552  
## [5,] 4 -0.288 0.561  
## [6,] 5 -0.293 0.559  
## [7,] 6 -0.274 0.565  
## [8,] 7 -0.267 0.567  
## Type 2: with drift no trend   
## lag ADF p.value  
## [1,] 0 -3.45 0.0100  
## [2,] 1 -2.72 0.0743  
## [3,] 2 -2.09 0.2928  
## [4,] 3 -2.11 0.2828  
## [5,] 4 -2.18 0.2565  
## [6,] 5 -2.21 0.2459  
## [7,] 6 -2.21 0.2423  
## [8,] 7 -2.13 0.2771  
## Type 3: with drift and trend   
## lag ADF p.value  
## [1,] 0 0.292 0.990  
## [2,] 1 -2.507 0.362  
## [3,] 2 -0.917 0.951  
## [4,] 3 -1.198 0.907  
## [5,] 4 -1.184 0.909  
## [6,] 5 -1.212 0.905  
## [7,] 6 -1.176 0.910  
## [8,] 7 -1.102 0.922  
## ----   
## Note: in fact, p.value = 0.01 means p.value <= 0.01

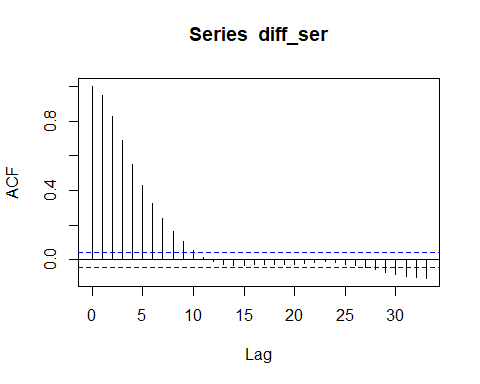
We see that null hypothesis cannot be rejected in this case and hence we need to difference the series.

Post-differencing, let’s look at the data and its respective ACF and PACF.

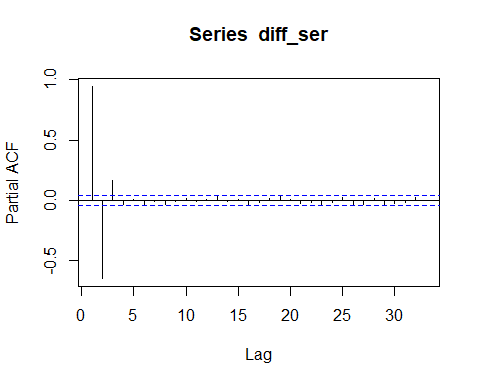
diff\_ser<-diff(xk)  
plot(diff\_ser)



acf(diff\_ser)



pacf(diff\_ser)



From ACF plot, it can be seen that it is an MA(10) process, but PACF plot shows that it is an AR(3) process. Lets try to model both.

ma10mod <- arima(diff\_ser,order=c(0,0,10))  
ma10mod

##   
## Call:  
## arima(x = diff\_ser, order = c(0, 0, 10))  
##   
## Coefficients:  
## ma1 ma2 ma3 ma4 ma5 ma6 ma7 ma8  
## 1.6839 1.8658 1.7602 1.4967 1.1871 0.8722 0.5992 0.3912  
## s.e. 0.0224 0.0438 0.0598 0.0685 0.0694 0.0657 0.0621 0.0574  
## ma9 ma10 intercept  
## 0.2039 0.0555 -0.1952  
## s.e. 0.0444 0.0234 0.2498  
##   
## sigma^2 estimated as 1.013: log likelihood = -2850.68, aic = 5725.37

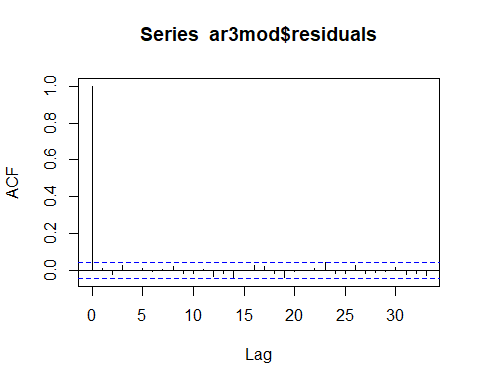
ar3mod <- arima(diff\_ser, order = c(3,0,0))  
ar3mod

##   
## Call:  
## arima(x = diff\_ser, order = c(3, 0, 0))  
##   
## Coefficients:  
## ar1 ar2 ar3 intercept  
## 1.674 -0.9213 0.174 -0.1897  
## s.e. 0.022 0.0382 0.022 0.3053  
##   
## sigma^2 estimated as 1.008: log likelihood = -2845.78, aic = 5701.57

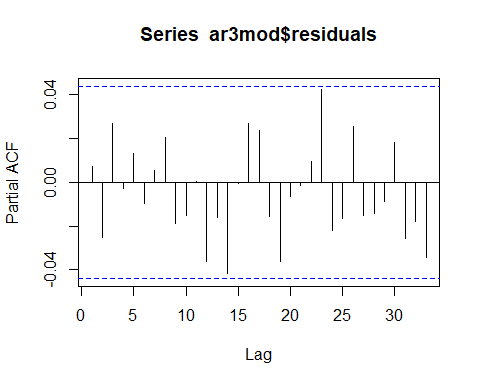
As one can see clearly that the **AIC(AR(3)) < AIC(MA(10))**, hence the preferred model is AR(3) here. On top of that, the **10th** coefficient of MA(10) overfits for a 99% confidence interval.

Now, let us look at the ACF and PACF of residuals of our AR(3) model.

acf(ar3mod$residuals)



pacf(ar3mod$residuals)



As expected, they show a white noise characteristic. Hence, we can conclude that our data is a **1-time differenced AR(3) process**.

## 1.c)

Given that,

We know that if we are given information about the past, then the best predicted value is the **Expectation E(.)** of the process.

Moving on for 2-more steps, we get

Now, let us see what is the true value of v[k+3],

This implies,

# Computing Correlations

## 2.a)

Given that,

where, u[k] and e[k] both are Gaussian White Noise processes. Hence, the **mean of y[k]=0.**

Expanding the denominator of first term to , we get

Now, given that for all l, hence,

This implies,

Since,

Here,  **since there is no term of u[k-1] in the y[k] term**.

For, , we will only consider the first term (n=0) of the y[k] sequence involving u[k] because the correlation with all other terms and e[k] will be 0.

## 2.b)

**One interesting observation that one can notice is that as we keep increasing the lag, our will only get multiplied by .**

So,

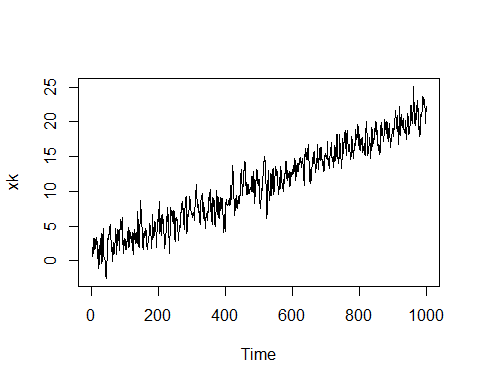
This is valid for all l>=2. Since , hence the maximum correlation will be seen at lag 2 only as the correlation will keep decreasing like an AR(1) process. The delay is seen at the point of maximum correaltion. Hence, the delay is **2**.

# Processes with Trend

## Fitting a linear trend

Here, in order to visualise the linear trend, let’s plot the series

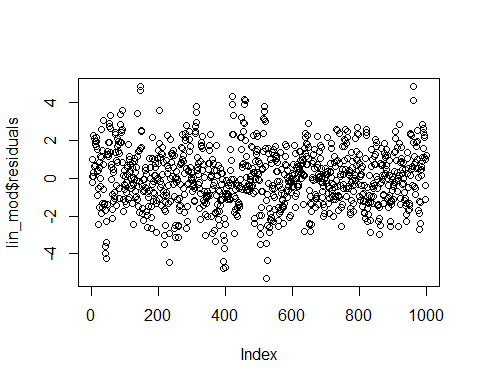
library(stats)  
  
load("a3\_q3.Rdata")  
  
plot(xk)



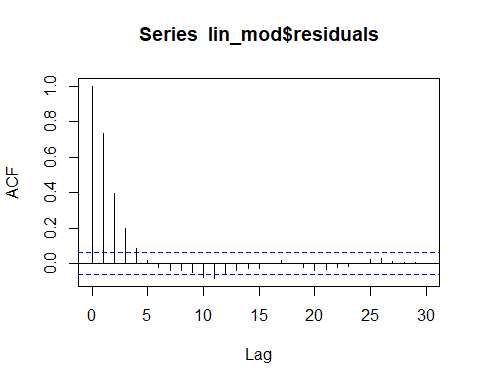
lin\_fit <- 1:1000   
  
lin\_mod <- lm(xk ~ lin\_fit)  
  
lin\_mod

##   
## Call:  
## lm(formula = xk ~ lin\_fit)  
##   
## Coefficients:  
## (Intercept) lin\_fit   
## 0.87462 0.02012

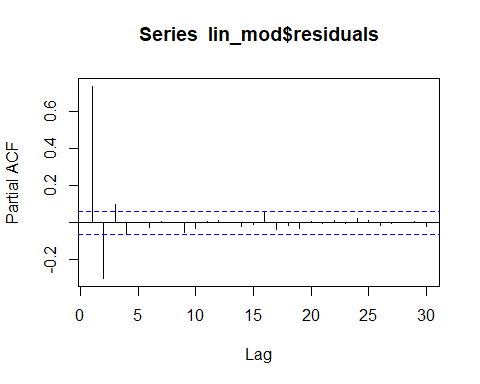
plot(lin\_mod$residuals)



acf(lin\_mod$residuals)



pacf(lin\_mod$residuals)



Here, we see that series perfectly fits a linear trend with **intercept=0.875** and **slope=0.02**. We then check the residuals of the series and we see that it is either an AR(3) or an AR(4) process as seen from PACF plot of residuals.

ar3mod <- arima(lin\_mod$residuals,order = c(3,0,0))  
ar3mod

##   
## Call:  
## arima(x = lin\_mod$residuals, order = c(3, 0, 0))  
##   
## Coefficients:  
## ar1 ar2 ar3 intercept  
## 0.9858 -0.3988 0.1001 0.0051  
## s.e. 0.0315 0.0425 0.0315 0.0999  
##   
## sigma^2 estimated as 0.9799: log likelihood = -1409.27, aic = 2828.53

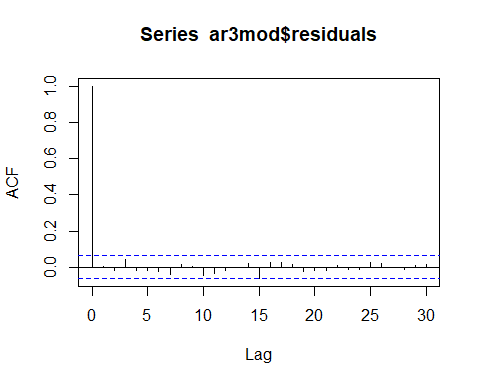
ar4mod <- arima(lin\_mod$residuals,order = c(4,0,0))  
ar4mod

##   
## Call:  
## arima(x = lin\_mod$residuals, order = c(4, 0, 0))  
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 intercept  
## 0.9919 -0.4237 0.1618 -0.0624 0.0045  
## s.e. 0.0316 0.0443 0.0443 0.0316 0.0939  
##   
## sigma^2 estimated as 0.976: log likelihood = -1407.32, aic = 2826.64

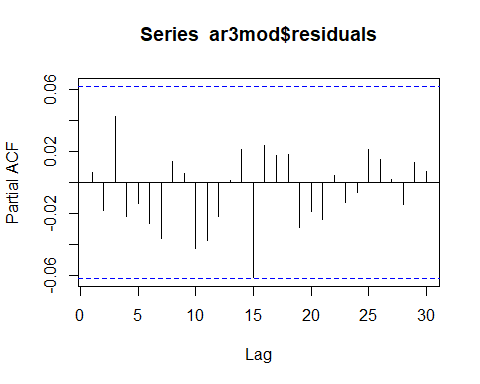
We see that the fourth coefficient of AR(4) model does not satisfy the 99% confidence interval. hence, our preferred model is AR(3) in this case.

We further check if ar3mod has its residues as white noise

acf(ar3mod$residuals)



pacf(ar3mod$residuals)



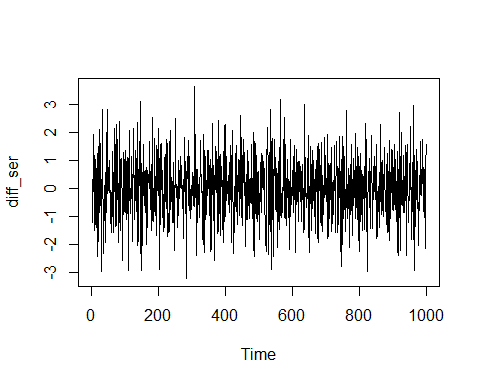
And it’s evident from above plots that the residues are of white-noise nature.

The major advantage of linear trend fitting is that the residuals will give white noise for an ARMA fit to linear tred, but while estimating the parameters of linear trend, if not done properly they will give a non-stationary process instead.

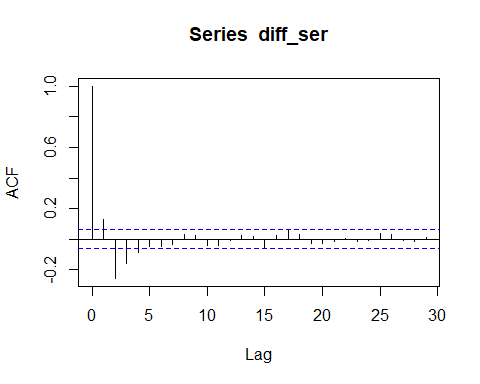
## Differencing the series

In this method, we difference the series once to accomodate the effect of linear trend and then, further fit an ARMA model as follows:

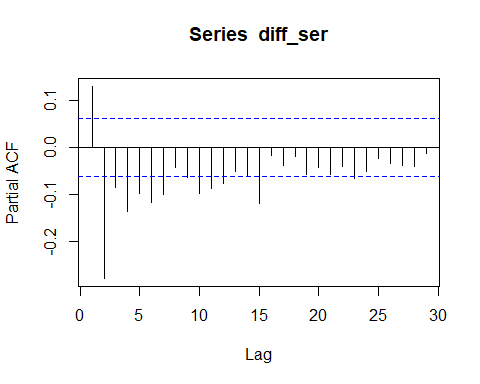
diff\_ser <- diff(xk)  
  
plot(diff\_ser)



acf(diff\_ser)



pacf(diff\_ser)

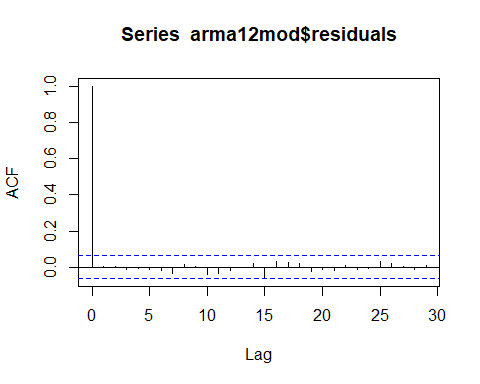


We see that the differenced series a stationary process, but one can’t make out it’s order through ACF or PACF. So, I tried various combinations of ARMA models, by keeping in mind the parsimony of the modelling technique, I was able to fit ARMA(1,2).

arma12mod <- arima(diff\_ser,c(1,0,2))  
  
arma12mod

##   
## Call:  
## arima(x = diff\_ser, order = c(1, 0, 2))  
##   
## Coefficients:  
## ar1 ma1 ma2 intercept  
## 0.5575 -0.5685 -0.4315 0.0201  
## s.e. 0.0327 0.0352 0.0350 0.0004  
##   
## sigma^2 estimated as 0.9781: log likelihood = -1409.24, aic = 2828.48

acf(arma12mod$residuals)



Hence, our final model satisfies both the overfit test (no parameter outside 99% CI) and underfit test (white noise nature of residues) as shown above, but we also see that we have an intercept term here.

A major advantage of modelling with differenced series is that it does not require parameter estimation of additional parameters since it’s a **non-parametric method**, but at the same time introduces a unit pole which needs to be accounted for and generally bring a bit complication with itself.

# Variance-type non-Stationarities

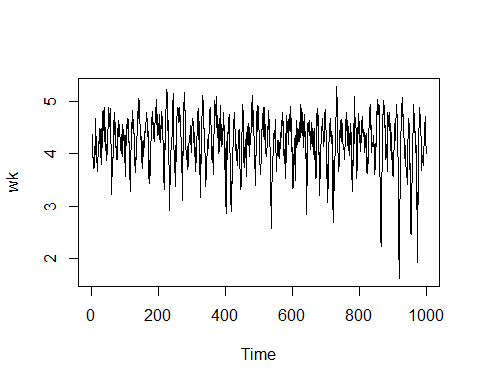
Let us first take a look at w[k] series given.

library(MLmetrics)

##   
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':  
##   
## Recall

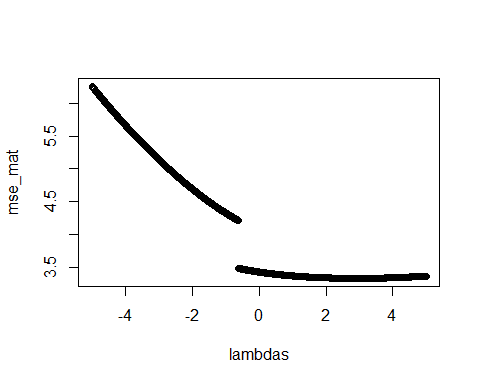
load("a3\_q4.Rdata")  
  
plot(wk)



mse\_mat <- {} #matrix for storing MSE  
  
lambdas <- {}  
  
k<-1:4999  
k<-append(k,5001:10000)  
  
#k includes all lambda [-5,5] except 0  
  
for (i in k){  
  
lambda <- -5 + (i\*0.001) #lambda changed with 0.1 increment  
   
lambdas[i] <- lambda  
  
#transforming both training and test data  
  
trans\_data <- ((wk^lambda)-1)/lambda  
  
train\_data <- trans\_data[1:800]  
  
test\_data <- wk[801:1000]  
  
#Fitting an AR model  
arPmod <- ar(train\_data,aic=TRUE, order.max=6)  
  
#Using filter, arPmod$ar gives coefficients  
new\_ser <- filter(test\_data,arPmod$ar,method = 'convolution',sides=1)  
  
#Calculating MSE  
mse\_i <- MSE(new\_ser[((arPmod$order)+1):200],test\_data[((arPmod$order)+1):200])  
  
mse\_mat[i]<-mse\_i  
  
}  
  
#For lambda=0  
  
trans\_data0 <- log(wk)  
  
train\_data0 <- trans\_data0[1:800]  
  
#Fitting an AR model  
arPmod0 <- ar(train\_data0,aic=TRUE, order.max=6)  
  
#Using filter, arPmod$ar gives coefficients  
new\_ser0 <- filter(test\_data,arPmod0$ar,method = 'convolution',sides=1)  
  
#Calculating MSE  
mse\_0 <- MSE(new\_ser0[((arPmod0$order)+1):200],test\_data[((arPmod0$order)+1):200])  
  
mse\_mat[5000] <- mse\_0  
lambdas[5000] <- 0

The above code snippet was used to evaluate the BoxCox transormation for with step size of 0.001. I evaluated MSE using MSE() of MLmetrics package and I plotted the graph as follows.

plot(lambdas,mse\_mat)



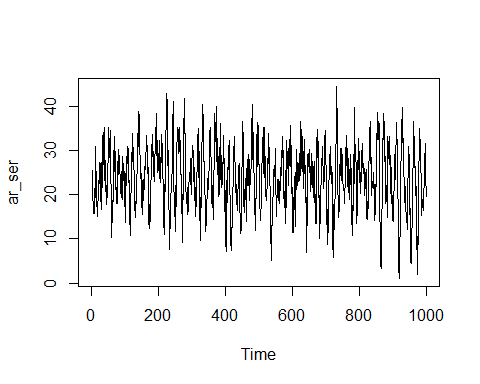
The minima is seen at  **= 2.928**. We now fit the AR model pertaining to this and see how well the coefficients are in enlighnment to BoxCox() in R as follows:

library(forecast)

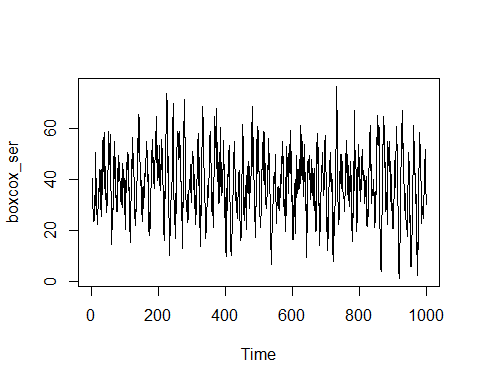
##   
## Attaching package: 'forecast'

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

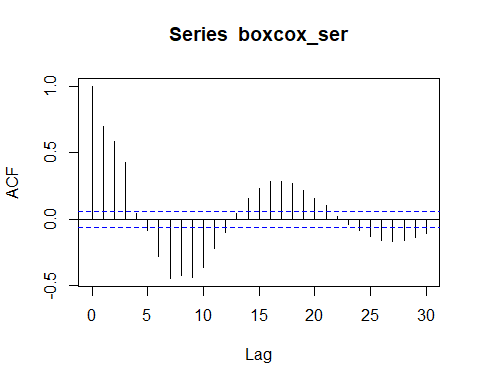
lambda\_original <- BoxCox.lambda(wk, method = 'guerrero', lower=-5,upper=5)  
boxcox\_ser<-BoxCox(wk,lambda\_original)  
  
ar\_ser<-((wk^2.928) -1)/2.928  
plot(ar\_ser)



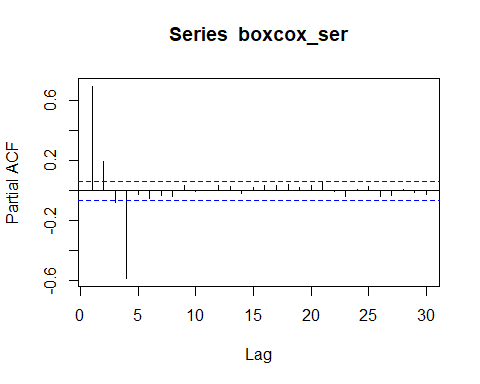
plot(boxcox\_ser)



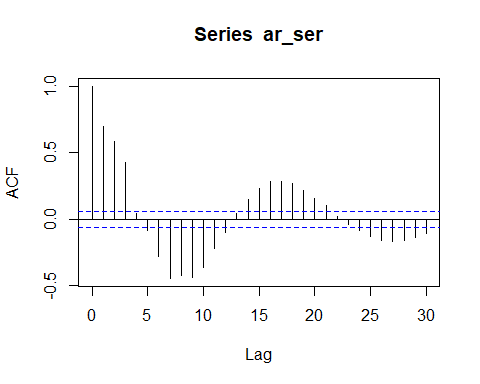
acf(boxcox\_ser)



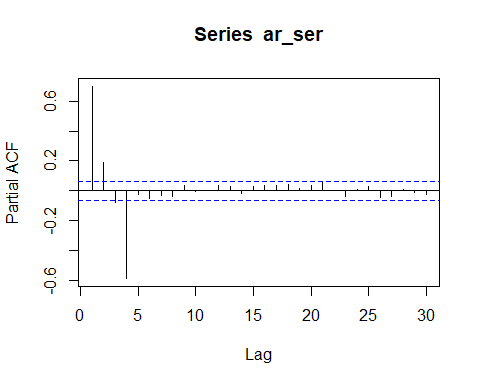
pacf(boxcox\_ser)



acf(ar\_ser)



pacf(ar\_ser)



arima(boxcox\_ser,c(4,0,0))

##   
## Call:  
## arima(x = boxcox\_ser, order = c(4, 0, 0))  
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 intercept  
## 0.5306 0.3752 0.2623 -0.5867 38.2313  
## s.e. 0.0256 0.0295 0.0295 0.0256 0.5385  
##   
## sigma^2 estimated as 50.76: log likelihood = -3383.68, aic = 6779.35

arima(ar\_ser,c(4,0,0))

##   
## Call:  
## arima(x = ar\_ser, order = c(4, 0, 0))  
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 intercept  
## 0.5321 0.3768 0.2616 -0.5882 23.9859  
## s.e. 0.0255 0.0295 0.0295 0.0256 0.3037  
##   
## sigma^2 estimated as 16.06: log likelihood = -2808.43, aic = 5628.86

As is evident from PACF plots for both series, we can model **AR(4)** processes for both series. We further find the lambda according to Guerrero’s method which is **3.3263**. We get around **12.5%** error for our value of estimated Also, the transformed series can be seen to have similar coefficients as that of the series transformed using BoxCox() routine.