Assignment 5: CH5350

Abhijeet Mavi (BE14B001)

## Sample Simulations for ARMA models

model\_aic=matrix(data=NA,nrow=3,ncol=100)  
model\_order=matrix(data=NA,nrow=4,ncol=100)  
  
for (i in 1:100){  
print(i)  
yk=arima.sim(n=600,list(ar=c(0.2,0,0.1)))  
  
#For AR model estimation  
  
armod\_aic={}  
for (j in 1:10){  
  
 armod=arima(yk,order = c(j,0,0))  
 armod\_aic[j]=armod$aic  
 }  
model\_aic[1,i]=min(armod\_aic)  
model\_order[1,i]=which.min(armod\_aic)  
  
  
# #For MA model estimation  
  
mamod\_aic={}  
for (k in 1:10){  
  
 mamod=arima(yk,order = c(0,0,k))  
 mamod\_aic[k]=mamod$aic  
}  
model\_aic[2,i]=min(mamod\_aic)  
model\_order[2,i]=which.min(armod\_aic)  
  
#For ARMA model estimation  
  
armamod <- auto.arima(yk,seasonal = FALSE,d=0,D=0,max.p=10,max.q=10,start.p = 1,start.q = 1)  
  
model\_aic[3,i] = armamod$aic  
  
model\_order[3,i] = armamod$arma[1]  
  
model\_order[4,i] = armamod$arma[2]  
}  
  
ar3\_indices = which(model\_order[1,]==3)  
  
ar3\_aic<-apply(model\_aic[,ar3\_indices], 2, min)  
  
count\_true<-length(intersect(ar3\_aic,model\_aic[1,]))  
  
print(count\_true)

I took AR, MA and ARMA models, ranging from 0 to 10th order.

For N=600 samples, we see that we get AR(3,0) **16 times** out of 300 possible orders, for the given R script. I used **forecast** package to evaluate the arma model estimations using function **auto.arima()**.

For N=100 samples, we could only get **2 reasonable models** with AR(3,0) for AIC being the model ranker.

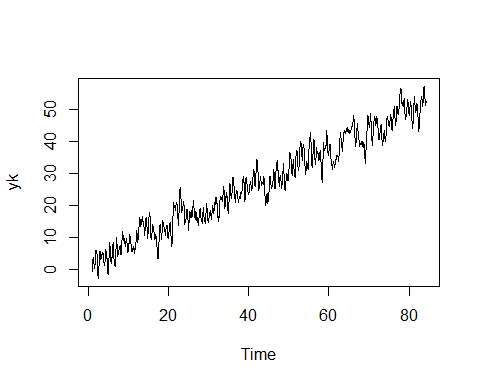
## Fitting a Seasonal Model

The seasonality in a model can be evaluated in 2 ways: in respect with **additive models** or **multiplicative models**

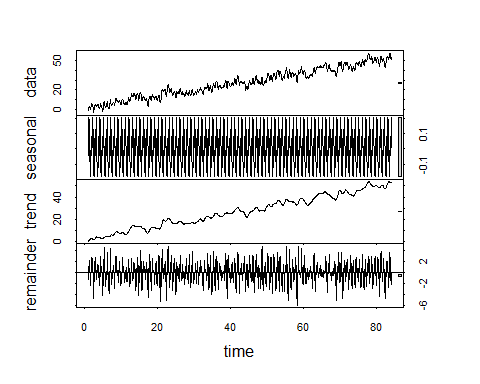
### Using stl() command

We see that the series is obviously seasonal in some way. Using **stl()**, we deduce the remainder for the decomposed series.

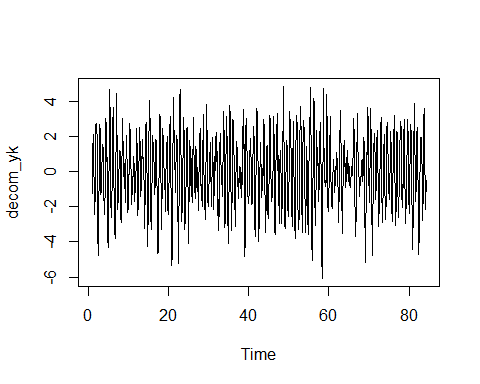
library('forecast')  
  
load('sarima\_data.Rdata')  
  
#Part 1  
  
plot(yk)



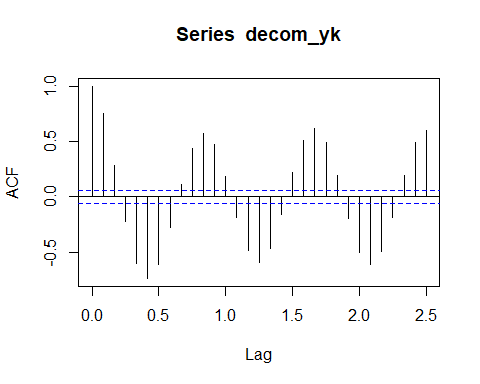
plot(stl(yk,'per'))



decom\_yk <- remainder(stl(yk,'per'))  
  
plot(decom\_yk)

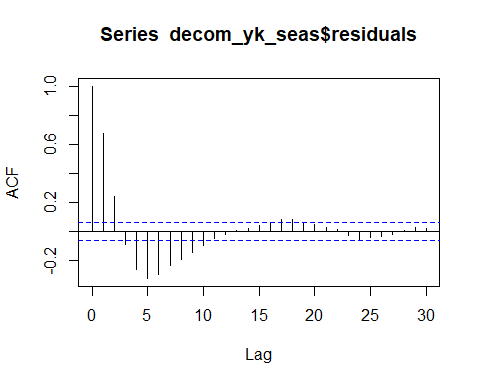


acf(decom\_yk)

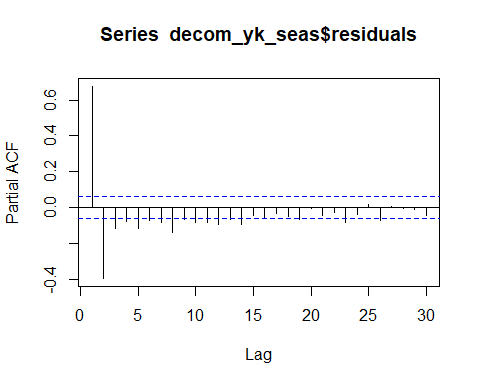


We see that the dcomposed series is seasonal with a period of about 10. We further fit a cosine wave using **lm()** and then check for its residuals.

tvec <- 1:1000  
  
decom\_yk\_seas <- lm(decom\_yk ~ I(sin(2\*pi\*(1/10)\*tvec))+ I(cos(2\*pi\*(1/10)\*tvec)))  
  
acf(decom\_yk\_seas$residuals)



pacf(decom\_yk\_seas$residuals)

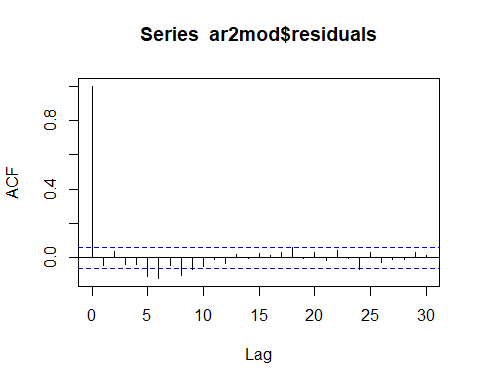


The residuals clearly show that the series is AR(2) with respect to the PACF plot. Further fitting the AR(2) model

ar2mod <- arima(decom\_yk\_seas$residuals,order=c(2,0,0))  
  
ar2mod

##   
## Call:  
## arima(x = decom\_yk\_seas$residuals, order = c(2, 0, 0))  
##   
## Coefficients:  
## ar1 ar2 intercept  
## 0.9434 -0.3940 0.0010  
## s.e. 0.0290 0.0291 0.0592  
##   
## sigma^2 estimated as 0.7116: log likelihood = -1249.31, aic = 2506.63

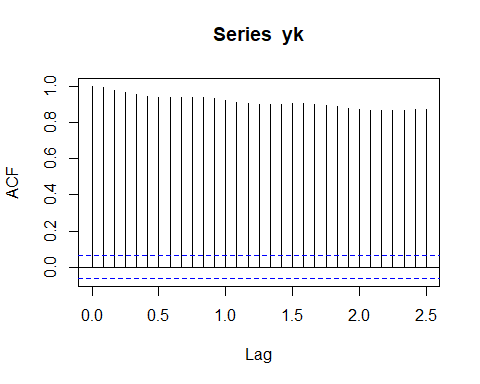
acf(ar2mod$residuals)



We see that the residuals of AR(2) model satisfy underfit overfit criteria with respect to the ACF of the plot. Hence, the final series is:

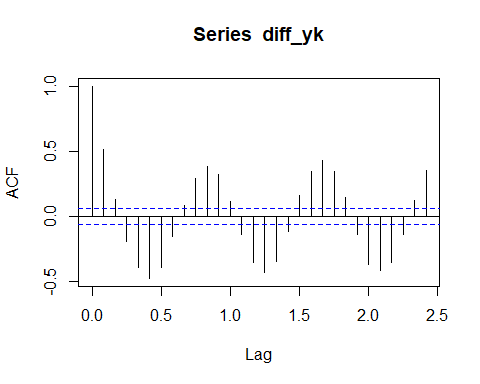
### Using SARIMA Models Now,

#Part 2  
  
acf(yk)



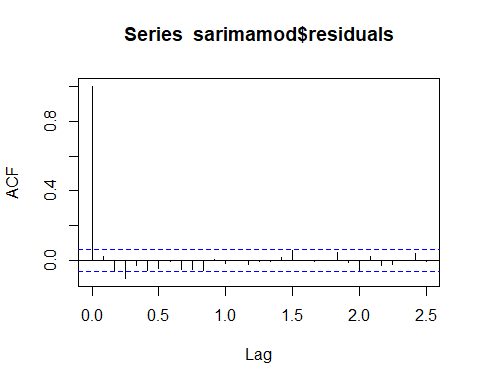
We see that the series clearly needs differencing. SO, after differencing once, we look at the acf of residuals and notice its periodicity of approximately 10.

diff\_yk <- diff(yk)  
  
acf(diff\_yk)

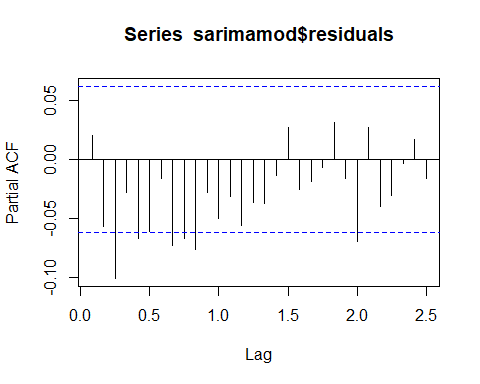


I then simulated various SARIMA models with periodicity and checked which one satisfies underfit and overfit arguements. I concluded that AR(2) with SARIMA(1,1) is the best model. The ACF and PACF plots show the same.

sarimamod=arima(yk,order=c(2,0,0),seasonal=list(period=10,order=c(1,0,1)))  
  
acf(sarimamod$residuals)



pacf(sarimamod$residuals)



## Maximum Likelihood Estimation

For a given sample set, suppose we have ranging from 1 to N, with , for some n being the maximum value of the set. So,

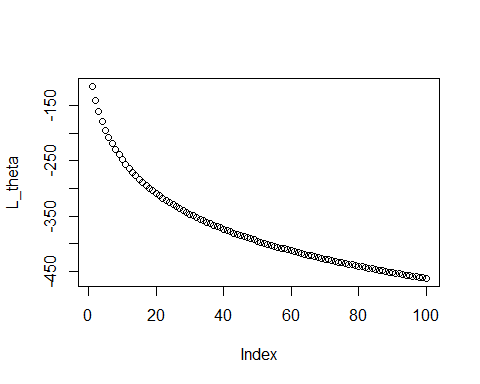
Assuming the events of picking a sample is independent.

NOw the log-likelihood function will take it’s maximum value at the minimum value of . should atleast be greater or equal to the maximum value of sample set .

load('mle\_unif.Rdata')  
  
max(xk)

## [1] 1.966975

theta = seq(max(xk),101, by=1)  
  
L\_theta={}  
  
for (i in 1:100) {  
   
 L\_theta[i] = -100\*(log(theta[i]))-(sum(xk)/theta[i])  
  
}  
plot(L\_theta)



One can deduce the same from the graph of likelihood vs .

Furthermore, we know that bias , with being the truth. We see that for any x in [0,] such that $x>= X\_n, $the probability of this sample space over N samples become

Hence,

Rather, if the distribution was uniform over we would have only got a limit tending to the maximum value for which is not a right estimate for our subsequent bias calculation since this limit could become far away from the real In order to solve this ambiguity, one can take large number of samples, then using the principles of consistency, one can say that the limit will tend to

## Fisher’s Information

For a given probability distribution function , we evaluate likelihood which can be considered equal to the the probability distribution function.

In order to estimate the most efficient estimator, the estimator should satisfy the following condition,

In the above equation $^\* $should only be a function of y, that is the data instants. With that in mind, let’s evaluate the two conditions given.

### is the parameter

This implies,

Clearly, the most efficient estimator of is a function of parameter itself, hence, it is not possible to determine the efficient estimator for Let’s try for a modified version of the same parameter.

### is the parameter

This happens because since the process is white noise, so that makes since y is plotted on an exponential distribution. Now,

As one can see, is purely a function of y, and hence it is the most efficient estimator for the given estimator.

## Variability of Sample Mean

For the vaiability of sample mean (), we have for stationary data that has N realisations,

The first term in the above expression can be easily seen as the , the second term where n is not equal to m can be shown as the auto-covariance function.

As, for a stationary process, so it will suffice us to only take one side of the double summation, when .

Now, let’s take as the lag for ACVF. For, l=1, we will see terms N-1 times, terms N-2 times and so on for , we will see it N-l times. We can rewrite the above expression as

Hence,

N=5000 #no. of samples  
  
#Generic method to calculate variance of sample mean estimator  
  
mean\_yk={}  
  
for (i in 1:N){  
   
 yk<-arima.sim(model=list(ma=0.4,order=c(0,0,1)),n=10000)  
 mean\_yk[i]=mean(yk)  
}  
  
hist(mean\_yk)  
  
var\_1={}  
for (i in 1:5000){  
 var\_1[i]=(mean(mean\_yk)-mean\_yk[i])^2  
}  
  
var\_1=sum(mean\_yk)/(N-1)  
  
#Calculation using the proved expression  
  
acvf\_data = acf(yk,lag.max = 10000, type='cov')  
  
acvf\_yk = acvf\_data$acf  
  
sum\_term=0  
  
for (i in 2:10000){  
   
 sum\_term=sum\_term+((1-(i/10000))\*acvf\_yk[i])  
}  
  
var\_2 = (1/10000)\*(acvf\_yk[1]+(2\*sum\_term))

The code written above simulates the given MA(1) model and I have evaluated variance with both the methods. Variance with the generic method comes out to be **4.149214e-05** and with our expression proved above, we get a variance of **1.728528e-05.** The values have a different scale altogether, but still are comparable to an extent that both are converging to 0 which is the general case for sample mean estimator. The variance would ideally converge to 0 as we increase the sample size to infinity.