Assignment 6: CH5350

Abhijeet Mavi

# MLE and Least Squares Estimate

We are given x[1] and x[2] here. So, the given equation is.

So, we have to evaluate and here. In order to do so, we will assume that the series is zero-mean stationary, we will get

Also,

Now for likelihood function, we will write the joint distribution of the following as follows:

We take the log-likelihood as follows,

Now, for maximum likelihoood of , we have and . Then , I solved the 2-equation, 2-variable system, we get:

For the Least-square estimates,

Hence, the least squares solution for this:

Hence, we get

Hence, there is no noise in variance, as .

As, the variance of LS estimate is much lower than the MLE estimator, I believe that **LS is a better estimator in this case.**

# Hannan-Rissanen Algorithm

arma\_hn=function(series,ar\_order,ma\_order){  
   
 len\_ser=length(series)  
   
 max\_order=5  
   
 ar5mod=arima(series,c(5,0,0))

I formulated an AR(5) model to first find the coefficients of e[k]. This formulated noise would then serve as regressors for moving average part of the ARMA models. Then, the z\_mat formulated in the code snippet below served as regressor to our overall problem statement, and we applied the least squares solution according to Projection theorem as follows.

where served as the estimated parameter set

res\_ek=ar5mod$residuals  
   
 res\_ek=res\_ek[which(is.na(res\_ek)==F)]  
   
 ser\_new=series[max(ar\_order,ma\_order):len\_ser]  
   
 z\_mat=matrix(NA,nrow=(len\_ser-(max(ma\_order,ar\_order))-1),ncol=(ar\_order+ma\_order))  
   
 for(i in 1:(ar\_order)){  
   
 z\_mat[,i]=series[(max(ar\_order,ma\_order)-i+2):(len\_ser-i)]  
   
 }  
   
   
   
   
 for(i in 1:(ma\_order)){  
   
 z\_mat[,(i+ar\_order)]=res\_ek[(max(ar\_order,ma\_order)-i+2):(len\_ser-i)]  
   
 }  
   
 par\_set=((qr.solve(t(z\_mat) %\*% z\_mat)) %\*% t(z\_mat)) %\*% series[(max(ar\_order,ma\_order)+2):len\_ser]  
   
 return(par\_set)  
}  
  
  
ak=arima.sim(n = 100000, list( ar=c(0.4),ma = c(0.7, 0.12)),sd = sqrt(1))  
ak1=arima.sim(n = 100000, list( ma = c(1, 0.21)),sd = sqrt(1))  
arma\_hn(ak1,0,2)  
  
arma\_hn(ak,1,2)

I then supplied MA(2) process with c1 = 1, c2 = 0.21 and ARMA(1,2) process with d1 = 0.4, c1 = 0.7, c2 = 0.12.

For MA(2), I got c1 as **0.99** and c2 as **0.20**. For ARMA(1,2), we got d1 as **0.41**, c1 as **0.69** and c2 as **0.11**. These values are quite close to the real values.

library(tseries)  
  
ak=arima.sim(n = 100000, list( ar=c(0.4),ma = c(0.7, 0.12)),sd = sqrt(1))  
ak1=arima.sim(n = 100000, list( ma = c(1, 0.21)),sd = sqrt(1))  
  
  
arma12mod=arma(ak,c(1,2))  
arma12mod$coef

## ar1 ma1 ma2 intercept   
## 0.38867869 0.71293418 0.13396257 0.00400538

ma2mod=arma(ak1,c(0,2))  
ma2mod$coef

## ma1 ma2 intercept   
## 0.99903690 0.20700842 -0.01039001

Again, the values are closer to the **arma()** routine as shown.

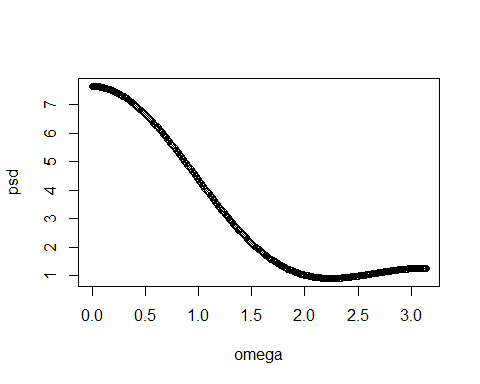
## Spectral Densities

We see that the given model is

We can find using the function

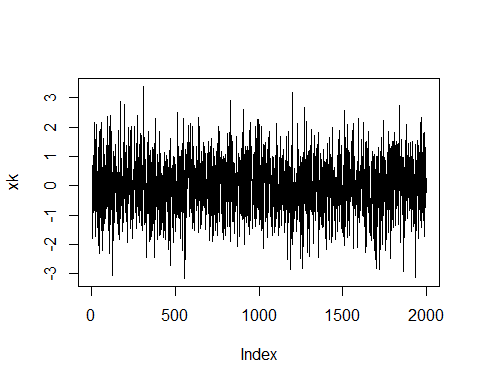
We find that,

psd={}  
omega=seq(0,3.14,by=0.01)  
for (i in 1:length(omega)){  
   
 psd[i]=(20+(20\*cos(omega[i]))+(8\*cos(2\*omega[i])))/6.28  
}   
plot(omega,psd)  
lines(omega,psd)



Now, we will generate 2000 samples of the said dataset as follows

ek=rnorm(10000)  
  
xk=ek[4001:6000]  
  
plot(xk, type='l')



We then apply periodogram to it as follows:

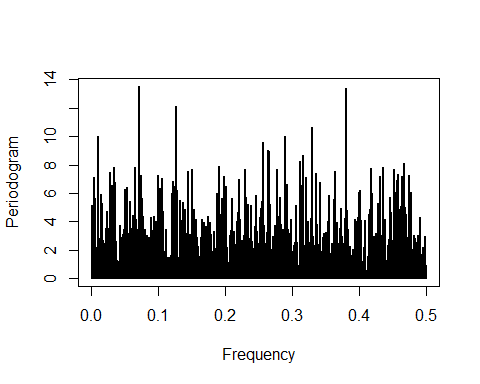
library(TSA)

##   
## Attaching package: 'TSA'

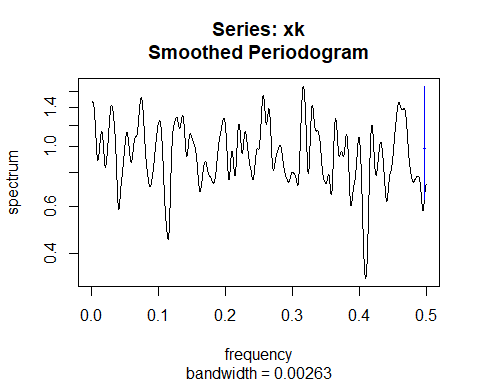
## The following objects are masked from 'package:stats':  
##   
## acf, arima

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

periodogram(xk)

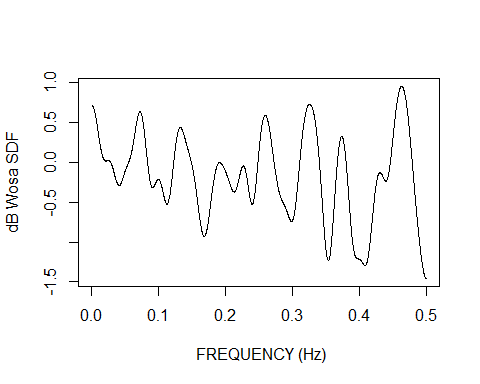


# For Daniell's method  
  
spec.pgram(xk , spans=c( 9 , 9 , 9 , 9 , 9 ))



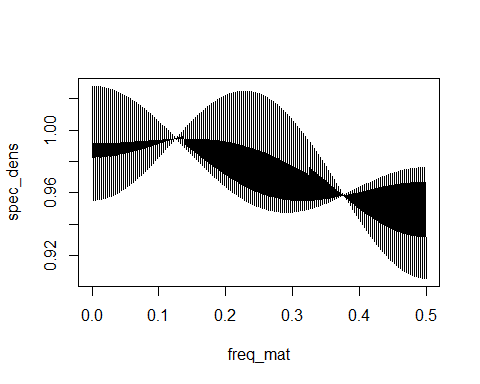
The Welch’s periodogram is plotted using **SDF()** function as follows:

library(sapa)  
plot(SDF(xk ,method="wosa" , blocksize= 70))



We will now ry to fit an MA(2) model for our process, and then try to calculate the PSD of it using parameters c1 and c2. We know that , and for an MA(2) process.

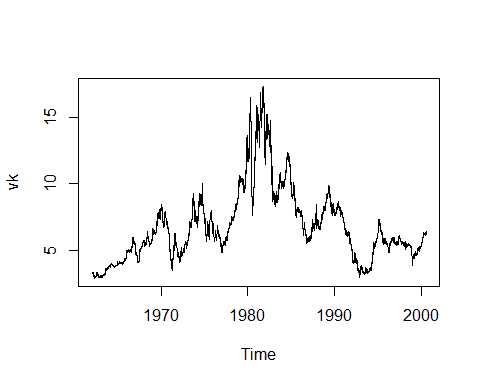
ma2mod=arima(xk,c(0,0,2))  
c1=ma2mod$coef[1]  
c2=ma2mod$coef  
  
err\_sig=ma2mod$sigma2  
  
freq\_mat=seq(0,0.5,by=0.001)  
  
spec\_dens= (err\_sig)\*((1+(c1^2)+(c2^2))+2\*(c1+(c1\*c2))\*cos(2\*pi\*freq\_mat)+2\*c2\*cos(2\*2\*pi\*freq\_mat))  
  
  
plot(freq\_mat,spec\_dens,type='l')



It is quite clear from the whole process that parametric method gives the best form of PSD, evn though the graph does not accurately describe ou process but it gives the smoothest approximation of all as compared to non-parametric ones. Note that, for the original process I have plotted and for subsequent processes, I plotted frequencies.

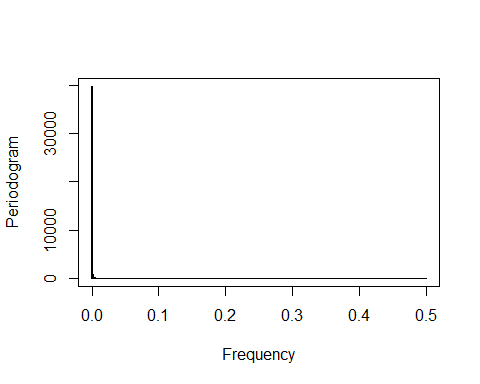
# Fitting a time-series model: tcm1yd

library(tseries)  
library(TSA)  
data(tcmd)  
vk=tcm1yd  
  
plot(vk)

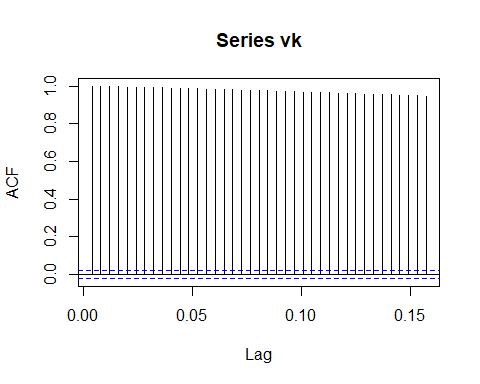


We first load the tcmd model and, and then take tcm1yd from it. We then plot it as shown above.

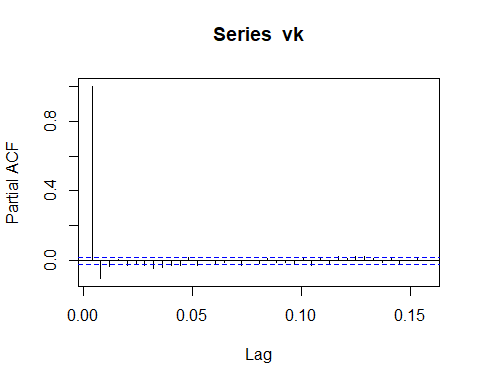
periodogram(vk)



acf(vk)



pacf(vk)



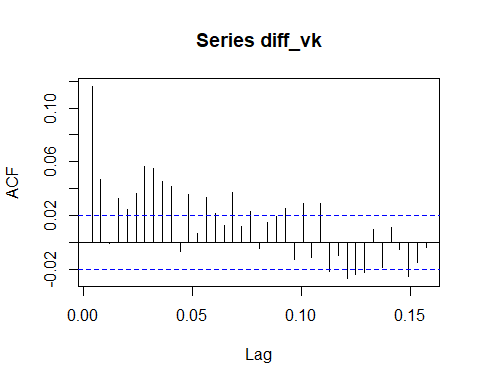
The periodogram, acf, and pacf clearly shows that there are trends in the series. Particularly differencing trends. This is because the decay in acf is too slow and the periodogram shows high power at lower frequency, a characteristic of differencing series.

diff\_vk= diff(vk)  
  
adf.test(diff\_vk)

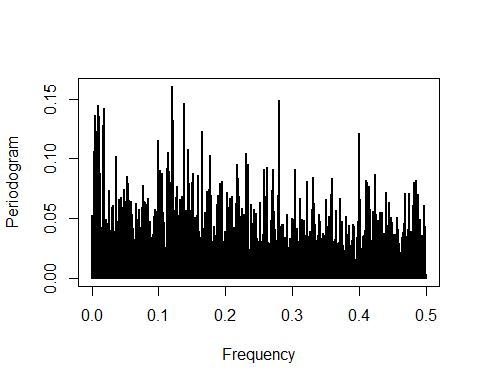
## Warning in adf.test(diff\_vk): p-value smaller than printed p-value

##   
## Augmented Dickey-Fuller Test  
##   
## data: diff\_vk  
## Dickey-Fuller = -17.642, Lag order = 21, p-value = 0.01  
## alternative hypothesis: stationary

acf(diff\_vk)



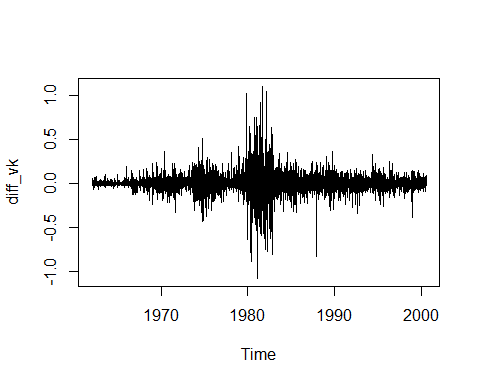
periodogram(diff\_vk)



I then differenced the series once, and then applied the Augmented Dickey-Fuller test using **adf.test()**. It clearly showed that null hypothesis can be rejected of the series requiring any more differencing.

The periodogram further reaffirmed my hypotheses as shown.

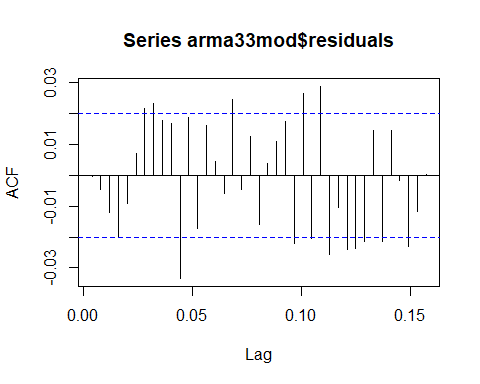
plot(diff\_vk)



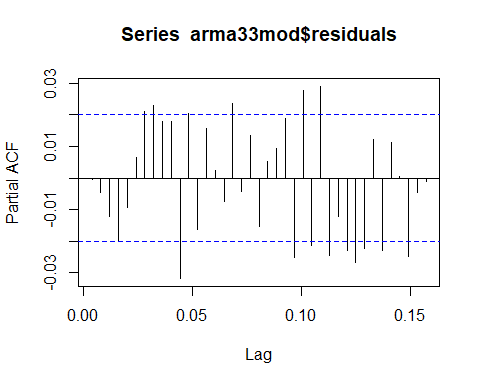
arma33mod=arima(diff\_vk, order=c(3,0,3))  
  
arma33mod

##   
## Call:  
## arima(x = diff\_vk, order = c(3, 0, 3))  
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 ma3 intercept  
## 0.2772 0.2861 0.2988 -0.1692 -0.2708 -0.3410 0.0003  
## s.e. 0.1891 0.1600 0.1007 0.1864 0.1437 0.0947 0.0015  
##   
## sigma^2 estimated as 0.00904: log likelihood = 8942.38, aic = -17870.76

acf(arma33mod$residuals)



pacf(arma33mod$residuals)



I then tried to fit various ARMA models to the series and was able to properly fit the ARMA(3,3) to the differenced series. Even though it showed a bit of underfitting discrepancies, it fit our model the best in all the possible models of 10th order.

Hence, our final series looks like as follows:

The estimated is **0.00904**.

## Predictions using Projection Theorem

For the given series, we have

For any process, the best predictor is the linear sum of its past realizations,

We see that and . Now, using Yule-Walker’s equations we construct the following matrix.

I solved the given matrix system to find that,

And hence,

Now, in order to calculate MSE for the above derived BLP estimate, we have

We will only see j for j=1, since all the other covariances are 0. Expanding, the expectations terms, we get

The third and fourth terms are happening hen l=0 and l=1.

Further, simplifying, we would get,

Hence, proved that MSE is .