Assignment\_6

Shritej Chavan (BE14B004)

November 29, 2018

### MLE and Least Squares

1. MLE method

The given two observations of the series are and . An AR(1) model as follows

The parameters to be estimated are and Assumption: zero-mean stationary process Therefore,

We know that the joint density function,

If all the density functions are conditional on another variable just as in our case we can write it as follows

Now calculating the variances and the expectation required for the above equation, we get

From ARMA model equation, we get

To be obvious we know that joint distribution is gaussian, Therefore,

Taking the log-likelihood, we get

Since we have two parameters to estimate, therefore calculating the maximum likelihood by below solving equations,

and $ = 0 $

We get,

1. Least Squares

The estimate of through least square solution is

Hence, we get

Hence, .

For the given process comparing with MLE, we can say that Least squares is the better estimate.

### Hannan-Rissanen Algorithm

For the Hannan-Rissanen algorithm, which is similar to the Durbin estimator where parameters are estimated linear least-squares regression of on the estimated past innovations.

Where the past innovations are obtained as a residual after fitting a high order AR model

hr\_fn = function(vk, ma, ar, max\_ar){  
   
 #Length of the series  
 N = length(vk)  
   
 #Fitting a high order AR model  
 arnmod = arima(vk, order = c(max\_ar,0,0))  
  
 #Extracting the residuals   
 ek = arnmod$residuals  
   
 #Omitting NA values   
 ek = ek[which(is.na(ek)==FALSE)]  
   
 ##Initializing z matrix   
 z = matrix(NA,nrow=(N-(max(ma,ar))-1),ncol=(ar+ma))  
   
 for (i in 1:ar){  
 z[,i] = vk[(max(ar,ma)-i+2):(N - i)]  
   
 }  
   
 for(j in 1:ma){  
 z[,j+ar] = ek[(max(ar,ma)-j+2) : (N-j)]  
   
 }  
  
#paramter set calculated using projection theorem  
   
 theta\_vec=((qr.solve(t(z) %\*% z)) %\*% t(z)) %\*% vk[(max(ar,ma)+2):N]  
   
 return(theta\_vec)  
   
}  
  
#b)  
  
#Simulating the MA series   
  
ma2 = arima.sim(n = 100000, list( ma = c(1, 0.21)),sd = 1)  
  
#Simulating the ARMA series   
  
arma12 = arima.sim(n=100000, list( ar=c(0.4),ma = c(0.7, 0.12)), sd = 1)  
  
  
mat\_ma2 = matrix(data = NA, nrow = 10, ncol = 2)  
  
  
  
for (i in 1:10){  
   
 mat\_ma2[i,] = t(hr\_fn(ma2,2,0,i))  
}  
  
mat\_ma2 = cbind(1:10, mat\_ma2)  
rownames(mat\_ma2) = 1:10  
colnames(mat\_ma2) = c("Initial AR order", "c\_1", "c\_2")  
  
mat\_ma2

## Initial AR order c\_1 c\_2  
## 1 1 0.8248252 -0.02964988  
## 2 2 0.9050023 0.09643681  
## 3 3 0.9508055 0.15421753  
## 4 4 0.9747864 0.18203492  
## 5 5 0.9863281 0.19562354  
## 6 6 0.9918193 0.20189814  
## 7 7 0.9946074 0.20470759  
## 8 8 0.9960276 0.20635959  
## 9 9 0.9966657 0.20707908  
## 10 10 0.9969806 0.20730653

#hr\_fn(ma2,2,0,5)   
mat\_arma12 = matrix(data = NA, nrow = 10, ncol = 3)  
  
#hr\_fn(arma12,2,1,5)   
  
for (j in 1:10){  
   
 mat\_arma12[j,] = t(hr\_fn(arma12,2,1,j))  
}  
  
mat\_arma12 = cbind(1:10, mat\_arma12)  
rownames(mat\_arma12) = 1:10  
colnames(mat\_arma12) = c("Initial AR order","d\_1", "c\_1", "c\_2")  
  
mat\_arma12

## Initial AR order d\_1 c\_1 c\_2  
## 1 1 0.6456768 0.4347106 -0.26496786  
## 2 2 0.5183181 0.5656281 -0.04792544  
## 3 3 0.4439056 0.6522774 0.06656809  
## 4 4 0.4186996 0.6819118 0.10404178  
## 5 5 0.4133739 0.6881393 0.11218118  
## 6 6 0.4123385 0.6893108 0.11369907  
## 7 7 0.4120878 0.6895794 0.11407004  
## 8 8 0.4120985 0.6895589 0.11406706  
## 9 9 0.4122022 0.6894602 0.11394749  
## 10 10 0.4121989 0.6894638 0.11395264

We can see that the estimates of the parameters get better with increasing order of the initial AR model, but reach a certain saturation point.

Also, with number data points required for simulating the process the parameters we are estimating gets better.

1. Now comparing the same with arma routine of the tseries package

library(tseries)  
arma\_pro = arima.sim(n = 100000, list( ar=c(0.4),ma = c(0.7, 0.12)),sd = 1)  
  
ma\_pro = arima.sim(n = 100000, list( ma = c(1, 0.21)),sd = 1)  
  
armamod = arma(arma\_pro, c(1,2))  
armamod$coef

## ar1 ma1 ma2 intercept   
## 0.400518869 0.707611984 0.127165650 0.004394084

mamod = arma(ma\_pro, c(0,2))  
mamod$coef

## ma1 ma2 intercept   
## 0.995807157 0.205704320 -0.004444554

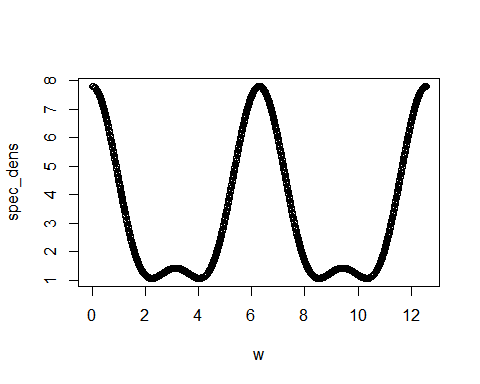
### Spectral Densities

1. Theoretical spectral density

The given series formed by

Therefore,

spec\_dens = {}  
w = seq(0,4\*pi,0.01 )  
for (i in 1:length(w)){  
 spec\_dens[i]=(21+(20\*cos(w[i]))+(8\*cos(2\*w[i])))/(2\*pi)  
}  
plot(w,spec\_dens)  
lines(w,spec\_dens)



1. Generating x[k]

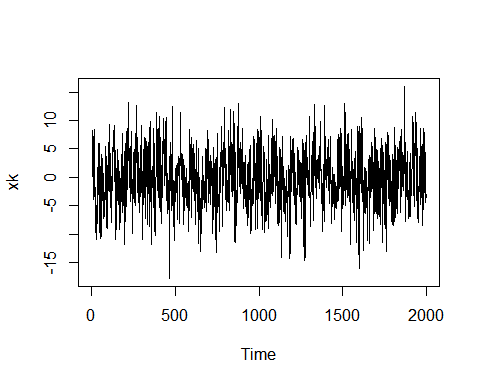
library(TSA)

##   
## Attaching package: 'TSA'

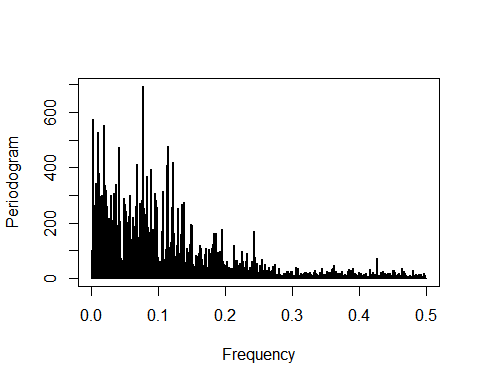
## The following objects are masked from 'package:stats':  
##   
## acf, arima

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

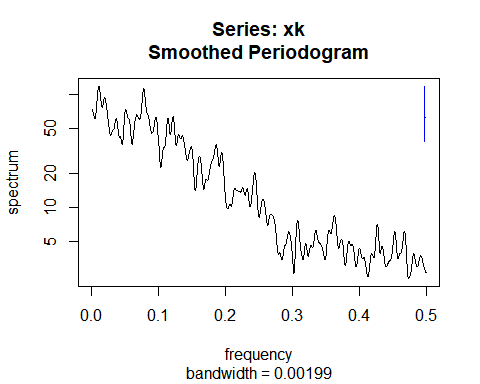
xk = arima.sim(model = list(ma = c(4,2,1)), 2000)  
plot(xk, type = 'l')



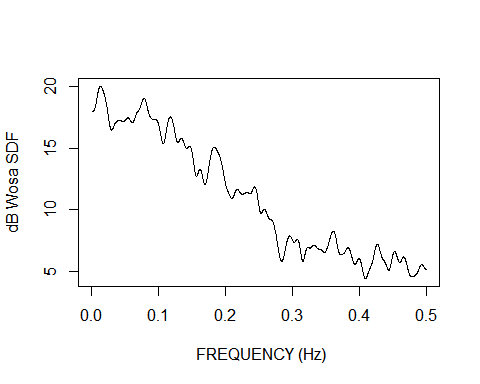
# Raw Periodogram  
periodogram(xk)



#Daniell's Smoother   
spec.pgram(xk, spans = rep(7,5))



#Welch averaged periodogram   
  
library(sapa)  
welch = SDF(xk ,method="wosa" , blocksize= 150)  
plot(welch)



1. Parametric method

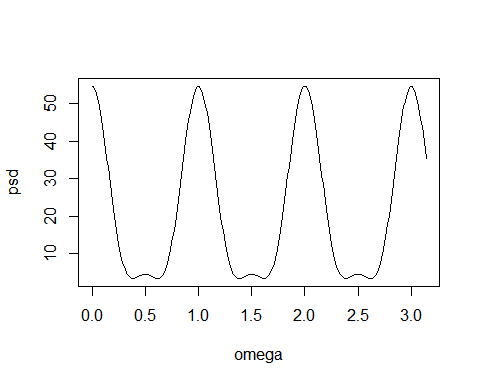
Estimating the parameters and by fitting the MA(2) model.

ma2mod = arima(xk, order = c(0,0,2))  
c1 = ma2mod$coef[1]  
c2 = ma2mod$coef[2]

Autocovariance functions for MA(2) models are as follows :

Now doing fourier transform on the autocovariance functions to obtain Spectral Density we get

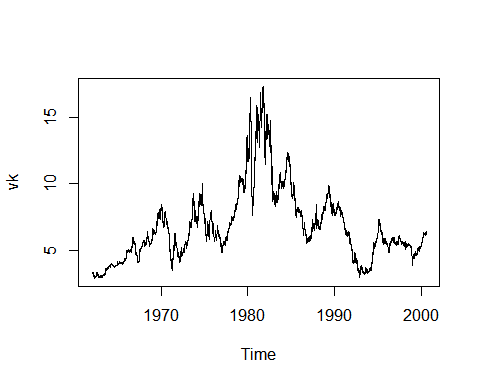
sig\_e = ma2mod$sigma2   
  
omega = seq(0,pi,by=0.01)   
  
psd= (sig\_e)\*((1+(c1^2)+(c2^2))+2\*(c1+(c1\*c2))\*cos(2\*pi\*omega)+2\*c2\*cos(2\*2\*pi\*omega))   
  
  
plot(omega,psd,type='l')



We can clearly see that the parametric method gives a better estimate of the Power spectral density than the other methods

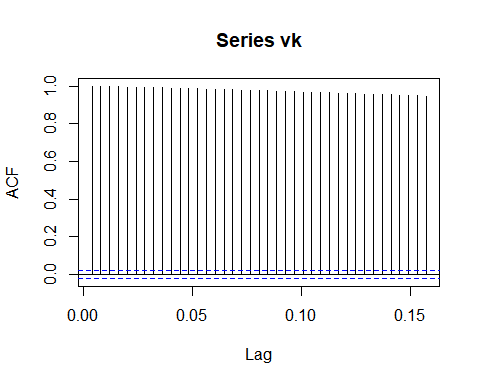
### Time Series model

library(tseries)   
library(TSA)   
data(tcmd)   
  
vk=tcm1yd   
# The given data set  
  
plot(vk)

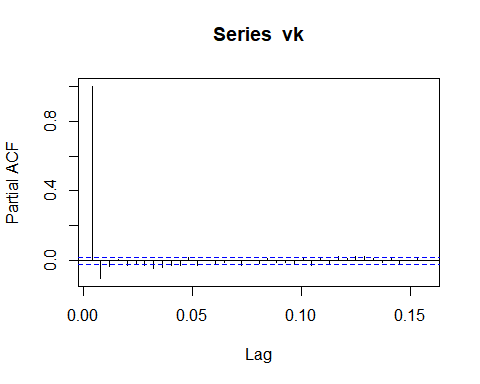


Now studying the ACF, PACF, we get

acf(vk)



pacf(vk)



We can see that the acf of the series decays slowly which one of the characteristics of the integrating effects in the given series. Therefore differencing the series once we get

diff\_vk = diff(vk)

Now using Augmented Dick Fuller test to check the need for differencing, where null hypothesis states that differencing is required to obtain a stationary series.

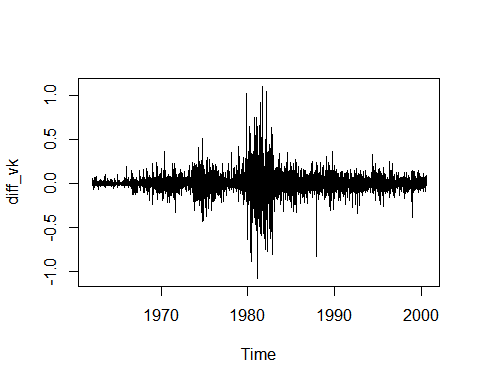
adf.test(diff\_vk)

## 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

Therefore null hypothesis is rejected

plot(diff\_vk)

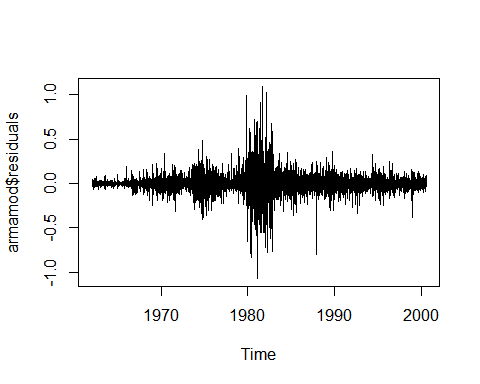


Now fitting the ARMA using the auto.arima function in the forecast package which searches through all the possible arma models and chooses the one with minimum aic value

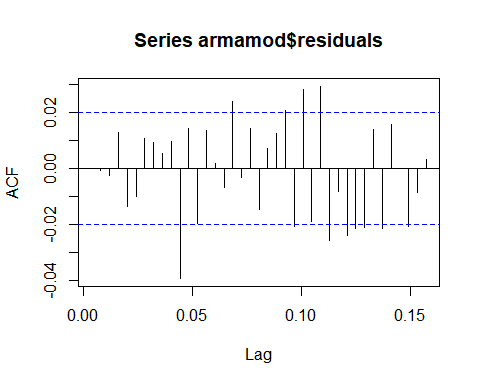
library(forecast)  
  
armamod <- auto.arima(diff\_vk,seasonal = FALSE,d=0,D=0,max.p=5,max.q=5,start.p = 1,start.q = 1)  
  
summary(armamod)

## Series: diff\_vk   
## ARIMA(4,0,2) with zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ma1 ma2  
## 1.4950 -0.5920 -0.0062 0.0494 -1.3894 0.4740  
## s.e. 0.1398 0.1414 0.0228 0.0115 0.1397 0.1255  
##   
## sigma^2 estimated as 0.009033: log likelihood=8948.71  
## AIC=-17883.43 AICc=-17883.41 BIC=-17833.26  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.0002196243 0.09501419 0.05393966 NaN Inf 0.6605652  
## ACF1  
## Training set 0.0002009539

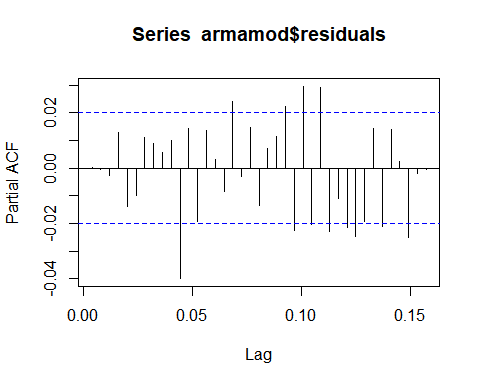
plot(armamod$residuals)



acf(armamod$residuals)



pacf(armamod$residuals)



ARMA model chosen by the function is ARMA(4,2) which is proper fit to the differenced series.

### Predictions

For the given MA(1) process,

For any process, the best prediction is the conditional expectation

We already proved that the conditional expectation of the RVs having joint distribution is the linear function of the independent RV.

Therefore in our case

Now,

Calculating the ACVF for the given MA (1) process, we get

Similarly we can calculate,

Also, $ = 0 l > 1 $

From Y-W’s equations we get the following equations,

Where,

and

the matrix of the coefficients of the process

$$d\_P = (

)

$$ Therefore, solving for we get

Therefore we get,

Calculating the mean square error in our above prediction of

By definition,

Sustituting the value of , we get

Expectation of second term in the above equation will only be valid for lag =1

Substituting in the first term and in the second term, we get