Term\_Project - CH5350

Shritej Chavan (BE14B004)

December 3, 2018

### Threshold Auto-Regressive Model

1. A two regime TAR (2, d=1) model, has the form:

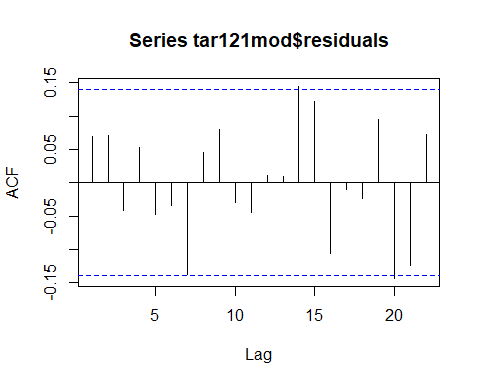
library(TSA)

##   
## Attaching package: 'TSA'

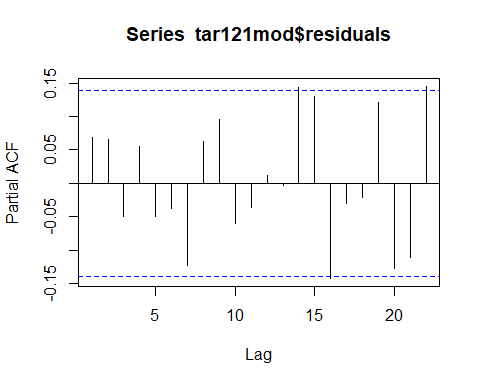
## The following objects are masked from 'package:stats':  
##   
## acf, arima

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

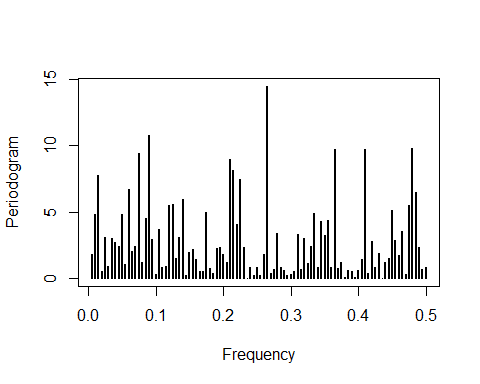
load("projq1a.Rdata")  
  
tar121mod = tar(vk,1,2,1)  
  
acf(tar121mod$residuals)



pacf(tar121mod$residuals)



periodogram(tar121mod$residuals)



TAR(1,2,1) model satisfies all residuals tests.

But let’s check model with lower number of parameters to satisfy parsimony conditions

tarmod = tar(vk,2,0,2)  
  
#Coefficients for the first regime of the fitted TAR(2,0,2) model  
tarmod$qr1$coefficients

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

#Sigma of the residuals of the regimes   
var(tarmod$qr1$residuals) # Regime 1

## [1] 1.53172

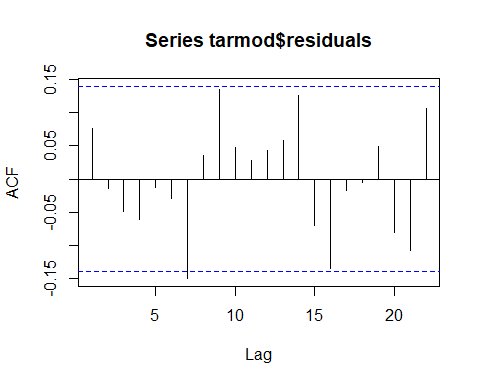
var(tarmod$qr2$residuals) # Regime 2

## [1] 0.9133471

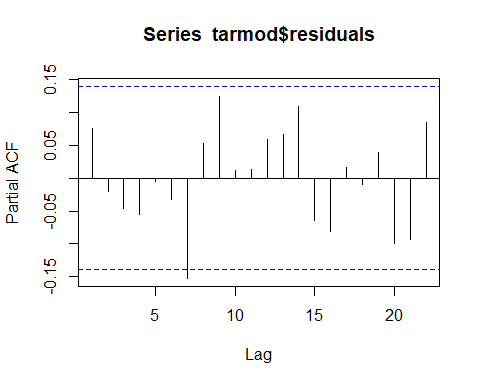
#Threshold   
  
tarmod$thd

##   
## 0.877825

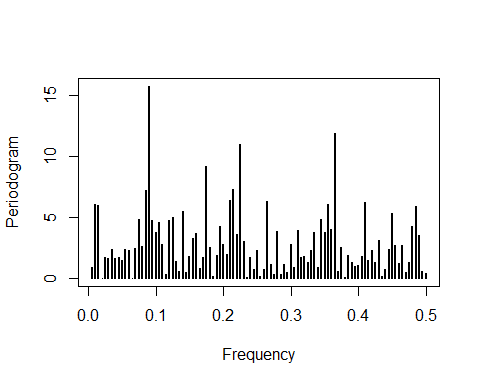
#Checking the residuals of the TAR model  
  
  
acf(tarmod$residuals)



pacf(tarmod$residuals)



periodogram(tarmod$residuals)



After viewing the ACF, PACF and periodogram of the residuals we can say that it passes the residual tests and the TAR(2,0,2) model perfectly fits on the given data set.

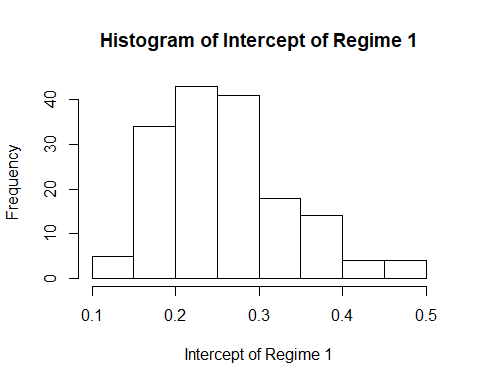
And the basis of parsimony TAR(2,0,2) model is chosen as the perfect fit for the given series.

The final two regime model can written as follows

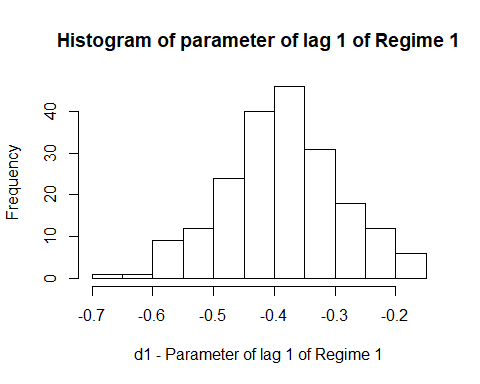
#Residual from the best model   
epsk = tarmod$residuals  
  
mod\_coeff\_reg1 = matrix(data = NA, nrow = 200, ncol = 3)  
  
mod\_coeff\_reg2 = {}  
  
#Number of realizations   
R = 1:200  
  
for (i in R){  
   
 #Resampling the residuals with replacement  
 epskr1 <- sample(epsk, size= length(epsk), replace=T)  
   
 # Generating new artificial realization of the series   
 vk\_new = tar.sim(tarmod, ntransient = 0, n = 198, epskr1)  
   
 #Restimating model parameters using same orders and delays   
 vmod = tar(vk\_new$y[1:196],2,0,2)  
   
 #Saving the model parameters  
 mod\_coeff\_reg1[i,1] = vmod$qr1$coefficients[1]  
 mod\_coeff\_reg1[i,2] = vmod$qr1$coefficients[2]  
 mod\_coeff\_reg1[i,3] = vmod$qr1$coefficients[3]  
   
 mod\_coeff\_reg2[i] = vmod$qr2$coefficients  
   
   
   
}  
  
#Omitting the NA values from the matrix of coefficients  
#Index of the number  
coef\_q1\_3 = which(is.na(mod\_coeff\_reg1[,3]) == F)  
coef\_q1\_2 = which(is.na(mod\_coeff\_reg1[,2]) == F)  
coef\_q1\_1 = which(is.na(mod\_coeff\_reg1[,1]) == F)

Now plotting the distribution (Histogram) of all the coefficients of the two regimes.

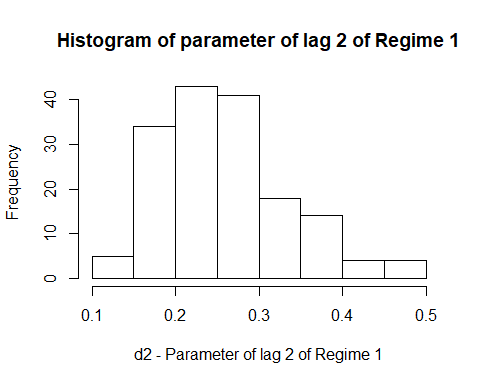
hist(mod\_coeff\_reg1[coef\_q1\_1,3], xlab = "Intercept of Regime 1", main = "Histogram of Intercept of Regime 1")



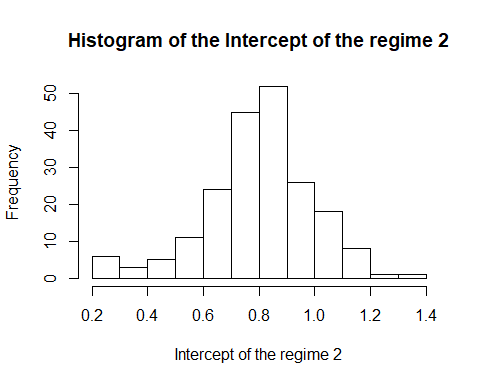
hist(mod\_coeff\_reg1[coef\_q1\_2,2], xlab = "d1 - Parameter of lag 1 of Regime 1", main = "Histogram of parameter of lag 1 of Regime 1")



hist(mod\_coeff\_reg1[coef\_q1\_3, 3], xlab = "d2 - Parameter of lag 2 of Regime 1", main = "Histogram of parameter of lag 2 of Regime 1")



hist(mod\_coeff\_reg2, xlab = "Intercept of the regime 2", main = "Histogram of the Intercept of the regime 2")



Now calculating mean and variances of the distribution of the respective coefficients

#Mean of all the parameters in Regime 1  
mean\_reg1 = colMeans(mod\_coeff\_reg1,na.rm = TRUE)  
  
#Mean of the parameters in Regime 2   
mean\_reg2 = mean(mod\_coeff\_reg2)  
  
#Standard error of the parameters in Regime 2  
std\_reg2 = sqrt(var(mod\_coeff\_reg2))  
  
#Standard error of the parameters in Regime 1  
std\_reg1 = matrix(data = c(sqrt(var(mod\_coeff\_reg1[coef\_q1\_1,1])), sqrt(var(mod\_coeff\_reg1[coef\_q1\_2,2])), sqrt(var(mod\_coeff\_reg1[coef\_q1\_3,3]))), nrow = 1, ncol = 3, byrow = TRUE )

means = c(mean\_reg1, mean\_reg2)  
stds = c(std\_reg1, std\_reg2)  
  
results = as.matrix(cbind(means, stds))  
  
rownames(results)=c('Regime1\_d0','Regime1\_d1','Regime1\_d2','Regime2\_d0')  
colnames(results)=c('Mean','Standard error')  
  
results

## Mean Standard error  
## Regime1\_d0 -0.7731765 0.17388259  
## Regime1\_d1 -0.3862817 0.09382737  
## Regime1\_d2 0.2552201 0.07498505  
## Regime2\_d0 0.8005845 0.19681567

The value of the parameters calculated from the distribution lie in the 99% confidence interval (as shown by the mean and ) of the true values which were obtained from the tar routine.

## (S)ARIMA Model

library(readr)

##   
## Attaching package: 'readr'

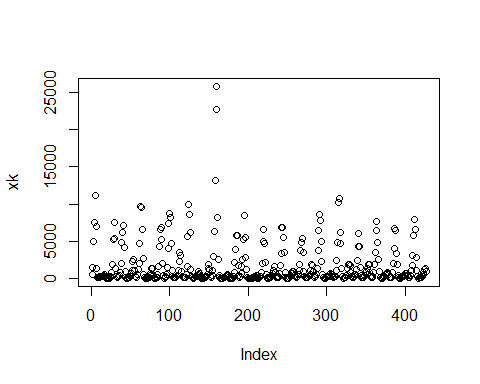
## The following object is masked from 'package:TSA':  
##   
## spec

library(TSA)  
  
monthly\_reported\_number\_of\_cases\_1\_ <- 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'

xk\_true = monthly\_reported\_number\_of\_cases\_1\_$`Monthly reported number of cases of measles, New York city, 1928-1972`  
  
# Distribution of dataset, 80% - Training and 20% - Test data  
  
xk = as.numeric(xk\_true[1:427])  
  
plot(xk)

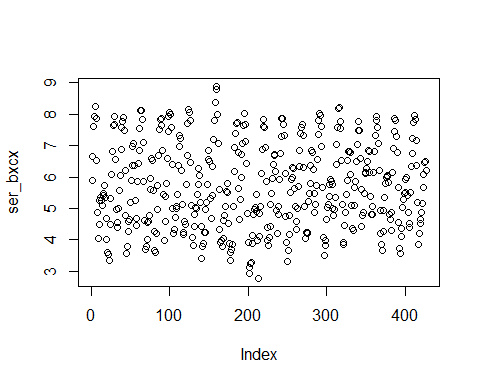


From the above plot of the series we can see that it is heteroskedastic i.e the variance is not constant throughout the series. Therefore to deal with this we will be using Box-Cox Transformation.

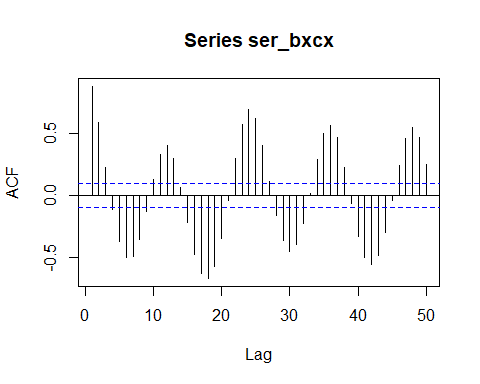
library(forecast)  
  
lambda\_bxcx = BoxCox.lambda(xk, method = 'guerrero', lower=-5,upper=5)  
lambda\_bxcx

## [1] -0.02713624

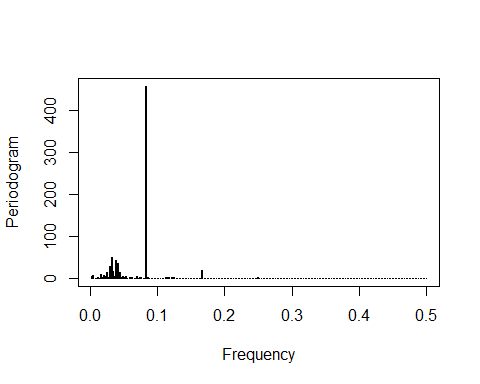
ser\_bxcx = BoxCox(xk,lambda\_bxcx)  
  
plot(ser\_bxcx)



acf(ser\_bxcx, lag.max = 50)



periodogram(ser\_bxcx)



Observing the periodogram we can say that the since there is no concentration of the psd at low frequency, there is no need for differencing.

Also, performing ADF (Augmented Dickey Fuller) test on the given series as follows

library(tseries)  
  
adf.test(ser\_bxcx)

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

##   
## Augmented Dickey-Fuller Test  
##   
## data: ser\_bxcx  
## Dickey-Fuller = -5.9884, Lag order = 7, p-value = 0.01  
## alternative hypothesis: stationary

Null hypothesis is rejected , hence there isn’t any need for differencing

Now observing the ACF plot of the series we can say that the series is periodic with period of 12.

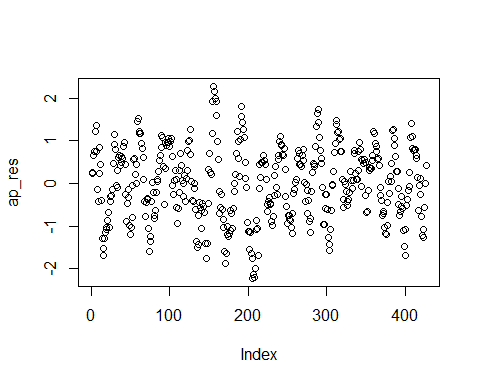
So there is a seasonal component but ‘stl’ function fails to detect seasonality.

Therefore we use ‘lm’ method to fit the seasonal component through sinusiodal functions.

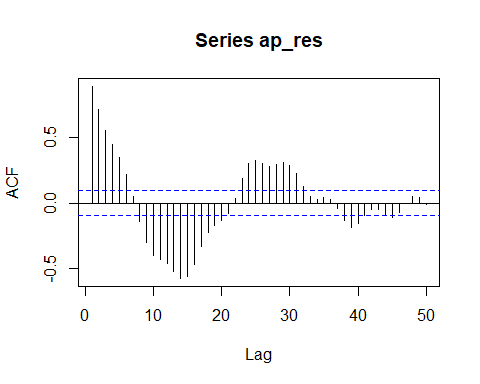
lin = 1:length(ser\_bxcx)  
  
periodic = lm(ser\_bxcx ~sin(2\*pi\*lin/12)+I(cos(2\*pi\*lin/12)))  
  
summary(periodic)

##   
## Call:  
## lm(formula = ser\_bxcx ~ sin(2 \* pi \* lin/12) + I(cos(2 \* pi \*   
## lin/12)))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.2268 -0.6148 0.0344 0.6538 2.2810   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.70829 0.04123 138.44 <2e-16 \*\*\*  
## sin(2 \* pi \* lin/12) 1.23435 0.05834 21.16 <2e-16 \*\*\*  
## I(cos(2 \* pi \* lin/12)) -0.79405 0.05827 -13.63 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8519 on 424 degrees of freedom  
## Multiple R-squared: 0.5985, Adjusted R-squared: 0.5966   
## F-statistic: 316 on 2 and 424 DF, p-value: < 2.2e-16

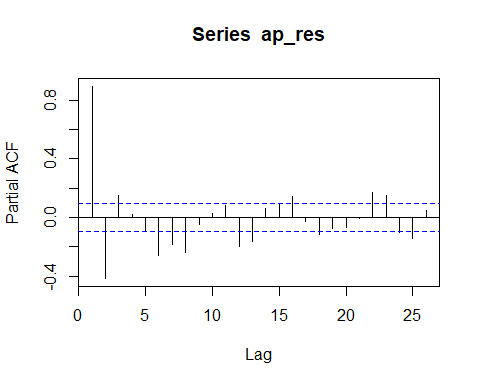
ap\_res = periodic$residuals  
  
plot(ap\_res)



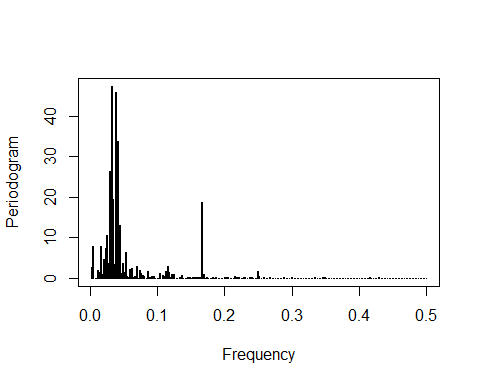
acf(ap\_res, lag.max = 50)



pacf(ap\_res)



periodogram(ap\_res)



Periodogram of the residual series clearly shows that there is no integrating effects present in the series but seasonal component is, as seen from the ACF plot.

Therefore, fitting multiplicative SARIMA model by trial and error by restricting the differencing term to be zero on both list

sarima = arima(ap\_res, order = c(0,0,1),c(1,0,2))  
  
#tsdiag(sarima)  
  
sarima

##   
## Call:  
## arima(x = ap\_res, order = c(0, 0, 1), seasonal = c(1, 0, 2))  
##   
## Coefficients:  
## ma1 sar1 sma1 sma2 intercept  
## 0.4312 0.7332 0.1740 0.1405 0.0076  
## s.e. 0.2366 0.0707 0.1876 0.1234 0.1157  
##   
## sigma^2 estimated as 0.1167: log likelihood = -148.21, aic = 306.43

We can clearly see that the model clearly overfits the series.

Therefore we will fit the constrained SARIMA model.

sarima2=arima(ap\_res,c(02,0,02),seasonal=list(order=c(0,0,2)) ,fixed = c(0,NA,NA,NA,NA,0,NA),transform.pars = F)  
  
sarima2

##   
## Call:  
## arima(x = ap\_res, 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 0.5427 0.4914 0.2422 0.8514 0 0.0074  
## s.e. 0 0.0692 0.0540 0.0739 0.0346 0 0.1145  
##   
## sigma^2 estimated as 0.1154: log likelihood = -145.98, aic = 301.97

Box.test(sarima2$residuals)

##   
## Box-Pierce test  
##   
## data: sarima2$residuals  
## X-squared = 0.00056788, df = 1, p-value = 0.981

The above series is simulated for the same number of points as of the original series (534) and since the 20% of the dataset is given to validation set, we will calculate the RMSE of the remaining dataset i.e. the last 107 data points.

library(CombMSC)

##   
## Attaching package: 'CombMSC'

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

library(MLmetrics)

##   
## Attaching package: 'MLmetrics'

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

#Fitting the multiplicative sesonal 'SARIMA' model   
  
mod1=sarima.Sim(n=534,period=12, model=list(ar=c(0,0.5427),ma=c(0.4914,0.2422)),seasonal=list(ma=c(0.8514,0)))   
  
tvec2=1:534  
  
# Additive Seasonal component  
mod2=(1.23435\*sin(2\*pi\*tvec2/12))-(0.79405\*cos(2\*pi\*tvec2/12))+ 5.70829  
  
#Lambda from the Box-Cox transformation  
  
mod =(mod1 + mod2)^(-.027)  
  
#Validation set from the predicted   
valid\_mod = mod[428:534]  
  
#Validation set from the true   
valid\_x = as.numeric(xk\_true[428:534])  
  
  
rmse = RMSE(valid\_mod, valid\_x)  
  
rmse

## [1] 770.3787

The above is the value of RMSE on the validation set.