Assignment5

Chanukya

11/17/2019

```
library(fpp2)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: forecast
```

```
## Warning: package 'forecast' was built under R version 3.5.2
```

```
## Loading required package: fma
```

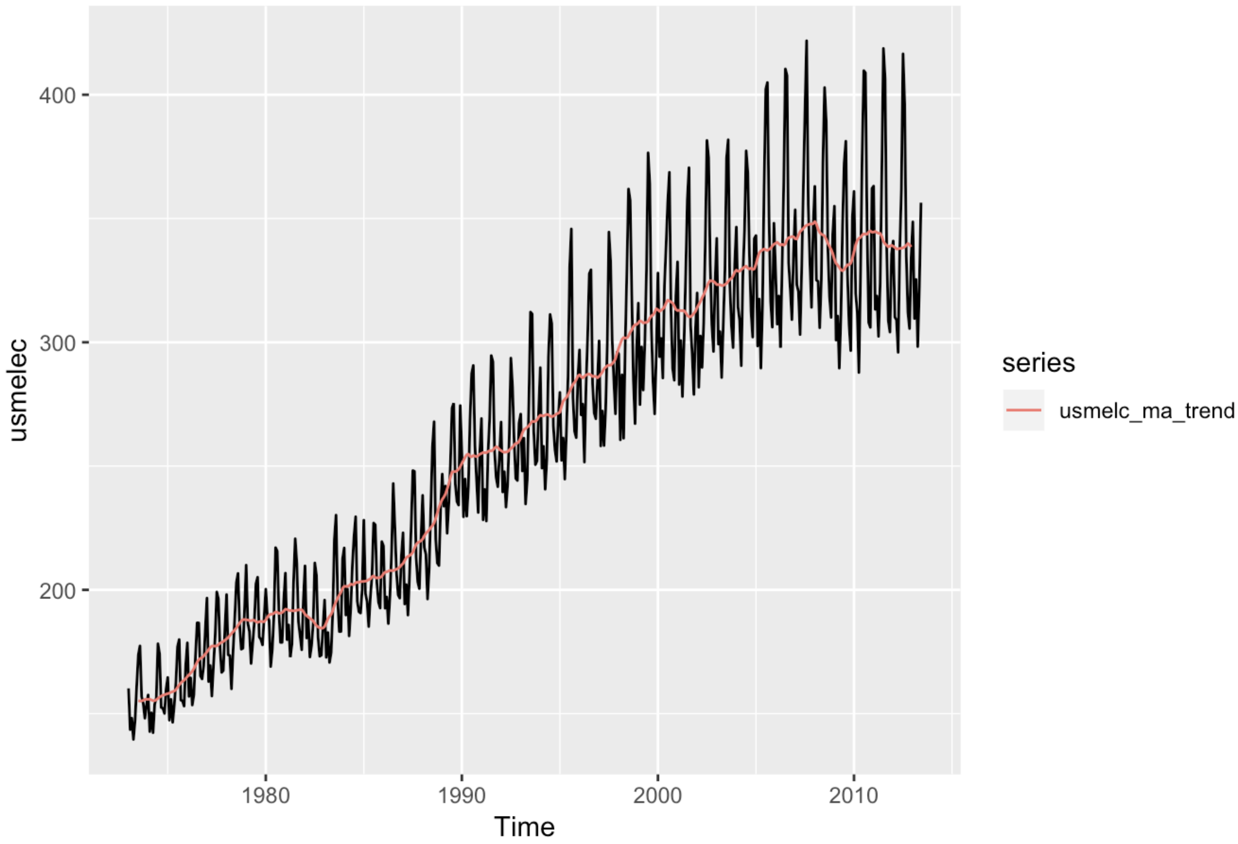
```
## Loading required package: expsmooth
```

```
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 3.5.2
```

```
#1
#a)
usmelc_ma_trend <- ma(usmelec, order =12)
autoplot(usmelec)+autolayer(usmelc_ma_trend)
```

```
## Warning: Removed 12 rows containing missing values (geom_path).
```



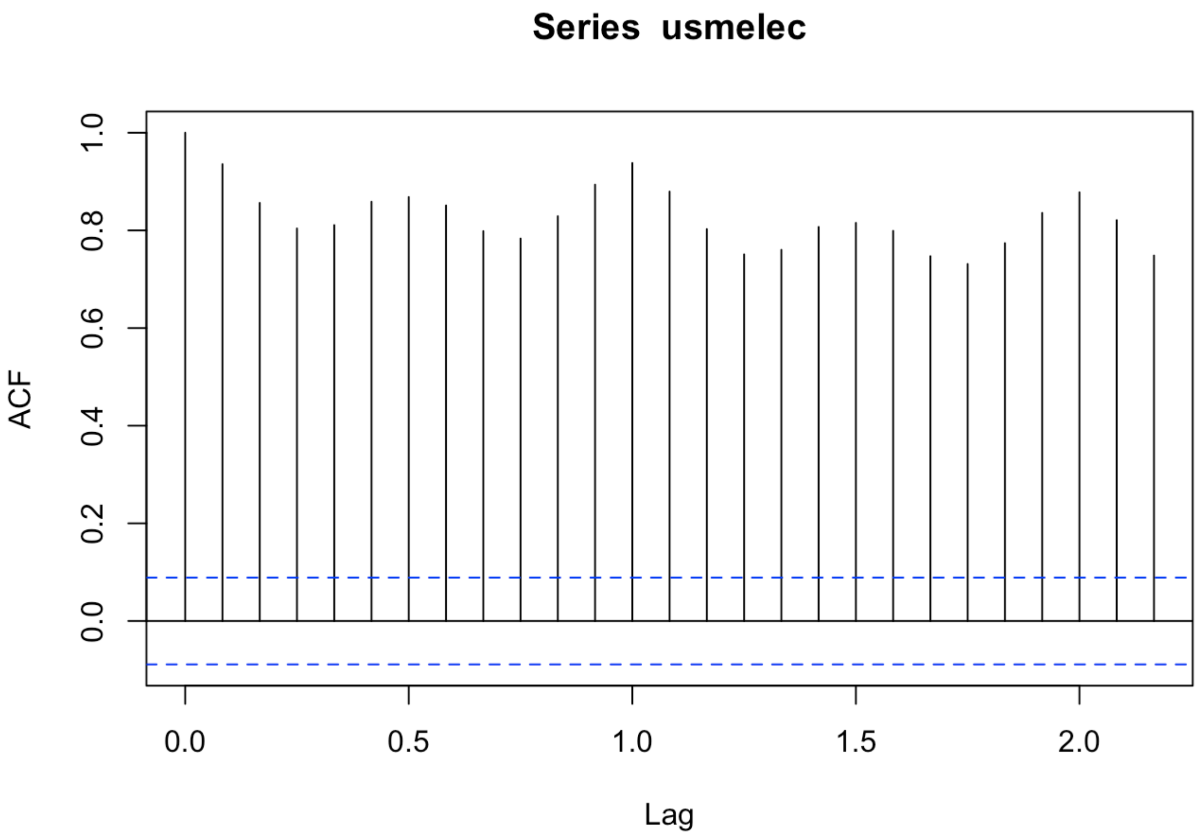

```
# Initally it is increasing, but in the end we can see it stopped increasing.

#b)
lambda <- BoxCox.lambda(usmelec)
lambda
```

```
## [1] -0.5738331
```

```
# Yes data needs transformation.

#c)
#from the graph we saw that data is not stationary
#the acf drops to zero for stationary time series
acf(usmelec)
```




```
#from the graph we can say the data is not stationary
ndiffs(usmelec)
```

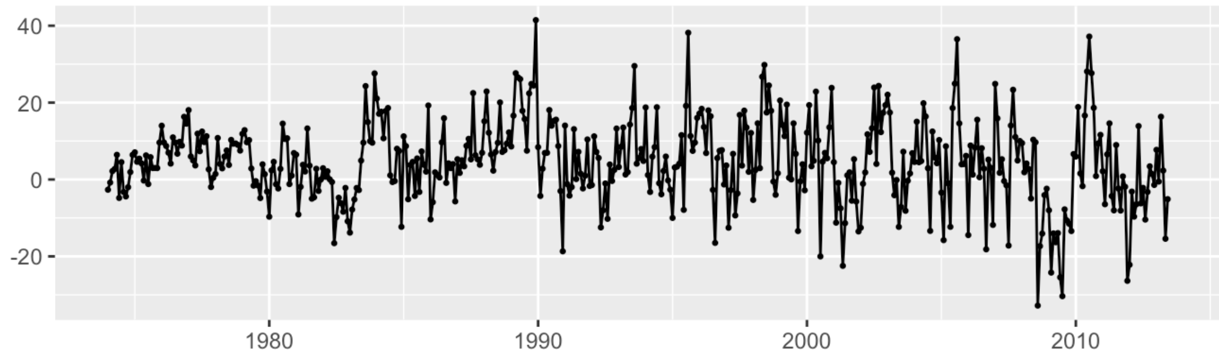
```
## [1] 1
```

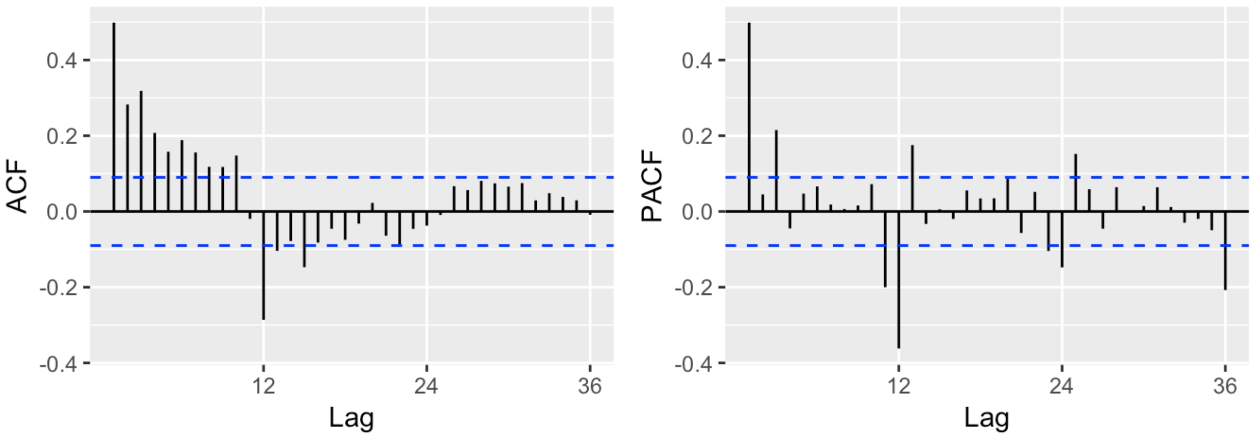
```
nsdiffs(usmelec)
```

```
## [1] 1
```

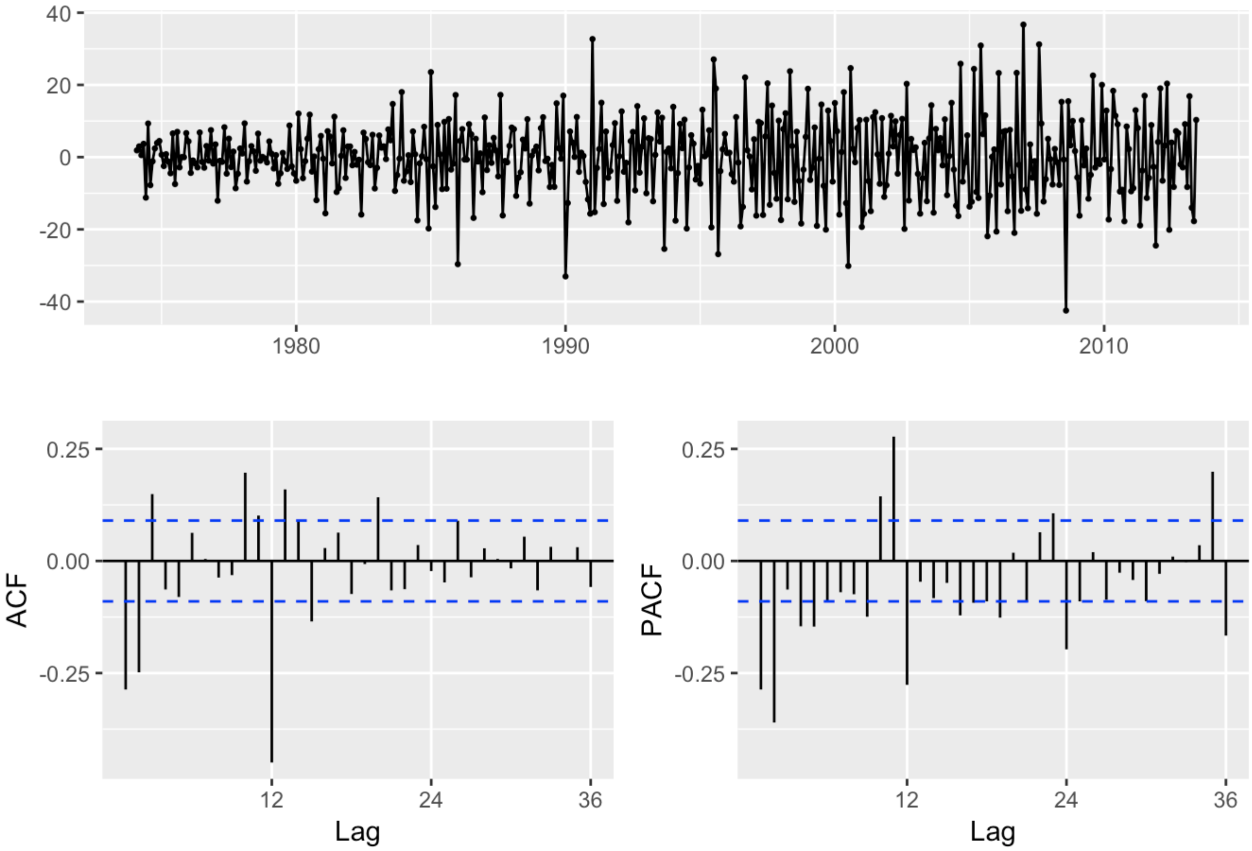
```
#require one seasonal differencing to make data stationary

#d)
#we see positive auto correlation for all the data
usmelec %>% diff(lag=12) %>% ggtsdisplay()
```



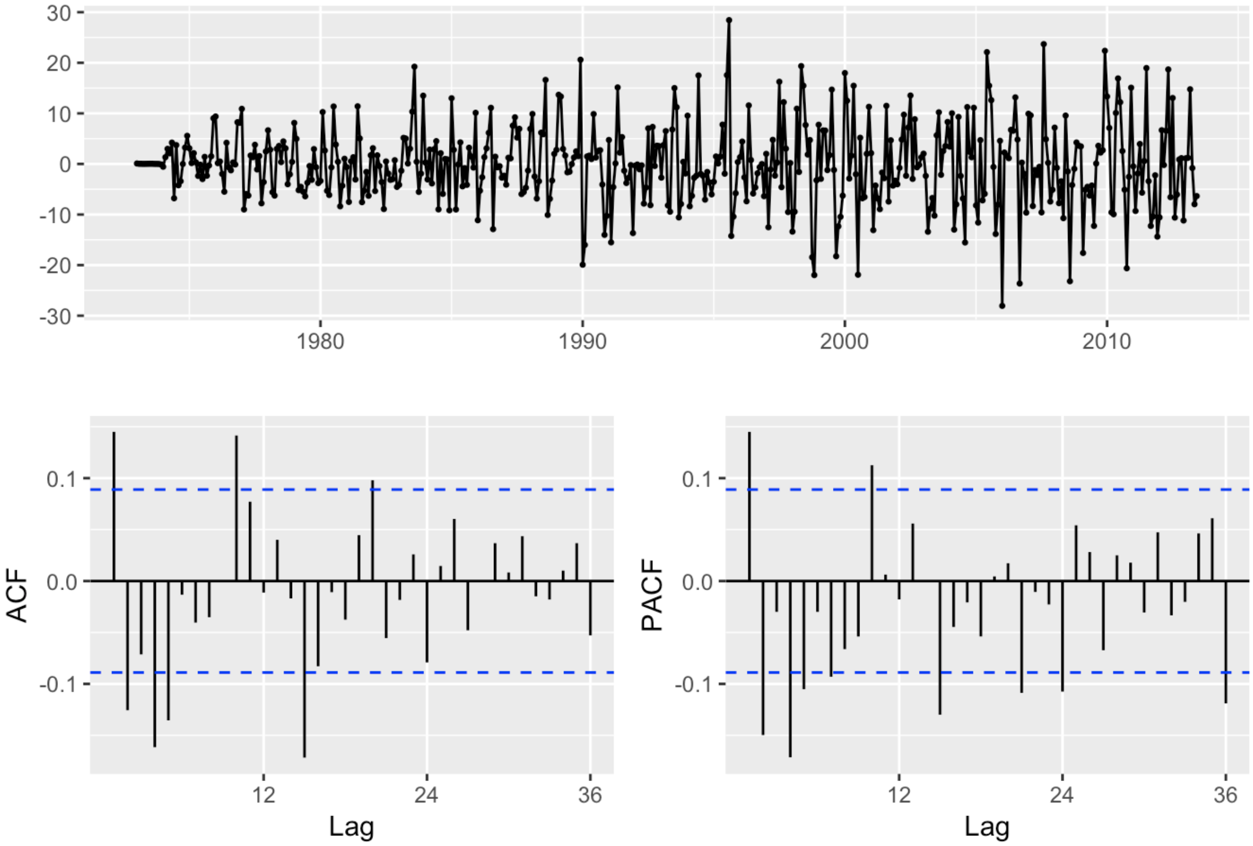



```
# we can say data is stationary, but we can see some seasonality so we take on
e more difference
usmelec %>% diff(lag=12) %>% diff() %>% ggtsdisplay()
```



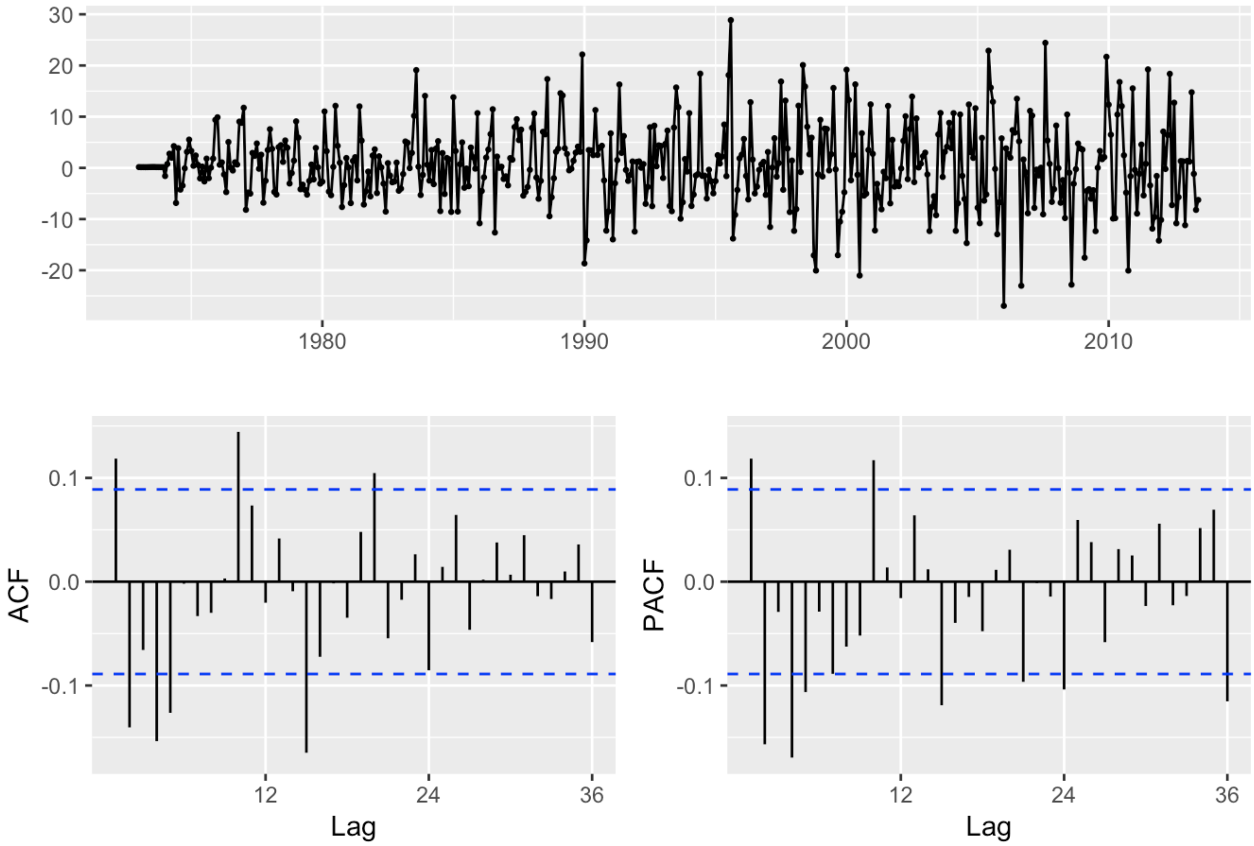

```
# we see a significant spike at lag1 in ACF and PACF

usmelec %>%
    Arima(order=c(0,1,1), seasonal=c(0,1,1)) %>%
    residuals() %>% ggtsdisplay()
```

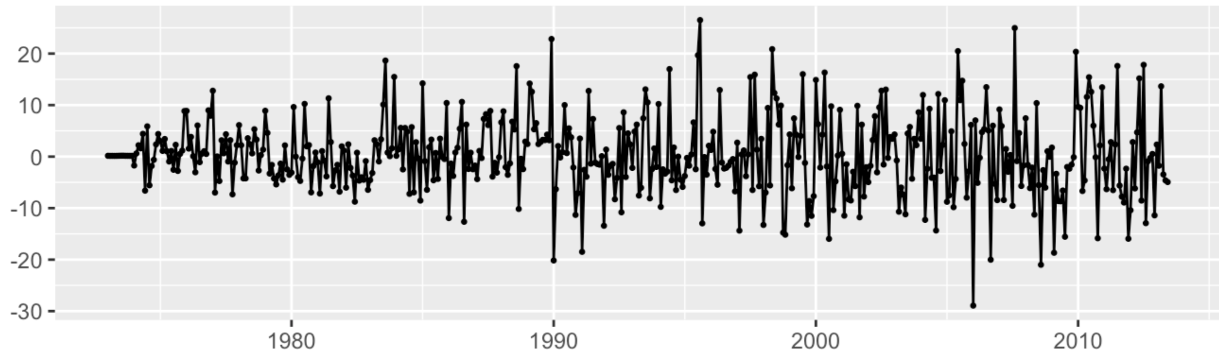


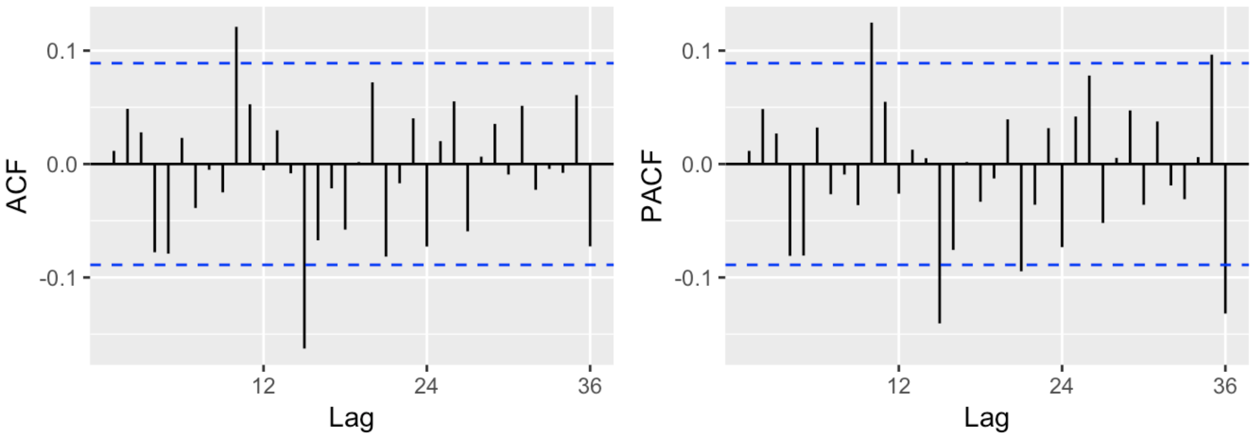

```
# we see spikes at lag1, almost significant spikes in lag 2, indication some a
dditional non-seasonal terms to be included

usmelec %>%
    Arima(order=c(1,0,1), seasonal=c(0,1,1)) %>%
    residuals() %>% ggtsdisplay()
```




```
usmelec %>%
    Arima(order=c(1,0,2), seasonal=c(0,1,1)) %>%
    residuals() %>% ggtsdisplay()
```



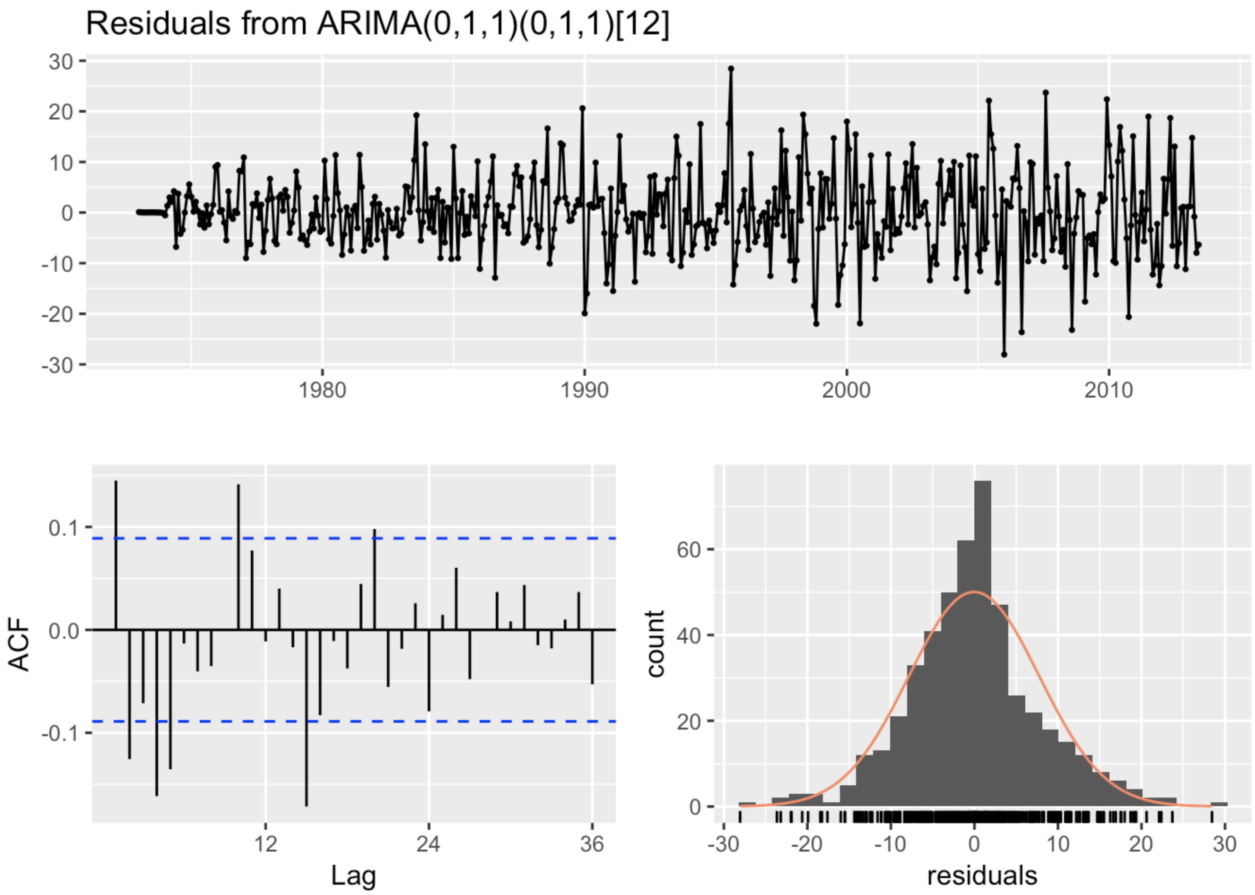



```
#this looks like a goo model among the selected one's

#lets try auto arima
auto.arima(usmelec)
```

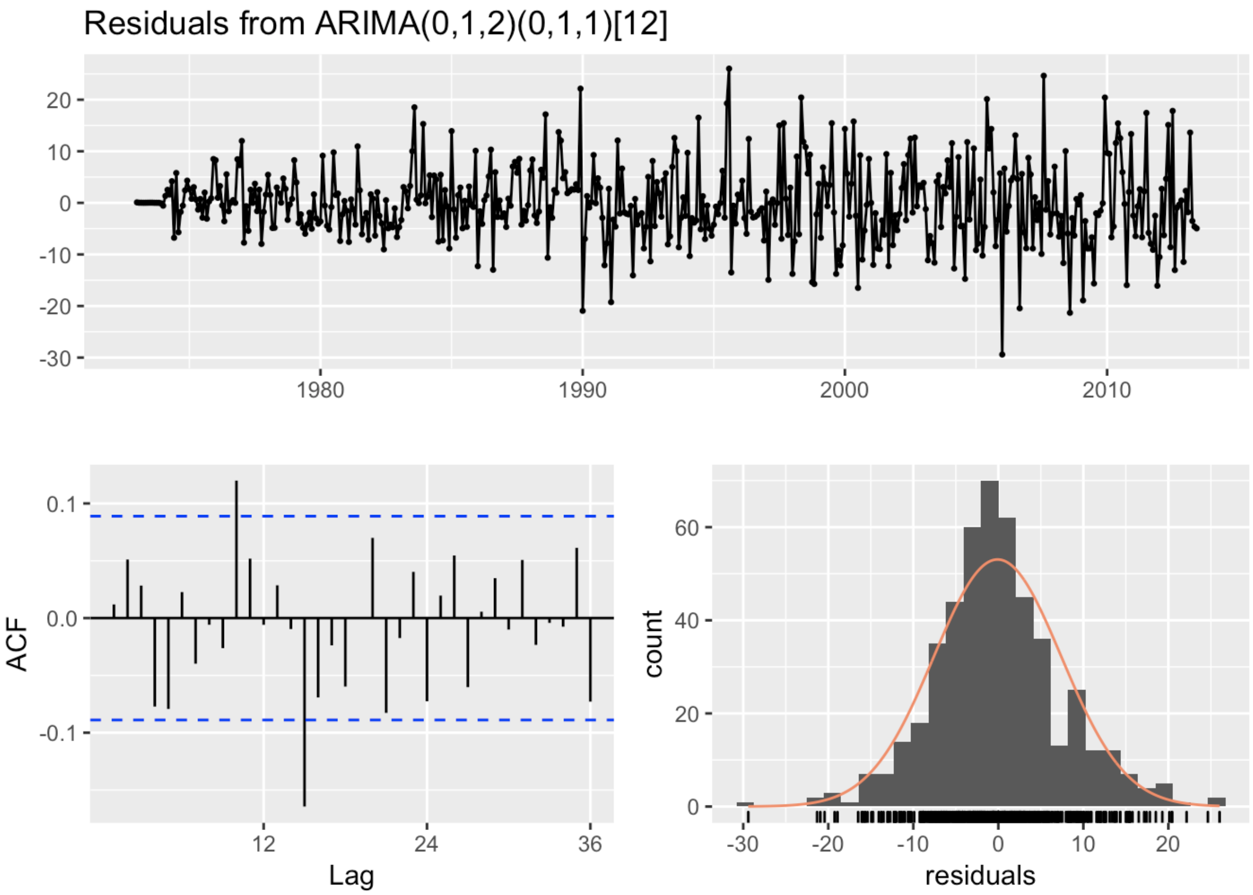
```
## Series: usmelec
## ARIMA(1,0,2)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1      ma1      ma2     sma1   drift
##       0.9717  -0.4374  -0.2774  -0.7061  0.3834
## s.e.  0.0163   0.0483   0.0493   0.0310  0.0868
##
## sigma^2 estimated as 57.67:  log likelihood=-1635.13
## AIC=3282.26   AICc=3282.44   BIC=3307.22
```

```
fit1 <- Arima(usmelec, order=c(0,1,1), seasonal=c(0,1,1))
checkresiduals(fit1)
```



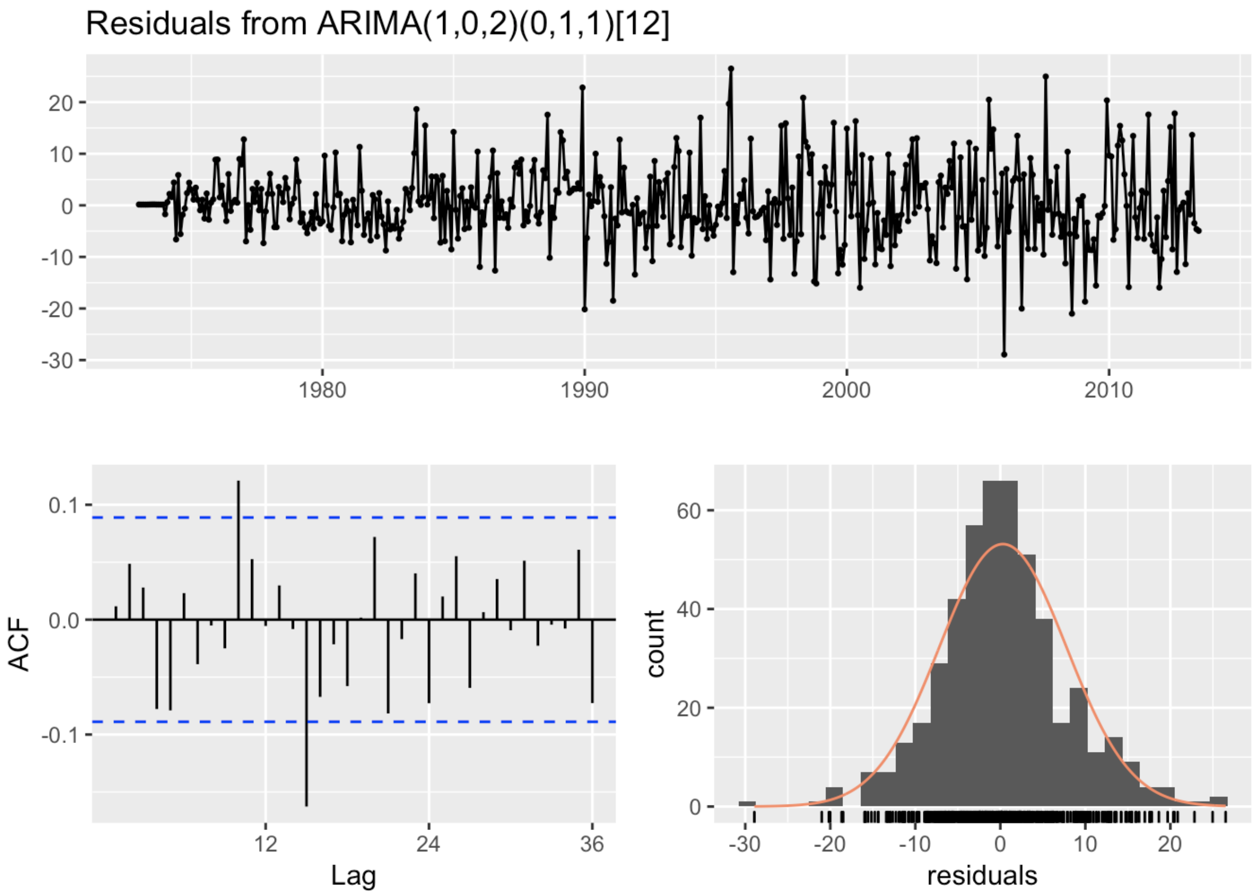

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(0,1,1)[12]
## Q* = 88.112, df = 22, p-value = 7.168e-10
##
## Model df: 2.   Total lags used: 24
```

```
fit2 <- Arima(usmelec, order=c(0,1,2), seasonal=c(0,1,1))
checkresiduals(fit2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)(0,1,1)[12]
## Q* = 45.966, df = 21, p-value = 0.001291
##
## Model df: 3.   Total lags used: 24
```

```
fit3 <- Arima(usmelec, order=c(1,0,2), seasonal=c(0,1,1))
checkresiduals(fit3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,2)(0,1,1)[12]
## Q* = 45.449, df = 20, p-value = 0.0009587
##
## Model df: 4.   Total lags used: 24
```

```
fit1$aic
```

```
## [1] 3313.701
```

```
fit2$aic
```

```
## [1] 3277.96
```

```
fit3$aic
```

```
## [1] 3284.601
```

```
# we get best value for out second model

#e)

#second model provides the least i
```

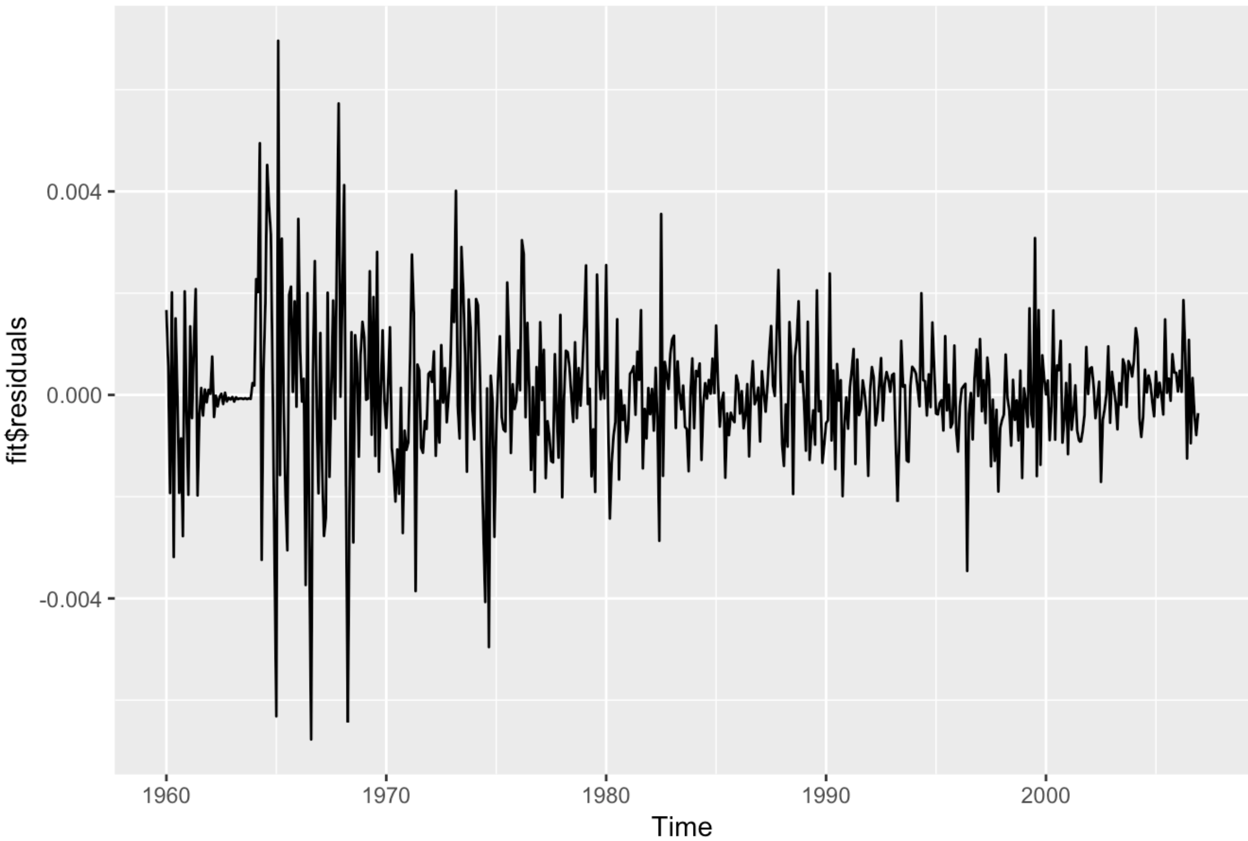
```
#2)
#a)
lambda_mcopper <- BoxCox.lambda(mcopper)
lambda_mcopper
```

```
## [1] 0.1919047
```

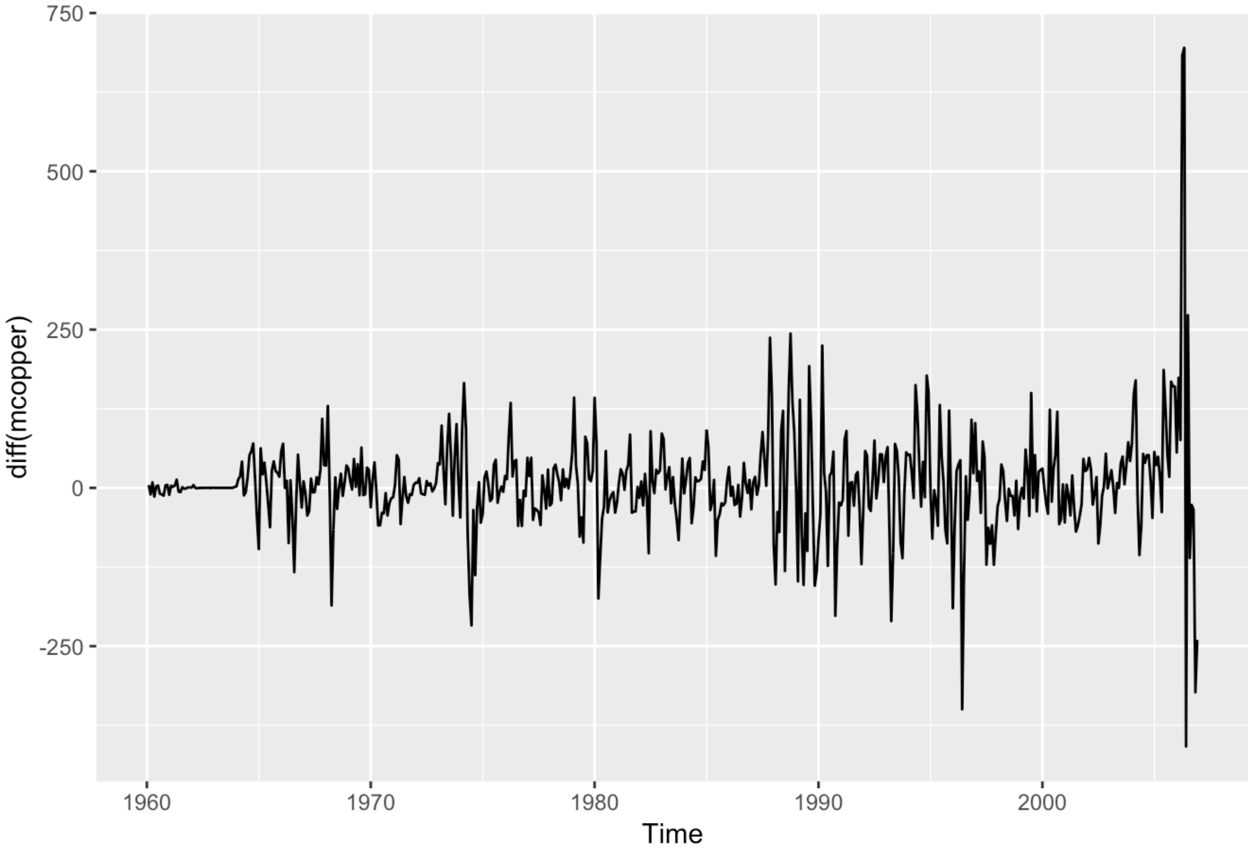
```
#b)
fit <- auto.arima(mcopper, lambda = lambda)
fit
```

```
## Series: mcopper
## ARIMA(2,1,2)(1,0,0)[12] with drift
## Box Cox transformation: lambda= -0.5738331
##
## Coefficients:
##           ar1      ar2     ma1     ma2    sar1  drift
##       -1.1530  -0.2806  1.5290  0.6461  0.0442  1e-04
## s.e.   0.1037   0.0961  0.0862  0.0800  0.0451  1e-04
##
## sigma^2 estimated as 1.902e-06:  log likelihood=2912.73
## AIC=-5811.46   AICc=-5811.26   BIC=-5781.13
```

```
autoplot(fit$residuals)
```




```
#Increasing trend
autoplot(diff(mcopper))
```



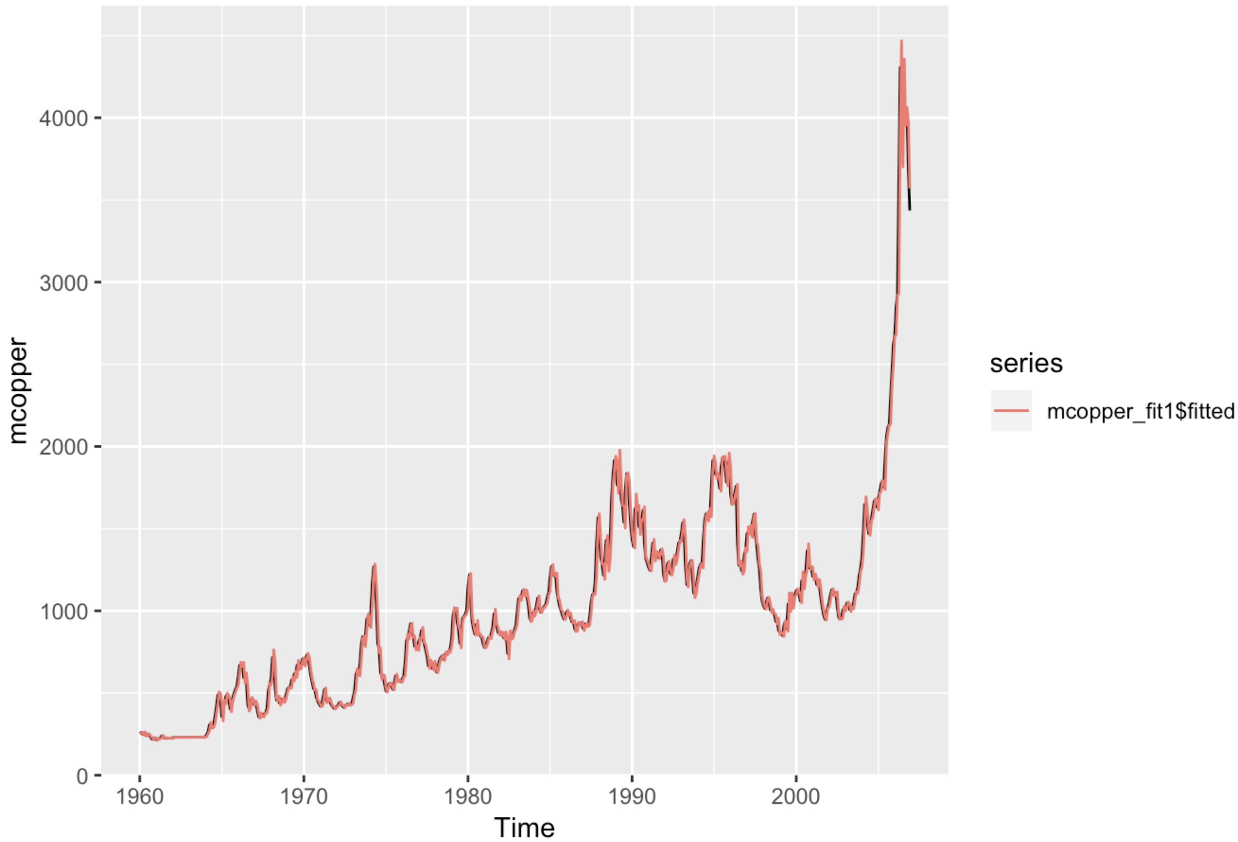
```
#data is stationary with signle differnecing

#c)

mcopper_fit1 <- Arima(mcopper, lambda = lambda_mcopper, order = c(0,1,1))
mcopper_fit1
```

```
## Series: mcopper
## ARIMA(0,1,1)
## Box Cox transformation: lambda= 0.1919047
##
## Coefficients:
##          ma1
##       0.3720
## s.e.  0.0388
##
## sigma^2 estimated as 0.04997:  log likelihood=45.05
## AIC=-86.1   AICc=-86.08   BIC=-77.43
```

```
autoplot(mcopper)+autolayer(mcopper_fit1$fitted)
```



```
#the model perfectly fits.

mcopper_fit2 <- Arima(mcopper, lambda = lambda_mcopper, order = c(1,0,1), incl
ude.drift = TRUE)



#d)

accuracy(mcopper_fit1)
```

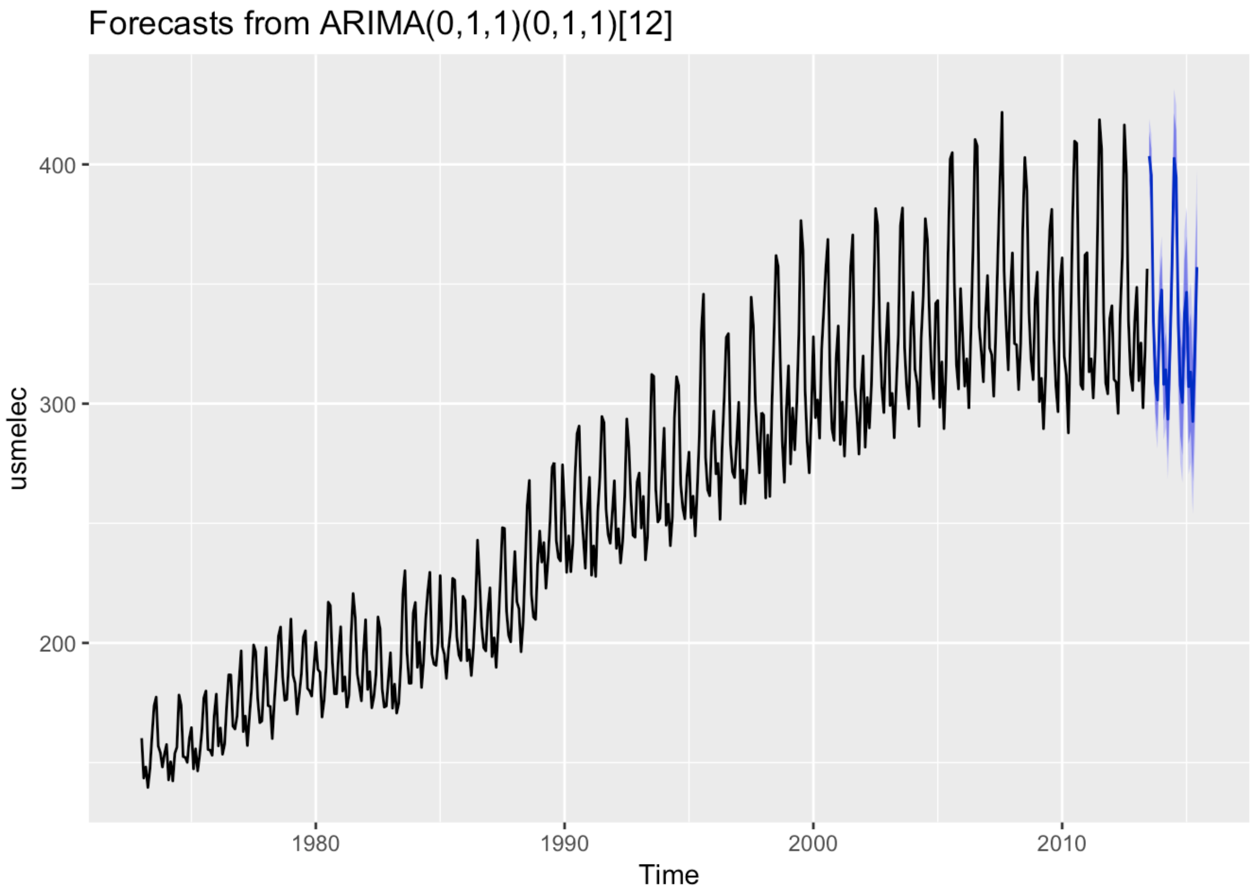
```
##                    ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 3.480533 77.27254 44.92858 0.166202 4.303677 0.2021433
##                     ACF1
## Training set -0.08442198
```

```
accuracy(mcopper_fit2)
```

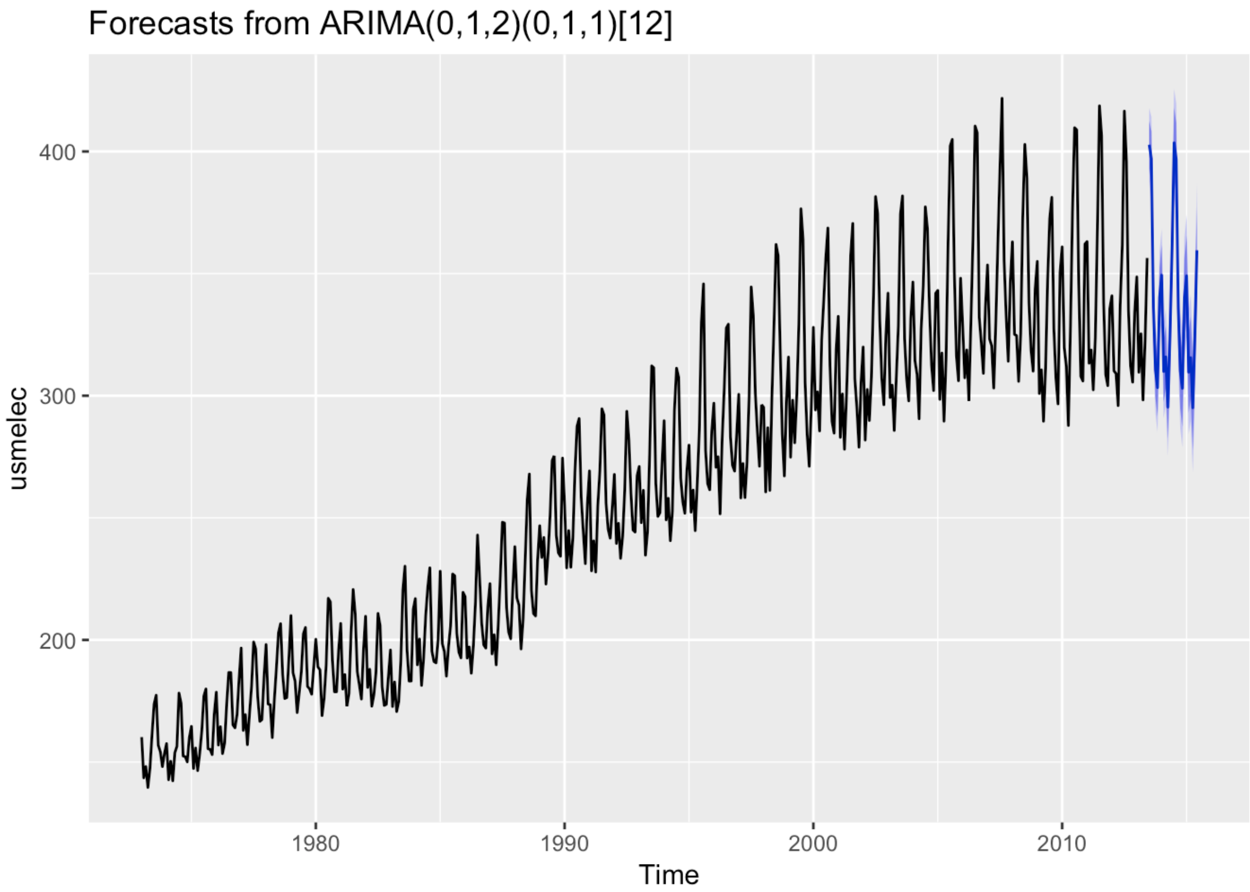
```
##                    ME     RMSE      MAE      MPE    MAPE      MASE
## Training set 1.733773 76.48637 44.54515 -0.19053 4.32756 0.2004182
##                     ACF1
## Training set -0.06861443
```

```
#Drift models works well with the data with less RMSE

#e)
fit1%>%
    forecast() %>%
    autoplot()
```

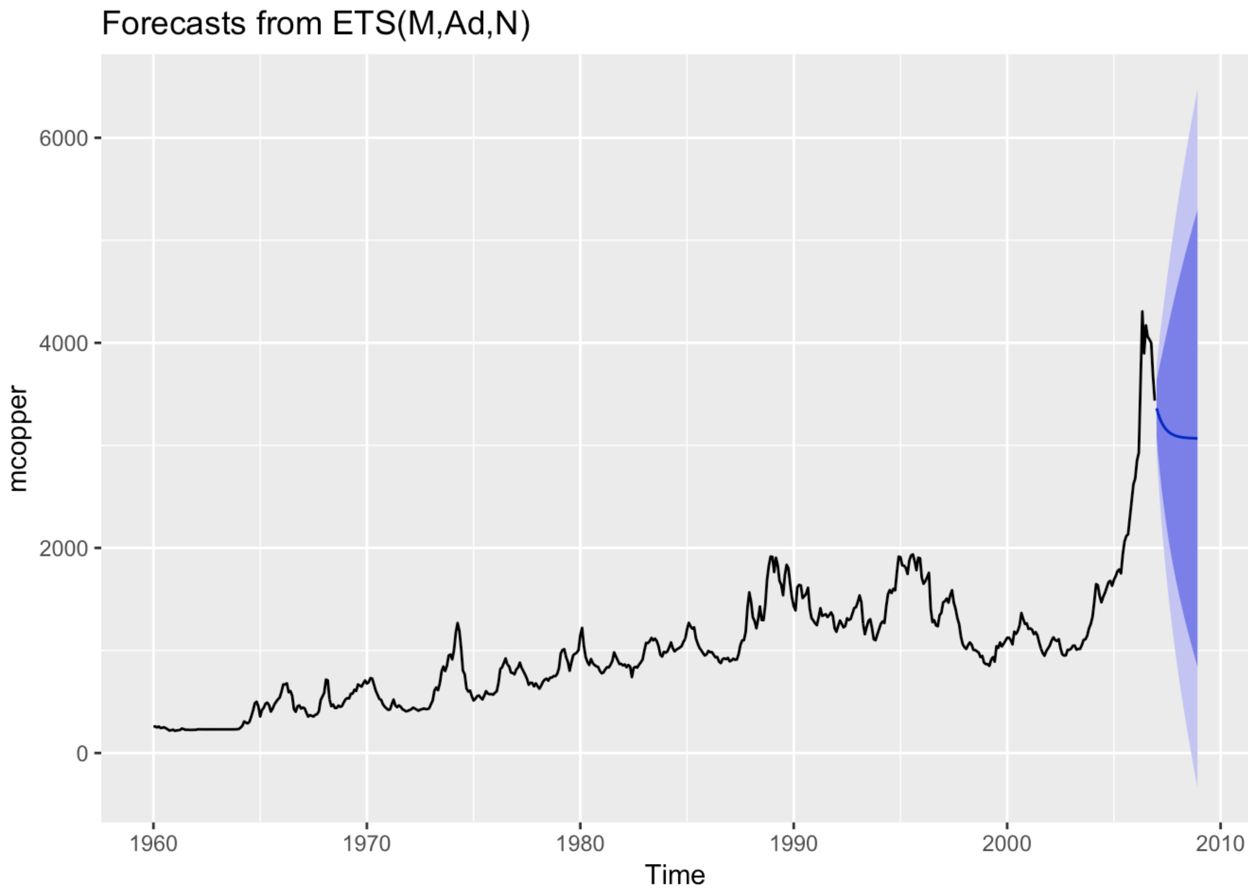



```
#Drift model forecast
fit2%>%
    forecast() %>%
    autoplot()
```




```
# Yes they look reasonable

#f)
mcopper_ets <- forecast(ets(mcopper))
autoplot(mcopper_ets)
```




```
#3)
library(caret)
```

```
## Loading required package: lattice
```

```
library(ISLR)
#Weekly$Direction

#a)

train_data <- Weekly[Weekly$Year <= 2008,]
test_data <- Weekly[Weekly$Year > 2008,]

logreg <- glm(Direction ~ Lag2, data = train_data, family = 'binomial')
pred <- predict(logreg, newdata = test_data, type = 'response')
class_prediction <- ifelse(pred > 0.50,"Down","Up")
class_prediction <- as.factor(class_prediction)
test_output   <- as.factor(test_data$Direction)
confusionMatrix(class_prediction,test_output)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down   34 56
##       Up      9  5
##
##                Accuracy : 0.375
##                  95% CI : (0.282, 0.4753)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.1097
##  Mcnemar's Test P-Value : 1.159e-08
##
##             Sensitivity : 0.79070
##             Specificity : 0.08197
##          Pos Pred Value : 0.37778
##          Neg Pred Value : 0.35714
##              Prevalence : 0.41346
##          Detection Rate : 0.32692
##    Detection Prevalence : 0.86538
##       Balanced Accuracy : 0.43633
##
##        'Positive' Class : Down
##
```

```
#b)
library(class)

knn_pred <- knn(train = data.frame(train_data$Lag2),
                                test = data.frame(test_data$Lag2),
                                cl = train_data$Direction, k = 5)
confusionMatrix(knn_pred,test_output)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down   16 21
##       Up     27 40
##
##                Accuracy : 0.5385
##                  95% CI : (0.438, 0.6367)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.8631
##
##                   Kappa : 0.0284
##  Mcnemar's Test P-Value : 0.4705
##
##             Sensitivity : 0.3721
##             Specificity : 0.6557
##          Pos Pred Value : 0.4324
##          Neg Pred Value : 0.5970
##              Prevalence : 0.4135
##          Detection Rate : 0.1538
##    Detection Prevalence : 0.3558
##       Balanced Accuracy : 0.5139
##
##        'Positive' Class : Down
##
```

```
#c)
values  <- c(0,0,0,0,0,0,0,0,0)
x <- c(1,2,3,4,5,6,7,8,9)
for (val in x) {
    knn_pred <- knn(train = data.frame(train_data$Lag2),
                                   test = data.frame(test_data$Lag2),
                                   cl = train_data$Direction, k = val)
    cm <- confusionMatrix(knn_pred,test_output)
    acc <- cm$overall['Accuracy']
    print(acc)
}
```

```
##  Accuracy
## 0.5096154
##  Accuracy
## 0.5480769
##  Accuracy
## 0.5384615
##  Accuracy
## 0.5192308
##  Accuracy
## 0.5384615
##  Accuracy
## 0.5288462
##  Accuracy
## 0.5480769
##  Accuracy
## 0.5673077
##  Accuracy
## 0.5576923
```

```
#for k =4 we are getting maximum accuracy


#d)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.5.2
```

```
classifier = svm(formula = Direction ~ Lag2,
                                 data = train_data,
                                 type = 'C-classification',
                                 kernel = 'linear')
pred <- predict(classifier, newdata = test_data)
confusionMatrix(pred,test_output)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down    0  0
##       Up     43 61
##
##                Accuracy : 0.5865
##                  95% CI : (0.4858, 0.6823)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.5419
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 1.504e-10
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.5865
##              Prevalence : 0.4135
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Down
##
```