STAT 460 HW5

Benjamin Pond and Dylan Borchert

10/21/2020

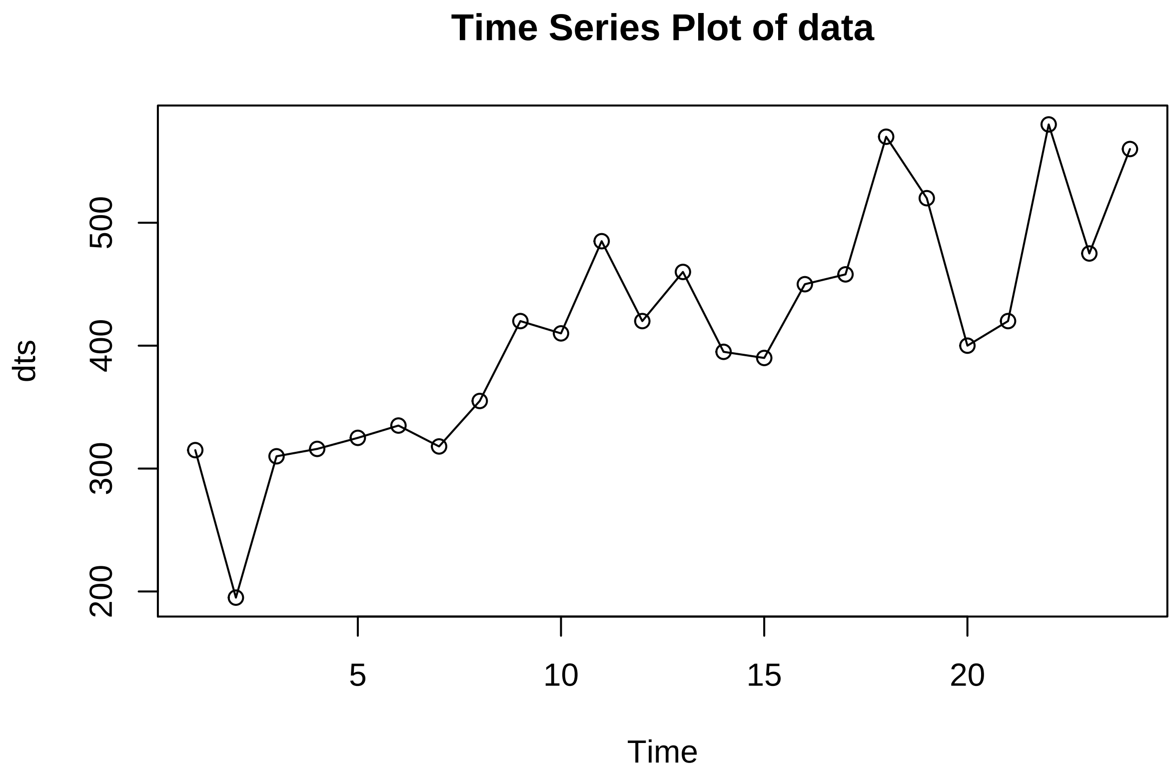
Homework 5

Exercise 4.8

The data in Tables E4.4 exhibit a linear trend

###(a) Verify there is a trend by plotting the data

```
# write data
period <- seq(from=1, to=24, by=1)
yt <- c(315, 195, 310, 316, 325, 335, 318, 355, 420, 410, 485, 420,
        460, 395, 390, 450, 458, 570, 520, 400, 420, 580, 475, 560)

# create data frame
d <- data.frame(period, yt)
# make into time series
dts<-ts(d[,2])

# plot time series
plot(dts, type='o', main='Time Series Plot of data')
```

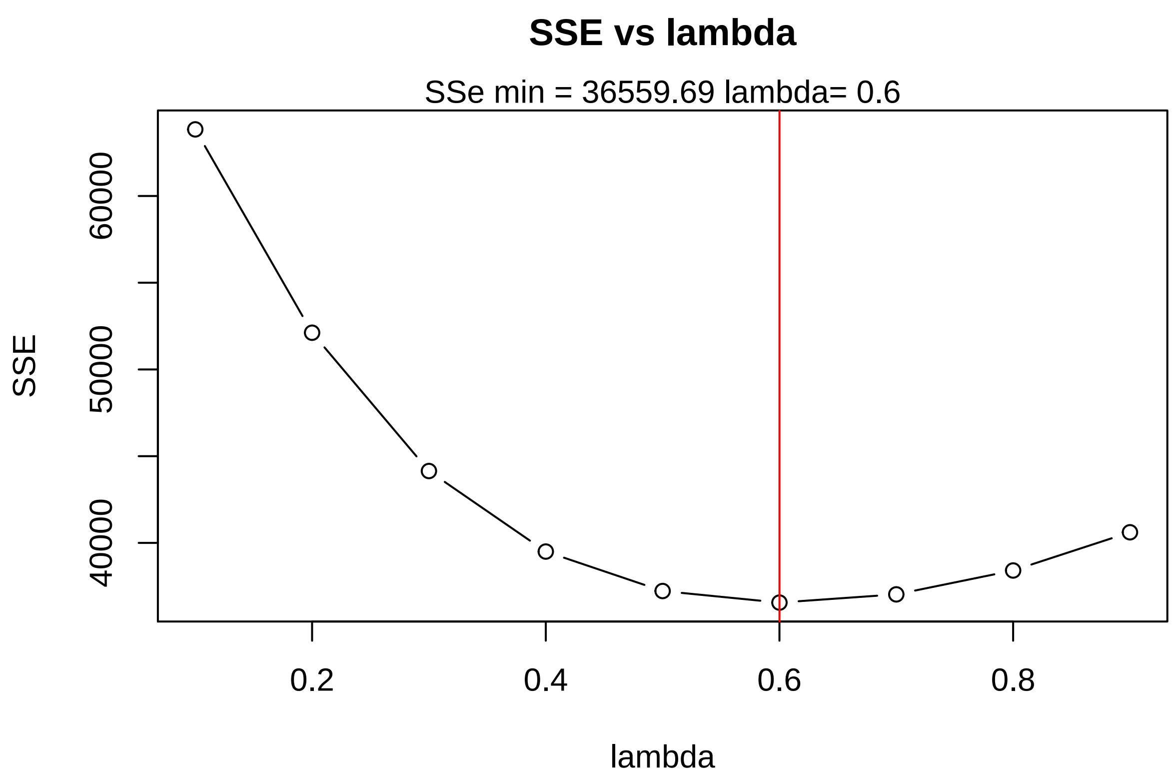


The data shows a positvie linear trend.

###(b) Using the first 12 observations, develope an appropriate preocedure for forecasting.

```r
# first smooth function
firstsmooth<-function(y,lambda,start=y[1]){
  ytilde<-y
  ytilde[1]<-lambda*y[1]+(1-lambda)*start
  for (i in 2:length(y)){
    ytilde[i]<-lambda*y[i]+(1-lambda)*ytilde[i-1]
  }
ytilde
}

# function calculate values of prediction, prediction error,
# SSE, MAPE, MAD, MSD at different steps in the model
measacc.fs<-function(y,lambda){
  out <- firstsmooth(y,lambda)
  T <- length(y)
  pred <- c(y[1], out[1:(T-1)])
  prederr <- y-pred
  SSE <- sum(prederr^2)
  MAPE <- 100*sum(abs(prederr/y))/T
  MAD <- sum(abs(prederr))/T
  MSD <- sum(prederr^2)/T
  ret1 <- c(SSE,MAPE,MAD,MSD)
  names(ret1)<-c('SSE', 'MAPE','MAD','MSD')
```

```r
  return(ret1)
}

# apply to data and plot SSE Vs Lambda
lambda.vec <- seq(0.1, 0.9, 0.1)
sse <- function(sc){measacc.fs(d[1:12,2],sc)[1]}
sse.vec <-sapply(lambda.vec, sse)
opt.lambda<-lambda.vec[sse.vec==min(sse.vec)]
plot(lambda.vec, sse.vec, type='b', main='SSE vs lambda',
     xlab='lambda', ylab='SSE')
abline(v=opt.lambda, col='red')
mtext(text=paste('SSe min =', round(min(sse.vec),2), 'lambda=', opt.lambda))
```



The value of lambda that minimizes SSE is $\lambda = 0.6$. So our forecast model is:

$$\hat{y}_{T+1}(T) = \left(2 + \frac{\lambda}{1-\lambda}\right) * \tilde{y}_T^{(1)} - \left(1 + \frac{\lambda}{1-\lambda}\right) * \tilde{y}_T^{(2)} = 3.5 * \tilde{y}_T^{(1)} + 2.5 * \tilde{y}_T^{(2)}$$

The forecast model predicts y using a linear combination of the first smoothed value, $\tilde{y}_T^{(1)}$, and second smoothed value $\tilde{y}_T^{(2)}$ of the prior observation. The weights involve out optimum value of lambda, $\lambda = 0.6$

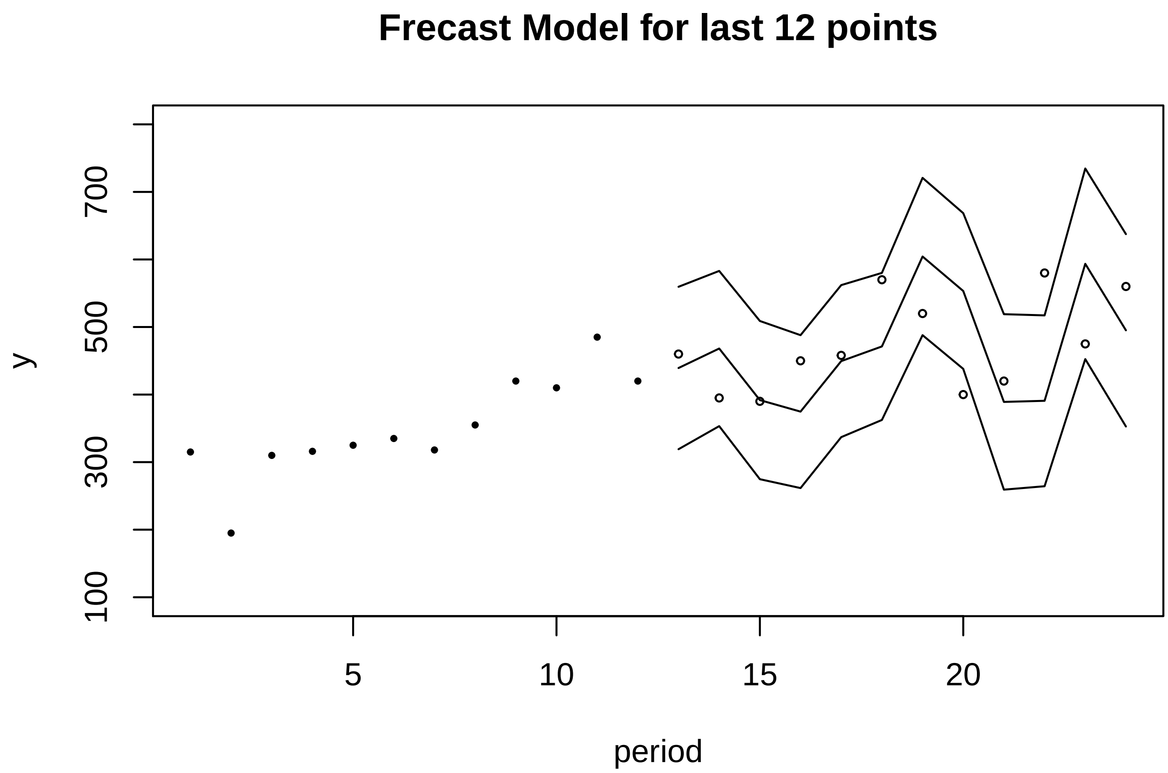
###(c) Forecast the last 12 observations and calculate the forecast errors. Does the frecasting procedure seem to be working satisfactorily?

```r
# initialize vectors and values
l<-0.6
```

```r
T<-12
tau<-12
alpha.lev<-0.05
cpi.forecast<-rep(0,tau)
cl<-rep(0,tau)
cpi.smooth1<-rep(0,tau+T)
cpi.smooth2<-rep(0,tau+T)

# forcast
for (i in 1:tau){
  cpi.smooth1[1: (T+i-1)] <-firstsmooth(y=d[1:(T+i-1),2], lambda=l)
  cpi.smooth2[1: (T+i-1)] <-firstsmooth(y=cpi.smooth1[1:(T+i-1)], lambda=l)
  cpi.forecast[i]<-(2+(l/(1-l)))*cpi.smooth1[T+i-1]-
    (1+(l/(1-l)))*cpi.smooth2[T+i-1]
  cpi.hat<-2*cpi.smooth1[1:(T+i-1)]-cpi.smooth2[1:(T+i-1)]
  sig.est<-sqrt(var(d[2:(T+i-1),2]-cpi.hat[1:(T+i-2)]))
  cl[i]<-qnorm(1-alpha.lev/2)*sig.est
}

# calculate forecast errors and display
forecast.error<-d[,2]-cpi.forecast
data.frame(Forecast=cpi.forecast, Forecast_Error<-forecast.error)[13:24,]
```

```
##    Forecast Forecast_Error....forecast.error
## 13 439.3167                        20.683340
## 14 468.1451                       -73.145141
## 15 391.8254                        -1.825447
## 16 374.7571                        75.242865
## 17 449.5136                         8.486363
## 18 471.2498                        98.750232
## 19 604.3576                       -84.357632
## 20 553.2861                      -153.286143
## 21 389.1317                        30.868307
## 22 390.7796                       189.220428
## 23 593.5626                      -118.562586
## 24 495.1253                        64.874662
```

```r
# plot forecast model with prediction intervals
plot(d[1:T,2], type='p', pch=16, cex=0.5, xlim=c(1,T+tau), ylim=c(100,800),
     ylab='y', xlab='period', main='Frecast Model for last 12 points')
points((T+1):(T+tau), d[(T+1):(T+tau),2], cex=0.5)
lines((T+1):(T+tau), cpi.forecast)
lines((T+1):(T+tau), cpi.forecast+cl)
lines((T+1):(T+tau), cpi.forecast-cl)
```



The forecast model seems to work pretty well to predict the last 12 data points in the data set. The 99% confidence interval encompasses most of the points, but there are two that lie outside. The points that are outside of the interval are still very close to the interval region. The model appears to be satisfactory.

Exercise 4.27

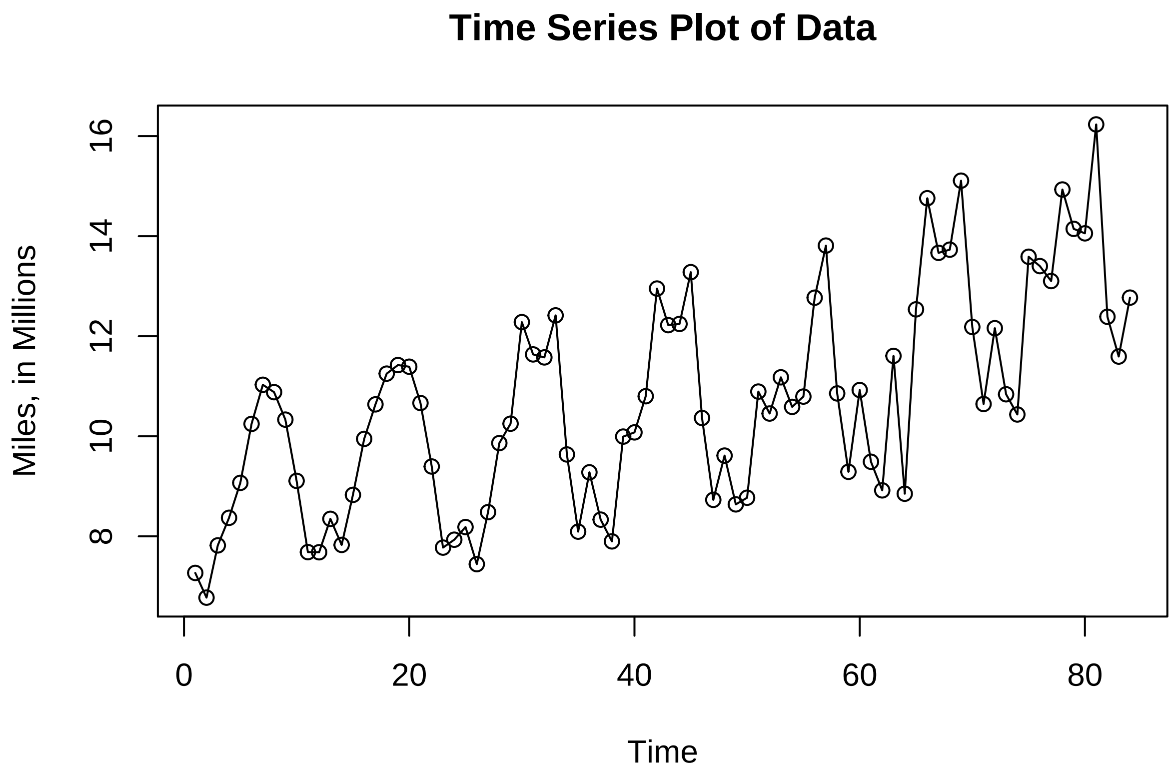
Table B.10 contains 7 years of monthly data on the number of airline miles flown in the United Kingdom. This is seasonal data.

###(a) Make a time series plot of the data and verify that it is seasonal.

```
# import Data
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 3.5.3
```

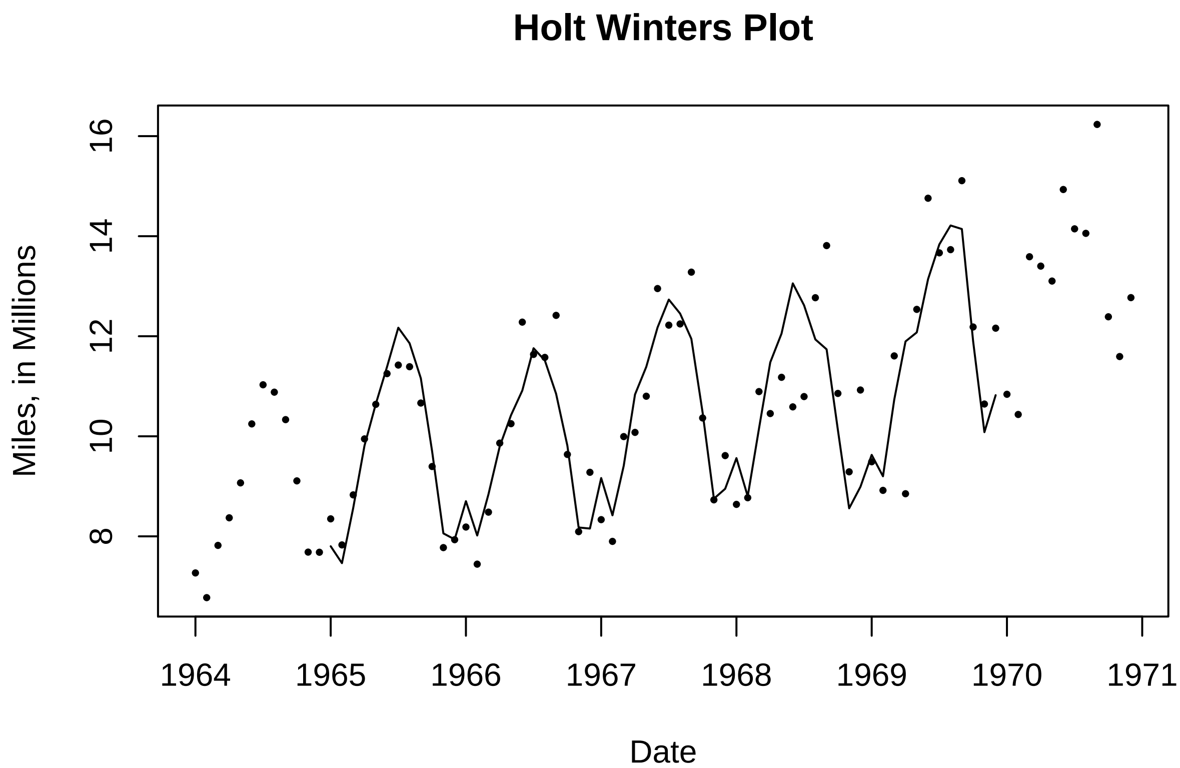
```
data <- read_excel("C:/College Stuff/Time series/AppendixB_datafile.xls",
    sheet = "B.10-FLOWN", col_types = c("date",
        "numeric"), skip = 2)
# create timeseries
data_ts<-ts(data[,2])

# plot time series
plot(data_ts, type='o',
     main='Time Series Plot of Data')
```



The data does appear to be seasonal. It also follows a linear trend. The seasonal occilations seems to get larger as time goes on, so using multiplicative seasonality is best.

###(b) Use Winter's multiplicative method for the first 6 years to develop a forecasting method for this data. How well does this smoothing procedure work?

```
# create time series for first 6 moths
yt <- ts(data[,2], start=c(1964,1), freq=12)
y_first6 <-  ts(data[1:72,2], start=c(1964,1), freq=12)

# holt winters model
air.model <- HoltWinters(y_first6,alpha=0.2,beta=0.2, gamma=0.2, seasonal='multiplicative')

# plot data and holt winters fit
plot(yt, type="p", pch=16, cex=0.5, xlab='Date', ylab='Miles, in Millions', main='Holt Winters Plot')
lines(air.model$fitted[,1])
```



The multiplicative seasonal model is:

$$y_t = L_t S_t + \epsilon_t$$

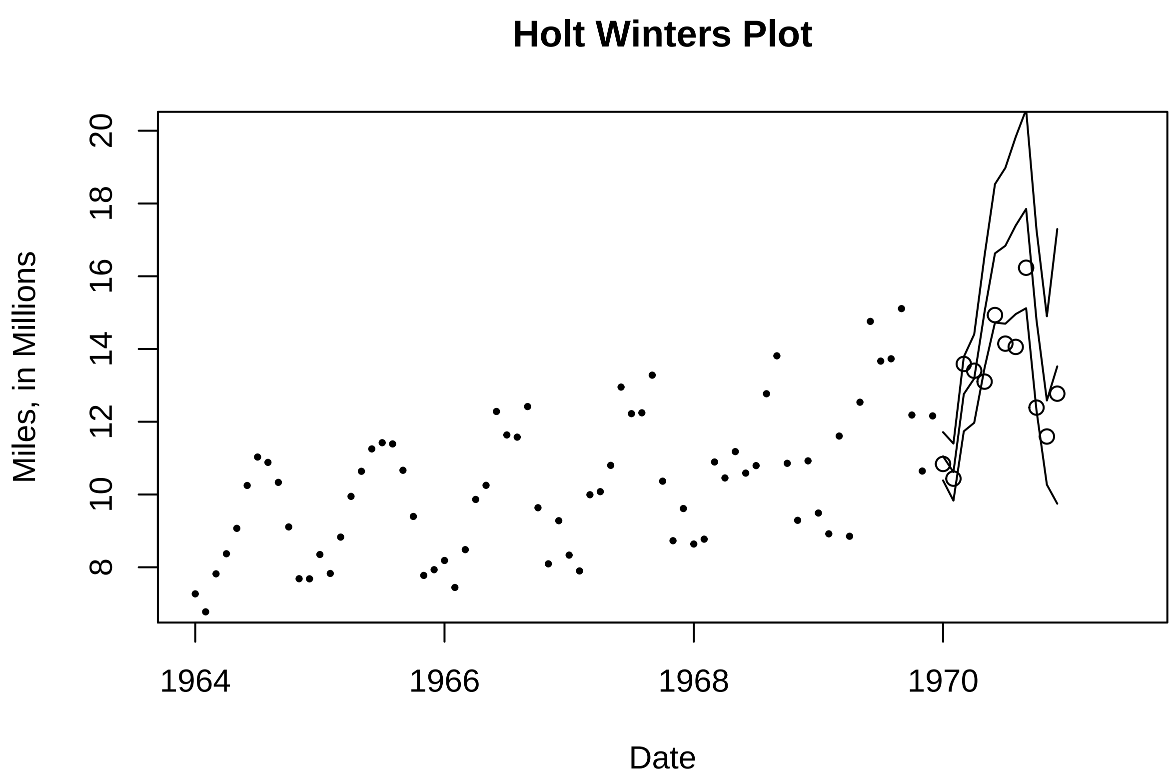
###(c) Make one-step-ahead forecasts of the last 12 months. Determine the forecast errors. How well did your procedure work in forecasting the new data?

```
# Forecast errors
y2 <- data[73:84,]
y2.ts <- ts(y2[,2], start=c(1970,1), freq=12)
y2.forecast <- predict(air.model, n.ahead=12, prediction.interval=TRUE)
Forecast_Error <- as.numeric(y2.ts)-as.numeric(y2.forecast[,1])

# display
data.frame(observation=1:12, Forecast_Error=Forecast_Error)
```

```
##    observation Forecast_Error
## 1            1     -0.2108528
## 2            2     -0.1799367
## 3            3      0.8288952
## 4            4      0.2142654
## 5            5     -1.9256487
## 6            6     -1.6959719
## 7            7     -2.6903735
## 8            8     -3.3378761
## 9            9     -1.6168380
## 10          10     -2.4004458
```

```
## 11          11     -0.9902873
## 12          12     -0.7506524
```

```
# plot the forecast
plot(y_first6, type="p", pch=16, cex=0.5, xlim=c(1964, 1971.5), ylim=c(7,20),
     xlab='Date', ylab='Miles, in Millions', main='Holt Winters Plot')
points(y2.ts)
lines(y2.forecast[,1])
lines(y2.forecast[,2])
lines(y2.forecast[,3])
```



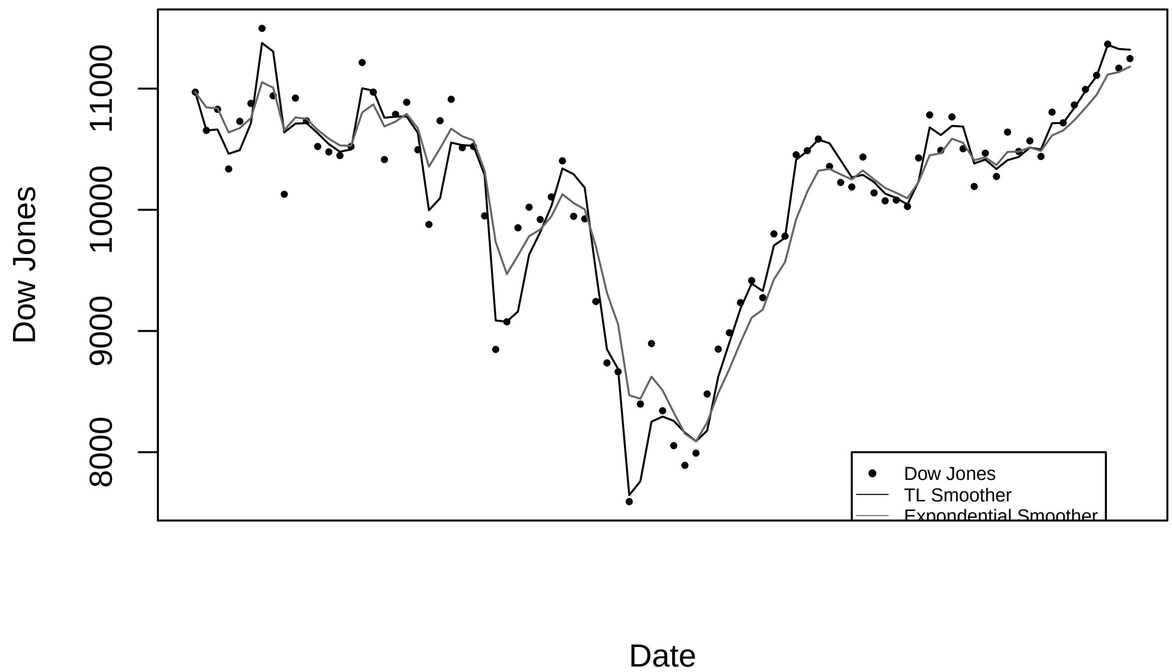
The forecast model seems to be predicting higher values than what the values actually are. We can see this with the negative forecast errors. Most of the data points lie within the confidence interval for the forecasted values. Although there are some points where the trend within the season changes from increasing to decreasing that lie outside of the confidence interval. The confidence interval is also very wide at the peak. The model works fairly well, but it overpredicts at the peak, it might be worth trying abother forecast model.

Example 4.6

Compare Trig-Leacher smoother with exponential smoother. Make a plot, and show the first 10 calculations for the Trigg-Leach smoother.

```
# import data
library(readxl)
dji.data <- read_excel("C:/College Stuff/Time series/DowJones.xlsx")
```

```r
# TL Smoother function
tlsmooth <- function(y, gamma, y.tilde.start=y[1], lambda.start=1){
  T <-length(y)
  Qt <-vector()
  Dt<-vector()
  y.tilde<-vector()
  lambda<-vector()
  err <- vector()

  lambda[1]=lambda.start
  y.tilde[1]=y.tilde.start
  Qt[1]<-0
  Dt[1]<-0
  err[1]<-0

  for (i in 2:T){
    err[i]<-y[i]-y.tilde[i-1]
    Qt[i]<-gamma*err[i]+(1-gamma)*Qt[i-1]
    Dt[i]<-gamma*abs(err[i])+(1-gamma)*Dt[i-1]
    lambda[i]<-abs(Qt[i]/Dt[i])
    y.tilde[i]=lambda[i]*y[i] + (1-lambda[i])*y.tilde[i-1]
  }
return(cbind(y.tilde,lambda,err,Qt,Dt))
}

# Apply TL smoother
out.tl.dji <-tlsmooth(dji.data$`Dow Jones`,0.4)

# Apply Eponential Smoother
dji.smooth1<-firstsmooth(y=dji.data$`Dow Jones`, lambda=0.4)

# plot
plot(dji.data$`Dow Jones`, type='p', pch=16, cex=0.5, xlab='Date', ylab='Dow Jones', xaxt='n')
lines(out.tl.dji[,1])
lines(dji.smooth1, col="grey40")
legend(60, 8000, c('Dow Jones', 'TL Smoother', 'Expondential Smoother'),
       pch=c(16,NA,NA), lwd=c(NA, .5, .5), cex=0.55, col=c('black', 'black', 'grey40'))
```




```r
# output tables for TL smoother model
data.frame(date = dji.data$Date, Dow_Jones=dji.data$`Dow Jones`, Lambda=out.tl.dji[,2],
           Error=out.tl.dji[,3], Qt=out.tl.dji[,4], Dt=out.tl.dji[,5])[1:10,]
```

```
##          date Dow_Jones     Lambda      Error         Qt       Dt
## 1  1999-06-01   10970.8 1.00000000     0.0000    0.00000   0.0000
## 2  1999-07-01   10655.2 1.00000000  -315.6000 -126.24000 126.2400
## 3  1999-08-01   10829.3 0.04198536   174.1000   -6.10400 145.3840
## 4  1999-09-01   10337.0 0.61566314  -325.5097 -133.86626 217.4343
## 5  1999-10-01   10729.9 0.11279687   267.7946   26.79810 237.5784
## 6  1999-11-01   10877.8 0.57381146   385.4882  170.27416 296.7423
## 7  1999-12-01   11497.1 0.84560785   783.5907  415.60076 491.4817
## 8  2000-01-01   10940.5 0.16010797  -435.6198   75.11256 469.1369
## 9  2000-02-01   10128.3 0.56616891 -1178.0736 -426.16189 752.7116
## 10 2000-03-01   10921.9 0.25271480   282.5151 -142.69111 564.6330
```

The TL smoother model seems to better fit the model better. The exponential smoother model does not seem to adapt quickly to large charnges in the data, where the TL smoother model can. The TL smoother can fit the large peaks and valleys better. The tables shows 10 rows of the values used to construct the TL smoother model.