Assignment 3

Be sure to mark each problem # properly and your student ID (last 4 digits) shows up - no names, remember to number your pages. The submitted file should be PDF (preferably typed). Name your file with the last four digits of your student id followed by '-A3'. For example, if the last four digits of your ID are 1234, then the file name should be the following: 1234-A3.pdf

3_1 [30 points]. This assignment extends from Assignment-2. Q-1, which is reproduced below. Your assignment is to extend Part-b and use SVM, and provide a comparison with a discussion (note - need to do it only for 2.1 part-b).

PROGRAM

```
#import the dataset and make some changes
library(readr)
kc_weather_srt <- read_csv("C:/Users/bvkka/Desktop/ISL-Deep Medhi/kc_weather_srt.csv")

kc_weather_srt=kc_weather_srt[,2:9]
```

```
> kc_weather_srt
# A tibble: 366 x 8
   Temp.F Dew_Point.F Humidity.percentage Sea_Level_Press.in Visibility.mi Wind.mph Precip.in Events
    <int>       <int>               <int>              <dbl>         <int>    <int>     <dbl>  <chr>
 1     26          12                  73              30.19             5        9      0.03   Snow
 2     31          18                  68              29.95             7       11      0.01   Snow
 3     10           1                  63              30.24             5       14      0.02   Snow
 4     38          35                  90              29.70             6        5      0.00   Rain
 5     40          30                  75              29.80             9        7      0.00   Rain
 6     49          29                  51              29.64            10       10      0.00   Rain
 7     36          19                  45              30.02            10        9      0.00   Rain
 8     29          11                  48              30.14            10        8      0.00   Rain
 9     26           2                  38              30.13            10       13      0.00   Snow
10     13          -3                  46              30.37            10       12      0.00   Snow
# ... with 356 more rows
>
```

```
#first make the response column to 0-snow, 1-rain and 2-rain_thunderstorm
#install.packages("plyr")
```

```r
library(plyr)
kc_weather_srt$Events <- revalue(kc_weather_srt$Events,c("Snow"=1))
kc_weather_srt$Events <- revalue(kc_weather_srt$Events,c("Rain"=0))
kc_weather_srt$Events <- revalue(kc_weather_srt$Events,c("Rain_Thunderstorm"=2))


#small changes to Events column , making it to numeric from character
kc_weather_srt$Events<-as.numeric(as.character(kc_weather_srt$Events))
```

```
> kc_weather_srt$Events
  [1] 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 2 0 1 1 1 1 0 2 1 1 0 2 2 2 2 0 0 2 2 1 0 0 0 2 2 0 0 0 0 0 2 0 2 0 0 0 0 2 2 0 0 0 2 2 2 2 2 0 0
 [66] 0 2 0 0 2 2 2 2 0 2 0 2 2 2 0 0 2 0 2 2 2 0 0 2 2 2 2 2 0 2 2 2 0 0 2 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1
[131] 0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 2 0 0 0 0 2 0 2 2 2 2 0 0 2 0 0 0 2 0 0 0 2 2 2 2 2 0 0 0 2 2 0
[196] 0 0 2 0 2 2 2 0 2 2 2 2 0 0 0 2 2 0 2 0 2 0 2 0 0 0 0 2 2 2 2 2 0 2 2 2 0 2 0 2 2 2 2 0 2 2 2 0 0 0 0 0 0 0 0 0 2 2 0 0 0 2 0 0 0
[261] 0 0 0 0 2 0 0 0 0 0 1 1 1 0 2 1 1 0 1 2 0 0 2 0 1 2 0 0 0 0 0 2 0 0 0 0 2 2 2 0 2 2 2 2 2 0 0 0 0 0 0 2 0 0 2 2 0 0 2 2 2 2 0 2 2
[326] 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 0 2 2 2 2 2 2 0 2 2 2 0 0 2 0 2 2 0 0 0 2 0 0 1 1 0
>
```

```r
#replications
rep=100



# newly added


accuracy1=dim(rep)



precision_snow1=dim(rep)
precision_rain1=dim(rep)
precision_rainThunderstorm1=dim(rep)

recall_snow1=dim(rep)
recall_rain1=dim(rep)
```

```
recall_rainThunderstorm1=dim(rep)



#splitting the dataset into training and test sets, also install caTools packages
#install.packages('caTools')
library(caTools)
set.seed(123)

for(k in 1:rep)
{


  split=sample.split(kc_weather_srt$Events,SplitRatio = 0.7923)
  training_set=subset(kc_weather_srt,split==TRUE)
  test_set=subset(kc_weather_srt,split==FALSE)
```

Data	
kc_weather_srt	366 obs. of 8 variables
test_set	76 obs. of 8 variables
training_set	290 obs. of 8 variables

```
#*****SVM*****#

  #fitting SVM to the training set
  #install.packages('e1071')
  library(e1071)
  classifier=svm(Events~.,data=training_set,type='C-
classification',kernel="radial",cost=1,gamma=0.04545455,coef.0=0,epsilon=0.1)
```

```
> classifier

Call:
svm(formula = Events ~ ., data = training_set, type = "C-classification", kernel = "radial", cost = 1, gamma = 0.04545455,
    coef.0 = 0, epsilon = 0.1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.04545455

Number of Support Vectors:  195
```

```
y_pred1=predict(classifier,newdata = test_set[-8])

#making the confusion matrix

cm1=table(test_set$Events,y_pred1)
```

```
> cm1
   y_pred1
     0  1  2
  0 24  0 13
  1  1  9  0
  2  7  0 22
>
```

```
#calculating the accuracy
accuracy1[k]=mean(y_pred1==test_set$Events)

#Precision of rain, rain_thunderstorm and snow results

precision1=precision1<-diag(cm1)/colSums(cm1)
```

```
  precision_rainThunderstorm1[k]=precision1[3]
  precision_snow1[k]=precision1[2]
  precision_rain1[k]=precision1[1]

  #Recall of rain, rain_thunderstorm and snow results

  recall1=recall1<-diag(cm1/rowSums(cm1))
  recall_rainThunderstorm1[k]=recall1[3]
  recall_snow1[k]=recall1[2]
  recall_rain1[k]=recall1[1]


}



#Calculating the end results using mean

  mean(accuracy1)
  mean(precision_rain1)
  mean(precision_rainThunderstorm1)
  mean(precision_snow1)

  mean(recall_rain1)
  mean(recall_rainThunderstorm1)
  mean(recall_snow1)
```

SVM Radial Kernel Results

```
> mean(accuracy1)
[1] 0.7736842
> mean(precision_rain1)
[1] 0.7938852
> mean(precision_rainThunderstorm1)
[1] 0.7231856
> mean(precision_snow1)
[1] 0.8981612
> mean(recall_rain1)
[1] 0.7278378
> mean(recall_rainThunderstorm1)
[1] 0.7931034
> mean(recall_snow1)
[1] 0.887
>
```

RESULTS:

I also changed the tuning parameters under SVM tuning to see the best results. I have used Kernels like linear, radial and sigmoid with different cost and gamma parameters. We see some differences.

SVM Linear Results

```
.6973684
 [64] 0.7631579 0.7631579 0.8157895 0.7631579 0.7763158 0.7763158 0
.6973684
 [71] 0.7763158 0.8026316 0.7631579 0.7631579 0.7631579 0.7500000 0
.8157895
 [78] 0.7631579 0.7763158 0.7631579 0.8421053 0.8684211 0.7500000 0
.7763158
 [85] 0.7631579 0.7763158 0.6973684 0.7763158 0.7763158 0.6710526 0
.7894737
 [92] 0.7894737 0.7763158 0.7236842 0.7368421 0.7763158 0.8157895 0
.7763158
 [99] 0.7368421 0.7368421
> mean(accuracy1)
[1] 0.7638158
> mean(precision_rain1)
[1] 0.7856708
> mean(precision_rainThunderstorm1)
[1] 0.7071105
> mean(precision_snow1)
[1] 0.9012634
> mean(recall_rain1)
[1] 0.7132432
> mean(recall_rainThunderstorm1)
[1] 0.7817241
> mean(recall_snow1)
[1] 0.899
>
```

SVM RADIAL WITH GAMMA =0 AND COST =1 RESULTS

```
+   precision_snow1[k]=precision1[2]
+   precision_rain1[k]=precision1[1]
+
+
+   recall1=recall1<-diag(cm1/rowSums(cm1))
+   recall_rainThunderstorm1[k]=recall1[3]
+   recall_snow1[k]=recall1[2]
+   recall_rain1[k]=recall1[1]
+
+
+
+ }
> mean(accuracy1)
[1] 0.7765789
> mean(precision_rain1)
[1] 0.7504617
> mean(precision_rainThunderstorm1)
[1] 0.7881519
> mean(precision_snow1)
[1] 0.8908958
> mean(recall_rain1)
[1] 0.8143243
> mean(recall_rainThunderstorm1)
[1] 0.7182759
> mean(recall_snow1)
[1] 0.806
> 
```

Model	Tuning Parameters	Accuracy	Precision Snow	Precision Rain	Precision Rain Thunderstorm	Recall Snow	Recall Rain	Recall ThunderStorm
SVM	`kernel="radial”,cost=1, gamma=0.04545455,coef.0=0,epsilon=0.1`	0.7736842	0.8981612	0.7938852	0.7231856	0.887	0.7278378	0.7931034
SVM	`kernel="linear"`	0.7638158	0.9012634	0.7856708	0.7071105	0.899	0.7132432	0.7817241
SVM	`kernel="radial",cost=1,gamma=0`	0.7765789	0.8908958	0.7504617	0.7881519	0.806	0.8143243	0.7182759
SVM	`kernel="sigmoid”,cost=1, gamma=0.04545455,coef.0=0,epsilon=0.1`	0.7515789	0.8825927	0.7829367	0.6924296	0.868	0.68	0.8027586

Comparing to the other models using in Assignment 2

Model	Accuracy	Precision Snow	Precision Rain	Precision Rain Thunderstorm	Recall Snow	Recall Rain	Recall ThunderStorm
LDA	0.9026316	0.6407459	0.9115871	0.9906168	0.911675	0.902705	0.9906152

QDA	0.7389474	0.945	0.697027	0.7213793	0.7950919	0.7514844	0.7115056
KNN (K=5)	0.745	0.895	0.7305405	0.7117241	0.9098042	0.7444542	0.701065

Discussion Note:

1. From Accuracy Results, we see that SVM model performs better than QDA and KNN, but LDA outperforms SVM too.
2. From Precision of Snow Results, SVM does better than LDA and KNN
3. From Precision of Rain Results, SVM does better than QDA and KNN, but less than LDA
4. From Precision of thunderstorm Results, SVM does better than QDA and KNN, but less than LDA.
5. From Recall of Snow Results, SVM does better than QDA and KNN
6. From Recall of Rain Results, SVM does better than QDA and KNN, but less than LDA
7. From Recall of thunderstorm Results, SVM does better than QDA and KNN, but less than LDA.
8. So, overall if we compare performance with respect to classifiers, LDA>SVM>KNN>QDA.

3_2. [45 points] Consider the time series on Milk production data milk-production(1).csv

it shows cow milk production per pound from 1962 to 1975.

a. Try at least three different values for window size with simple moving average (SMA) for forecasting

b. Apply exponential moving average using HoltWinters for forecasting

c. For the above, discuss how the forecasting differs in terms of MAD and MFE and why one approach or the other is better.

PROGRAM

```
#import the dataset and make some changes
library(readr)
milk_production <- read_csv("C:/Users/bvkka/Desktop/ISL-Deep Medhi/assignment3/milk-production(1).csv")
```

```
> milk_production
# A tibble: 168 x 3
    Month `Pounds_per_Cow" \r\n"1962-01` `589`
    <chr>                          <int> <chr>
 1 1962-01                           589  <NA>
 2 1962-02                           561  <NA>
 3 1962-03                           640  <NA>
 4 1962-04                           656  <NA>
 5 1962-05                           727  <NA>
 6 1962-06                           697  <NA>
 7 1962-07                           640  <NA>
 8 1962-08                           599  <NA>
 9 1962-09                           568  <NA>
10 1962-10                           577  <NA>
# ... with 158 more rows
>
```

```r
milk_production<-milk_production[,2]
```

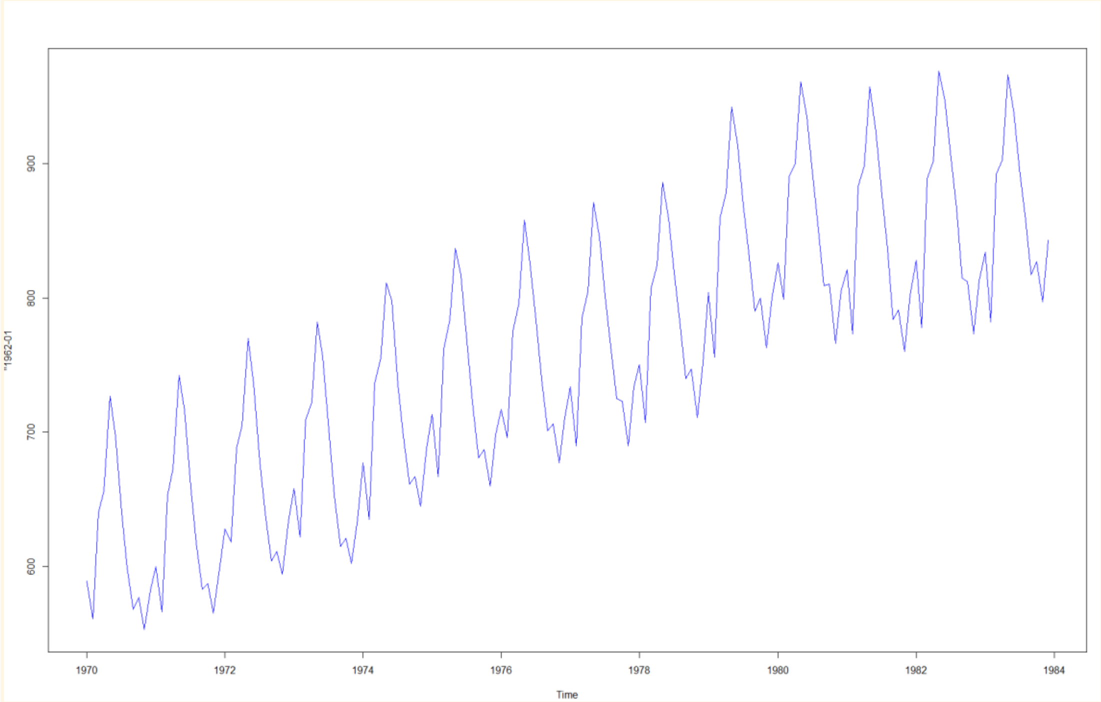
```
> milk_production
# A tibble: 168 x 1
   `Pounds_per_Cow" \r\n"1962-01`
                             <int>
 1                             589
 2                             561
 3                             640
 4                             656
 5                             727
 6                             697
 7                             640
 8                             599
 9                             568
10                             577
# ... with 158 more rows
>
```

```r
milk_production_timeseries<-ts(milk_production)

#contains monthly milk productions for January 1970-Decemeber 1983
mp_TS<-ts(milk_production,frequency = 12,start=c(1970,1))
```

```
> mp_TS
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1970 589 561 640 656 727 697 640 599 568 577 553 582
1971 600 566 653 673 742 716 660 617 583 587 565 598
1972 628 618 688 705 770 736 678 639 604 611 594 634
1973 658 622 709 722 782 756 702 653 615 621 602 635
1974 677 635 736 755 811 798 735 697 661 667 645 688
1975 713 667 762 784 837 817 767 722 681 687 660 698
1976 717 696 775 796 858 826 783 740 701 706 677 711
1977 734 690 785 805 871 845 801 764 725 723 690 734
1978 750 707 807 824 886 859 819 783 740 747 711 751
1979 804 756 860 878 942 913 869 834 790 800 763 800
1980 826 799 890 900 961 935 894 855 809 810 766 805
1981 821 773 883 898 957 924 881 837 784 791 760 802
1982 828 778 889 902 969 947 908 867 815 812 773 813
1983 834 782 892 903 966 937 896 858 817 827 797 843
>
```

```r
#plotting time series

plot.ts(milk_production_TS)
lines(mp_TS,col="blue")
```

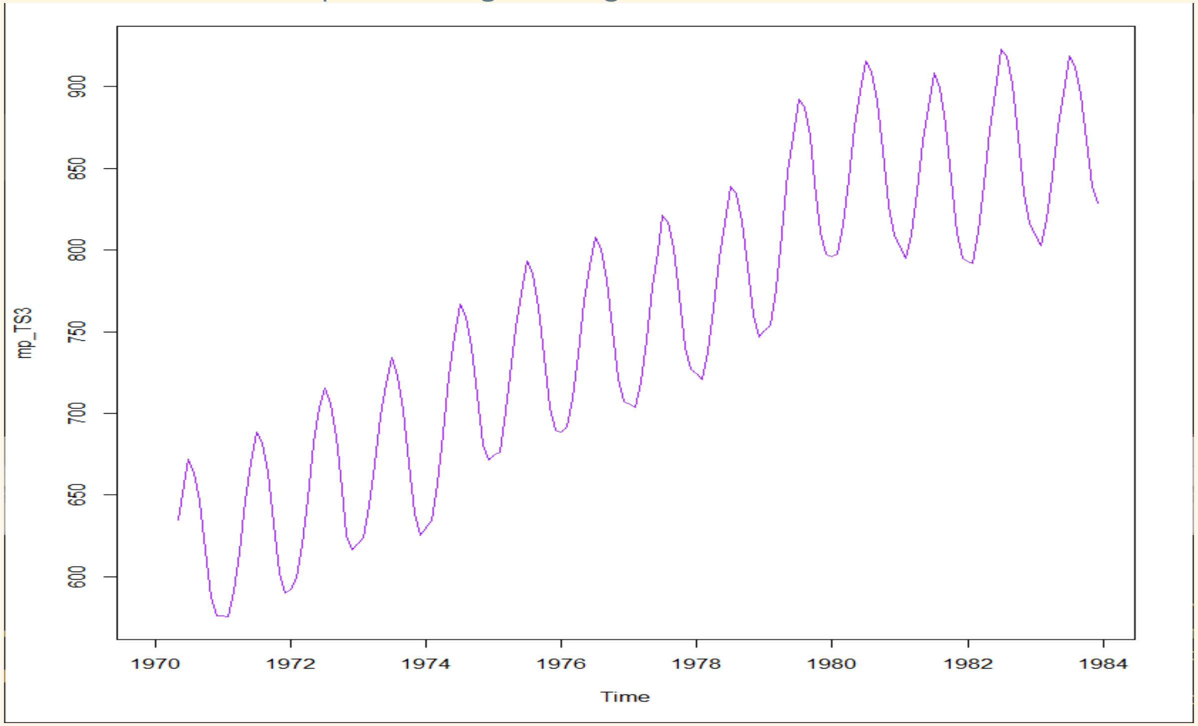
Plot of Timeseries



Question #2 – Part A

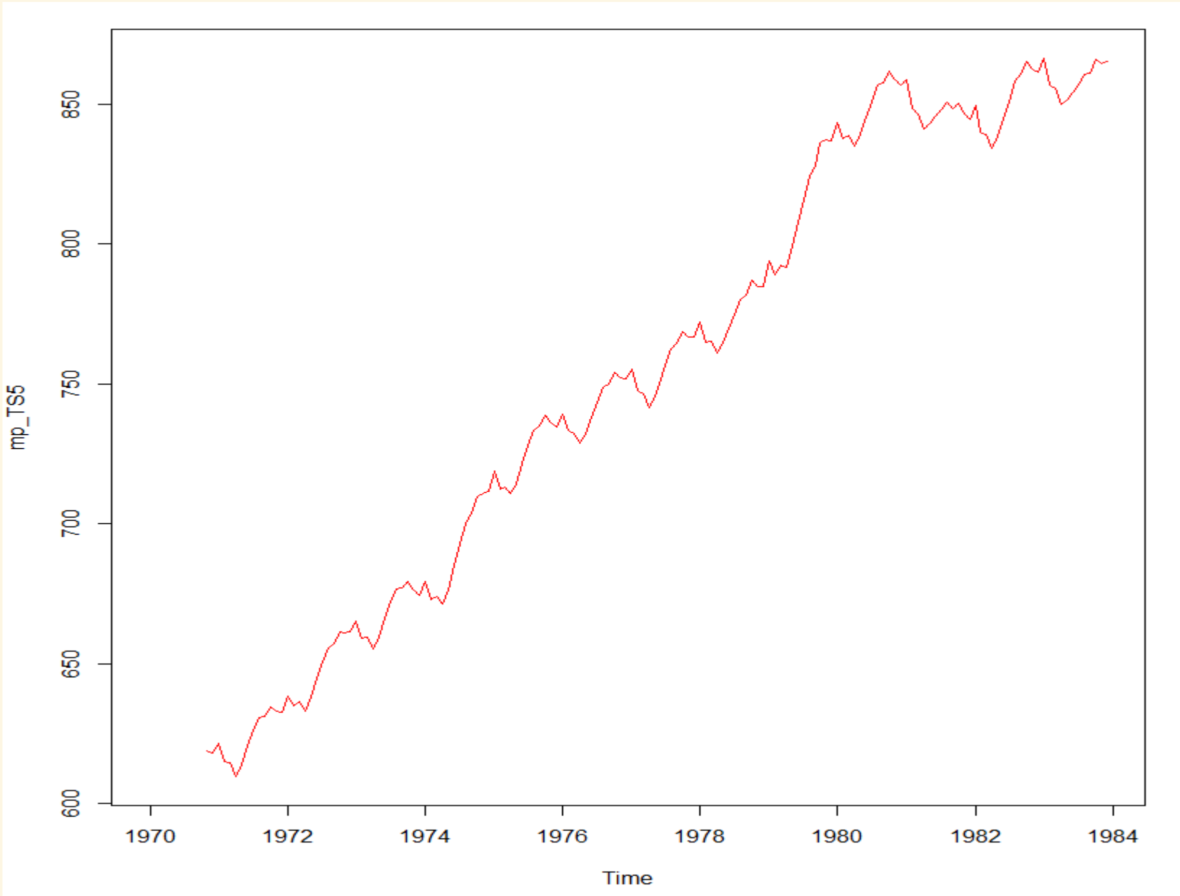
```
#***Simple Moving Average(SMA)***# ->it is used to smooth time series data
#install.packages("TTR")
library("TTR")
mp_TS5<-SMA(milk_production_TS,n=5)
plot.ts(mp_TS5)
lines(mp_TS5,col="purple")
```

Simple Moving Average: Window size 5



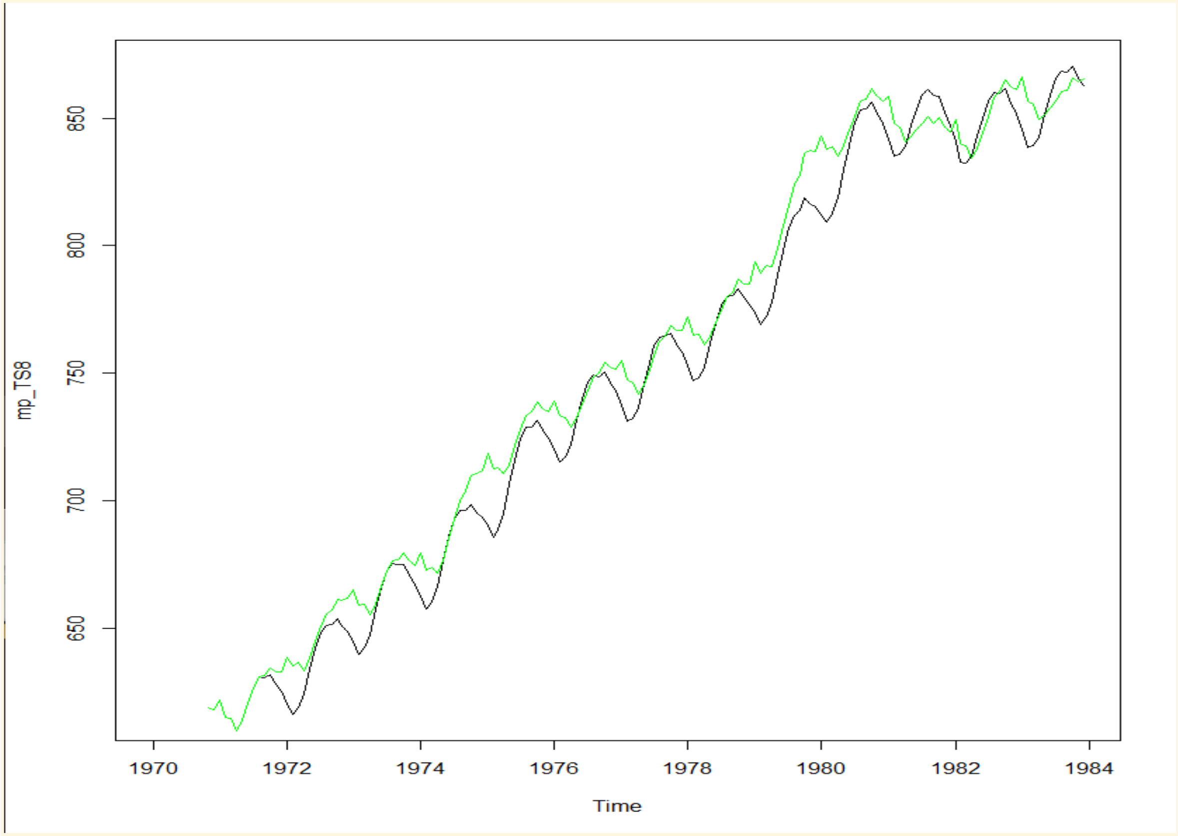
```
mp_TS5<-SMA(milk_production_TS,n=11)
plot.ts(mp_TS11)
lines(mp_TS11,col="red")
```

Simple Moving Average: Window size 11

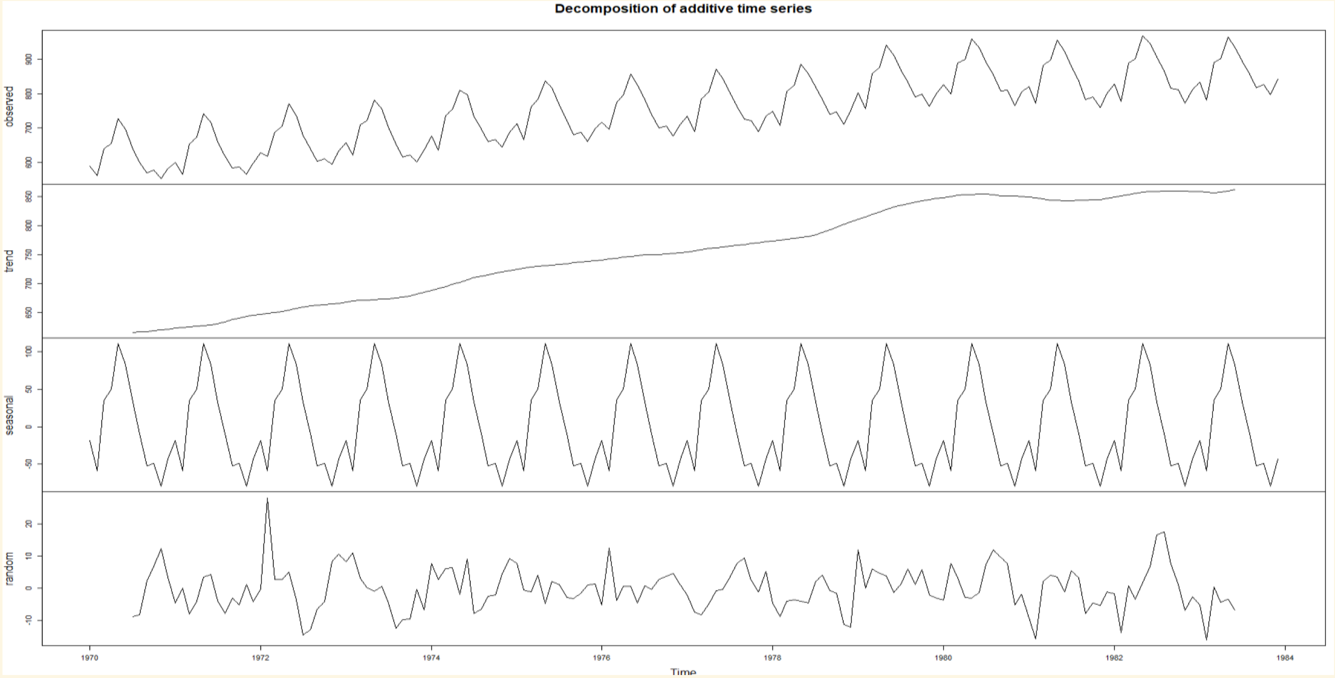


```
mp_TS20<-SMA(milk_production_TS,n=20)
plot.ts(mp_TS20)
lines(mp_TS20,col="green")
```

Simple Moving Average: Window size 20



```r
#To estimate the trend component and seasonal component of a seasonal time series that can be described using an additive model, we can use the "decompose ()" function in R. This function estimates the trend, seasonal, and irregular components of a time series that can be described using an additive model###

mp_decompose=decompose(milk_production_TS)
plot(mp_decompose) #The plot above shows the original time series (top), the estimated trend component (second from top), the estimated seasonal component (third from top), and the estimated irregular component (bottom)##
```

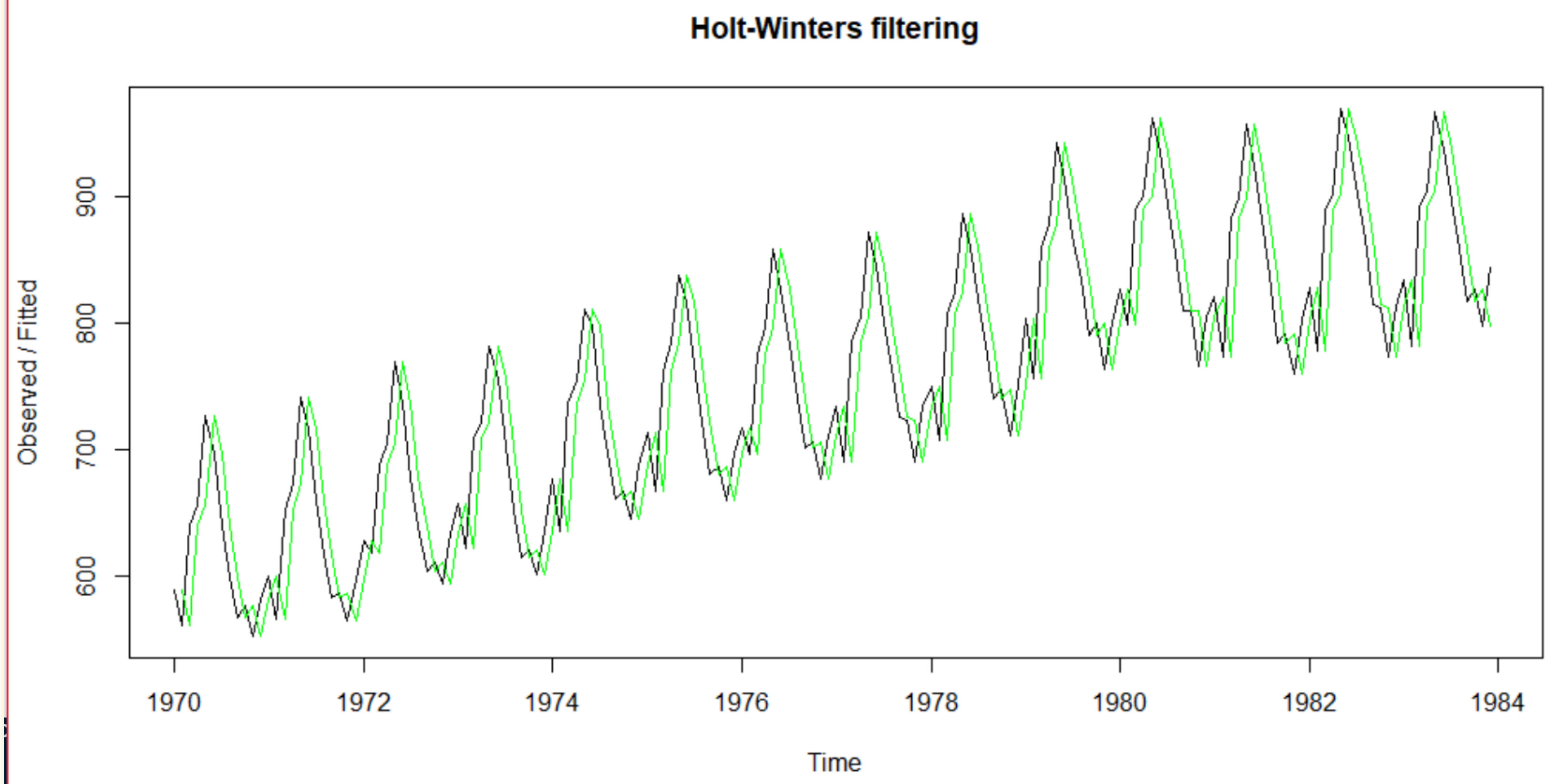


Discussion and observations:

- The moving average model uses the last t periods to predict demand in period t+1.
- SMA is an arithmetic moving average calculated by adding the actual forecasts for many time periods and then dividing this total by the number of time periods.
- In the above program, simple moving average of milk-production dataset has been calculated, with three different windows, 5,11,20. I chose this window sizes because,
- When window size is 5, the graph is changing, but we can see that it is changing in in equally distributed patterns. Whereas when window size is 11, the graph is increasing but with some non-linearity. When the window size is 20, the graph is almost linearly increasing.

*****Question #2 – Part B*****

```
#****Forecasts suing Exponential Smoothing***###

mp_exp=HoltWinters(mp_TS,beta=FALSE,gamma = FALSE)
plot(mp_exp) #The plot shows the original time series in black, and the forecasts as a red line. The time series of forecasts is much smoother than the time series of the original data here.
lines(mp_exp$fitted[,1],col="red")
```



Holt-Winters filtering
Observed / Fitted
600
700
800
900
1970
1972
1974
1976
1978
1980
1982
1984
Time

```
mp_exp
```

```
> mp_exp
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = mp_TS)

Smoothing parameters:
 alpha: 0.68933
 beta : 0
 gamma: 0.8362592

Coefficients:
          [,1]
a   885.775547
b     1.278118
s1  -16.743296
s2  -59.730034
s3   47.492731
s4   56.203890
s5  115.537545
s6   84.554817
s7   39.580306
s8   -4.702033
s9  -54.554684
s10 -51.582594
s11 -85.953466
s12 -42.907363
```

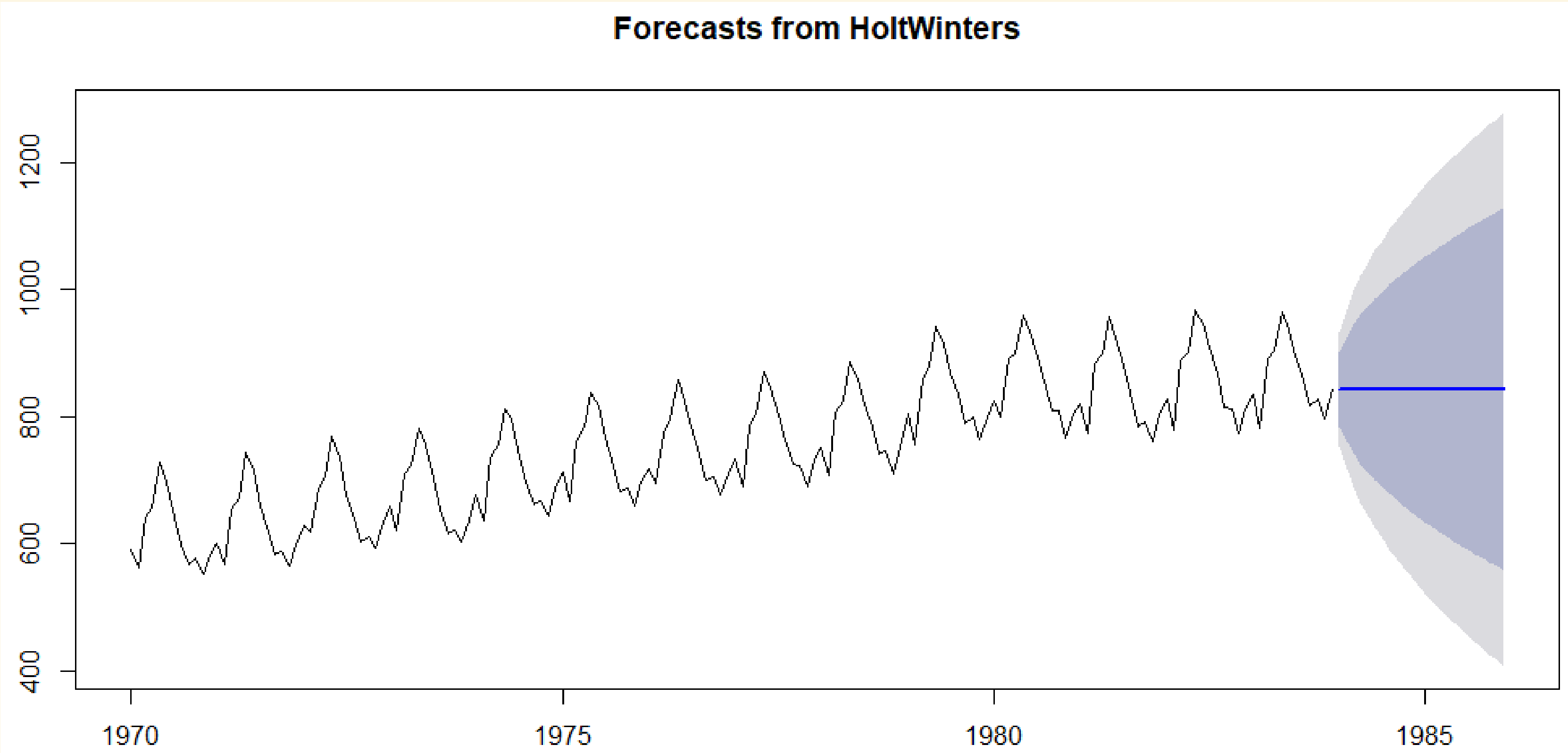
```r
#install.packages("TTR")
library(forecast)
forecast_holtwinter=forecast(mp_exp)
forecast_holtwinter
```

```
> forecast_holtwinter
         Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
Jan 1984        842.997 784.7725  901.2214 753.9504  932.0436
Feb 1984        842.997 760.6579  925.3360 717.0702  968.9237
Mar 1984        842.997 742.1537  943.8402 688.7705  997.2234
Apr 1984        842.997 726.5539  959.4401 664.9126 1021.0813
May 1984        842.997 712.8100  973.1839 643.8932 1042.1007
Jun 1984        842.997 700.3846  985.6093 624.8902 1061.1037
Jul 1984        842.997 688.9583  997.0356 607.4152 1078.5788
Aug 1984        842.997 678.3229 1007.6710 591.1497 1094.8442
Sep 1984        842.997 668.3339 1017.6600 575.8729 1110.1211
Oct 1984        842.997 658.8861 1027.1079 561.4236 1124.5703
Nov 1984        842.997 649.8999 1036.0940 547.6806 1138.3134
Dec 1984        842.997 641.3138 1044.6801 534.5492 1151.4447
Jan 1985        842.997 633.0786 1052.9154 521.9545 1164.0394
Feb 1985        842.997 625.1544 1060.8395 509.8356 1176.1584
Mar 1985        842.997 617.5086 1068.4853 498.1423 1187.8517
Apr 1985        842.997 610.1136 1075.8803 486.8326 1199.1613
May 1985        842.997 602.9464 1083.0475 475.8713 1210.1226
Jun 1985        842.997 595.9870 1090.0069 465.2278 1220.7661
Jul 1985        842.997 589.2184 1096.7755 454.8761 1231.1178
Aug 1985        842.997 582.6257 1103.3682 444.7935 1241.2004
Sep 1985        842.997 576.1959 1109.7981 434.9599 1251.0340
Oct 1985        842.997 569.9174 1116.0766 425.3578 1260.6362
Nov 1985        842.997 563.7800 1122.2139 415.9715 1270.0224
Dec 1985        842.997 557.7747 1128.2192 406.7871 1279.2068
```

```r
Accuracy(forecast_holt_winters)
```

```
> accuracy(forecast_holtwinter)
                  ME     RMSE     MAE        MPE     MAPE     MASE       ACF1
Training set 1.52104 45.32207 39.0064 0.03434009 5.159602 1.832279 0.02263573
>
```

```r
plot(forecast_holt_winters)
```



Discussion and observations:

- The main idea is an exponential smoothing is that the prediction mostly depends on most recent observation and on the error of the latest forecast.
- If the time series can be described using an additive model with increasing or decreasing trend and seasonality, Holt-Winters exponential smoothing to make short-term forecasts.
- Smoothing is controlled by three parameters: alpha, beta, and gamma, for the estimates of the level, slope b of the trend component, and the seasonal component, respectively, at the current time point. The coefficients alpha, beta and gamma, usually ranges between 0 and 1.
-
```
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = mp_TS)

Smoothing parameters:
 alpha: 0.68933
 beta : 0
 gamma: 0.8362592
```
- In the above forecast, α is 0.68933 indicating that the estimate of the level at the current time is based upon observations in the more distant past as well as some recent observations.
- The value of beta is 0, which indicates that the estimate of the slope b of the trend component is not updated over the time series, and instead is set equal to its initial value. This makes a conclusion that as the level changes over the time series, slope b of the trend component remains almost constant.
- In contrast, the value of gamma (0.8362592) is high, indicating that the estimate of the seasonal component at the current time point is just based upon very recent observations.

***Question #2 – Part C**

```
## Discuss how the forecasting differs in terms of MAD and MFE and##
                 ##why one approach or the other is better##
count=168
x=mean(mp_TS3[-(1:7)])
for (k in 8:count) {
  mean_abs_dev3=mean(abs(mp_TS3[k]-x))
  mfe3=mean(mp_TS3[k]-x)
  mad3=mad(mp_TS3[k], centre, constant = 1.4826, na.rm = FALSE,  low = FALSE, high = FALSE)
}
y=mean(mp_TS5[-(1:12)])
for (k in 13:count) {
  mean_abs_dev4=mean(abs(mp_TS5[k]-y))
  mfe4=mean(mp_TS5[k]-y)
  mad4=mad(mp_TS5[k], centre, constant = 1.4826, na.rm = FALSE,  low = FALSE, high = FALSE)
}
z=mean(mp_TS8[-(1:23)])
for (k in 24:count) {
  mean_abs_dev5=mean(abs(mp_TS8[k]-z))
  mfe5=mean(mp_TS8[k]-z)
  mad(mp_TS8[k], centre, constant = 1.4826, na.rm = FALSE,  low = FALSE, high = FALSE)
}
```

```
> mfe3
[1] -93.8795
> mfe4
[1] -135.6247
> mfe5
[1] -135.1197
> mean_abs_dev3
[1] 93.8795
> mean_abs_dev4
[1] 135.6247
> mean_abs_dev5
[1] 135.1197
>
```

```
> accuracy(forecast_holtwinter)
                    ME     RMSE     MAE        MPE     MAPE     MASE       ACF1
Training set 1.52104 45.32207 39.0064 0.03434009 5.159602 1.832279 0.02263573
> 
```

Observations:

- According to the above values of MAD and MFE of Simple moving average (three different windows) and MAE of Holt-winter forecasting, I find Holt-winters exponential smoothing and forecasting more precise than SMA.