Assignment 3

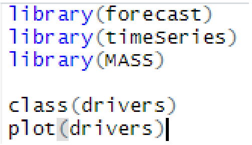
By Riya Gupta (17BCS026)

Predictive Modeling: Time Series Analysis

1. Apply Time Series methods to analyze the data in any one time series data set other than AirPassengers or Atmospheric CO2. (Hint: You may find and choose your own data set, but choose one that gives reasonably good results when you analyze it using the methods of time series analysis)

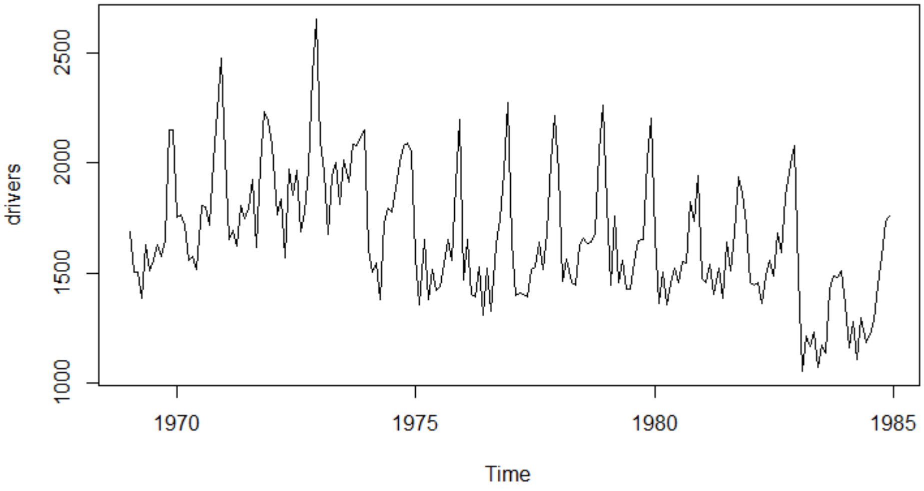
Data: **Deaths of Car Drivers in Great Britain 1969-84**
drivers{MASS}
Description: A regular time series giving the monthly totals of car drivers in Great Britain killed or seriously injured Jan 1969 to Dec 1984. Compulsory wearing of seat belts was introduced on 31 Jan 1983
Source: Harvey, A.C. (1989) Forecasting, Structural Time Series Models and the Kalman Filter. Cambridge University Press, pp. 519–523.

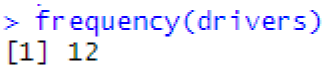
Importing libraries *forecast* and *timeSeries*. The library *MASS* is for 'drivers' data.



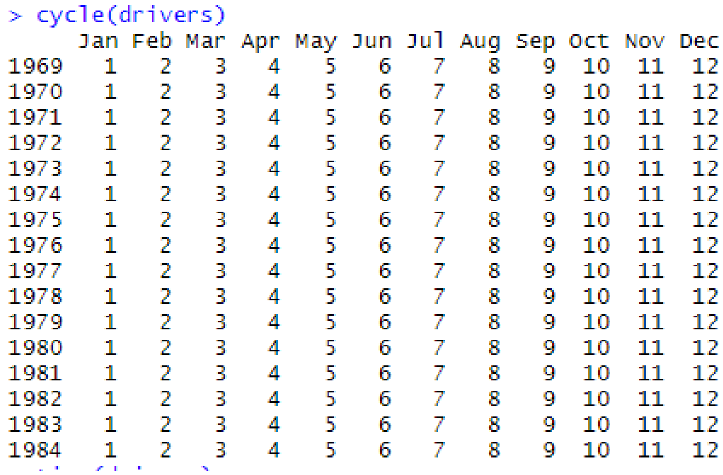
```
library(forecast)
library(timeSeries)
library(MASS)

class(drivers)
plot(drivers)
```

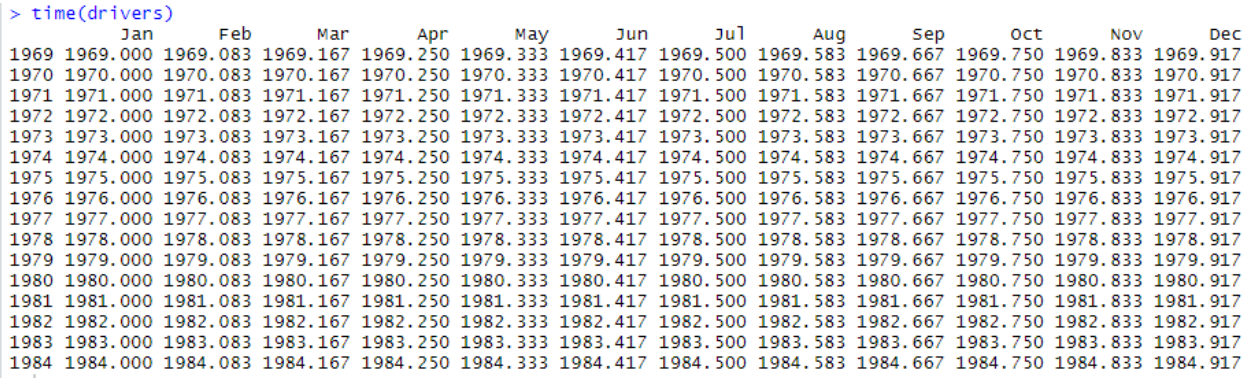




```
> frequency(drivers)
[1] 12
```



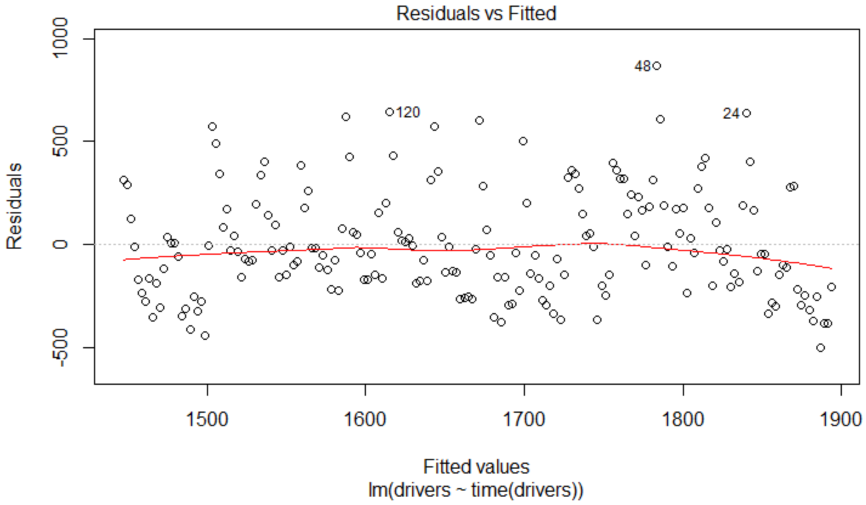
```
> cycle(drivers)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1969   1   2   3   4   5   6   7   8   9  10  11  12
1970   1   2   3   4   5   6   7   8   9  10  11  12
1971   1   2   3   4   5   6   7   8   9  10  11  12
1972   1   2   3   4   5   6   7   8   9  10  11  12
1973   1   2   3   4   5   6   7   8   9  10  11  12
1974   1   2   3   4   5   6   7   8   9  10  11  12
1975   1   2   3   4   5   6   7   8   9  10  11  12
1976   1   2   3   4   5   6   7   8   9  10  11  12
1977   1   2   3   4   5   6   7   8   9  10  11  12
1978   1   2   3   4   5   6   7   8   9  10  11  12
1979   1   2   3   4   5   6   7   8   9  10  11  12
1980   1   2   3   4   5   6   7   8   9  10  11  12
1981   1   2   3   4   5   6   7   8   9  10  11  12
1982   1   2   3   4   5   6   7   8   9  10  11  12
1983   1   2   3   4   5   6   7   8   9  10  11  12
1984   1   2   3   4   5   6   7   8   9  10  11  12
```

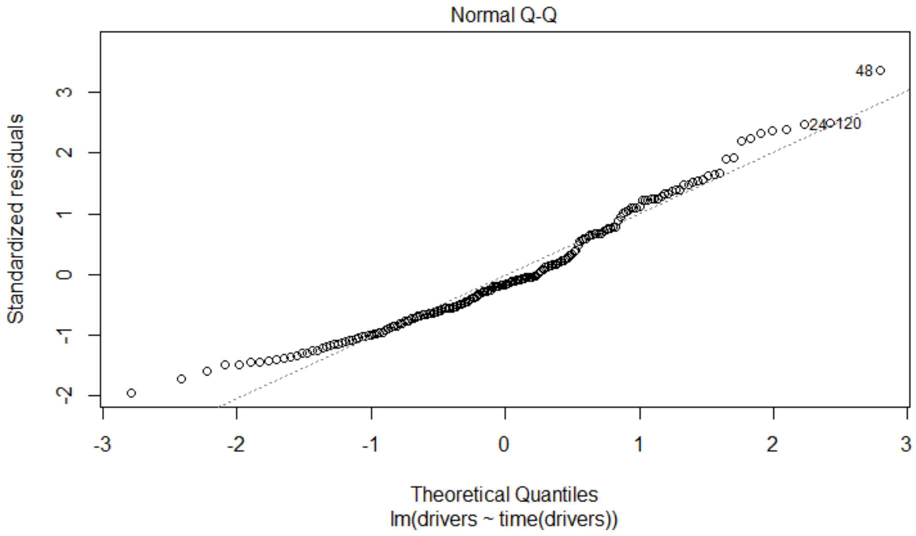


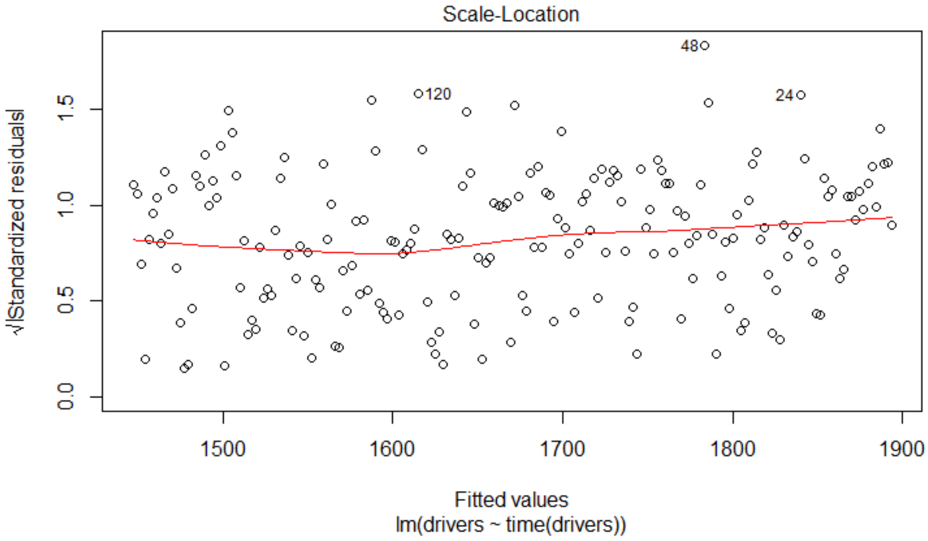
```
> time(drivers)
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
1969 1969.000 1969.083 1969.167 1969.250 1969.333 1969.417 1969.500 1969.583 1969.667 1969.750 1969.833 1969.917
1970 1970.000 1970.083 1970.167 1970.250 1970.333 1970.417 1970.500 1970.583 1970.667 1970.750 1970.833 1970.917
1971 1971.000 1971.083 1971.167 1971.250 1971.333 1971.417 1971.500 1971.583 1971.667 1971.750 1971.833 1971.917
1972 1972.000 1972.083 1972.167 1972.250 1972.333 1972.417 1972.500 1972.583 1972.667 1972.750 1972.833 1972.917
1973 1973.000 1973.083 1973.167 1973.250 1973.333 1973.417 1973.500 1973.583 1973.667 1973.750 1973.833 1973.917
1974 1974.000 1974.083 1974.167 1974.250 1974.333 1974.417 1974.500 1974.583 1974.667 1974.750 1974.833 1974.917
1975 1975.000 1975.083 1975.167 1975.250 1975.333 1975.417 1975.500 1975.583 1975.667 1975.750 1975.833 1975.917
1976 1976.000 1976.083 1976.167 1976.250 1976.333 1976.417 1976.500 1976.583 1976.667 1976.750 1976.833 1976.917
1977 1977.000 1977.083 1977.167 1977.250 1977.333 1977.417 1977.500 1977.583 1977.667 1977.750 1977.833 1977.917
1978 1978.000 1978.083 1978.167 1978.250 1978.333 1978.417 1978.500 1978.583 1978.667 1978.750 1978.833 1978.917
1979 1979.000 1979.083 1979.167 1979.250 1979.333 1979.417 1979.500 1979.583 1979.667 1979.750 1979.833 1979.917
1980 1980.000 1980.083 1980.167 1980.250 1980.333 1980.417 1980.500 1980.583 1980.667 1980.750 1980.833 1980.917
1981 1981.000 1981.083 1981.167 1981.250 1981.333 1981.417 1981.500 1981.583 1981.667 1981.750 1981.833 1981.917
1982 1982.000 1982.083 1982.167 1982.250 1982.333 1982.417 1982.500 1982.583 1982.667 1982.750 1982.833 1982.917
1983 1983.000 1983.083 1983.167 1983.250 1983.333 1983.417 1983.500 1983.583 1983.667 1983.750 1983.833 1983.917
1984 1984.000 1984.083 1984.167 1984.250 1984.333 1984.417 1984.500 1984.583 1984.667 1984.750 1984.833 1984.917
```

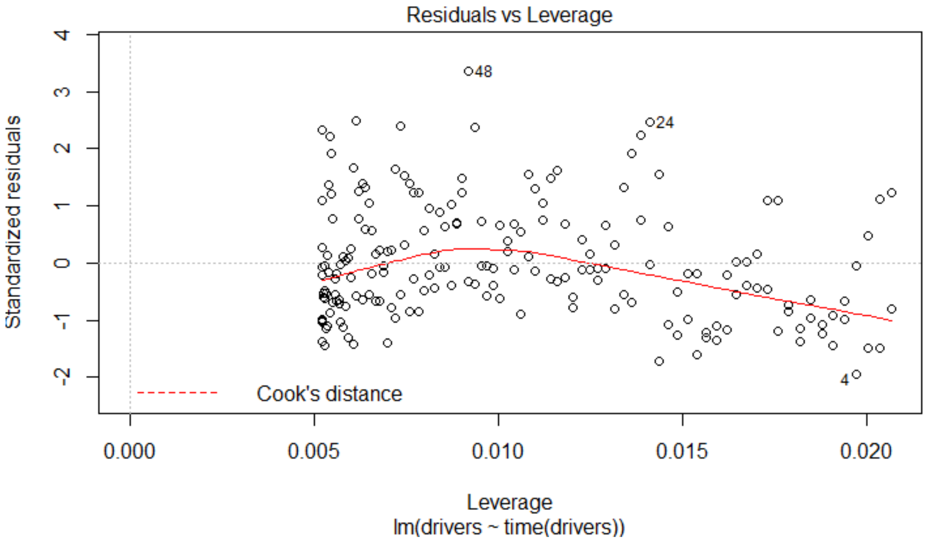
2. Build a timeSeries object with the data.

m <- lm(drivers~time(drivers))
plot(m)

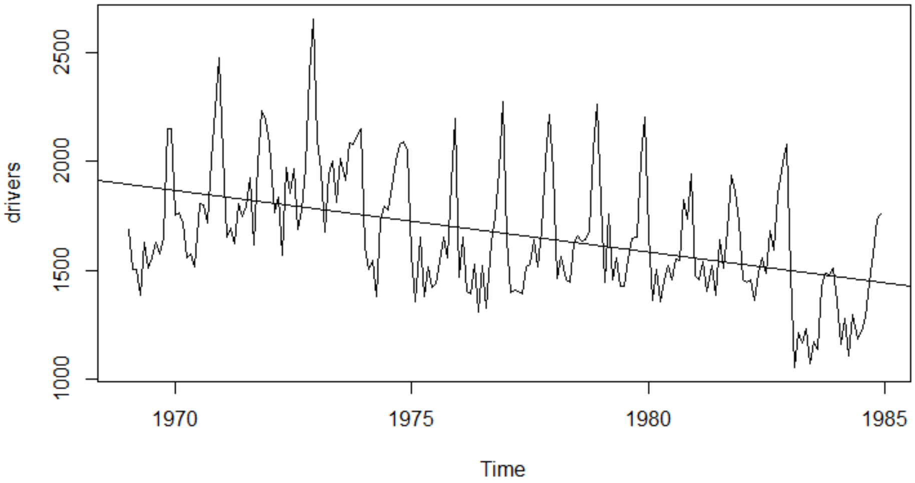



Normal Q-Q
Standardized residuals
-2
-1
0
1
2
3
48
24
120
-3
-2
-1
0
1
2
3
Theoretical Quantiles
lm(drivers ~ time(drivers))


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
48
120
24
1500
1600
1700
1800
1900
Fitted values
lm(drivers ~ time(drivers))

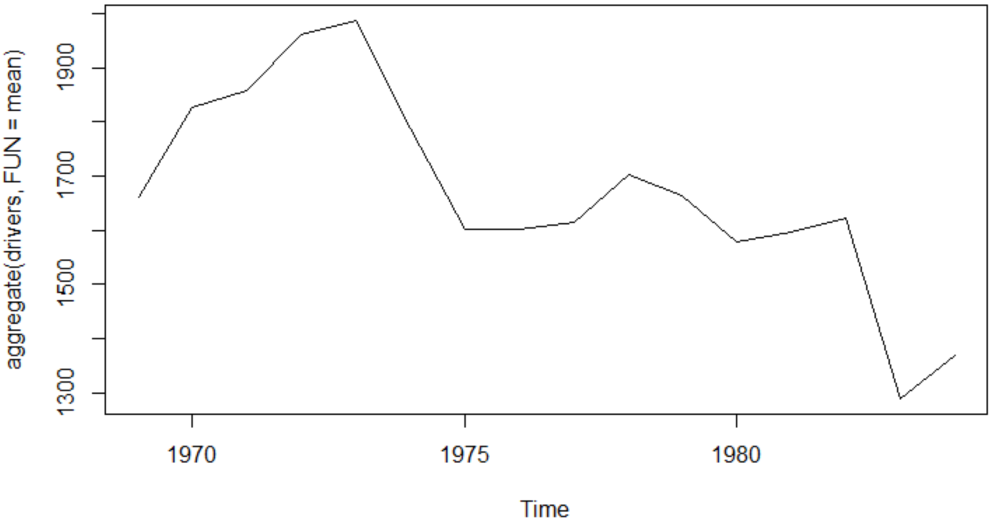

Residuals vs Leverage
Standardized residuals
-2
-1
0
1
2
3
4
48
24
4
Cook's distance
0.000
0.005
0.010
0.015
0.020
Leverage
lm(drivers ~ time(drivers))

```
plot(drivers)
abline(m)
```



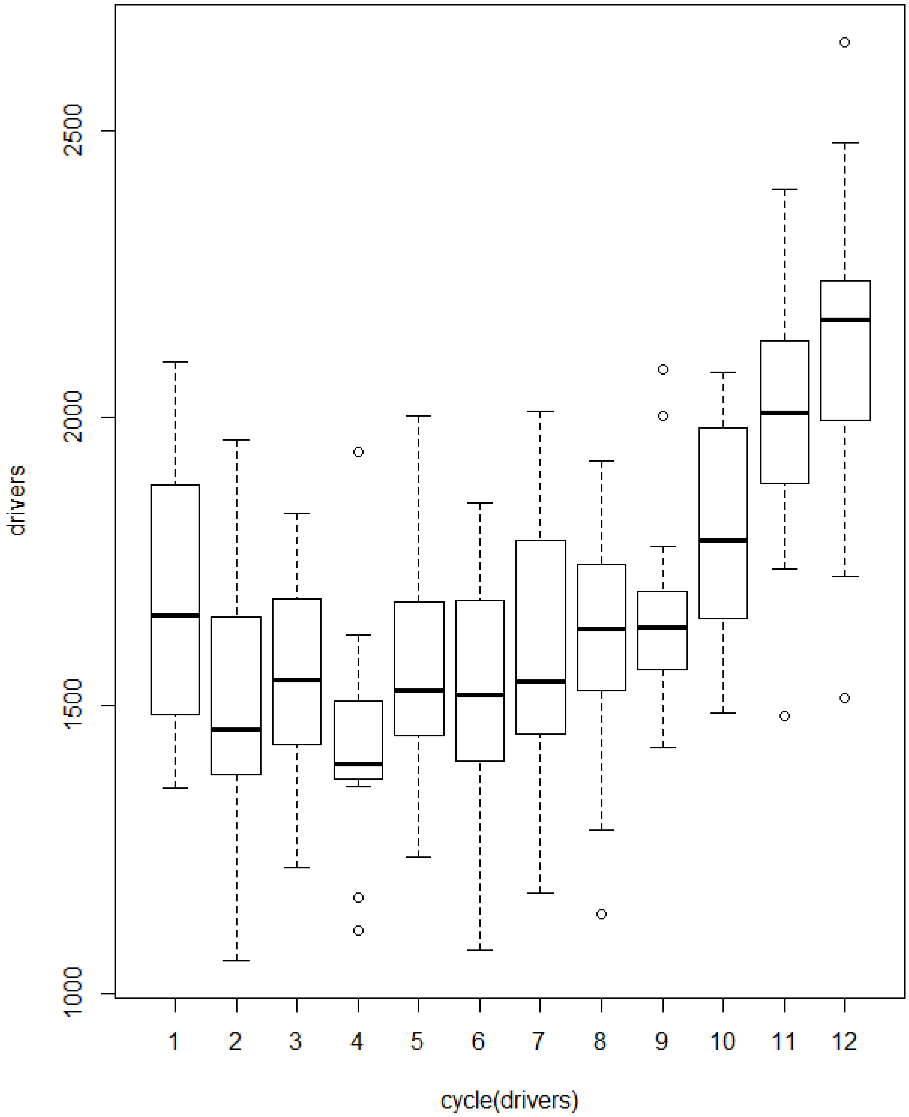
3. Plot the yearly (or other suitable periodic) mean values

```
plot(aggregate(drivers, FUN=mean))
```



4. Plot the monthly (or other suitable periodic) boxplots

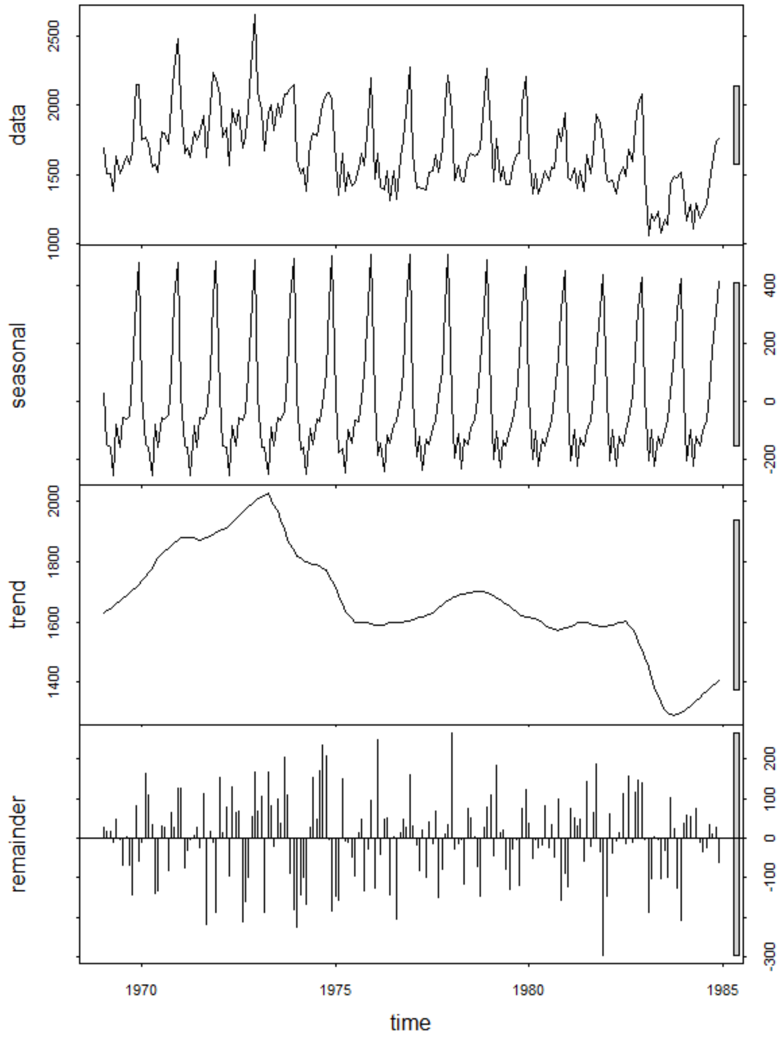
```
boxplot(drivers ~ cycle(drivers))
```



5. Decompose the time series using the stl function. What type of trend does it show?

d <- stl(drivers, s.window = 12)

plot(d)



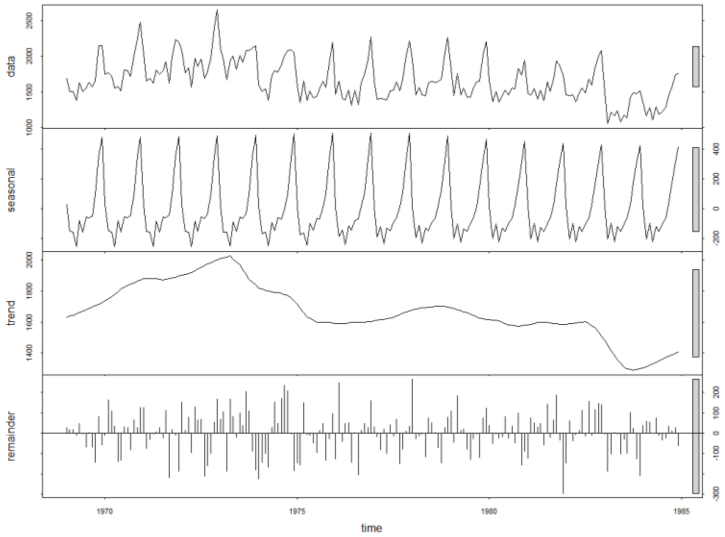
We know from the plots that the trend is not specific. It grows till around 1973 then decreases with ups and downs.

6. What type of seasonality?

Seasonality is monthly variation (sine format). As shown in the last question, in a year the trend first decreased then increased rapidly.

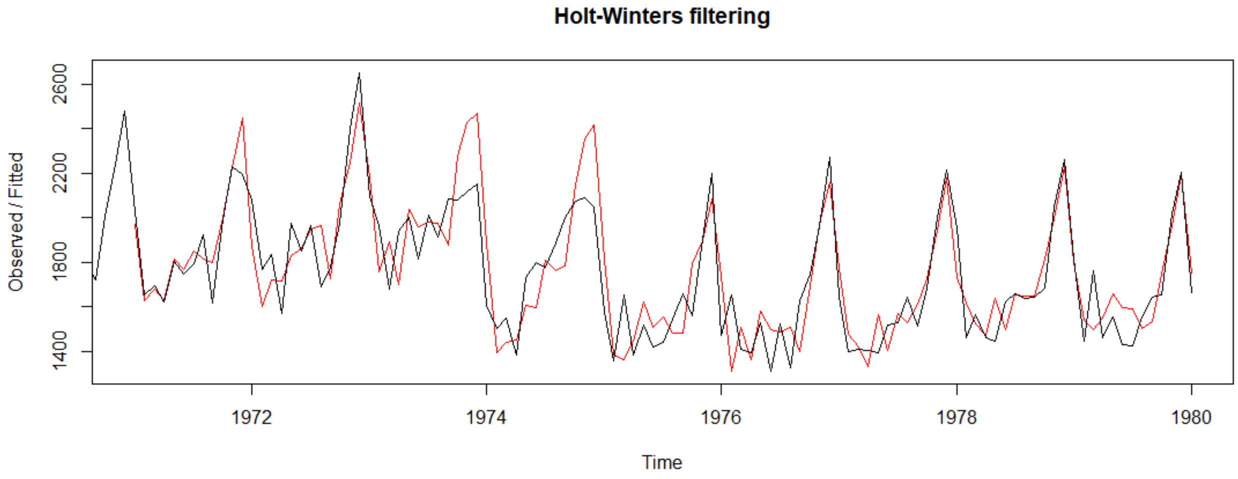
7. How is the residue after you remove trend and seasonality?

Remainder graph is obtained as Residue on removing trend and seasonality.



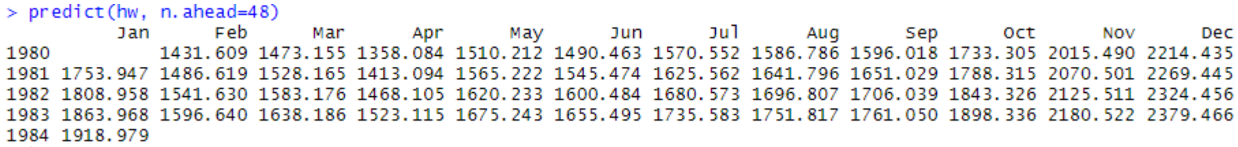
8. Build a model of the data using the HoltWinters method for the period upto about 75% of the data (e.g., up to December 2015 if it were for the CO2 data set). Use suitable values of alpha, beta and gamma.

Drivers <- window(drivers,start=1970,end=1980)
hw <- HoltWinters(Drivers, alpha='0.5', beta=NULL, seasonal='additive')
plot(hw)



9. Predict the values for the next 25% of the time (e.g., for the CO2 data set, all of 2016 and the first 3 months of 2017).

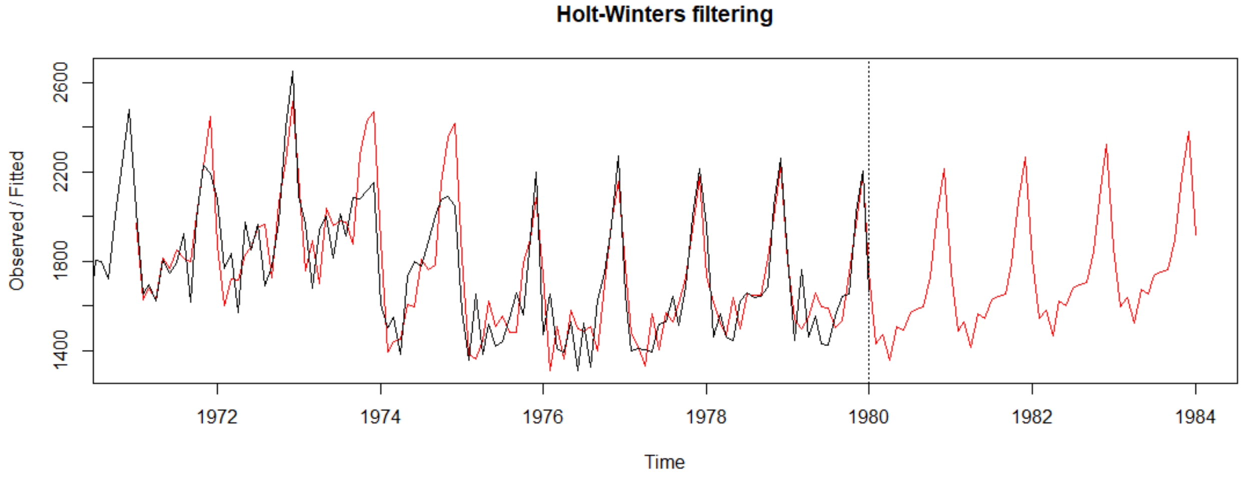
predict(hw, n.ahead=48)



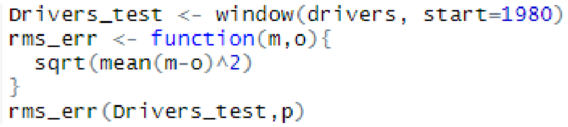
```
> predict(hw, n.ahead=48)
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
1980          1431.609 1473.155 1358.084 1510.212 1490.463 1570.552 1586.786 1596.018 1733.305 2015.490 2214.435
1981 1753.947 1486.619 1528.165 1413.094 1565.222 1545.474 1625.562 1641.796 1651.029 1788.315 2070.501 2269.445
1982 1808.958 1541.630 1583.176 1468.105 1620.233 1600.484 1680.573 1696.807 1706.039 1843.326 2125.511 2324.456
1983 1863.968 1596.640 1638.186 1523.115 1675.243 1655.495 1735.583 1751.817 1761.050 1898.336 2180.522 2379.466
1984 1918.979
```

10. Plot the predicted values along with the actual values to compare them.

p<-predict(hw, n.ahead=48)
plot(hw,p)



11. Compute the rms error between the predicted and actual values.

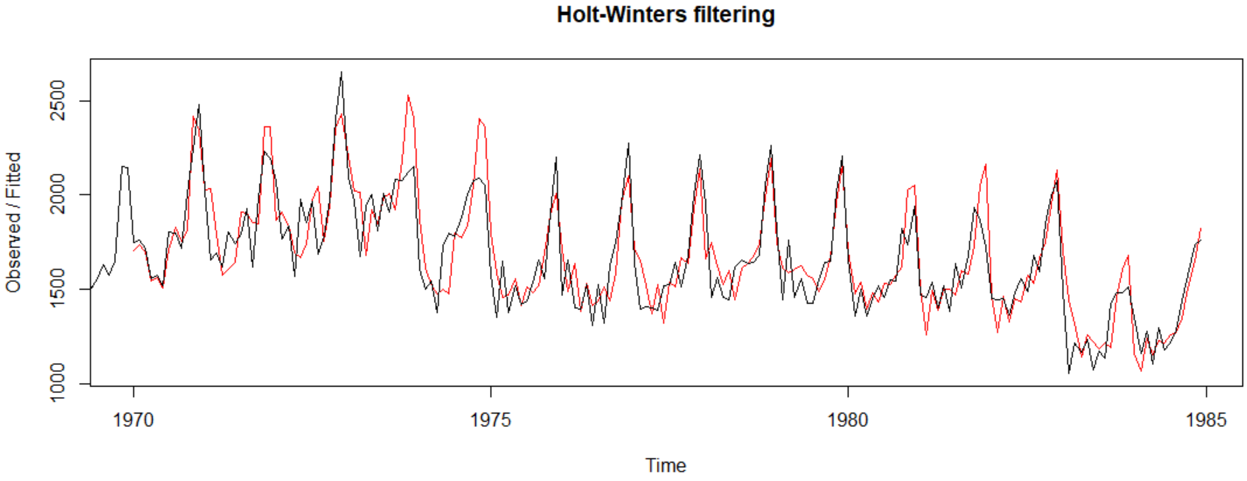


```
Drivers_test <- window(drivers, start=1980)
rms_err <- function(m,o){
  sqrt(mean(m-o)^2)
}
rms_err(Drivers_test,p)
```

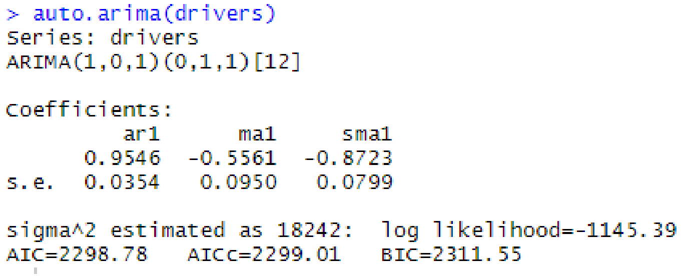
Output: 212.3328

12. Try to fine tune the model by changing alpha, beta and gamma. Are you able to improve the model (i.e., get a lower rms error)?

hw <- HoltWinters(drivers,alpha='0.5',beta=NULL, gamma=NULL, seasonal='additive')
plot(hw)



13. Build an ARIMA model for the period up to about 75% of the data (e.g., for the CO2 data, up to December 2015) using auto.arima()

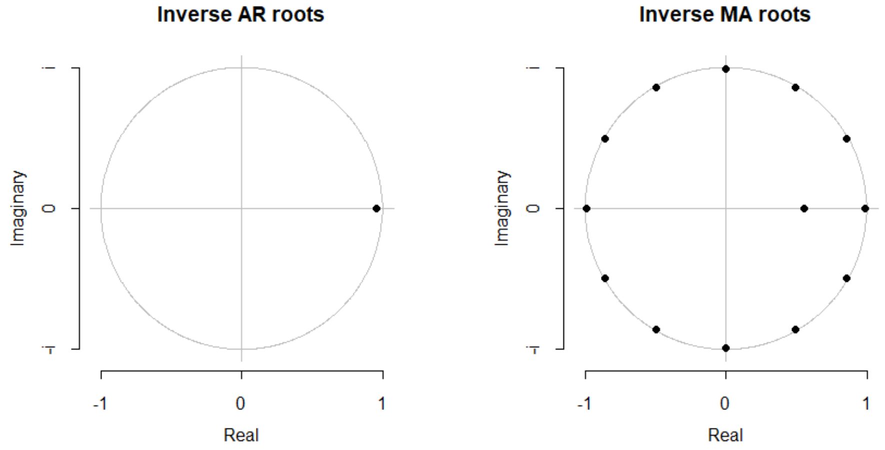


```
> auto.arima(drivers)
Series: drivers
ARIMA(1,0,1)(0,1,1)[12]

Coefficients:
         ar1      ma1     sma1
      0.9546  -0.5561  -0.8723
s.e.  0.0354   0.0950   0.0799

sigma^2 estimated as 18242:  log likelihood=-1145.39
AIC=2298.78   AICc=2299.01   BIC=2311.55
```

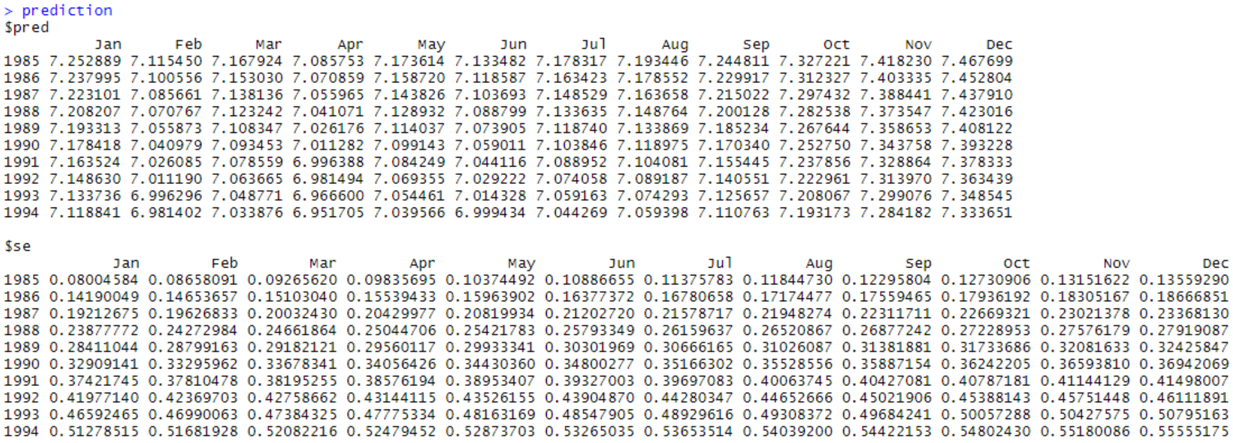
On plotting it we get,



14. Predict the values for the next 15 months (e.g., for the CO2 data, all of 2016 and the first 3 months of 2017).

data1 <- arima(log(drivers),c(0,1,1),seasonal=list(order=c(0,1,1),period=12))

prediction <- predict(data1, n.ahead=12*10)

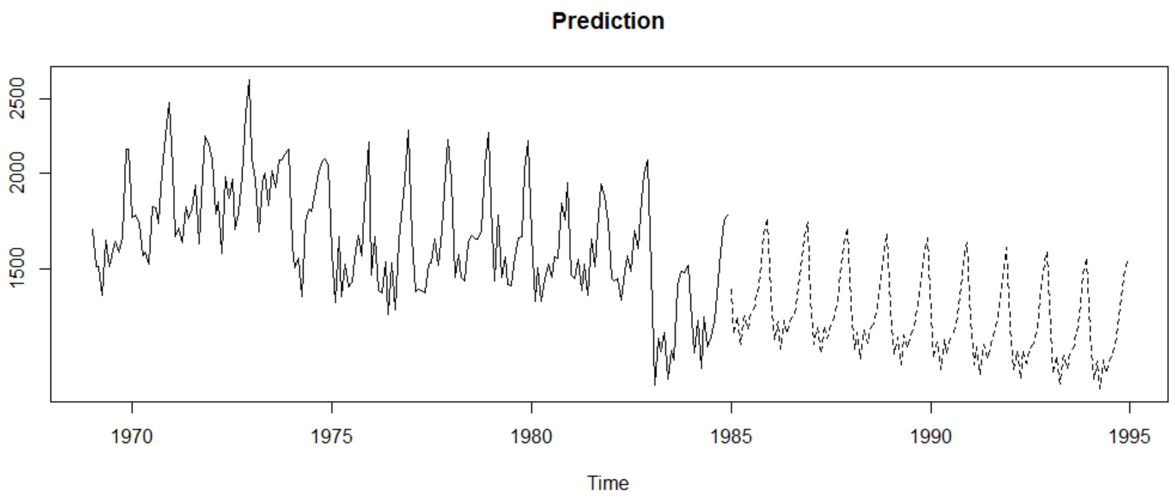


```
> prediction
$pred
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
1985 7.252889 7.115450 7.167924 7.085753 7.173614 7.133482 7.178317 7.193446 7.244811 7.327221 7.418230 7.467699
1986 7.237995 7.100556 7.153030 7.070859 7.158720 7.118587 7.163423 7.178552 7.229917 7.312327 7.403335 7.452804
1987 7.223101 7.085661 7.138136 7.055965 7.143826 7.103693 7.148529 7.163658 7.215022 7.297432 7.388441 7.437910
1988 7.208207 7.070767 7.123242 7.041071 7.128932 7.088799 7.133635 7.148764 7.200128 7.282538 7.373547 7.423016
1989 7.193313 7.055873 7.108347 7.026176 7.114037 7.073905 7.118740 7.133869 7.185234 7.267644 7.358653 7.408122
1990 7.178418 7.040979 7.093453 7.011282 7.099143 7.059011 7.103846 7.118975 7.170340 7.252750 7.343758 7.393228
1991 7.163524 7.026085 7.078559 6.996388 7.084249 7.044116 7.088952 7.104081 7.155445 7.237856 7.328864 7.378333
1992 7.148630 7.011190 7.063665 6.981494 7.069355 7.029222 7.074058 7.089187 7.140551 7.222961 7.313970 7.363439
1993 7.133736 6.996296 7.048771 6.966600 7.054461 7.014328 7.059163 7.074293 7.125657 7.208067 7.299076 7.348545
1994 7.118841 6.981402 7.033876 6.951705 7.039566 6.999434 7.044269 7.059398 7.110763 7.193173 7.284182 7.333651

$se
            Jan        Feb        Mar        Apr        May        Jun        Jul        Aug        Sep        Oct        Nov        Dec
1985 0.08004584 0.08658091 0.09265620 0.09835695 0.10374492 0.10886655 0.11375783 0.11844730 0.12295804 0.12730906 0.13151622 0.13559290
1986 0.14190049 0.14653657 0.15103040 0.15539433 0.15963902 0.16377372 0.16780658 0.17174477 0.17559465 0.17936192 0.18305167 0.18666851
1987 0.19212675 0.19626833 0.20032430 0.20429977 0.20819934 0.21202720 0.21578717 0.21948274 0.22311711 0.22669321 0.23021378 0.23368130
1988 0.23877772 0.24272984 0.24661864 0.25044706 0.25421783 0.25793349 0.26159637 0.26520867 0.26877242 0.27228953 0.27576179 0.27919087
1989 0.28411044 0.28799163 0.29182121 0.29560117 0.29933341 0.30301969 0.30666165 0.31026087 0.31381881 0.31733686 0.32081633 0.32425847
1990 0.32909141 0.33295962 0.33678341 0.34056426 0.34430360 0.34800277 0.35166302 0.35528556 0.35887154 0.36242205 0.36593810 0.36942069
1991 0.37421745 0.37810478 0.38195255 0.38576194 0.38953407 0.39327003 0.39697083 0.40063745 0.40427081 0.40787181 0.41144129 0.41498007
1992 0.41977140 0.42369703 0.42758662 0.43144115 0.43526155 0.43904870 0.44280347 0.44652666 0.45021906 0.45388143 0.45751448 0.46111891
1993 0.46592465 0.46990063 0.47384325 0.47775334 0.48163169 0.48547905 0.48929616 0.49308372 0.49684241 0.50057288 0.50427575 0.50795163
1994 0.51278515 0.51681928 0.52082216 0.52479452 0.52873703 0.53265035 0.53653514 0.54039200 0.54422153 0.54802430 0.55180086 0.55555175
```

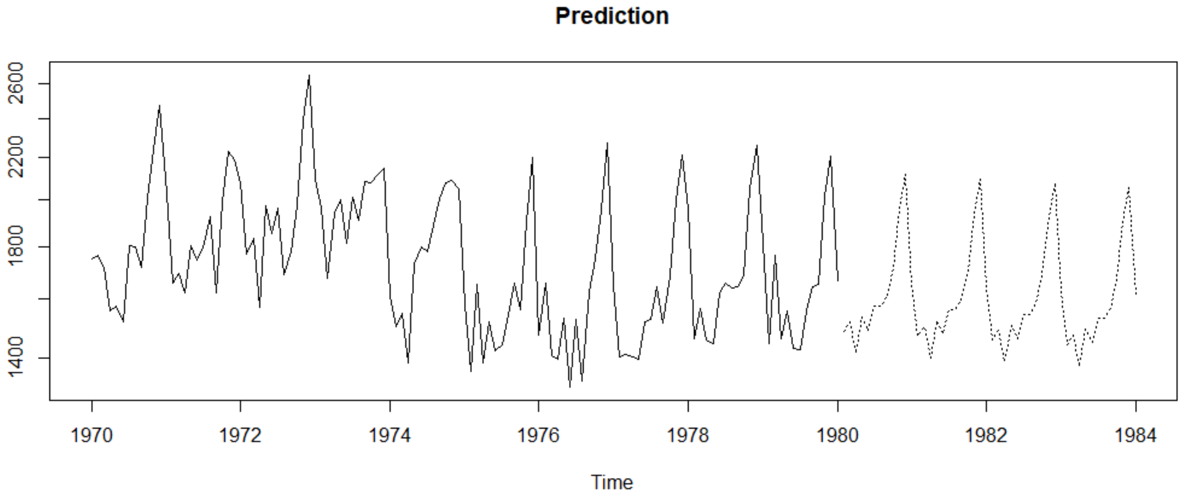
15. Plot the predicted values along with the actual values to compare them.

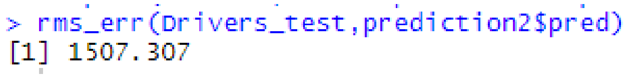
ts.plot(drivers, 2.718^prediction$pred, log='y', lty=c(1,2), main='Prediction')



16. Compute the rms error between the predicted and actual values.

```
data2 <- arima(log(Drivers),c(0,1,1),seasonal=list(order=c(0,1,1),period=12))
prediction2 <- predict(data2, n.ahead=48)
ts.plot(Drivers, 2.718^prediction2$pred, log='y', lty=c(1,3), main='Prediction')
```



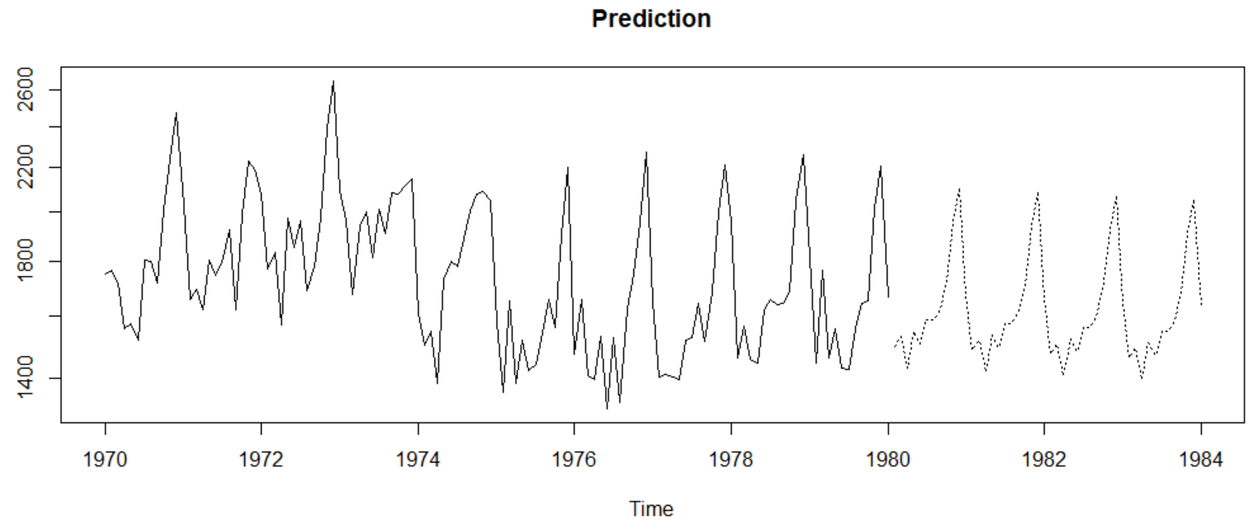


```
> rms_err(Drivers_test,prediction2$pred)
[1] 1507.307
```

17. Try to fine tune the model by manually changing the values of p, d, and q in ARIMA. Are you able to improve the model (i.e., get a lower rms error)?

For p=1, d=0.1 and q=1,

```
data2 <- arima(log(Drivers),c(0,1,1),seasonal=list(order=c(1,0.5,1),period=12))
prediction2 <- predict(data2, n.ahead=48)
ts.plot(Drivers, 2.718^prediction2$pred, log='y', lty=c(1,3), main='Prediction')
```



Prediction

rms_err(Drivers_test,prediction2$pred)

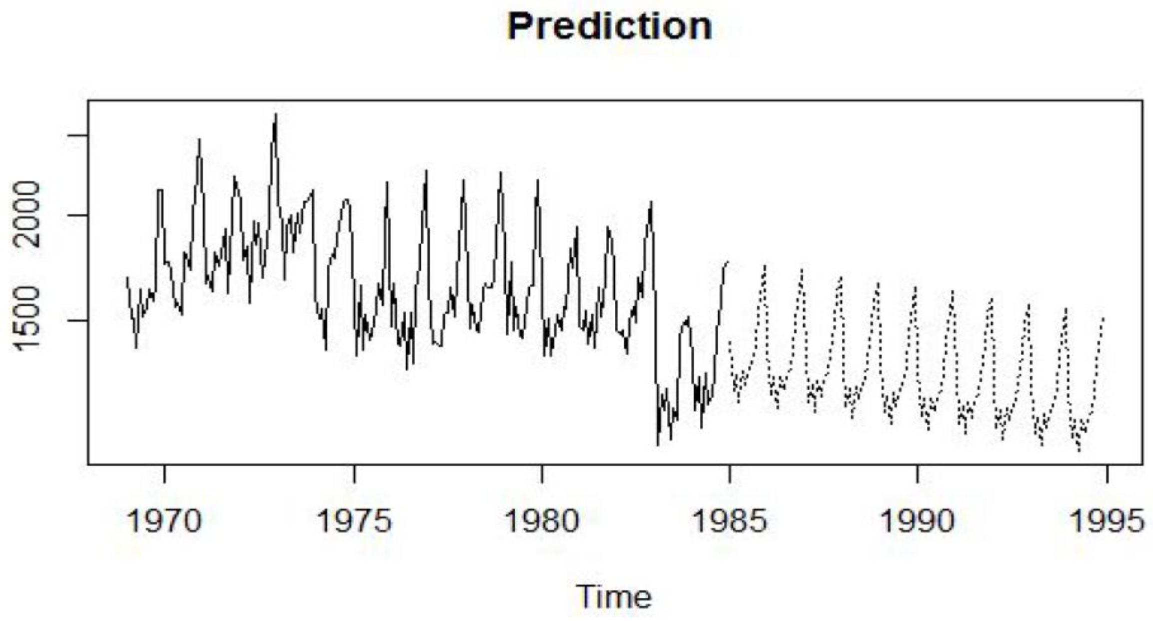
Output: 1507.299

Above graph is after tuning for multiple times.
For ex. p=0,q=1,r=0.5 will give wrong prediction

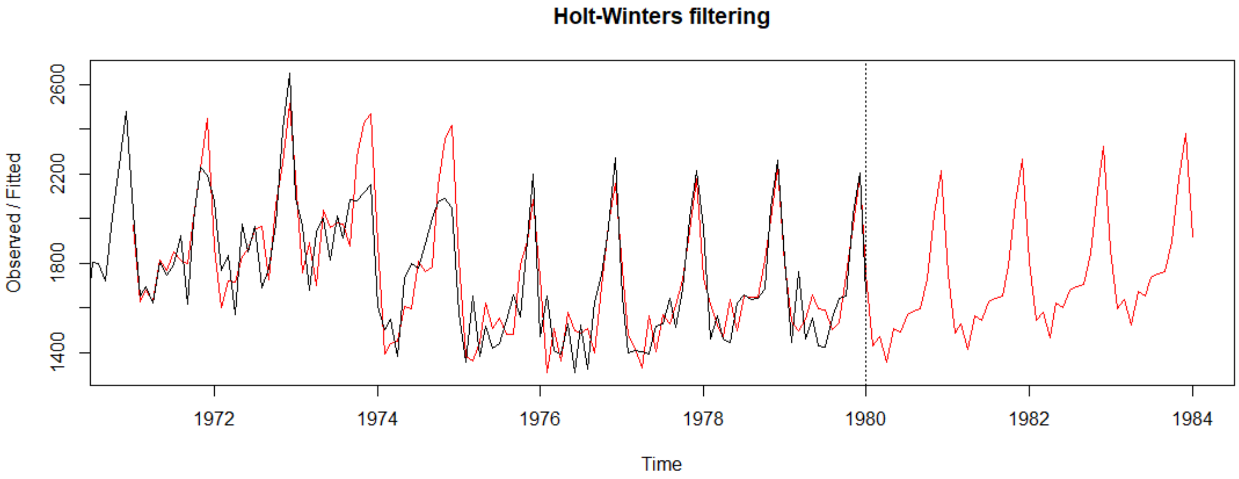
For p=0.5, d=0.5 and q=1,
RMS Error = 1507.212

18. Based on your experiment, which method is better and why? HoltWinters or ARIMA?

ARIMA is better because in ARIMA, the trend went on decreasing till 1990 and further following previous trend. All this is not in HoltWinters.



Prediction



19. Did you try anything like detrending or cleaning up the data outside of these methods? Did it help?

The given dataset gave good outputs without any cleaning up of data so did not try any other methods of detrending.

20. If at any point you are not getting good results, consider changing the data set! (You don't have that liberty in real life, but for this lab assignment, you can!)

The same dataset was used in the entire assignment.