Assignment_3

Prashant Kubsad

5/28/2020

Question 7.1

Describe a situation or problem from your job, everyday life, current events, etc., for which exponential smoothing would be appropriate. What data would you need? Would you expect the value of μ (the first smoothing parameter) to be closer to 0 or 1, and why?

Answer : I am working as a developer in a mortgage insurance company where we get a lot of new quotes and new loan applications from our business partners. We keep monitoring the daily volumes of new quotes and applications that come from our business partners. The number of applications coming in depends on variuos factors like prevailing interest rate, competitior pricing, quality of the insurance applications. We model this time series data year on year to set our annual targets and on a weekly basis to measure to see if we are on track to meet the target. Whenever there is a fluctuation in the volume trends we dont assume it as randomness because there could be factors that are changing the trend. For ex.: Feds slashing interest rates can cause increase in the volume of the applications. If we apply exponential smoothing equation $St = \alpha X_t + (1-\alpha)S_{t-1}$, we would keep alpha closer to 1 as the changes most of the times are not random but there could be a trend.

The volume of business (loan applications) that comes in is a cyclic in nature. In a week to week observation, the traffic is slow during the start of the week and then reaches peak around Wednesday-Thrusday and fall down during weekends. We can apply cyclical exponential equation, $S_t = \alpha/C_{t-L} + (1-\alpha)(S_{t-1} + T_{t-1})$ to model the time series data for the year.

Question 7.2

The first step to use Holt-Winters function is to convert the data into timeseries data. Lets use *ts()* function of R to convert the daily tempratures of Atlanta into timeseries data. I have plotted the timeseries data

```
#cleaning environment and starting fresh.
rm(list = ls())
#setting seed for consistent results
set.seed(1)
library(smooth)
```

```
## Loading required package: greybox
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Package "greybox", v0.6.0 loaded.
```

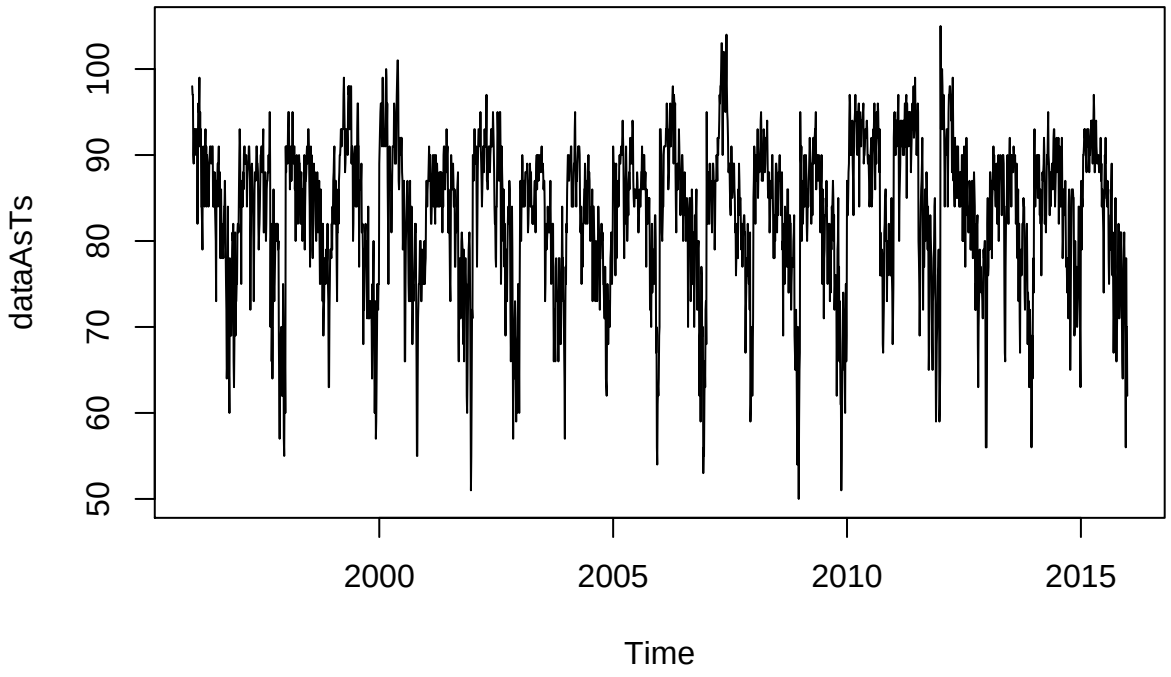
```
## This is package "smooth", v2.5.6
```

```
#Read the text file into data.
data<-read.table("temps.txt", header=TRUE)
```

```r
#Converting the data into a vector and then to time-series data
dataAsVector <- as.vector(as.matrix(data[,2:21]))
dataAsTs <- ts(data = dataAsVector, frequency=123, start=1996)
head(dataAsTs)
```

```
## Time Series:
## Start = c(1996, 1)
## End = c(1996, 6)
## Frequency = 123
## [1] 98 97 97 90 89 93
```

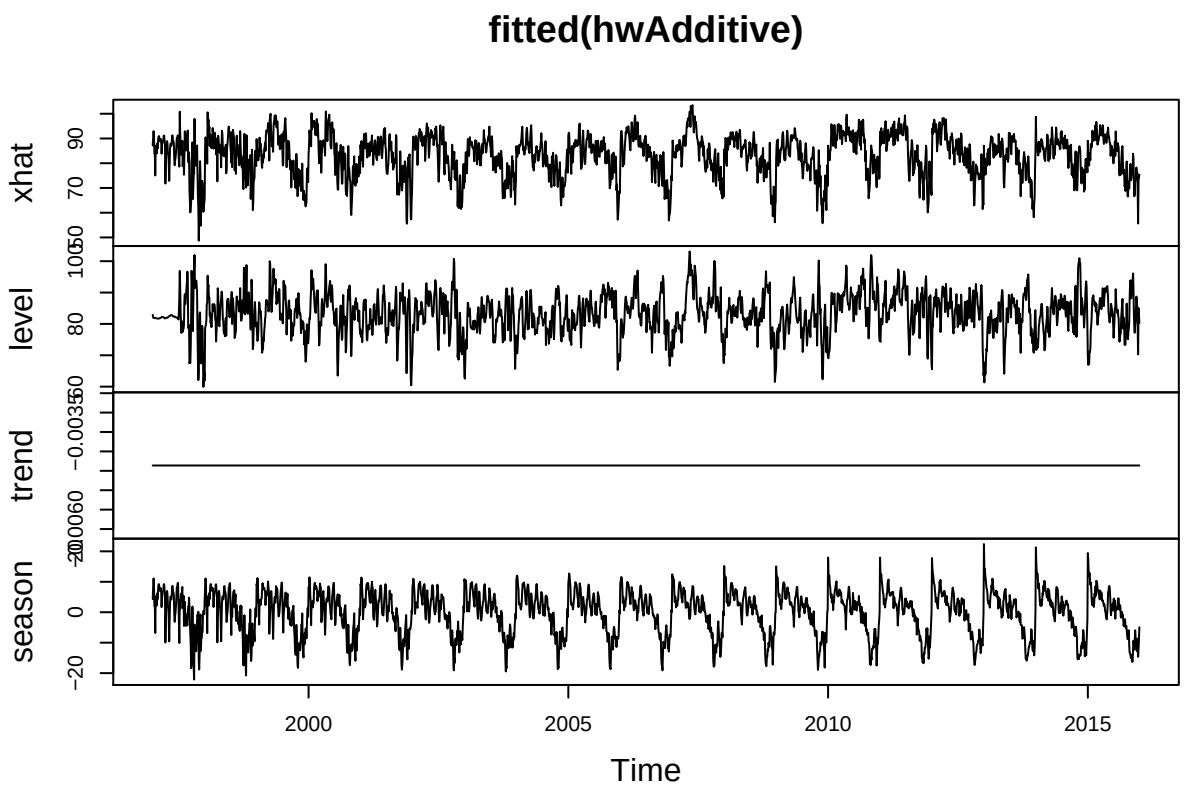
```r
ts.plot(dataAsTs)
```



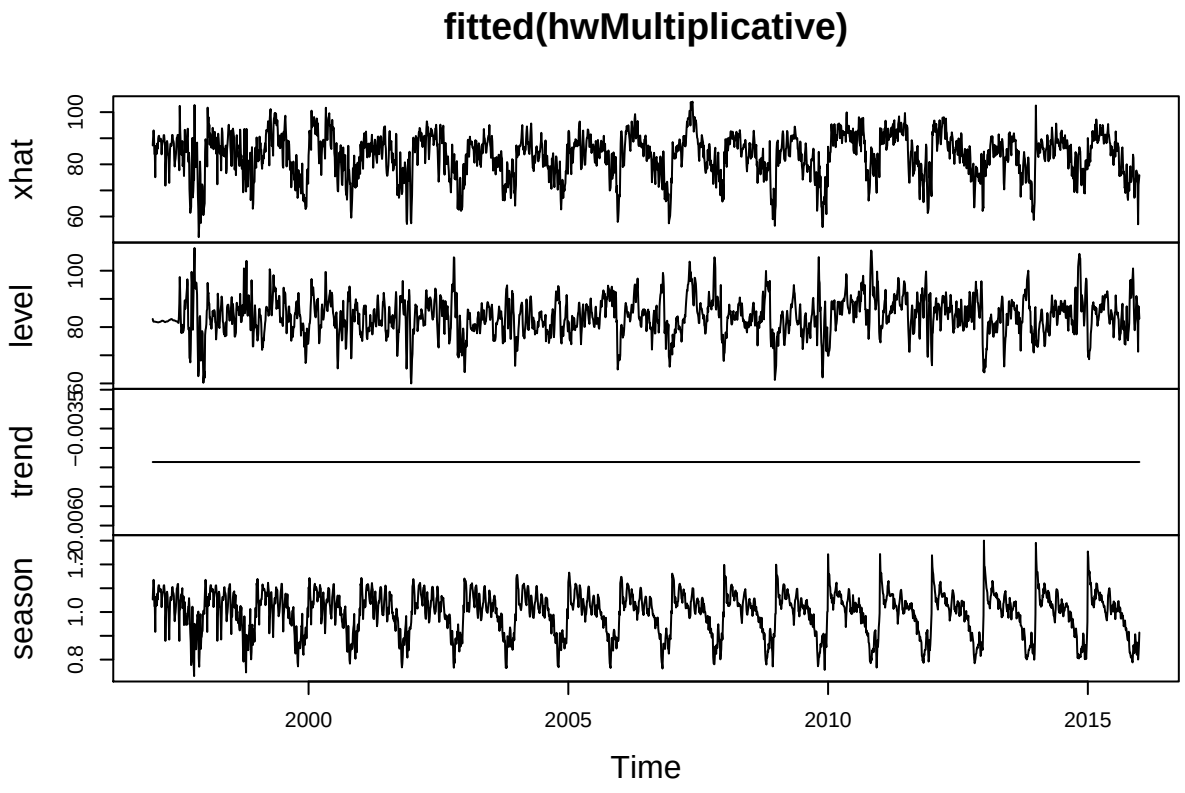
Experiments of Holt-Winters with different parameters

Using this as the input lets run the Holt-Winters filtering function without any parameters. First I will run Holt-Winters for *additive* and then *multiplicative*. One good rule of thumb to use is additive model is best used when the seasonal trend is of the same magnitude throughout the data set, while the Multiplicative Model is preferred when the magnitude of seasonality changes as time increases. If we look at the plots for both additive and multiplicative models, we see seasonal trend keeps increasing through time. So based on this , am going with with multiplicative parameter for the function. If we dont pass the α, β, γ values to the function, the function returns the best values we can use. The best values for multiplicative structure are : $\alpha = 0.615003, \beta = 0, \gamma = 0.5495256$.

```r
#Experiment to test holt-winters with different parameters.
hwAdditive <- HoltWinters(x=dataAsTs, seasonal="additive")
hwMultiplicative <- HoltWinters(x=dataAsTs, seasonal="multiplicative")
plot(fitted(hwAdditive))
```




```
plot(fitted(hwMultiplicative))
```




```
cat("suggested smoothing values: ")
```

```
## suggested smoothing values:
```

```
cat("alpha: ",hwMultiplicative$alpha)
```

```
## alpha:  0.615003
```

```
cat("beta: ",hwMultiplicative$beta)
```

```
## beta:  0
```

```
cat("gamma: ",hwMultiplicative$gamma)
```

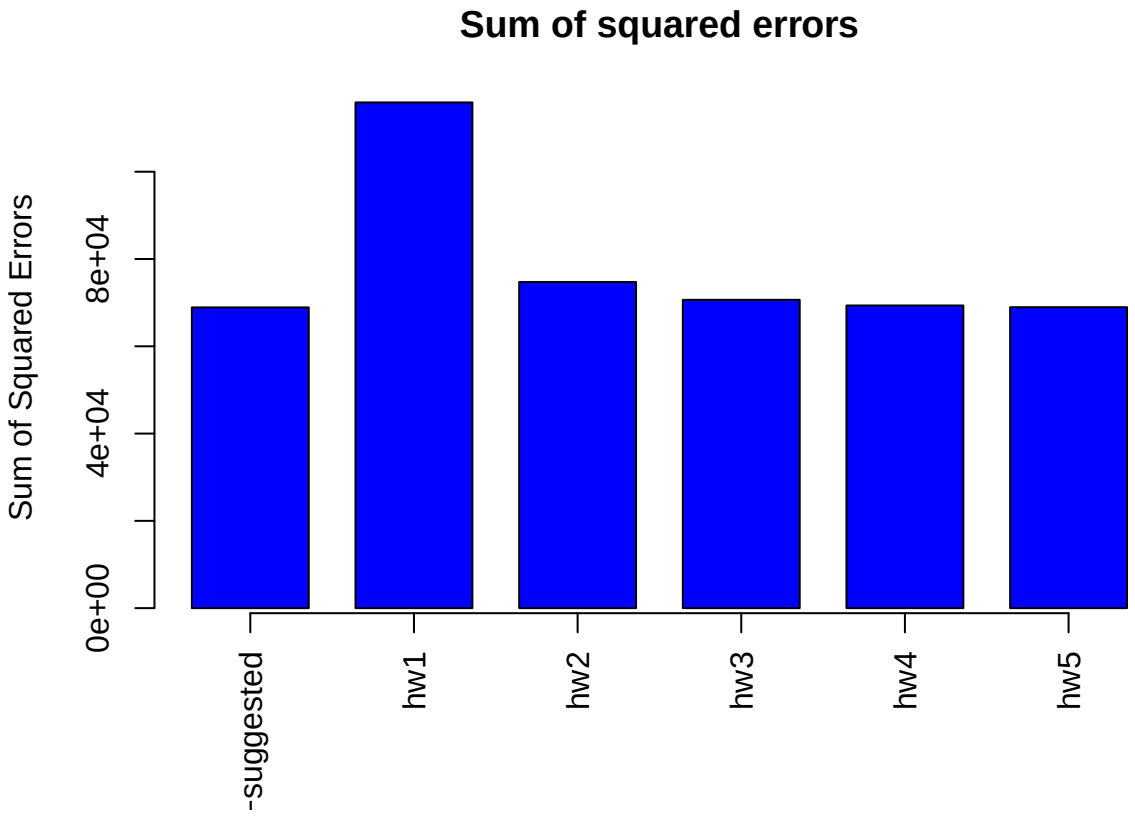
```
## gamma:  0.5495256
```

I tried manually with different values for α, β, γ but the best sum of squared errors was lowest for the R suggested values. I am going with R-suggested values for the final exponential smoothing.

```
# Experiments with different values for alpha, beta, gamma
#initializing list to capture sum of squared errors
sseList <- vector()

#adding the sum of squared errors for the R-suggested model.
sseList<-append(sseList,hwMultiplicative$SSE)

#computing new models for various different values of alpha, beta, gamma
hw1 <-  HoltWinters(x=dataAsTs,alpha=0.1,beta = 0.1,gamma=0.1,seasonal = "multiplicative")
hw2 <-  HoltWinters(x=dataAsTs,alpha=0.2,beta = 0,gamma=0.3,seasonal = "multiplicative")
hw3 <-  HoltWinters(x=dataAsTs,alpha=0.4,beta = 0,gamma=0.5,seasonal = "multiplicative")
hw4 <-  HoltWinters(x=dataAsTs,alpha=0.5,beta = 0,gamma=0.5,seasonal = "multiplicative")
hw5 <-  HoltWinters(x=dataAsTs,alpha=0.6,beta = 0,gamma=0.5,seasonal = "multiplicative")
sseList<-append(sseList,hw1$SSE)
sseList<-append(sseList,hw2$SSE)
sseList<-append(sseList,hw3$SSE)
sseList<-append(sseList,hw4$SSE)
sseList<-append(sseList,hw5$SSE)
barPlot <- barplot(sseList, space=0.4, xaxt='n', xlab="", ylab="Sum of Squared Errors",
main="Sum of squared errors", col = "blue")
axis(1,las=2, at=barPlot, labels=c('R-suggested','hw1','hw2','hw3','hw4','hw5'))
```



Recreating HoltWinters function with the best parameters suggested. The response of HW function has fitted parameter with *xhat,level,trend,season*.

- xhat is the smooted value for every value in input file.

- level is the base line value.
- trend is the trend value.
- Season is the seasonal trend value.

xhat = (level + trend) * season.

The daily values of the year 1996 is considered as baseline and xhat values from 1997 is given in the output.

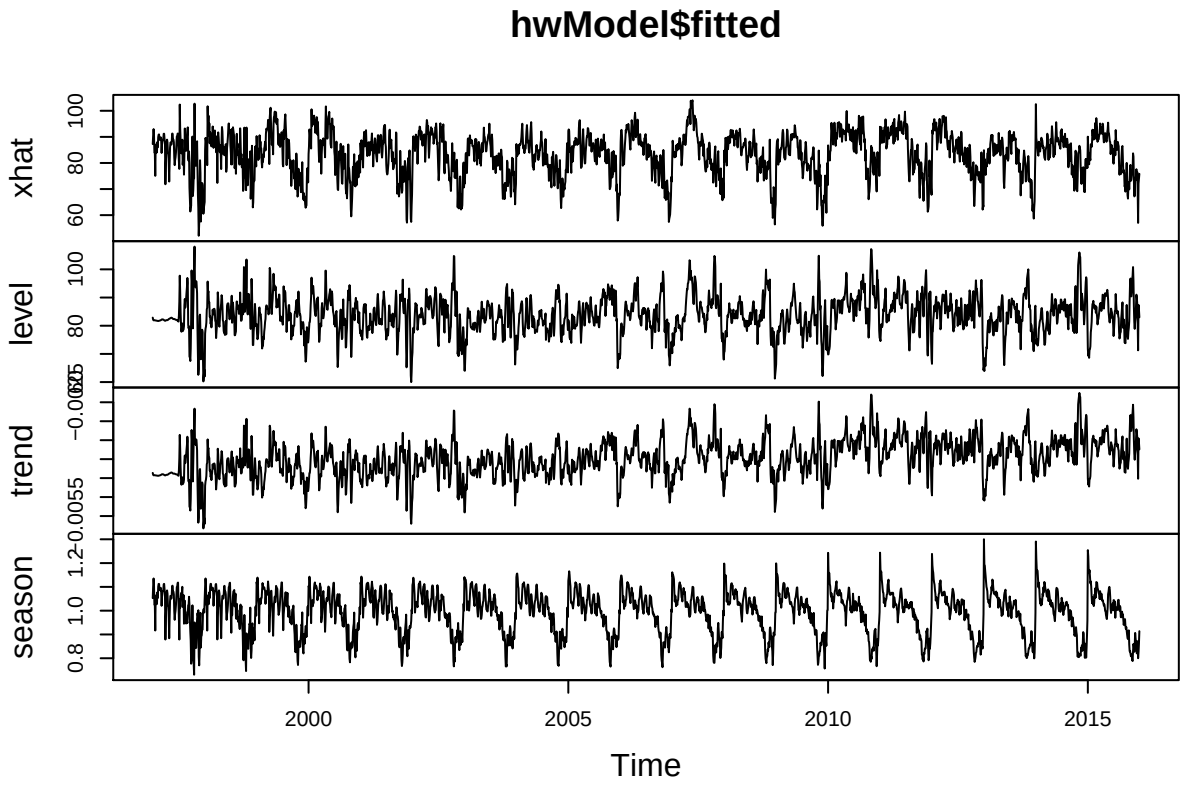
To answer if the summers ended later in the recent years we need to do some kind of change detection using CUSUM. In the previous example we did CUSUM on the actual values, now we can do the same on the seasonal factors or xhat values. Whenver we see the seasonal trend factor drops beyond a threshold, we can consider that as the unofficial end of the summer.

```
# Recreating the Holt-Winters function with best parameters suggested.
hwModel <- HoltWinters(x=dataAsTs,
                       alpha=0.615003,
                       #beta=FALSE,
                       gamma=0.5495256,
                       seasonal = "multiplicative")
head(hwModel$fitted)
```

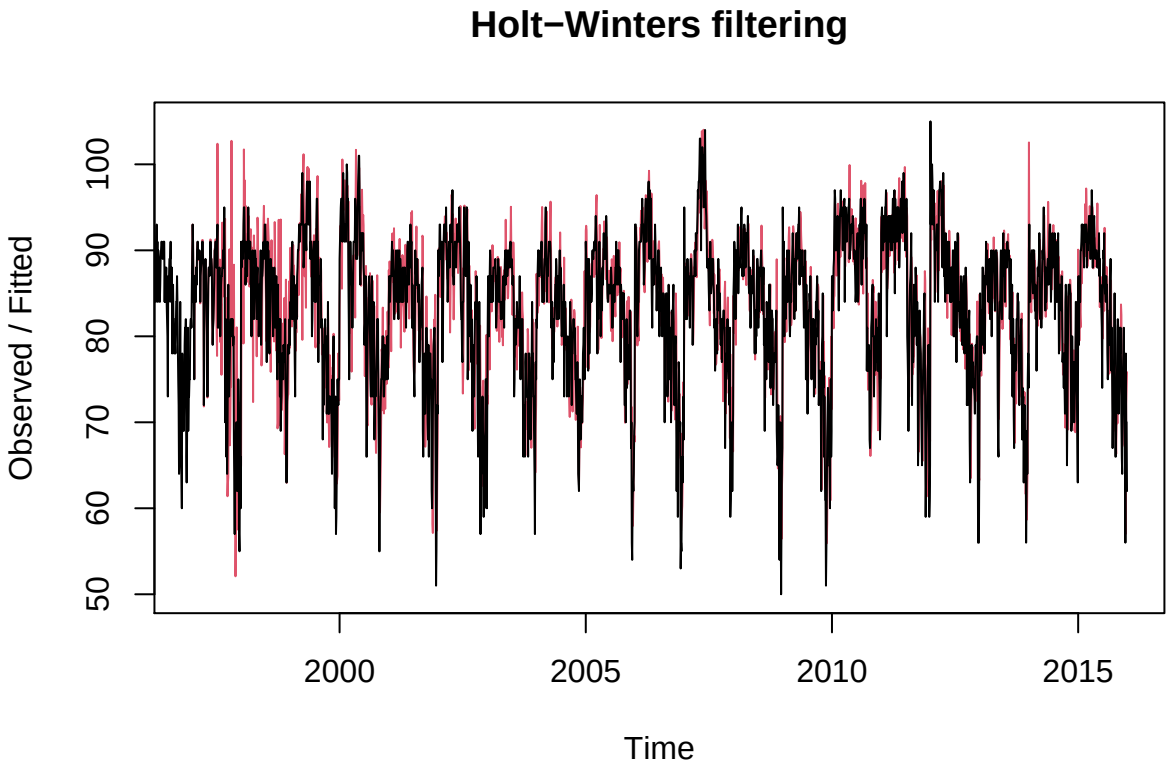
```
## Time Series:
```

```
## Start = c(1997, 1)
## End = c(1997, 6)
## Frequency = 123
##              xhat    level        trend   season
## 1997.000 87.23653 82.87739 -0.004362918 1.052653
## 1997.008 90.42177 82.15059 -0.004410675 1.100742
## 1997.016 92.99725 81.91054 -0.004426253 1.135413
## 1997.024 90.94019 81.90760 -0.004426155 1.110338
## 1997.033 83.99907 81.93630 -0.004423965 1.025231
## 1997.041 84.04486 81.93243 -0.004423928 1.025838
```

```
plot(hwModel$fitted)
```




```
#plot of Holt-Winters overlaying the smoothed curve on top of actual values.
plot(hwModel)
```




```
# converting time series values of seasonal trend into matrix of 123 X 19.
# from 1997 to 2015.
seasonalMatrix <- (matrix(hwModel$fitted[,4],nrow = 123, ncol = 19))
head(as.data.frame(seasonalMatrix))
```

```
##         V1       V2       V3       V4       V5       V6       V7       V8
## 1 1.052653 1.049468 1.120614 1.103343 1.118399 1.108182 1.140920 1.140593
## 2 1.100742 1.099653 1.108029 1.098328 1.110191 1.116222 1.126838 1.154092
## 3 1.135413 1.135420 1.139098 1.142836 1.143207 1.138502 1.129686 1.156106
## 4 1.110338 1.110492 1.117080 1.125778 1.134543 1.126123 1.130765 1.137733
## 5 1.025231 1.025233 1.044685 1.067294 1.084728 1.097244 1.115061 1.103886
## 6 1.025838 1.025722 1.028169 1.042341 1.053954 1.067496 1.080206 1.094317
##         V9      V10      V11      V12      V13      V14      V15      V16
## 1 1.125457 1.122080 1.161435 1.198124 1.198935 1.243041 1.243811 1.238463
## 2 1.142206 1.131907 1.144568 1.134681 1.153456 1.165456 1.172962 1.190763
## 3 1.165675 1.147999 1.149476 1.135774 1.153330 1.155219 1.157310 1.169797
## 4 1.150653 1.147007 1.142512 1.150179 1.151187 1.157770 1.163866 1.159364
## 5 1.120829 1.133745 1.132179 1.142729 1.139260 1.112924 1.132453 1.132063
## 6 1.102687 1.092186 1.075775 1.088558 1.082197 1.103105 1.115085 1.118590
##        V17      V18      V19
## 1 1.300234 1.290678 1.254545
## 2 1.191984 1.219221 1.228855
## 3 1.189939 1.172337 1.169072
## 4 1.166626 1.168018 1.158981
## 5 1.145247 1.168183 1.170472
## 6 1.121610 1.134978 1.145493
```

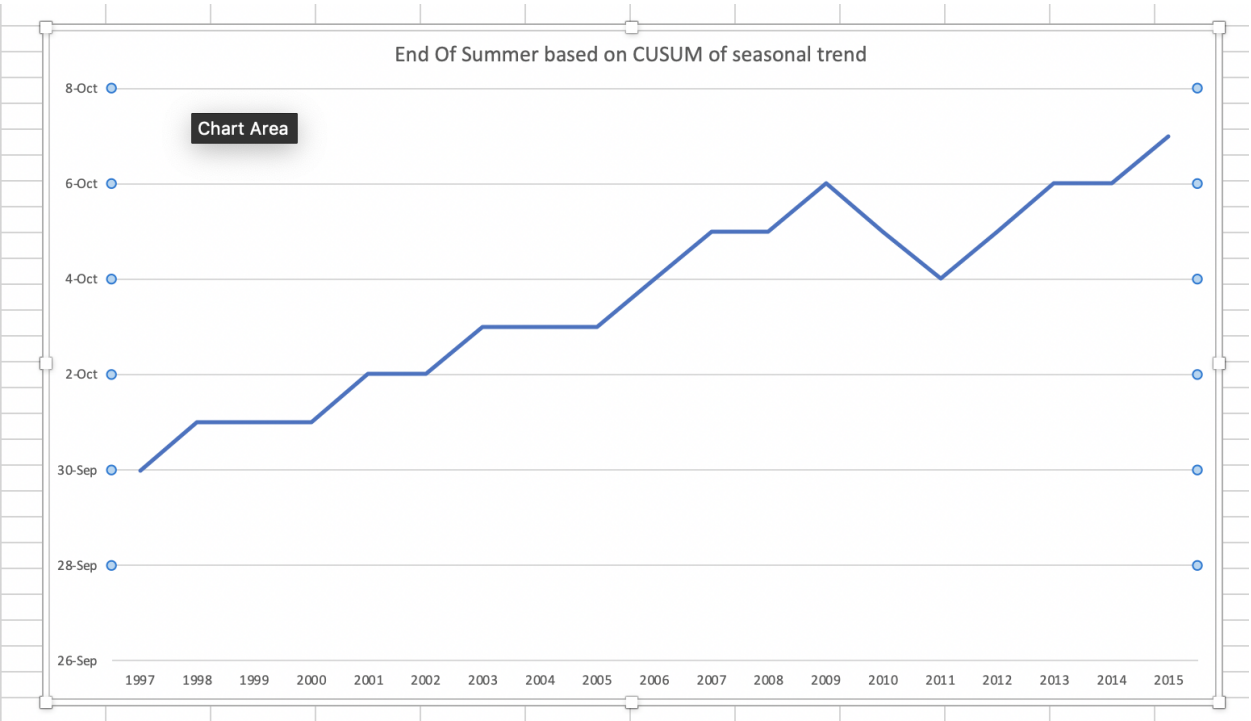
```
xhatMatrix<- (matrix(hwModel$fitted[,1],nrow = 123, ncol = 19))
head(as.data.frame(xhatMatrix))
```

```
##         V1       V2       V3       V4       V5       V6       V7       V8
```

```
## 1 87.23653 65.04295 90.29596 83.39868 87.68840 78.07383 73.09911 87.27051
## 2 90.42177 84.87508 85.44813 86.44384 84.78798 86.02302 72.13057 85.01816
## 3 92.99725 89.61485 85.65856 92.85742 88.70510 90.22969 77.77567 82.68540
## 4 90.94019 88.47552 84.80660 91.55291 86.98688 87.27882 83.52297 83.37206
## 5 83.99907 83.11152 81.14231 88.80206 81.40606 86.06705 83.86001 83.64823
## 6 84.04486 88.00068 85.21649 91.04501 81.83688 87.87734 78.93379 86.79087
##          V9       V10       V11       V12       V13       V14       V15       V16
## 1 92.29768 78.50742 81.58640 84.72857 79.51740 86.74572 93.88434 82.30580
## 2 92.85652 88.18103 88.52612 80.39464 85.65650 81.47246 87.43851 92.55014
## 3 92.33885 92.43564 86.72265 84.53314 88.31310 82.29220 90.24833 91.18751
## 4 87.29552 92.69778 83.30506 89.62791 88.56562 82.90476 93.69366 95.13157
## 5 84.25159 90.58912 84.18887 89.26978 89.12480 80.92705 90.14676 94.60957
## 6 85.75606 86.91475 82.21692 84.28933 79.32507 84.51972 88.67074 96.75518
##         V17       V18      V19
## 1 84.88715 102.54876 90.07919
## 2 76.18605  89.57556 85.16914
## 3 81.46101  88.15102 82.09151
## 4 76.56644  87.11594 79.49257
## 5 75.41942  85.20641 83.69606
## 6 78.42344  83.25366 82.08768
```

First lets do the CUSUM on the seasonal trend to see when there was a change detected. The drop in seasonal trend can be marked as unofficial end of the summer.As discussed in previous office hours am going with good rule of thumb,C= 0.5 times std deviation and T = 3 times std deviation for each year. I am using the excel spreadsheet to calculate CUSUM on seasonal trend factor. I am attaching image of the cusum calculations below. I have hidden columns and rows that are not relevant to capture everything in one screen shot. The detailed calculations are attached in the submission. Based on the dates that cross the threshold, we can say the summers are ending a bit later in the recent years.

	A	E	I	M	Q	U	Y	AC	AG	AK	AO	AS	AW	BA	BE	BI	BM	BQ	BU	BY
1		1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
2	T= 3*std.dev	0.27599155	0.26813173	0.26155603	0.2609549	0.26006481	0.25675292	0.25842642	0.26057199	0.25888702	0.25924193	0.26455466	0.26839785	0.27359573	0.27773687	0.27996292	0.28404548	0.29317767	0.29679307	0.29674521
3	C=0.5%*std-	0.04599859	0.04468862	0.04359267	0.04349248	0.04334413	0.04279215	0.04307107	0.04342867	0.04314784	0.04320699	0.04409244	0.04473297	0.04559929	0.04628948	0.04666049	0.04734091	0.04886294	0.04946551	0.04945753
4	std-dev	0.09199718	0.08937724	0.08718534	0.08698497	0.08668827	0.08558431	0.08614214	0.08685733	0.08629567	0.08641398	0.08818489	0.08946595	0.09119858	0.09257896	0.09332097	0.09468183	0.09772589	0.09893102	0.09891507
5	mu	1	0.99818849	0.99805443	0.99750879	0.99712556	0.99619352	0.99550798	0.99573591	0.99563188	0.99496119	0.99480933	0.99464057	0.99431958	0.9941094	0.99408031	0.99337352	0.99322783	0.99349994	0.99285808
6		st	st	st	st	st	st	st	st	st	st	st	st	st	st	st	st	st	st	st
7																				
67	29-Aug	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
68	30-Aug	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
69	31-Aug	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
70	1-Sep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
82	13-Sep	0.00880951	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
83	14-Sep	0.00521981	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
84	15-Sep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
85	16-Sep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
86	17-Sep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
87	18-Sep	0.00816091	0	0.00349386	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
88	19-Sep	0.00353791	0	0.00094802	0.0018924	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
89	20-Sep	0	0	0	0.0210098	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
90	21-Sep	0.00648901	0.01235727	0.00694556	0.00689551	0.00973393	0	0	0	0	0	0	0	0.00045069	0	0	0	0	0	0
91	22-Sep	0	0.04317794	0.01892652	0.02816831	0.04316725	0.02175366	0.03188731	0.04288465	0.01944515	0.0205275	0	0	0	0	0	0	0	0	0
92	23-Sep	0	0	0	0	0.00511558	0	0.00384061	0.01643299	0	0	0	0	0	0	0	0	0	0	0
93	24-Sep	0	0	0	0	0	0.00110776	0.02214202	0.02813834	0.00712935	0	0	0	0	0	0	0	0	0.01022243	0.00328094
94	25-Sep	0	0	0	0	0	0	0.02405582	0.02593639	0.0086958	0.006442	0.02549569	0.0164378	0.00072269	0	0	0	0	0	0
95	26-Sep	0	0	0	0	0	0	0	0	0	0.0130707	0.03108227	0.0293	0.01250538	0.00894792	0.04067002	0.02580721	0.00099328	0	0
96	27-Sep	0	0	0	0	0	0	0	0	0	0	0.00911456	0.0132465	0.00942677	0.01969944	0.05074425	0.04046281	0.00307107	0	0
97	28-Sep	0.04391491	0.02399777	0.02873126	0.0132998	0	0	0	0	0	0	0	0	0	0	0.03773907	0.04605112	0.00929285	0.00509563	0.00808944
98	29-Sep	0.12414751	0.06714964	0.08273912	0.0633367	0.04070133	0.04309376	0.03897981	0.05211885	0.03524815	0.0225098	0.03930389	0.0353225	0.01646159	0.03422362	0.0584501	0.05397462	0.02061863	0.01728156	0.02772389
99	30-Sep	0.30123692	0.1961884	0.19906177	0.17961521	0.13556055	0.12752503	0.09861281	0.09866539	0.0718646	0.0605023	0.06659177	0.068025	0.04184088	0.05871164	0.08713822	0.08972943	0.08085242	0.0651879	0.03960183
100	1-Oct	0.45286673	0.35816727	0.32819573	0.30094411	0.24314588	0.21864429	0.17185982	0.16153174	0.12313785	0.1064634	0.08714616	0.0891456	0.09184987	0.09766787	0.11700245	0.15738433	0.1426037	0.11015643	0.06705867
101	2-Oct	0.53103723	0.46206514	0.45004449	0.41631521	0.34987541	0.30771386	0.24964442	0.24493099	0.19808709	0.1696813	0.14134314	0.13144349	0.14021546	0.14411959	0.16060927	0.20355564	0.19787428	0.16020576	0.11127152
102	3-Oct	0.46263264	0.43363691	0.46381535	0.42876051	0.37419313	0.34033302	0.28946743	0.30105163	0.26155534	0.235098	0.19992943	0.19598679	0.18723205	0.18759951	0.2299886	0.24976944	0.24137777	0.20397979	0.17024136
103	4-Oct	0.56496424	0.50942788	0.52301041	0.50992771	0.44779516	0.40892069	0.35718203	0.34073668	0.30536109	0.2788523	0.24038022	0.23481199	0.21695674	0.22429693	0.28117692	0.28302015	0.25567485	0.21738643	0.2330085
104	5-Oct	0.71580465	0.63466285	0.63296116	0.61239821	0.54068719	0.49795115	0.43032674	0.39994163	0.36448894	0.3434192	0.2943418	0.28247179	0.25858273	0.30011225	0.34604754	0.33074205	0.29230093	0.25107486	0.26461845
105	6-Oct	0.89060006	0.79300881	0.79908842	0.76877782	0.70209041	0.66413082	0.57699995	0.53359217	0.49433379	0.4809195	0.43397439	0.40037499	0.36157762	0.37886007	0.41214737	0.39377446	0.34469982	0.29885659	0.29071629
106	7-Oct	1.11380806	1.00033058	0.98199598	0.93981542	0.88867144	0.85172378	0.76407385	0.71198382	0.66233334	0.6434239	0.60045968	0.55100739	0.49932931	0.4826346	0.48776819	0.46997286	0.4502945	0.41421342	0.37622523
107	8-Oct	1.11712287	1.04949165	1.09206904	1.06088052	1.06042257	1.00232755	0.94266076	0.88894627	0.84017679	0.8144265	0.77055296	0.71388818	0.6758677	0.63686802	0.61979122	0.59202197	0.58640688	0.55212556	0.49046438
108	9-Oct	1.21760758	1.15123502	1.1994814	1.17538312	1.19731689	1.16983681	1.12933996	1.07322211	1.02787374	1.0038041	0.94039725	0.88455548	0.84273599	0.78691234	0.75614594	0.73660647	0.71294097	0.67186659	0.60462902
109	10-Oct	1.29223248	1.22701159	1.28111755	1.26515882	1.27375162	1.25374217	1.22898997	1.19138806	1.16229978	1.1336577	1.07258883	1.02479848	0.98279788	0.94965336	0.90898087	0.89793628	0.85626085	0.80037562	0.72879196
110	11-Oct	1.40274859	1.33022536	1.36965741	1.35257143	1.33776945	1.31801914	1.28288897	1.28268351	1.25127223	1.2187302	1.16779582	1.16625768	1.12099017	1.08987628	1.04990869	1.05174029	0.99928533	0.93977985	0.863874



I have run the similar cusum model on the Xhat in the attached spread sheet.

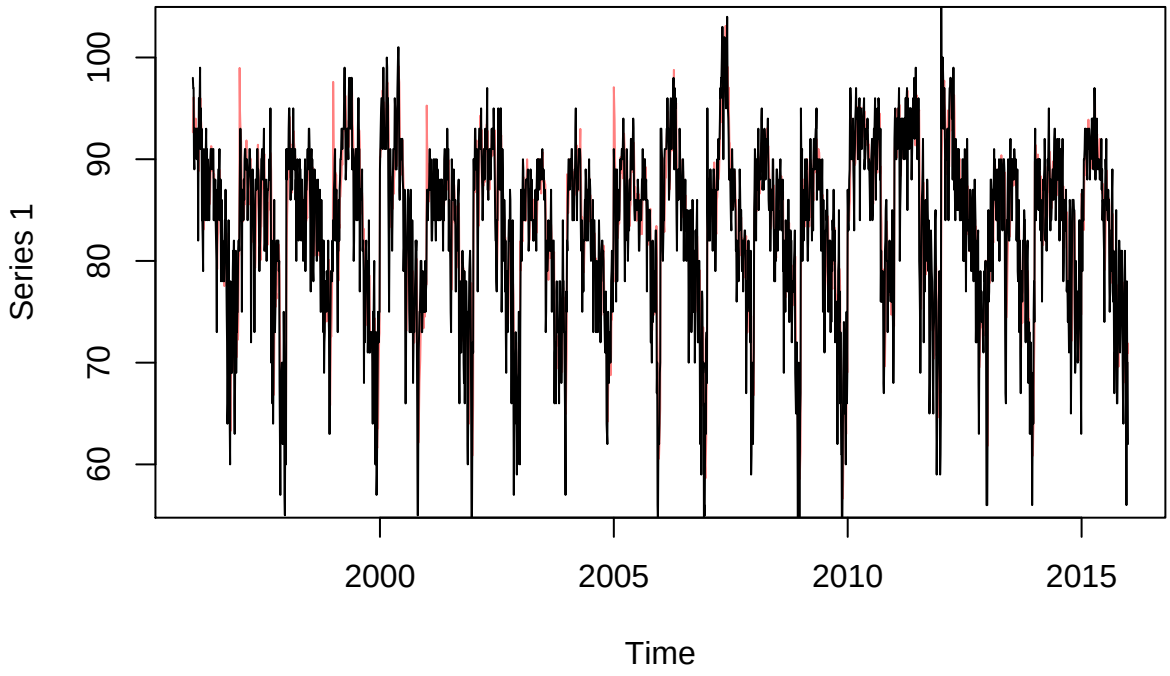
Exponential smoothing using es function from smooth package:

I have run the exponential smoothing using es function from smooth package. As given in assignment, i have used *model="AAM"* parameter to emulate HoltWinters method in es. The response of this function is interestingly different from HoltWinters function from above. The best set of parameters suggested by es function are $\alpha = 0.3572, \beta = 0.0023 and \gamma = 0.0102$. I have plotted the fitted values overlayed on top of the actual data.

```
esModel<-es(y=dataAsTs,
   model="AAM")
esModel
```

```
## Time elapsed: 1.89 seconds
## Model estimated: ETS(AAM)
## Persistence vector g:
##  alpha   beta  gamma
## 0.3572 0.0023 0.0102
## Initial values were optimised.
##
## Loss function type: MSE; Loss function value: 22.0773
## Error standard deviation: 4.8269
## Sample size: 2460
## Number of estimated parameters: 129
## Number of degrees of freedom: 2331
## Information criteria:
##      AIC     AICc      BIC     BICc
## 14851.78 14866.17 15601.00 15657.19
```

```
plot(esModel$fitted, col=rgb(red = 1, green = 0, blue = 0, alpha = 0.5))
lines(dataAsTs,col = "black")
```



Question 8.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a linear regression model would be appropriate. List some (up to 5) predictors that you might use.

Answer: We use linear regression model in our current work to predict the number of new loan applications for mortgage insurance that come up in next few quarters. One measure of this is also used to set the targets to achieve with respect to new insurance that should be underwritten. There are multiple factors that determine the predictions of new loan applications.I am listing few of the macro economical factors:

- Current federal interest rates.: If the interest rates stay low, it encourages more buyers to purchase new homes or refinance their existing loan. Since federal interest rates themselves keep changing, it is a very interesting puzzle to rely on the prediction of changing interest rate, like aiming at a moving target.
- Employment rate: More number of people are employed, more the number of loan applications.
- House price volatility index: If the house prices are more volatile, the risk index associated with new business increases.
- Quote Volume: Quotes are precursor to the new loan applications. If the quote volume is high, usually translates to increased application numbers.
- House price index: This index shows the quality of the housing market in general.

Other than above mentioned predictive use of linear regression, there are other uses of model to determine the value of entities like how much is a refinance of a loan actually worth.

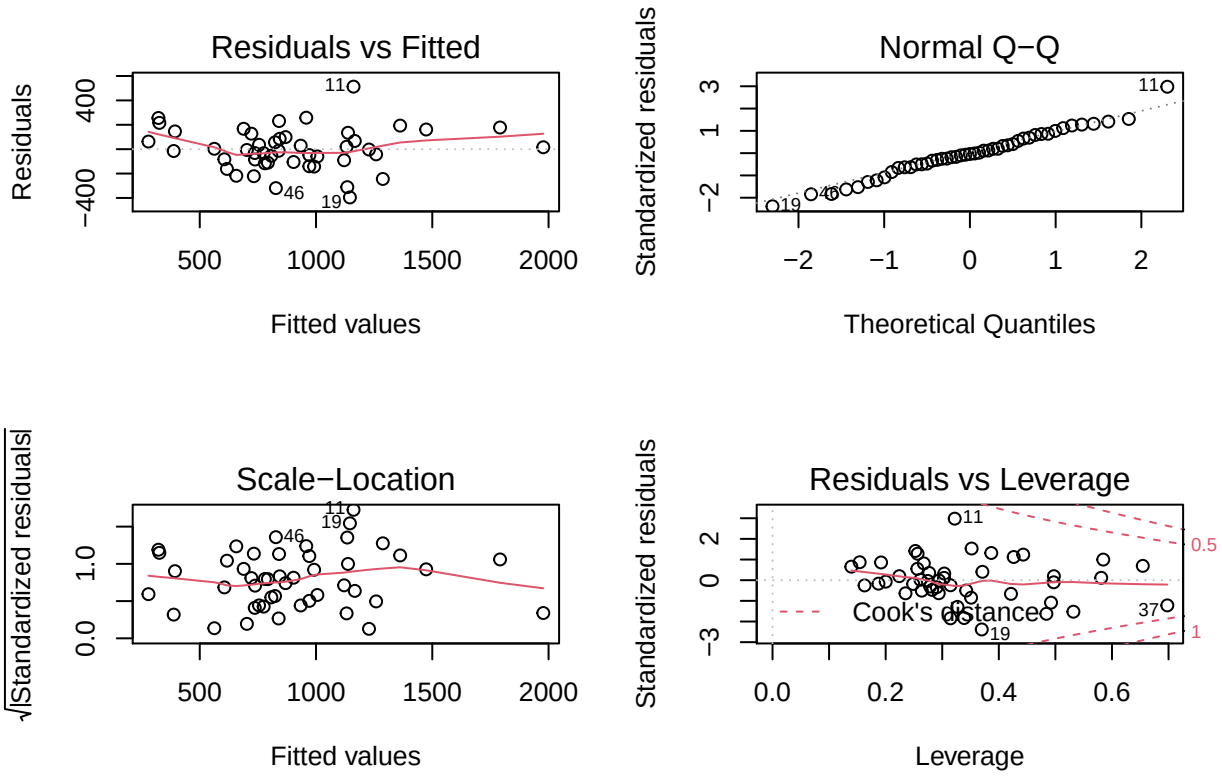
Question 8.2

I have used lm function to perfom linear regression on the input file data. Later used the model to predict for the values provided in question. The predicted value is *155.4349*. This observed value is well out of the

bounds of given vvalues of crimes column in the input. This kind of points out that all the predictors are not significant towards the result.

```
#cleaning environment and starting fresh.
rm(list = ls())
#setting seed for consistent results
set.seed(1)
#Read the text file into data.
data<-read.table("uscrime.txt", header=TRUE)
#creating model with all the predictors
model<-lm(formula = Crime ~ M+So+Ed+Po1+Po2+LF+M.F+Pop+NW+U1+U2+Wealth+Ineq+Prob+Time,
   data=data)

#Configuration to show all the plots in one image.
par(mfrow=c(2,2))
plot(model)
```




```
#Predicting the outcome with params given in the hw.
p<-predict(model,data.frame(M=14,So=0,Ed=10.0,Po1=12.0,Po2=15.5,LF=0.640,M.F=94.0,Pop=150,NW=1.1,U1=0.1
p
```

```
##        1
## 155.4349
```

Lets take a look at the response of the lm function. Coefficients listed in the response has intercept which will correspond to a_0 in the equation in the lecture, $y = a_0 + \sum_{j=1}^{m} a_j x_j$. Each of the predictor's coefficient is listed in the R output. Against each co-efficient, there is a p-value with signif code. This p-value will help us identify which are all the factors that are statistically significant and are co related to the response. For

every factor there are 2 hypothesis:

- Null hypothesis: This hypothesis says that there is no corelation between the result and each parameter.
- Alternative hypothesis: This hypothesis says there is a corelation between the result and each parameter.

For each field, we can take a look at p-value to select which parameters are significant. If *p-value<0.05* we can conclude that predictor is a significant towards the result. For the model we see the following fields have p-value less than 0.05. M, Ed,Ineq and Prob.

```
summary(model)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop +
##     NW + U1 + U2 + Wealth + Ineq + Prob + Time, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

We can re run a new model with only significant predictors and predict value for given input. We see that predicted value is *897.2302*. The new model's adjusted R-squared is less than previous model and Residual standard error on new model is more than previous model. This tells the predictors which we excluded are not directly impacting the output by themselves, but there is corelation between multiple predictors towards final outcome.We can run feature selection using recursive feature elimination method to determine the optimal number of predictors that impact the outcome. I have used caret library's rfe function to perform recursive feature elimination. Based on the output, we pick top 5 predictors and recreate the model. The new prediction comes around *1388.401*

```r
#creating new model with only significant predictors based on p-value
newModel<-lm(formula = Crime ~ M+Ed+Ineq+Prob,data)
p2<-predict(newModel,data.frame(M=14,Ed=10.0,Ineq=20.1,Prob=0.04))
summary(newModel)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Ineq + Prob, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -532.97 -254.03  -55.72  137.80  960.21
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1339.35    1247.01  -1.074  0.28893
## M              35.97      53.39   0.674  0.50417
## Ed            148.61      71.92   2.066  0.04499 *
## Ineq           26.87      22.77   1.180  0.24458
## Prob        -7331.92    2560.27  -2.864  0.00651 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 347.5 on 42 degrees of freedom
## Multiple R-squared:  0.2629, Adjusted R-squared:  0.1927
## F-statistic: 3.745 on 4 and 42 DF,  p-value: 0.01077
```

```r
cat("\n predicted value for given input:",p2)
```

```
##
##  predicted value for given input: 897.2307
```

```r
#Running recursive feature elimination to determine the significant factors
library(caret)
```

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

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

```
## Registered S3 method overwritten by 'lava':
##   method     from
##   print.pcor greybox
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:greybox':
##
##     MAE
```

```r
set.seed(0)
# creating control with cross validation params.
ctrl <- rfeControl(functions = lmFuncs,
                   method = "repeatedcv",
                   number=15)
#running recursive feature elimintion using above control.
rfe <- rfe(data[,-16], data[[16]],
```

```
                sizes = c(1:15),
                rfeControl = ctrl)

rfe
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (15 fold, repeated 1 times)
##
## Resampling performance over subset size:
##
##  Variables  RMSE Rsquared   MAE RMSESD RsquaredSD  MAESD Selected
##          1 351.1   0.7700 290.0 129.34     0.2957 103.10
##          2 346.6   0.6419 286.9 161.44     0.3477 131.19
##          3 349.7   0.5856 295.2 161.67     0.3840 131.30
##          4 329.2   0.7711 286.1 123.35     0.2755 118.32
##          5 321.4   0.7651 277.3 120.44     0.3192 121.17
##          6 305.5   0.6971 259.9 118.64     0.3855 118.44
##          7 302.0   0.6594 256.3 119.95     0.4010 114.14
##          8 257.6   0.6903 216.0  94.47     0.3852  83.87
##          9 214.9   0.7504 181.2 109.91     0.3642  93.12
##         10 208.7   0.7659 170.5  91.04     0.3565  74.93        *
##         11 209.5   0.7584 173.2  86.93     0.3504  66.85
##         12 220.4   0.7558 184.0  93.13     0.3603  79.44
##         13 229.5   0.7158 190.9  90.52     0.3722  78.35
##         14 230.3   0.7178 190.3  94.03     0.3775  79.75
##         15 243.3   0.7036 197.5  86.83     0.3719  73.29
##
## The top 5 variables (out of 10):
##    U1, Prob, LF, Po1, Ed
```

```
#re-running new model wiht predictors suggested by rfe function.
newModel2<-lm(formula = Crime ~ U1+Prob+LF+Ed+Po1, data=data)
#predicting the value based on the predictors suggested by rfe function.
p5<-predict(newModel2,data.frame(Ed=10.0,Po1=12.0,LF=0.640,U1=0.120,Prob=0.04))
summary(newModel2)
```

```
##
## Call:
## lm(formula = Crime ~ U1 + Prob + LF + Ed + Po1, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -643.80 -131.22   47.91  139.85  537.26
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -126.00     777.27  -0.162    0.872
## U1            382.39    2493.65   0.153    0.879
## Prob        -2055.23    2195.78  -0.936    0.355
## LF           1537.12    1381.31   1.113    0.272
## Ed            -49.66      54.83  -0.906    0.370
## Po1            88.63      18.19   4.873 1.68e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 288.8 on 41 degrees of freedom
## Multiple R-squared:  0.503,  Adjusted R-squared:  0.4424
## F-statistic: 8.299 on 5 and 41 DF,  p-value: 1.736e-05
```

```
cat("\n predicted value for given input:",p5)
```

```
##
##  predicted value for given input: 1388.401
```

Experiment of glm() for linear regression.

I have re run the linear regression with glm function instead of lm function and we get similar results as we got with lm.

```
modelGlm <- glm(Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 + Wealth + Ineq + Prob +
summary(modelGlm)
```

```
##
## Call:
## glm(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop +
##     NW + U1 + U2 + Wealth + Ineq + Prob + Time, family = gaussian,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -395.74   -98.09    -6.69   112.99   512.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 43707.93)
##
##     Null deviance: 6880928  on 46  degrees of freedom
## Residual deviance: 1354946  on 31  degrees of freedom
## AIC: 650.03
##
```

```
## Number of Fisher Scoring iterations: 2
```

```
newdata = data.frame(M=14,So=0,Ed=10.0,Po1=12.0,Po2=15.5,LF=0.640,M.F=94.0,Pop=150,NW=1.1,U1=0.120,U2=3
p3<-predict(modelGlm, newdata, type="response")
p3
```

```
##        1
## 155.4349
```

```
modelGlm2 <- glm(Crime ~ M + Ed + Ineq + Prob, data = data, family=gaussian)
summary(modelGlm2)
```

```
##
## Call:
## glm(formula = Crime ~ M + Ed + Ineq + Prob, family = gaussian,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -532.97  -254.03   -55.72   137.80   960.21
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1339.35    1247.01  -1.074  0.28893
## M              35.97      53.39   0.674  0.50417
## Ed            148.61      71.92   2.066  0.04499 *
## Ineq           26.87      22.77   1.180  0.24458
## Prob        -7331.92    2560.27  -2.864  0.00651 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 120758.8)
##
##     Null deviance: 6880928  on 46  degrees of freedom
## Residual deviance: 5071868  on 42  degrees of freedom
## AIC: 690.07
##
## Number of Fisher Scoring iterations: 2
```

```
newdata2 = data.frame(M=14,Ed=10.0,Ineq=20.1,Prob=0.04)
p4<-predict(modelGlm2, newdata2, type="response")
p4
```

```
##        1
## 897.2307
```