ISYE 6501 HW Wk 3 Solution

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 alpha (the first smoothing parameter) to be closer to 0 or 1, and why?

Working in the supply chain industry, we use exponential smoothing to estimate the package volume level that arrives to the processing terminals. We will need the historical weekly volume at each terminal, with seasonal trend across the year. With the estimated volume level, will allocate the resource to meet the capacity accordingly.

The parameter should be closer to 1 because the volume on weekly level is considered stable with low level randomness.

Question 7.2

Using the 20 years of daily high temperature data for Atlanta (July through October) from Question 6.2 (file temps.txt), build and use an exponential smoothing model to help make a judgment of whether the unofficial end of summer has gotten later over the 20 years. (Part of the point of this assignment is for you to think about how you might use exponential smoothing to answer this question. Feel free to combine it with other models if you'd like to. There's certainly more than one reasonable approach.)

Note: in R, you can use either HoltWinters (simpler to use) or the smooth package's es function (hardertouse,butmoregeneral). If you use es,the Holt-Winters model uses model="AAM" in the function call (the first and second constants are used "A"dditively, and the third (seasonality) is used "M"ultiplicatively; the documentation doesn't make that clear).

```
# read the file temps.txt into R
df_raw <- read.table("Wk3/temps.txt", header=TRUE)

#preview and explore df_raw

head(df_raw)
```

```
##     DAY X1996 X1997 X1998 X1999 X2000 X2001 X2002 X2003 X2004 X2005 X2006
## 1 1-Jul    98    86    91    84    89    84    90    73    82    91    93
## 2 2-Jul    97    90    88    82    91    87    90    81    81    89    93
## 3 3-Jul    97    93    91    87    93    87    87    87    86    86    93
## 4 4-Jul    90    91    91    88    95    84    89    86    88    86    91
## 5 5-Jul    89    84    91    90    96    86    93    80    90    89    90
## 6 6-Jul    93    84    89    91    96    87    93    84    90    82    81
##   X2007 X2008 X2009 X2010 X2011 X2012 X2013 X2014 X2015
## 1    95    85    95    87    92   105    82    90    85
## 2    85    87    90    84    94    93    85    93    87
## 3    82    91    89    83    95    99    76    87    79
## 4    86    90    91    85    92    98    77    84    85
## 5    88    88    80    88    90   100    83    86    84
## 6    87    82    87    89    90    98    83    87    84
```

1. To prepare the input data for exponential modeling

```r
# remove the first column to use the data from 1996 to 2015 as input
df_input <- df_raw[,2:21]

# by checking function HoltWinters, the data input x should ba an object of class ts - to convert input
# by checking function ts, the data should be a vector or matrix - so to convert the input data first t

# first to convert the dataframe to 1-dimension to connect the data across all years
df_input <- unlist(df_input)

# convert 1-dimensional array to vector
df_input_1 <- as.vector(df_input)

# convert to ts, data from 1996 with 123 obs
data_input <- ts(df_input_1, start=1996, frequency=123)

# structure of model input data
str(data_input)
```

```
##  Time-Series [1:2460] from 1996 to 2016: 98 97 97 90 89 93 93 91 93 93 ...
```

2. To build model

2.1 To start from single exponential smoothing

```r
model_single <- HoltWinters(data_input, beta=FALSE, gamma=FALSE)
model_single
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = data_input, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.8388021
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 63.30952
```

From single exponential smoothing model, the best alpha is 0.8388021. The Coefficients is 63.30952.

2.2 To build double exponential smoothing model with trend

```r
model_double <- HoltWinters(data_input, gamma=FALSE)
model_double
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = data_input, gamma = FALSE)
```

```
##
## Smoothing parameters:
##  alpha: 0.8445729
##  beta : 0.003720884
##  gamma: FALSE
##
## Coefficients:
##          [,1]
## a 63.2530022
## b -0.0729933
```

From the above model result, the beta is ~0.0037 and the Coefficients of betta is ~-0.07 which are all close to 0. The beta itself is close to 0 that means the trend is more close to randomness instead of actual trend. And the Coefficients b is also close to zero meaning the trend is not significant that we can ignore.

2.3 To build Holt-Winter model (with trend and cyclic pattern)

```
model_triple <- HoltWinters(data_input)
model_triple
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = data_input)
##
## Smoothing parameters:
##  alpha: 0.6610618
##  beta : 0
##  gamma: 0.6248076
##
## Coefficients:
##                [,1]
## a     71.477236414
## b     -0.004362918
## s1    18.590169842
## s2    17.803098732
## s3    12.204442890
## s4    13.233948865
## s5    12.957258705
## s6    11.525341233
## s7    10.854441534
## s8    10.199632666
## s9     8.694767348
## s10    5.983076192
## s11    3.123493477
## s12    4.698228193
## s13    2.730023168
## s14    2.995935818
## s15    1.714600919
## s16    2.486701224
## s17    6.382595268
## s18    5.081837636
## s19    7.571432660
## s20    6.165047647
```

```
## s21    9.560458487
## s22    9.700133847
## s23    8.808383245
## s24    8.505505527
## s25    7.406809208
## s26    6.839204571
## s27    6.368261304
## s28    6.382080380
## s29    4.552058253
## s30    6.877476437
## s31    4.823330209
## s32    4.931885957
## s33    7.109879628
## s34    6.178469084
## s35    4.886891317
## s36    3.890547248
## s37    2.148316257
## s38    2.524866001
## s39    3.008098232
## s40    3.041663870
## s41    2.251741386
## s42    0.101091985
## s43   -0.123337548
## s44   -1.445675315
## s45   -1.802768181
## s46   -2.192036338
## s47   -0.180954242
## s48    1.538987281
## s49    5.075394760
## s50    6.740978049
## s51    7.737089782
## s52    8.579515859
## s53    8.408834158
## s54    4.704976718
## s55    1.827215229
## s56   -1.275747384
## s57    1.389899699
## s58    1.376842871
## s59    0.509553410
## s60    1.886439429
## s61   -0.806454923
## s62    5.221873550
## s63    5.383073482
## s64    4.265584552
## s65    3.841481452
## s66   -0.231239928
## s67    0.542761270
## s68    0.780131779
## s69    1.096690727
## s70    0.690525998
## s71    2.301303414
## s72    2.965913580
## s73    4.393732595
## s74    2.744547070
```

```
## s75    1.035278911
## s76    1.170709479
## s77    2.796838283
## s78    2.000312540
## s79    0.007337449
## s80   -1.203916069
## s81    0.352397232
## s82    0.675108103
## s83   -3.169643942
## s84   -1.913321175
## s85   -1.647780450
## s86   -5.281261301
## s87   -5.126493027
## s88   -2.637666754
## s89   -2.342133004
## s90   -3.281910970
## s91   -4.242033198
## s92   -2.596010530
## s93   -7.821281290
## s94   -8.814741200
## s95   -8.996689798
## s96   -7.835655534
## s97   -5.749139155
## s98   -5.196182693
## s99   -8.623793296
## s100 -11.809355220
## s101 -13.129428554
## s102 -16.095143067
## s103 -15.125436350
## s104 -13.963606549
## s105 -12.953304848
## s106 -16.097179844
## s107 -15.489223470
## s108 -13.680122300
## s109 -11.921434142
## s110 -12.035411347
## s111 -12.837047727
## s112  -9.095808127
## s113  -5.433029341
## s114  -6.800835107
## s115  -8.413639598
## s116 -10.912409484
## s117 -13.553826535
## s118 -10.652543677
## s119 -12.627298331
## s120  -9.906981556
## s121 -12.668519900
## s122  -9.805502547
## s123  -7.775306633
```

From the output, the beta and Coefficients b are still close to 0 meaning there is no clear trend of change across 20 years.

2.4 With Seasonal Trend

```r
model_season <- HoltWinters(data_input,seasonal="multiplicative")
model_season
```

```
## Holt-Winters exponential smoothing with trend and multiplicative seasonal component.
##
## Call:
## HoltWinters(x = data_input, seasonal = "multiplicative")
##
## Smoothing parameters:
##  alpha: 0.615003
##  beta : 0
##  gamma: 0.5495256
##
## Coefficients:
##             [,1]
## a    73.679517064
## b    -0.004362918
## s1    1.239022317
## s2    1.234344062
## s3    1.159509551
## s4    1.175247483
## s5    1.171344196
## s6    1.151038408
## s7    1.139383104
## s8    1.130484528
## s9    1.110487514
## s10   1.076242879
## s11   1.041044609
## s12   1.058139281
## s13   1.032496529
## s14   1.036257448
## s15   1.019348815
## s16   1.026754142
## s17   1.071170378
## s18   1.054819556
## s19   1.084397734
## s20   1.064605879
## s21   1.109827336
## s22   1.112670130
## s23   1.103970506
## s24   1.102771209
## s25   1.091264692
## s26   1.084518342
## s27   1.077914660
## s28   1.077696145
## s29   1.053788854
## s30   1.079454300
## s31   1.053481186
## s32   1.054023885
## s33   1.078221405
## s34   1.070145761
## s35   1.054891375
## s36   1.044587771
```

```
## s37   1.023285461
## s38   1.025836722
## s39   1.031075732
## s40   1.031419152
## s41   1.021827552
## s42   0.998177248
## s43   0.996049257
## s44   0.981570825
## s45   0.976510542
## s46   0.967977608
## s47   0.985788411
## s48   1.004748195
## s49   1.050965934
## s50   1.072515008
## s51   1.086532279
## s52   1.098357400
## s53   1.097158461
## s54   1.054827180
## s55   1.022866587
## s56   0.987259326
## s57   1.016923524
## s58   1.016604903
## s59   1.004320951
## s60   1.019102781
## s61   0.983848662
## s62   1.055888360
## s63   1.056122844
## s64   1.043478958
## s65   1.039475693
## s66   0.991019224
## s67   1.001437488
## s68   1.002221759
## s69   1.003949213
## s70   0.999566344
## s71   1.018636837
## s72   1.026490773
## s73   1.042507768
## s74   1.022500795
## s75   1.002503740
## s76   1.004560984
## s77   1.025536556
## s78   1.015357769
## s79   0.992176558
## s80   0.979377825
## s81   0.998058079
## s82   1.002553395
## s83   0.955429116
## s84   0.970970220
## s85   0.975543504
## s86   0.931515830
## s87   0.926764603
## s88   0.958565273
## s89   0.963250387
## s90   0.951644060
```

```
## s91   0.937362688
## s92   0.954257999
## s93   0.892485444
## s94   0.879537700
## s95   0.879946892
## s96   0.890633648
## s97   0.917134959
## s98   0.925991769
## s99   0.884247686
## s100  0.846648167
## s101  0.833696369
## s102  0.800001437
## s103  0.807934782
## s104  0.819343668
## s105  0.828571029
## s106  0.795608740
## s107  0.796609993
## s108  0.815503509
## s109  0.830111282
## s110  0.829086181
## s111  0.818367239
## s112  0.863958784
## s113  0.912057203
## s114  0.898308248
## s115  0.878723779
## s116  0.848971946
## s117  0.813891909
## s118  0.846821392
## s119  0.819121827
## s120  0.851036184
## s121  0.820416491
## s122  0.851581233
## s123  0.874038407
```

From all 4 exponential models, there is no significant trend across 20 years. In conclustion, we can not prove that the unofficial summer ends is getting later across years.

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.

Use linear regression to build the cost model of logistics. To estimated cost, will need parameter like the weight, the requested time of labor to process for different destination, the distance of delivery, and based rate of vendor. Use historical data to find the best parameter and then use the model to estimate the future cost.

Question 8.2

Using crime data from http://www.statsci.org/data/general/uscrime.txt (file uscrime.txt, description at http://www.statsci.org/data/general/uscrime.html), use regression (a useful R function is lm or glm) to predict the observed crime rate in a city with the following data: M = 14.0 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 = 3.6 Wealth = 3200 Ineq = 20.1

Prob = 0.04 Time = 39.0 *Show your model (factors used and their coefficients), the software output, and the quality of fit.*

Note that because there are only 47 data points and 15 predictors, you'll probably notice some overfitting. We'll see ways of dealing with this sort of problem later in the course.

```
# Read us crime data file into R dataframe
df_crime <- read.table("Wk3/uscrime.txt", header=TRUE)

# Data Explore
head(df_crime)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6
##       Prob    Time Crime
## 1 0.084602 26.2011   791
## 2 0.029599 25.2999  1635
## 3 0.083401 24.3006   578
## 4 0.015801 29.9012  1969
## 5 0.041399 21.2998  1234
## 6 0.034201 20.9995   682
```

```
str(df_crime)
```

```
## 'data.frame':    47 obs. of  16 variables:
##  $ M     : num  15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...
##  $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
##  $ Ed    : num  9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...
##  $ Po1   : num  5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...
##  $ Po2   : num  5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...
##  $ LF    : num  0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553 0.632 ...
##  $ M.F   : num  95 101.2 96.9 99.4 98.5 ...
##  $ Pop   : int  33 13 18 157 18 25 4 50 39 7 ...
##  $ NW    : num  30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...
##  $ U1    : num  0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081 0.1 ...
##  $ U2    : num  4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...
##  $ Wealth: int  3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...
##  $ Ineq  : num  26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...
##  $ Prob  : num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
##  $ Time  : num  26.2 25.3 24.3 29.9 21.3 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

```
summary(df_crime)
```

```
##        M               So               Ed             Po1
##  Min.   :11.90   Min.   :0.0000   Min.   : 8.70   Min.   : 4.50
##  1st Qu.:13.00   1st Qu.:0.0000   1st Qu.: 9.75   1st Qu.: 6.25
##  Median :13.60   Median :0.0000   Median :10.80   Median : 7.80
```

```
##  Mean   :13.86   Mean   :0.3404   Mean   :10.56   Mean   : 8.50
##  3rd Qu.:14.60   3rd Qu.:1.0000   3rd Qu.:11.45   3rd Qu.:10.45
##  Max.   :17.70   Max.   :1.0000   Max.   :12.20   Max.   :16.60
##       Po2              LF              M.F              Pop
##  Min.   : 4.100   Min.   :0.4800   Min.   : 93.40   Min.   :  3.00
##  1st Qu.: 5.850   1st Qu.:0.5305   1st Qu.: 96.45   1st Qu.: 10.00
##  Median : 7.300   Median :0.5600   Median : 97.70   Median : 25.00
##  Mean   : 8.023   Mean   :0.5612   Mean   : 98.30   Mean   : 36.62
##  3rd Qu.: 9.700   3rd Qu.:0.5930   3rd Qu.: 99.20   3rd Qu.: 41.50
##  Max.   :15.700   Max.   :0.6410   Max.   :107.10   Max.   :168.00
##        NW              U1                U2            Wealth
##  Min.   : 0.20   Min.   :0.07000   Min.   :2.000   Min.   :2880
##  1st Qu.: 2.40   1st Qu.:0.08050   1st Qu.:2.750   1st Qu.:4595
##  Median : 7.60   Median :0.09200   Median :3.400   Median :5370
##  Mean   :10.11   Mean   :0.09547   Mean   :3.398   Mean   :5254
##  3rd Qu.:13.25   3rd Qu.:0.10400   3rd Qu.:3.850   3rd Qu.:5915
##  Max.   :42.30   Max.   :0.14200   Max.   :5.800   Max.   :6890
##       Ineq            Prob              Time           Crime
##  Min.   :12.60   Min.   :0.00690   Min.   :12.20   Min.   : 342.0
##  1st Qu.:16.55   1st Qu.:0.03270   1st Qu.:21.60   1st Qu.: 658.5
##  Median :17.60   Median :0.04210   Median :25.80   Median : 831.0
##  Mean   :19.40   Mean   :0.04709   Mean   :26.60   Mean   : 905.1
##  3rd Qu.:22.75   3rd Qu.:0.05445   3rd Qu.:30.45   3rd Qu.:1057.5
##  Max.   :27.60   Max.   :0.11980   Max.   :44.00   Max.   :1993.0
```

```r
# Return the column names of data frame in order to build lm model next
colnames(df_crime)
```

```
##  [1] "M"      "So"     "Ed"     "Po1"    "Po2"    "LF"     "M.F"
##  [8] "Pop"    "NW"     "U1"     "U2"     "Wealth" "Ineq"   "Prob"
## [15] "Time"   "Crime"
```

```r
# In the data, the crime column is the response as Y, all the rest columns are variable X
# To build the model use LM
model_lm <- lm(Crime ~ M+So+Ed+Po1+Po2+LF+M.F+NW+U1+U2+Wealth+Ineq+Prob+Time, data=df_crime)

model_lm
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + NW +
##     U1 + U2 + Wealth + Ineq + Prob + Time, data = df_crime)
##
## Coefficients:
## (Intercept)            M           So           Ed          Po1
##  -6.216e+03    8.978e+01   -4.105e+00    1.888e+02    1.896e+02
##         Po2           LF          M.F           NW           U1
##  -1.122e+02   -7.845e+02    2.188e+01    4.335e+00   -6.253e+03
##          U2       Wealth         Ineq         Prob         Time
##   1.695e+02    9.103e-02    6.745e+01   -4.858e+03   -4.456e+00
```

```r
# evaluate model lm
summary(model_lm)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + NW +
##     U1 + U2 + Wealth + Ineq + Prob + Time, data = df_crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -380.91 -101.89  -14.77  110.87  505.40
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.216e+03  1.560e+03  -3.986 0.000364 ***
## M            8.978e+01  4.113e+01   2.183 0.036491 *
## So          -4.105e+00  1.472e+02  -0.028 0.977921
## Ed           1.888e+02  6.142e+01   3.074 0.004293 **
## Po1          1.896e+02  1.048e+02   1.808 0.079978 .
## Po2         -1.122e+02  1.161e+02  -0.966 0.341042
## LF          -7.845e+02  1.439e+03  -0.545 0.589392
## M.F          2.188e+01  1.857e+01   1.178 0.247367
## NW           4.335e+00  6.408e+00   0.676 0.503623
## U1          -6.253e+03  4.099e+03  -1.525 0.136993
## U2           1.695e+02  8.140e+01   2.083 0.045351 *
## Wealth       9.103e-02  1.022e-01   0.891 0.379612
## Ineq         6.745e+01  2.176e+01   3.099 0.004026 **
## Prob        -4.858e+03  2.248e+03  -2.161 0.038299 *
## Time        -4.456e+00  6.882e+00  -0.648 0.521894
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 206.8 on 32 degrees of freedom
## Multiple R-squared:  0.801,  Adjusted R-squared:  0.714
## F-statistic: 9.202 on 14 and 32 DF,  p-value: 1.301e-07
```

```r
# Fit the input data into model

# Create predict input as given
predict_input <-data.frame(M = 14.0,So = 0, Ed = 10.0, Po1 = 12.0, Po2 = 15.5,LF = 0.640, M.F = 94.0, P

# Fit into lm model
predict_crime <- predict(model_lm, predict_input)

predict_crime
```

```
##        1
## 162.4041
```

From above, the predicted crime is 162.4041.

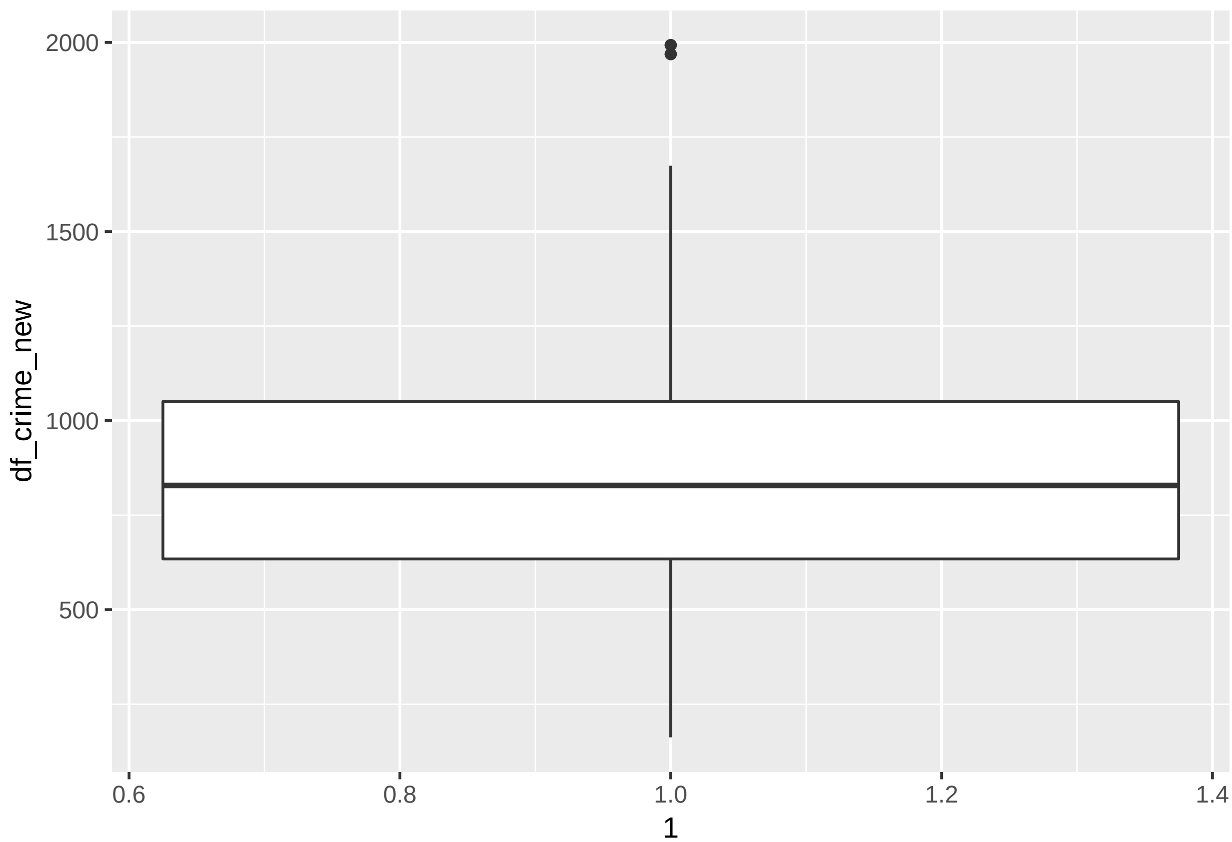
To use the solution of 5.1 to evaluate if the predicted value is outlier because the value seems too low

First to add the new value to the existing crime data to draw boxplot

```r
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from 
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```r
df_crime_new <-c(df_crime[,16], 162.4041)
qplot(y=df_crime_new, x= 1, geom = "boxplot")
```



Then use Grubbs Test to find the outliers

```r
# use type = 10 to test for one outlier
library(outliers)
test_11 <- grubbs.test(df_crime_new, type = 11)
test_11
```

```
## 
##  Grubbs test for two opposite outliers
## 
## data:  df_crime_new
## G = 4.60691, U = 0.76427, p-value = 0.7254
## alternative hypothesis: 162.4041 and 1993 are outliers
```

From the above test, it shows the predicted value has a good chance to be as an outlier because it is too low.

Update the lm model, noticed from the summary that p value of different factors are all different. Choose the factors with small p-value: M, So, Ed, Po1, U2, Ineq, Prob

```
model_lm_2 <- lm(Crime ~ M+So+Ed+Po1+U2+Ineq+Prob, data=df_crime)
model_lm_2
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + U2 + Ineq + Prob, data = df_crime)
##
## Coefficients:
## (Intercept)            M           So           Ed          Po1
##    -4959.30        99.51        73.01       205.24       111.96
##          U2         Ineq         Prob
##       88.48        63.87     -4223.04
```

```
summary(model_lm_2)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + U2 + Ineq + Prob, data = df_crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -480.55  -79.12  -14.81  122.64  562.66
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4959.30     914.45  -5.423 3.27e-06 ***
## M              99.51      34.55   2.880 0.006422 **
## So             73.01     110.11   0.663 0.511200
## Ed            205.24      46.97   4.370 8.94e-05 ***
## Po1           111.96      14.60   7.666 2.65e-09 ***
## U2             88.48      41.22   2.147 0.038105 *
## Ineq           63.87      15.15   4.216 0.000143 ***
## Prob        -4223.04    1664.87  -2.537 0.015310 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.1 on 39 degrees of freedom
## Multiple R-squared:  0.7685, Adjusted R-squared:  0.7269
## F-statistic: 18.49 on 7 and 39 DF,  p-value: 1.368e-10
```

Use the new model to do the prediction

```
predict_crime_2 <- predict(model_lm_2, predict_input)

predict_crime_2
```

```
##        1
## 1263.083
```

The new data seems a better range. Double check on the outlier

```
df_crime_new <-c(df_crime[,16], 1263.083)
test_2 <- grubbs.test(df_crime_new, type = 11)
test_2
```

```
##
##  Grubbs test for two opposite outliers
##
## data:  df_crime_new
## G = 4.27610, U = 0.78612, p-value = 1
## alternative hypothesis: 342 and 1993 are outliers
```

From above, the new predicted value is not an outlier.

*Show model output and quality of fit

```
summary(model_lm_2)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + U2 + Ineq + Prob, data = df_crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -480.55  -79.12  -14.81  122.64  562.66
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4959.30     914.45  -5.423 3.27e-06 ***
## M              99.51      34.55   2.880 0.006422 **
## So             73.01     110.11   0.663 0.511200
## Ed            205.24      46.97   4.370 8.94e-05 ***
## Po1           111.96      14.60   7.666 2.65e-09 ***
## U2             88.48      41.22   2.147 0.038105 *
## Ineq           63.87      15.15   4.216 0.000143 ***
## Prob        -4223.04    1664.87  -2.537 0.015310 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.1 on 39 degrees of freedom
## Multiple R-squared:  0.7685, Adjusted R-squared:  0.7269
## F-statistic: 18.49 on 7 and 39 DF,  p-value: 1.368e-10
```

The predicted crime is 1263.083

The R-squared is over 0.72 which means the linear model can explain over 72% of the prediction.