HW4

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

I work in the automotive industry for a truck manufacturing company. We have different customers: from Fedex, JB Hunt, Amazon etc… Among our customers, let's take a company who is an ice cream cones supplier in the United States. If we focus on the northeast region, there will be little to no ice cream cones from October to March and the peak season might be during the summer (June to August). To forecast ice cream cone orders for the next year, this supplier can implement an exponential smoothing model. He is in a niche market and has heavy exposure to seasonality. Therefore, exponential smoothing will be able to factor in seasonality accurately. At a minimum the following data will be needed: temperature for major US cities in the northeast, order history, and historical weather forecast. Since the randomness of natural factors such as temperature and weather conditions impact the demand for the ice cream cones, I believe alpha will be closer to 0.

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 (harder to use, but more general). 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).

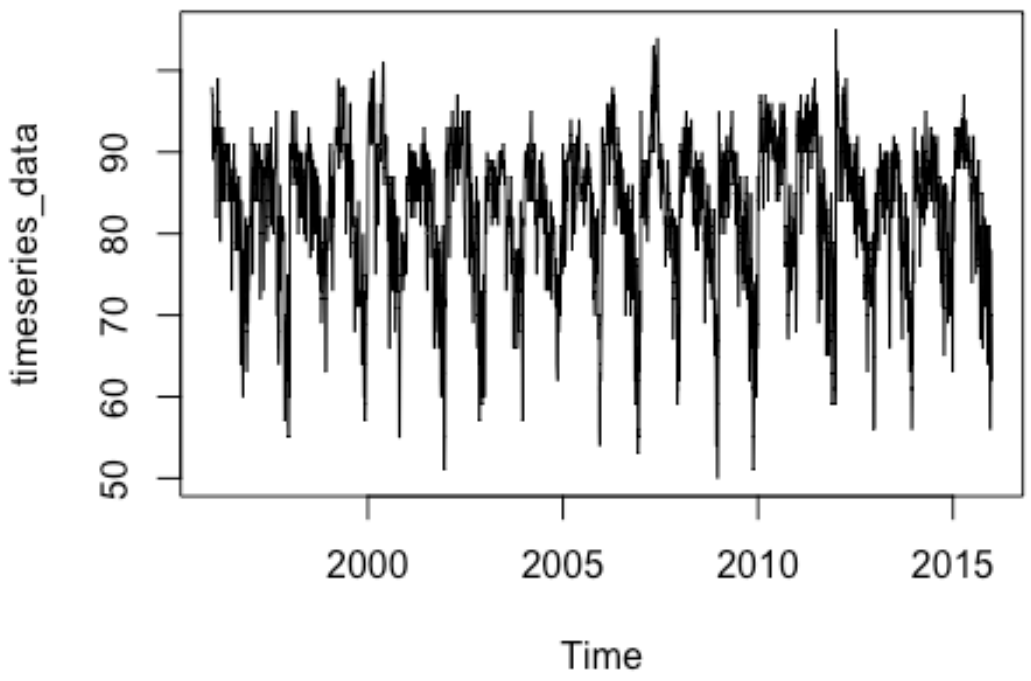
If I understand correctly, the main objective of this question is to use the HoltWinters algorithm, to perform exponential smoothing and better answer Q6.2(b) from Week 3 homework. Now we know that the main aim of exponential smoothing is to remove noise (dealing with outliers in data). We need to use exponential smoothing to somehow find out if the unofficial end of summer has gotten later over the 20 years.

I performed a single, double, and triple exponential smoothing, plotted the graphs, and printed the SSE for each model. One thing I noticed was that the SSE value was lowest for the Single exponential smoothing model. I don't know if it means the model is more accurate, the low SSE value could be due to overfitting. Alpha is the lowest for the triple exponential model multiple variants, while beta is 0 for both the additive and multiplicative variant of the triple exponential smoothing model. In other words, the trend is 0, suggesting there is not a significant increase or decrease over the years.

As commented above, there are two variants to the triple exponential model that differ in the nature of the seasonal component. The additive method is preferred when seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series(link). Given our data, representing temperatures

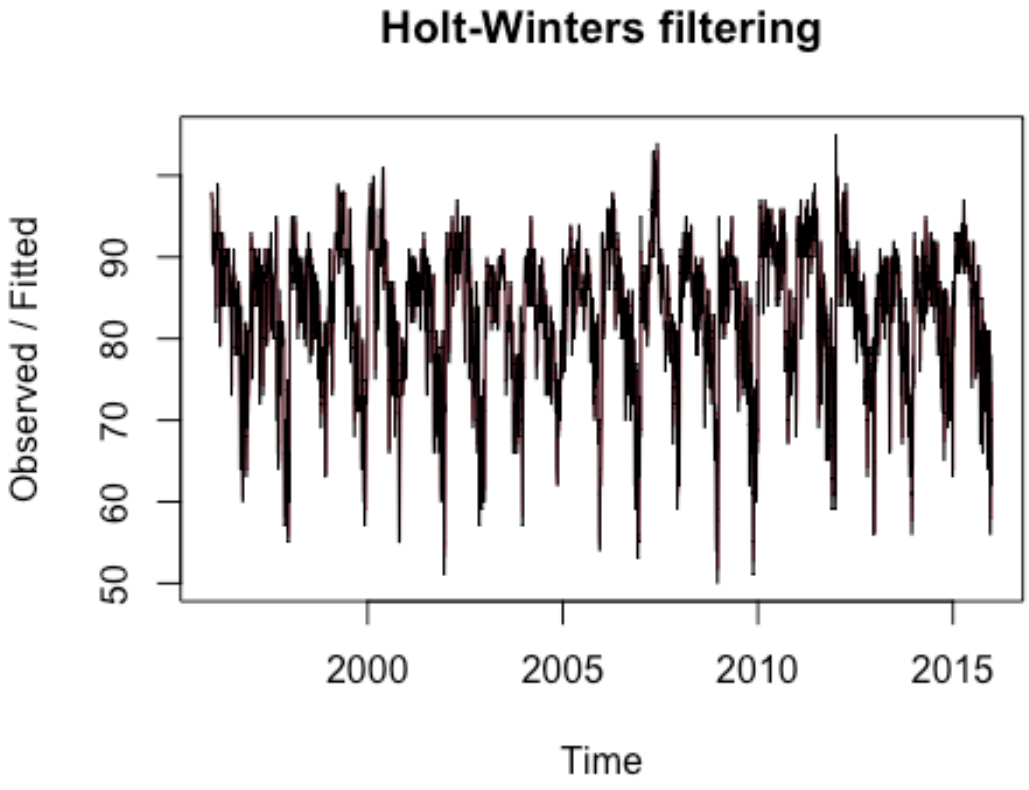
between July and October, I will adopt the additive method since there seem to be roughly constant seasonal variations. After smoothing, I will proceed with the cusum of the xhat value, then the level value and compare the plots of the approximative summer dates with my raw data (note that xhat = lebel+season+trend)

```
#Loading data, using the unlist function in R to convert the data into a vect
or before converting it into a timeseries object. This is basically stringing
all the years to obtain a single longer time series buy carefully setting the
frequency to 123 (there are 123 days between July 1st and October 31st)
temps_data<-read.table("temps.txt",header = TRUE,stringsAsFactors = FALSE,che
ck.names = FALSE)
temps_data_timeseries = as.vector( unlist( temps_data[ , 2:21 ] ) )
timeseries_data = ts(temps_data_timeseries,start = c(1996,1), frequency = 123
)
plot(timeseries_data)
```



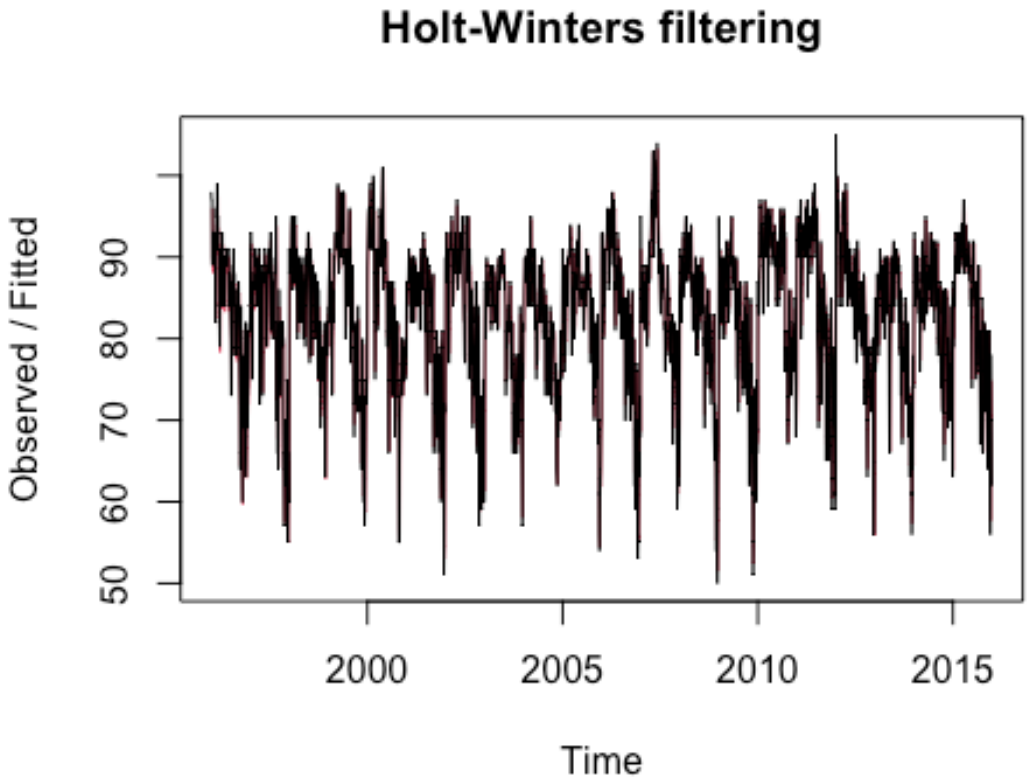

```
#Performing a single, double and triple exponential smoothing on our time ser
ies data. I know the triple exponential smoothing is actually the HoltWinters
algotrithm, but I would like to try them all to see how my time series data i
s performing, and I will print the sum of squares error for each model.
single_exp <- HoltWinters(timeseries_data, beta = F, gamma = F)
#single_exp$fitted
```

```
Fitted_data_single_exp = ts(timeseries_data[1:123],single_exp$fitted[,1], frequency=123)
print(single_exp$SSE)
## [1] 56198.1
print(single_exp)
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = timeseries_data, beta = F, gamma = F)
##
## Smoothing parameters:
##  alpha: 0.8388021
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 63.30952
plot(single_exp)
```




```
#plot(single_expo)
```

```
double_exp <- HoltWinters(timeseries_data,gamma = F)
#double_exp$fitted
Fitted_data_double_exp = ts(timeseries_data[1:123],double_exp$fitted[,1], fre
quency=123)
print(double_exp$SSE)
## [1] 56572.54
print(double_exp)
## Holt-Winters exponential smoothing with trend and without seasonal compone
nt.
##
## Call:
## HoltWinters(x = timeseries_data, gamma = F)
##
## Smoothing parameters:
##  alpha: 0.8445729
##  beta : 0.003720884
##  gamma: FALSE
##
## Coefficients:
##         [,1]
## a 63.2530022
## b -0.0729933
plot(double_exp)
```

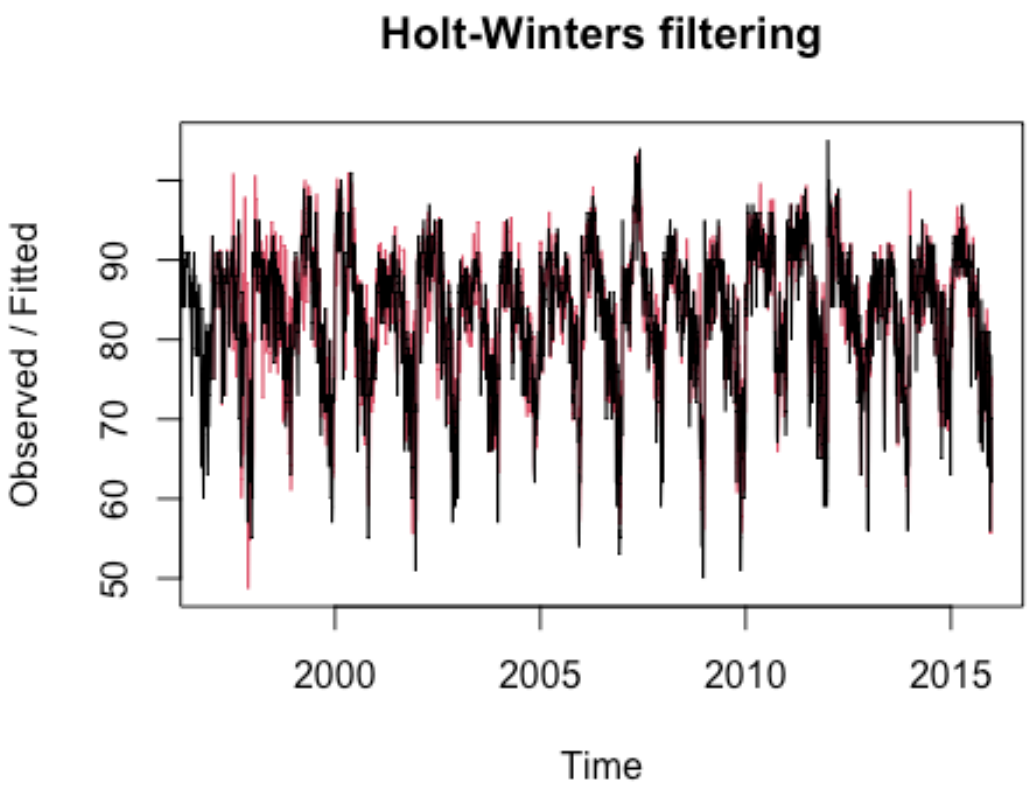


```r
#plot(double_exp)

triple_exp_a <- HoltWinters(timeseries_data, seasonal = 'additive')
#triple_exp_a$fitted
Fitted_data_triple_exp_a=ts(timeseries_data[1:123],triple_exp_a$fitted[,1], f
requency=123)
print(triple_exp_a$SSE)
## [1] 66244.25
print(triple_exp_a)
## Holt-Winters exponential smoothing with trend and additive seasonal compon
ent.
##
## Call:
## HoltWinters(x = timeseries_data, seasonal = "additive")
##
## 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
plot(triple_exp_a)
```



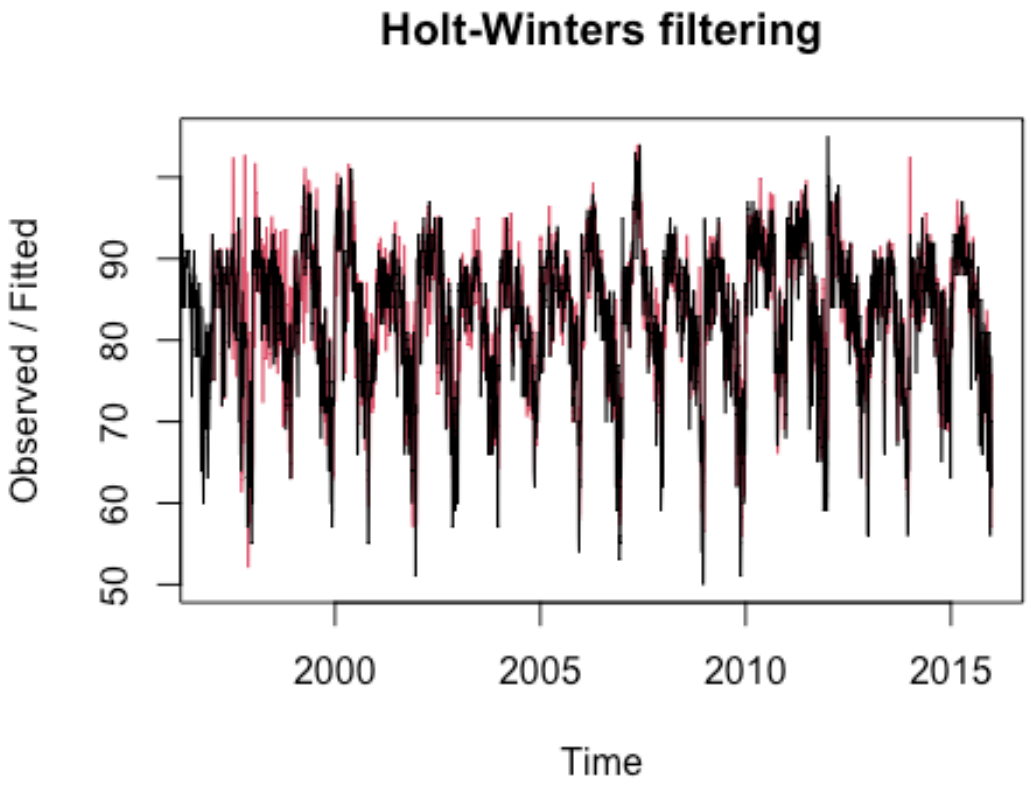

```
#plot(triple_expo_a)

triple_exp_m <- HoltWinters(timeseries_data, seasonal = 'multiplicative')
#triple_expo_m$fitted
Fitted_data_triple_exp_m=ts(timeseries_data[1:123],triple_exp_m$fitted[,1], f
requency=123)
print(triple_exp_m$SSE)
## [1] 68904.57
print(triple_exp_m)
## Holt-Winters exponential smoothing with trend and multiplicative seasonal
component.
##
## Call:
## HoltWinters(x = timeseries_data, 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
plot(triple_exp_m)
```




```
#I used the same code from my homework 3, in this case, I added the rows numb
er for the approx end of summer as an output in order to make a graph later
temps_data2<-read.table("temps.txt",header = TRUE,stringsAsFactors = FALSE,ch
eck.names = FALSE)
Std_dev = array()
i=1
#This value is the average temperature for each year, I used July for my cont
rol data, because I am aassuming that July is still in the summer usually.
data_avg= colMeans(temps_data2[1:31,2:21])
Std_dev=sd(data_avg)
#Setting my C and my T arbitrary, I did not want a very large C and I chose a
T big enough to trigger when S>T
#Given that we want to evaluate when summer ends unofficially, I assumed that
the threshold should be big enough to account for those small temperature var
iations during the summer
Std_dev
## [1] 2.486898
C=Std_dev/2
T=5*Std_dev
C
## [1] 1.243449
T
## [1] 12.43449
```

```r
#Looping to run a CUSUM on each year from 1996 to 2015
for(col in names(temps_data2)[2:ncol(temps_data2)]){
  data=temps_data2[col]
  #average temp in july for each year
  mu=mean(data[1:31,])
  S=0
  for(i in seq(1:dim(data)[1]))
  {
    x=data[i,]
    #using this equation to assess decrease
    S=max(0,S+(mu-x-C))

    if(S>T)
    {
      cat(col,'average temperature',mu,'Approx end summer date: ',temps_data2
[i,1],'row',i,'\n')
      break
    }
  }
}
```

```
## 1996 average temperature 91.19355 Approx end summer date:  27-Jul row 27
## 1997 average temperature 87.25806 Approx end summer date:  7-Jul row 7
## 1998 average temperature 89.70968 Approx end summer date:  1-Aug row 32
## 1999 average temperature 87.64516 Approx end summer date:  12-Jul row 12
## 2000 average temperature 91.74194 Approx end summer date:  25-Jul row 25
## 2001 average temperature 86.74194 Approx end summer date:  1-Sep row 63
## 2002 average temperature 89.25806 Approx end summer date:  12-Jul row 12
## 2003 average temperature 85.58065 Approx end summer date:  2-Jul row 2
## 2004 average temperature 87.83871 Approx end summer date:  8-Aug row 39
## 2005 average temperature 86.93548 Approx end summer date:  7-Jul row 7
## 2006 average temperature 90.19355 Approx end summer date:  7-Jul row 7
## 2007 average temperature 86.41935 Approx end summer date:  16-Sep row 78
## 2008 average temperature 89.16129 Approx end summer date:  12-Aug row 43
## 2009 average temperature 86.64516 Approx end summer date:  22-Jul row 22
## 2010 average temperature 91.25806 Approx end summer date:  3-Jul row 3
## 2011 average temperature 91.93548 Approx end summer date:  16-Jul row 16
## 2012 average temperature 94.09677 Approx end summer date:  13-Jul row 13
## 2013 average temperature 84.70968 Approx end summer date:  4-Jul row 4
## 2014 average temperature 86.6129 Approx end summer date:  20-Jul row 20
## 2015 average temperature 90.06452 Approx end summer date:  3-Jul row 3
```

```r
#Using the xhat data to compute the triple exponential matrix, I noticed that
that the 1996 column was removed since HoltWinters only gives predicted value
s after the first column
head(triple_exp_a$fitted)
```

```
##          xhat    level        trend    season
## [1,] 87.17619 82.87739 -0.004362918  4.303159
## [2,] 90.32925 82.09550 -0.004362918  8.238119
```

```
## [3,] 92.96089 81.87348 -0.004362918 11.091777
## [4,] 90.93360 81.89497 -0.004362918  9.042997
## [5,] 83.99752 81.93450 -0.004362918  2.067387
## [6,] 84.04358 81.93177 -0.004362918  2.116168
triple_exp_a_matrix <- matrix(triple_exp_a$fitted[,1], nrow = 123)
colnames(triple_exp_a_matrix)<-c(1997:2015)
rownames(triple_exp_a_matrix)<-as.vector(t(temps_data2[,1]))
dim(triple_exp_a_matrix)
## [1] 123  19
head(triple_exp_a_matrix)
##           1997     1998     1999     2000     2001     2002     2003     2
004
## 1-Jul 87.17619 65.99606 89.75466 83.65738 87.44361 79.13415 74.12006 86.89
937
## 2-Jul 90.32925 86.63516 85.05435 86.86490 84.58717 86.97737 72.37889 84.72
274
## 3-Jul 92.96089 90.46471 85.78686 93.25382 88.90811 90.78638 78.44689 82.61
629
## 4-Jul 90.93360 88.77120 84.90009 91.79686 87.08965 87.47605 84.41830 83.69
406
## 5-Jul 83.99752 83.25107 81.12482 89.31287 81.18354 86.29410 84.37192 84.18
777
## 6-Jul 84.04358 88.40825 85.51084 91.53539 81.69877 88.16103 78.52036 87.15
484
##           2005     2006     2007     2008     2009     2010     2011     2
012
## 1-Jul 91.15242 79.17602 82.08626 85.64527 81.25471 87.07795 92.70889 83.33
264
## 2-Jul 92.36299 88.94595 89.18548 80.16002 86.86767 81.30538 87.10638 94.13
220
## 3-Jul 91.99930 92.92709 86.87669 84.99336 89.07355 82.54299 90.64593 91.82
838
## 4-Jul 87.06963 93.05548 83.28236 90.20184 88.94787 83.21820 94.17789 95.83
964
## 5-Jul 84.32874 90.87864 84.51007 89.66314 89.58972 81.21594 90.61737 95.47
744
## 6-Jul 85.91322 86.97635 82.41612 84.39436 78.92613 85.11707 89.01132 97.60
932
##           2013     2014     2015
## 1-Jul 87.07333 98.76579 89.08086
## 2-Jul 75.36736 87.73317 84.11698
## 3-Jul 81.93827 88.12055 81.57585
## 4-Jul 76.22184 87.01128 79.10413
## 5-Jul 75.44617 85.16544 83.96091
## 6-Jul 78.70689 83.29378 82.17153
tail(triple_exp_a_matrix)
##            1997     1998     1999     2000     2001     2002     2003
2004
## 26-Oct 72.87474 73.16935 63.23398 79.11577 76.66589 62.28070 76.05719 77.3
8524
```

```
## 27-Oct 71.63919 74.32439 64.10169 74.53788 68.04062 64.88274 69.21285 77.5
3701
## 28-Oct 67.90864 83.72860 75.62795 78.28388 57.86329 71.69443 69.68487 82.1
6193
## 29-Oct 60.33835 85.07493 77.86605 82.70795 57.26434 76.99448 63.29186 85.5
0999
## 30-Oct 62.73017 80.28623 75.73769 80.44802 64.79360 79.72510 69.71478 80.6
7287
## 31-Oct 63.83046 78.85423 72.35307 75.98847 68.53345 69.94630 72.87967 76.1
6107
##            2005     2006     2007     2008     2009     2010     2011
2012
## 26-Oct 57.21330 59.27248 67.59135 70.55272 72.70234 82.72033 80.72747 83.6
6667
## 27-Oct 59.19225 60.55571 67.91700 69.96211 65.18098 82.33165 74.54407 79.2
5647
## 28-Oct 62.43034 63.97815 65.77415 62.26793 61.03195 80.63008 79.91056 73.3
6289
## 29-Oct 66.90088 67.56336 73.94622 56.11181 72.78482 79.76873 78.43663 61.4
5383
## 30-Oct 68.75353 74.93308 69.13694 63.18631 79.18687 73.66484 69.19173 63.8
3377
## 31-Oct 67.97415 73.08971 66.49032 64.55886 70.77600 74.62618 67.14295 63.3
0258
##            2013     2014     2015
## 26-Oct 61.94611 79.17236 76.29092
## 27-Oct 58.15061 81.69161 68.94343
## 28-Oct 63.83154 79.46450 55.62316
## 29-Oct 66.03378 81.78931 73.03021
## 30-Oct 77.44226 79.67961 74.11555
## 31-Oct 84.47681 76.28772 75.38342
Std_dev_2 = array()
#This value is the average temperature for each year, I used July for my cont
rol data, because I am aassuming that July is still in the summer usually.
data_avg_2= colMeans(triple_exp_a_matrix[1:31,1:19])
Std_dev_2=sd(data_avg_2)
Std_dev=sd(data_avg)
#Setting my C and my T arbitrary, I did not want a very large C and I chose a
T big enough to trigger when S>T
#Given that we want to evaluate when summer ends unofficially, I assumed that
the threshold should be big enough to account for those small temperature var
iations during the summer
Std_dev_2
## [1] 2.36593
C_2=Std_dev_2/2
T_2=5*Std_dev_2
C_2
## [1] 1.182965
T_2
## [1] 11.82965
```

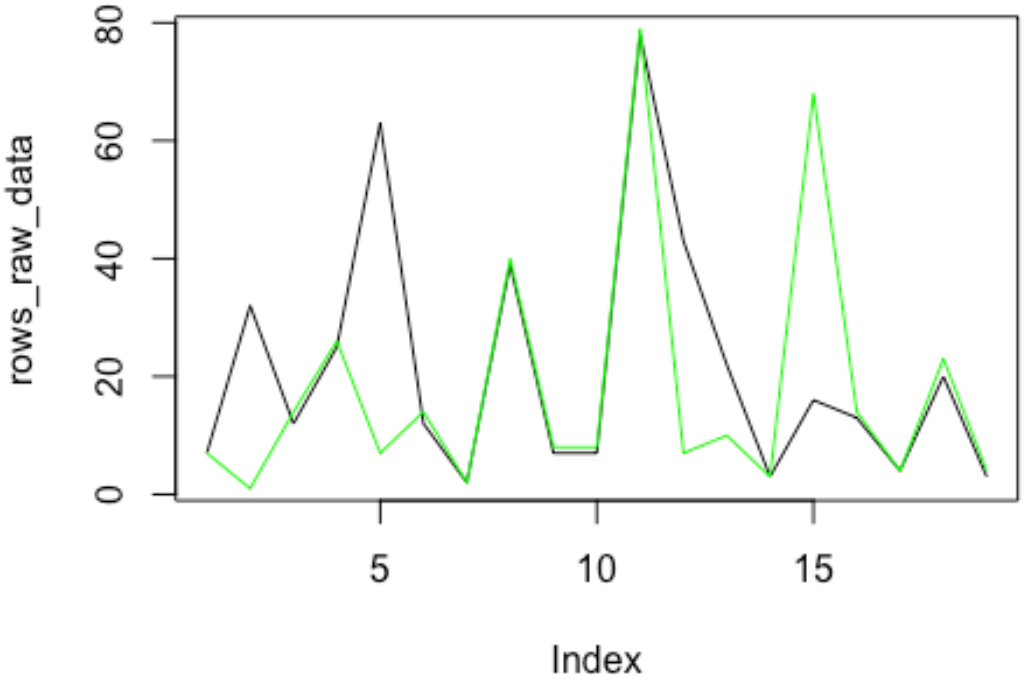
```
k=0
#Looping to run a CUSUM on each year from 1996 to 2015
for(k in 1: ncol(triple_exp_a_matrix)){
  #average temp in july for each year
  data_new=triple_exp_a_matrix[,k]
  mu_2=mean(data_new[1:31])
  S_2=0
  for(i in 1:123)
  {
    x_2=data_new[i]
    #-using this equation to assess decrease
    S_2=max(0,S_2+(mu_2-x_2-C_2))

    if(S_2>T_2)
    {
      cat("year",k,'average temperature',mu_2,'Approx end summer date: ',rown
ames(triple_exp_a_matrix)[i],'row',i,'\n')
      break
    }
  }
}
## year 1 average temperature 87.30024 Approx end summer date:  7-Jul row 7
## year 2 average temperature 88.26678 Approx end summer date:  1-Jul row 1
## year 3 average temperature 86.68264 Approx end summer date:  14-Jul row 14
## year 4 average temperature 91.22706 Approx end summer date:  26-Jul row 26
## year 5 average temperature 86.37348 Approx end summer date:  7-Jul row 7
## year 6 average temperature 88.47667 Approx end summer date:  14-Jul row 14
## year 7 average temperature 84.62334 Approx end summer date:  2-Jul row 2
## year 8 average temperature 87.45237 Approx end summer date:  9-Aug row 40
## year 9 average temperature 87.17211 Approx end summer date:  8-Jul row 8
## year 10 average temperature 89.36313 Approx end summer date:  8-Jul row 8
## year 11 average temperature 85.74592 Approx end summer date:  17-Sep row 7
9
## year 12 average temperature 88.59173 Approx end summer date:  7-Jul row 7
## year 13 average temperature 86.0606 Approx end summer date:  10-Jul row 10
## year 14 average temperature 90.31285 Approx end summer date:  3-Jul row 3
## year 15 average temperature 91.23196 Approx end summer date:  6-Sep row 68
## year 16 average temperature 93.20074 Approx end summer date:  14-Jul row 1
4
## year 17 average temperature 84.09303 Approx end summer date:  4-Jul row 4
## year 18 average temperature 86.48797 Approx end summer date:  23-Jul row 2
3
## year 19 average temperature 89.15983 Approx end summer date:  4-Jul row 4

#Table with rows numbers raw data, fitted data and corresponding years
```

Year	Unofficial end of summer (raw data) and corresponding row	Unofficial end of summer (fitted da ta) and corresponding row
1997	: 7-Jul row 7	: 7-Jul row 7
1998	1-Aug row 32	: 1-Jul row 1
1999	12-Jul row 12	14-Jul row 14
2000	25-Jul row 25	26-Jul row 26
2001	1-Sep row 63	: 7-Jul row 7
2002	12-Jul row 12	14-Jul row 14
2003	: 2-Jul row 2	: 2-Jul row 2
2004	8-Aug row 39	9-Aug row 40
2005	:7-Jul row 7	: 8-Jul row 8
2006	: 7-Jul row 7	: 8-Jul row 8
2007	16-Sep row 78	17-Sep row 79
2008	12-Aug row 43	: 7-Jul row 7
2009	22-Jul row 22	10-Jul row 10
2010	: 3-Jul row 3	: 3-Jul row 3
2011	16-Jul row 16	6-Sep row 68
2012	13-Jul row 13	14-Jul row 14
2013	: 4-Jul row 4	: 4-Jul row 4
2014	20-Jul row 20	23-Jul row 23
2015	3-Jul row 3	e 4-Jul row 4

```
#computing my rows for the fitted data and the raw data from my results above and making a plot, 1996 was removed from fitted data so removing 1996 column in my raw data
rows_fitted_data<-c(7,1,14,26,7,14,2,40,8,8,79,7,10,3,68,14,4,23,4)
rows_raw_data<-c(7,32,12,25,63,12,2,39,7,7,78,43,22,3,16,13,4,20,3)
plot(rows_raw_data, type ="l",col="black")
lines(rows_fitted_data,col="green")
```



This plot shows, that although the CUSUM of the fitted data was a little smoother than the raw data, the cyclical trend is still presents in relationship to the unofficial end of the summer across the 20 years. For example, in some years summer ended very early (like in 1997, in July), while in other years, summer was ending way later (like in 2007, September). I am not sure this data supports that the unofficial date of summer has gotten later over the last 20 years.

I chose to plot the raw data against the level data of the triple exponential matrix to see what I would look like.

```
#Using the level data to compute the triple exponential matrix
triple_exp_a_matrix <- matrix(triple_exp_a$fitted[,2], nrow = 123)
colnames(triple_exp_a_matrix)<-c(1997:2015)
rownames(triple_exp_a_matrix)<-as.vector(t(temps_data2[,1]))
dim(triple_exp_a_matrix)
## [1] 123  19
head(triple_exp_a_matrix)
##           1997     1998     1999     2000     2001     2002     2003     2
004
## 1-Jul 82.87739 61.94635 80.40983 75.53122 78.18604 70.60583 63.29068 76.30
718
```

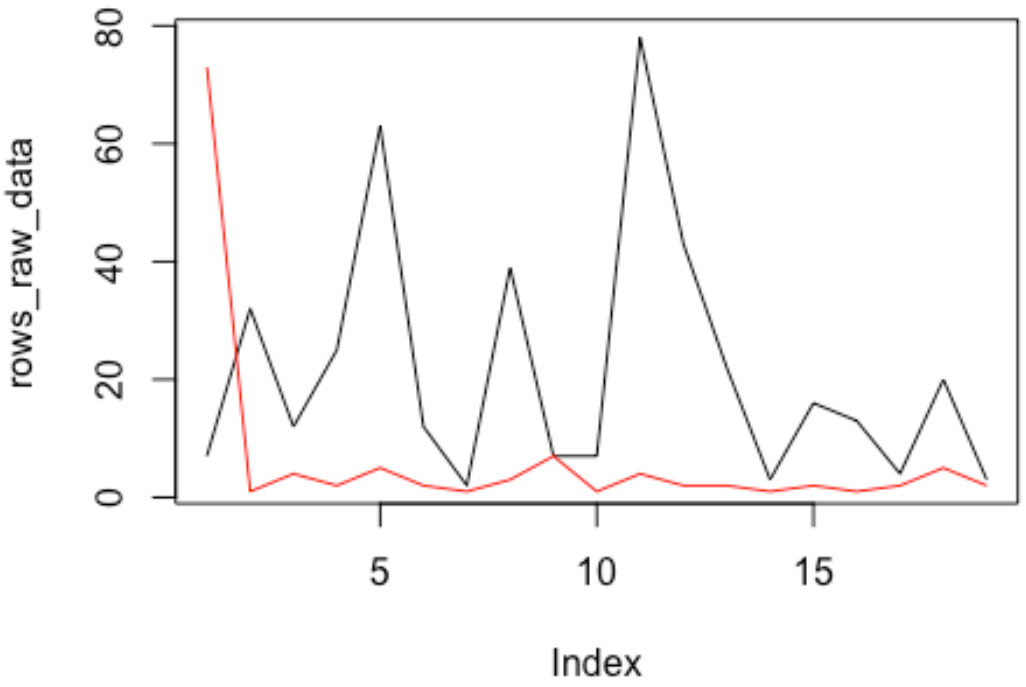
```
## 2-Jul 82.09550 78.47113 76.60128 79.05866 75.90524 77.78447 62.54588 73.06
403
## 3-Jul 81.87348 79.36901 74.57781 81.78785 77.49590 79.77825 68.24061 70.59
871
## 4-Jul 81.89497 79.71851 75.37540 81.61570 76.23016 77.27086 73.89038 72.83
118
## 5-Jul 81.93450 81.18752 77.42027 83.72881 74.18335 78.27392 74.93162 75.67
331
## 6-Jul 81.93177 86.30568 83.28295 88.14505 77.36296 82.70257 72.03715 79.51
119
##           2005     2006     2007     2008     2009     2010     2011     2
012
## 1-Jul 81.59778 69.65366 69.63638 70.46063 66.20672 69.11910 74.76655 65.54
042
## 2-Jul 81.49265 78.78780 78.16879 70.02970 75.28884 69.06321 74.29357 79.85
952
## 3-Jul 79.26514 81.46341 75.39757 74.54698 77.35514 70.84016 78.84631 79.10
670
## 4-Jul 75.29487 81.50725 72.16941 78.51338 77.30216 71.13790 81.72026 83.84
322
## 5-Jul 74.58342 80.14408 73.96158 78.37559 78.65437 72.31142 80.27618 85.26
699
## 6-Jul 77.66705 79.55888 76.26428 77.27178 72.31061 76.79174 79.86370 88.25
232
##           2013     2014     2015
## 1-Jul 64.69259 77.45943 69.63084
## 2-Jul 61.33444 71.66034 66.92878
## 3-Jul 67.69785 75.13768 68.83028
## 4-Jul 63.76792 74.39256 67.12312
## 5-Jul 64.27797 72.39756 71.01629
## 6-Jul 69.26716 72.94489 71.03777
tail(triple_exp_a_matrix)
##            1997     1998     1999     2000     2001     2002     2003
2004
## 26-Oct 79.37269 80.06432 68.89418 83.55490 81.97661 70.27370 82.83900 85.8
7333
## 27-Oct 78.12901 83.91437 72.70151 80.82976 73.59932 74.05015 77.50834 86.9
3643
## 28-Oct 68.44724 87.00088 79.90161 83.11407 62.33005 76.76755 74.05796 89.2
2132
## 29-Oct 59.90947 83.87062 78.16001 84.24416 60.43288 78.94837 65.66812 86.4
6566
## 30-Oct 62.32569 79.18929 74.27782 79.14437 64.22014 77.62553 70.09825 79.5
1355
## 31-Oct 64.48288 80.31783 73.78580 76.86066 68.97965 69.87016 74.90987 77.7
4225
##            2005     2006     2007     2008     2009     2010     2011
2012
## 26-Oct 65.35943 66.61670 74.35796 77.23279 79.49947 91.14859 88.67296 92.4
0153
```

```
## 27-Oct 67.85831 68.41540 74.62374 76.86305 74.40338 92.65123 86.20451 89.9
7327
## 28-Oct 70.37111 72.01005 74.01319 69.61205 70.97407 88.46125 89.14579 83.8
4981
## 29-Oct 70.08226 71.35907 76.80238 61.49783 77.55920 84.07400 85.23419 72.3
6750
## 30-Oct 68.16024 74.28760 68.90082 63.40274 79.01921 76.28978 72.38101 68.7
5782
## 31-Oct 68.31881 73.00535 67.48381 64.59734 70.29751 74.52379 66.96140 63.5
7485
##            2013     2014     2015
## 26-Oct 71.45746 88.88407 84.98028
## 27-Oct 70.82766 92.07108 78.83404
## 28-Oct 77.99542 93.59270 70.27328
## 29-Oct 78.10242 91.95915 85.06139
## 30-Oct 84.02528 86.14452 83.05386
## 31-Oct 84.38962 78.41921 80.32887
#Observing that the filtered series does not have year 1996
Std_dev_2 = array()
#This value is the average temperature for each year, I used July for my cont
rol data, because I am aassuming that July is still in the summer usually.
data_avg_2= colMeans(triple_exp_a_matrix[1:31,1:19])
Std_dev_2=sd(data_avg_2)

#Setting my C and my T arbitrary, I did not want a very large C and I chose a
T big enough to trigger when S>T
#Given that we want to evaluate when summer ends unofficially, I assumed that
the threshold should be big enough to account for those small temperature var
iations during the summer
Std_dev_2
## [1] 2.394791
C_2=Std_dev_2/2
T_2=5*Std_dev_2
C_2
## [1] 1.197396
T_2
## [1] 11.97396
k=0
#rows<-vector("numeric",123)
#Looping to run a CUSUM on each year from 1996 to 2015
for(k in 1: ncol(triple_exp_a_matrix)){
  #average temp in july for each year
  data_new=triple_exp_a_matrix[,k]
  mu_2=mean(data_new[1:31])
  S_2=0
  #print(S_2)
  for(i in 1:123)
  {
    x_2=data_new[i]
    #-using this equation to assess decrease
```

```
    S_2=max(0,S_2+(mu_2-x_2-C_2))

    if(S_2>T_2)
    {
      cat("year",k,'average temperature',mu_2,'Approx end summer date: ',rown
ames(triple_exp_a_matrix)[i],'row',i,'\n')
      break
    }
  }
}
## year 1 average temperature 81.94506 Approx end summer date:  11-Sep row 73
## year 2 average temperature 82.92053 Approx end summer date:  1-Jul row 1
## year 3 average temperature 81.03082 Approx end summer date:  4-Jul row 4
## year 4 average temperature 85.37141 Approx end summer date:  2-Jul row 2
## year 5 average temperature 80.40879 Approx end summer date:  5-Jul row 5
## year 6 average temperature 82.43396 Approx end summer date:  2-Jul row 2
## year 7 average temperature 78.41515 Approx end summer date:  1-Jul row 1
## year 8 average temperature 81.04145 Approx end summer date:  3-Jul row 3
## year 9 average temperature 80.67938 Approx end summer date:  7-Jul row 7
## year 10 average temperature 82.9205 Approx end summer date:  1-Jul row 1
## year 11 average temperature 79.12744 Approx end summer date:  4-Jul row 4
## year 12 average temperature 81.83064 Approx end summer date:  2-Jul row 2
## year 13 average temperature 79.17889 Approx end summer date:  2-Jul row 2
## year 14 average temperature 83.30735 Approx end summer date:  1-Jul row 1
## year 15 average temperature 84.02628 Approx end summer date:  2-Jul row 2
## year 16 average temperature 85.84608 Approx end summer date:  1-Jul row 1
## year 17 average temperature 76.54861 Approx end summer date:  2-Jul row 2
## year 18 average temperature 78.81296 Approx end summer date:  5-Jul row 5
## year 19 average temperature 81.45837 Approx end summer date:  2-Jul row 2
rows_fitted_data_level<-c(73,1,4,2,5,2,1,3,7,1,4,2,2,1,2,1,2,5,2)
plot(rows_raw_data, type ="l",col="black")
lines(rows_fitted_data_level,col="red")
```



From the graph, it is clearly noticeable that the level data alone does not accurately shape the data as accurately as xhat. This helped me understand the following explanation from one of the TAs in regard to the difference between xhat and level:

“The reason that xhat is used is it emulated a total (not detreneded and not de-seasoned) view of the temperature and retain the summer fall shape you are looking for. you can definitely make a case of maybe using the St level to assess whether or not temperature as a whole as changed (did the "water level" change as a collective). but in terms of trying to answer the question... you would likely notice that level actually loses some of the "seasonal" shapes when it's separated ”