Homework4

02/04/2019

Contents

Question 7.1

Describe a situation or problem from your job for which exponential smoothing would be appropiate. What data you need? Would you expect the value of alpha to be closer to 0 or 1 and why?

Response to Question 7.1

For instance, a magazine company wants to study the best channels to publicize their magazine subscriptions. The company has multiple channels to set the advertising. For the study they measure each channel every week and measure new subscriptions.

For each channel the company builds an exponential smoothing model, with weekly subscriptions as the value to measure (response). It would include cyclic effects where a cycle could be the renewing subscriptions every year and the trend would show if the annually subscriptors are increasing or decreasing. The company could have variability for other variables that are not taken into account here and the estimate would be better if it is updated so I would expect the value of α to be closer to 1. Why ? This parameter controls the rate at which the influence of the observations at prior time steps decay exponentially. Alpha is often set to a value between 0 and 1. Large values mean that the model pays attention mainly to the most recent past observations, whereas smaller values mean more of the history is taken into account when making a prediction.

A value close to 1 indicates fast learning (that is, only the most recent values influence the forecasts), whereas a value close to 0 indicates slow learning (past observations have a large influence on forecasts)

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.

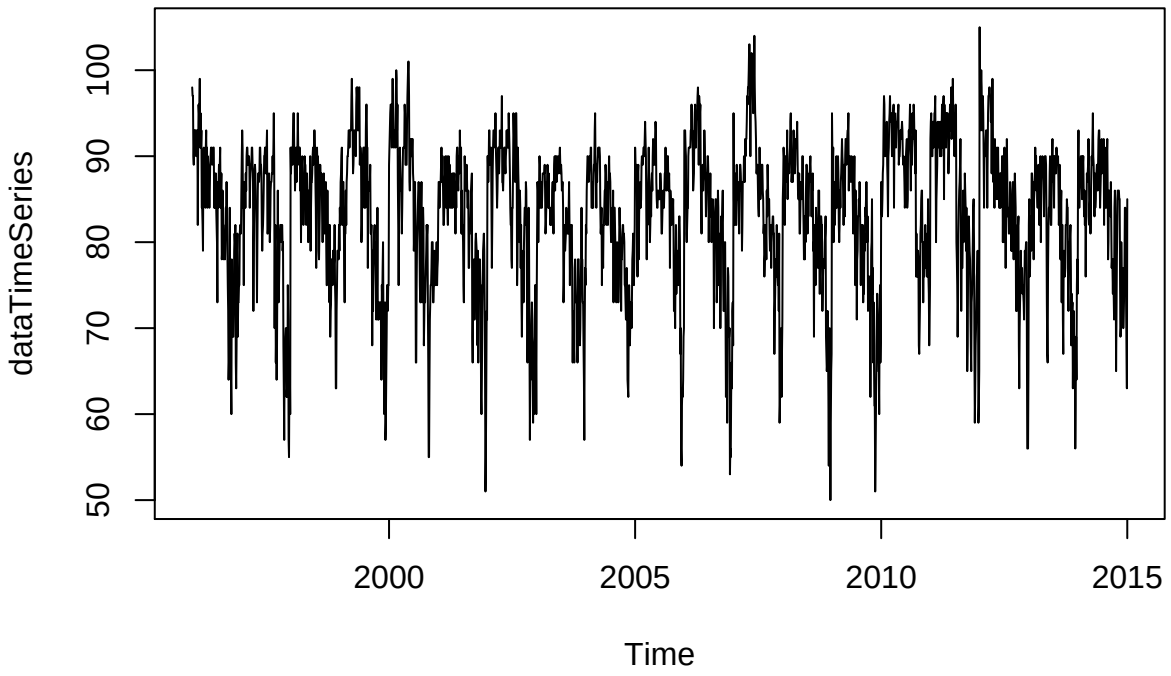
```
set.seed(42) #answer of life
# import the data
temps <- read.delim("~/Documents/R/GeorgiaTech/TimeSeries/temps.txt")
# to use Holt Winter, first it is needed to create a time series object

# the frequency is the data points per year
# start is a vector by year and month of starting data
```

```r
data <- as.vector(unlist(temps[,2:ncol(temps)])) #unlists flattens the list
dataTimeSeries <- ts(data, start = c(1996,7), frequency = 123, end = c(2015, 10))
dataTimeSeries <- ts(data, start = 1996, frequency = 123, end = 2015)

#dataTimeSeries <- ts(data, start = 1996, frequency = 123)

# lets plot the data to see
ts.plot(dataTimeSeries, col = 1:23)
```




```r
# it is difficult to see here much information because it is too much data to visualize
# but at least temps look consistent through the years.

# now test Holt Winters with single exponential smoothing without trend
result1 <- HoltWinters(dataTimeSeries, beta=FALSE, gamma = FALSE)
print (result1)
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = dataTimeSeries, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.8396301
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 81.62444
```

```r
# test double exponential smoothing with trend
result2 <- HoltWinters(dataTimeSeries, gamma = FALSE)
print (result2)
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = dataTimeSeries, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.8455303
##  beta : 0.003777803
##  gamma: FALSE
##
## Coefficients:
##           [,1]
## a 81.729657393
## b -0.004838906
```

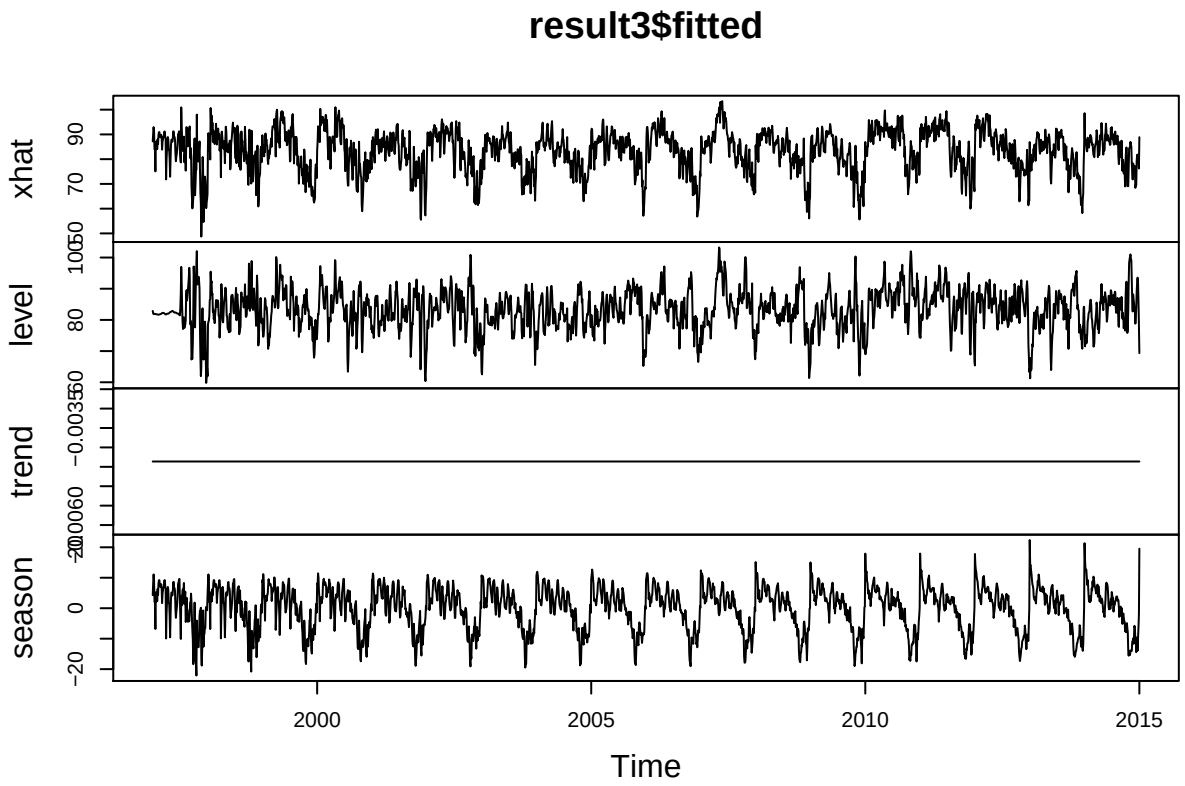
```
# test triple exponential smoothing with trend and additive seasonality
result3 <- HoltWinters(dataTimeSeries)
print (result3)
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = dataTimeSeries)
##
## Smoothing parameters:
##  alpha: 0.6677614
##  beta : 0
##  gamma: 0.6297674
##
## Coefficients:
##               [,1]
## a     66.739214602
## b     -0.004362918
## s1    17.167113056
## s2    12.692593452
## s3    11.926233267
## s4    12.862822489
## s5    11.026083880
## s6     8.860499089
## s7     9.547553333
## s8     7.755384526
## s9     4.419013466
## s10    2.272689626
## s11    4.628251667
## s12    2.396834852
## s13    3.512957136
## s14    1.734948091
## s15    3.035023890
## s16    6.257944053
## s17    5.086362292
## s18    8.599153274
## s19    5.507486014
## s20   10.404819396
## s21   10.115801978
## s22    9.628840064
```

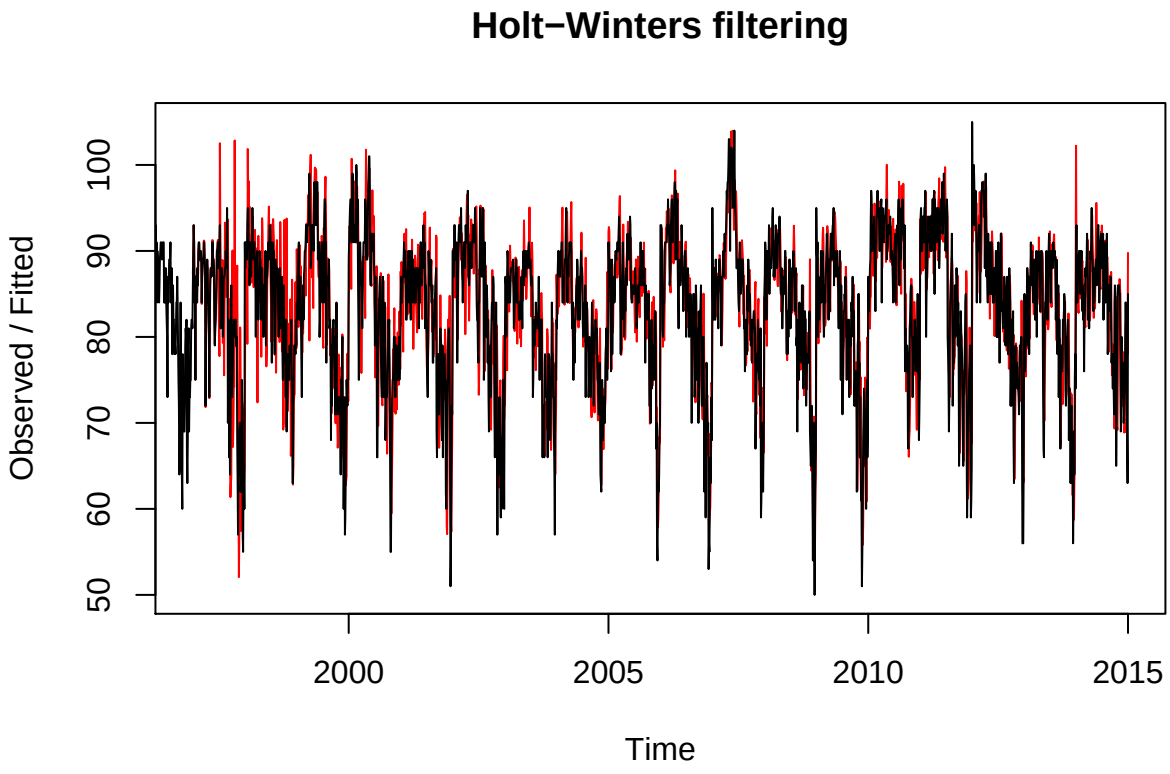
```
## s23    7.658623118
## s24    7.150473636
## s25    6.306599371
## s26    5.850691115
## s27    5.770487458
## s28    4.280481134
## s29    7.229771199
## s30    4.632381095
## s31    6.006248308
## s32    6.443645890
## s33    5.701166527
## s34    3.546887269
## s35    3.879569716
## s36    3.517339384
## s37    2.828550977
## s38    2.122971410
## s39    2.627923984
## s40    1.658896597
## s41    0.165866282
## s42   -0.001574460
## s43   -1.557500303
## s44   -2.159601227
## s45   -2.260609558
## s46    0.474052766
## s47    2.501631056
## s48    6.552191593
## s49    7.240238719
## s50    8.395899120
## s51    8.633263084
## s52    7.504540260
## s53    4.804135812
## s54    0.449902809
## s55   -1.045831475
## s56    1.562077049
## s57    1.632745190
## s58    0.857309158
## s59    2.909614779
## s60    0.626594899
## s61    4.491805650
## s62    4.567058619
## s63    3.065433531
## s64    3.787652805
## s65   -2.147135463
## s66    1.759895146
## s67    1.541155061
## s68    1.278521842
## s69    0.895959617
## s70    2.009912430
## s71    3.695537344
## s72    4.675235988
## s73    4.535880359
## s74    1.710420810
## s75    0.822675780
## s76    2.363162195
```

```
## s77    1.925012161
## s78   -1.656914701
## s79   -1.809929506
## s80   -0.427021203
## s81    0.056812125
## s82   -1.137248149
## s83   -1.037423821
## s84   -2.817503990
## s85   -4.578240308
## s86   -3.080091372
## s87   -2.710719111
## s88   -2.255335538
## s89   -4.518502545
## s90   -5.159556421
## s91   -4.440834373
## s92   -5.790113744
## s93   -7.461163074
## s94   -8.882612687
## s95   -8.619859733
## s96   -6.200719796
## s97   -6.055889182
## s98  -11.167287691
## s99  -13.489975101
## s100 -13.615536188
## s101 -14.373453486
## s102 -15.142110213
## s103 -14.419874185
## s104 -14.023613348
## s105 -16.187082843
## s106 -15.999259045
## s107 -12.074075053
## s108  -9.199729415
## s109 -10.403127076
## s110 -12.075113349
## s111  -9.722863134
## s112  -5.846856763
## s113  -8.047801338
## s114  -9.636669876
## s115 -10.510269852
## s116 -12.876648138
## s117  -8.657362442
## s118  -9.828539578
## s119 -14.522204766
## s120 -11.852457644
## s121  -8.714763993
## s122  -4.711332904
## s123  18.737998957
```

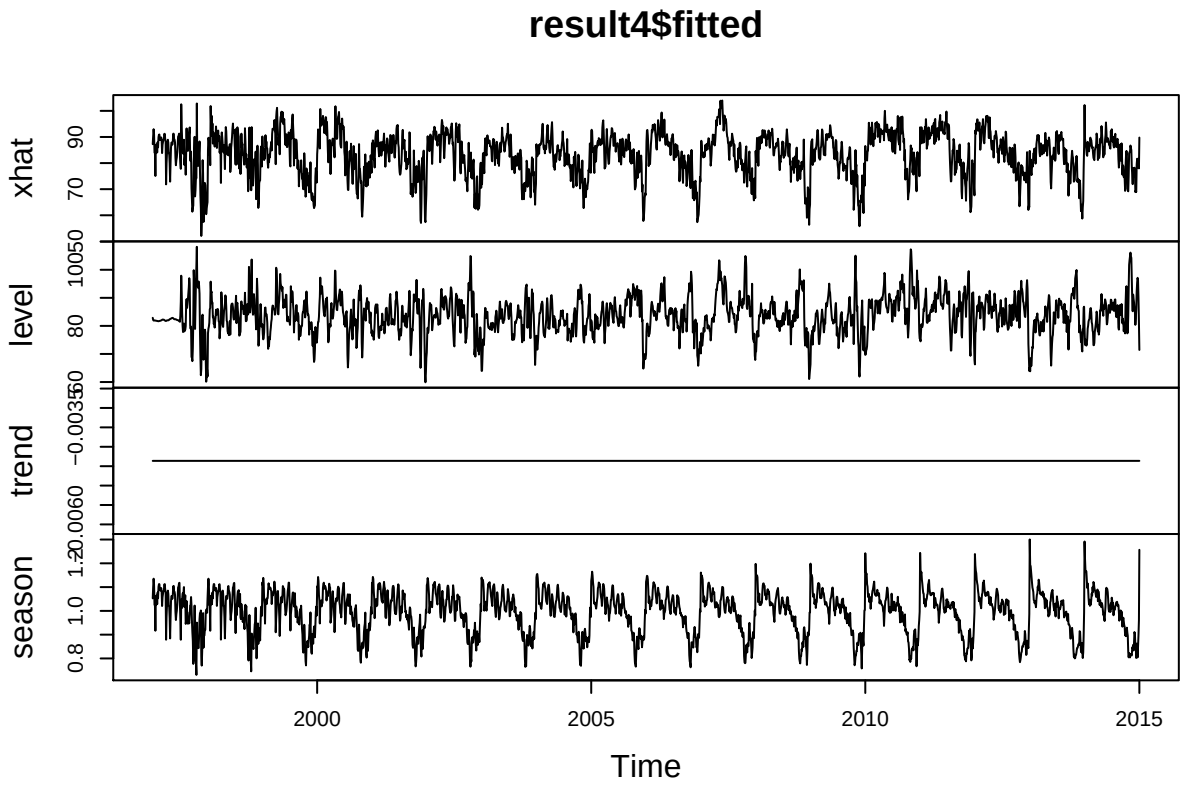
```
# the red color indicates the estimated values
plot(result3$fitted)
```




```
# triple exponential smoothing with mutiplicative seasonality
result4 <- HoltWinters(dataTimeSeries, seasonal="multiplicative")
plot(result4)
```



```r
plot(result4$fitted)
```



Analysis 7.2

First I converted the data to a times series data because the model needs it in that special format. Then applied different versions HoltWinters to apply the exponential smoothing. Here α (alpha) is the smoothing coefficient for the level, β (beta) is the smoothing coefficient for the trend and γ (gamma) is the smoothing coefficient for the seasonal component.

According to the graphs, we don't have enough evidence to assume that the unofficial end of summer hast gotten later over the years. In the result of HoltWinters, there is no trend detected and that's why the trend line is flat and the level line looks constant through the time.

To validate this statement, now it will be applied the CUSUM model to test the seasonal factors of the fitted model.

See below the discussion on which HoltWinters model I choose.

```r
options(warn=-1)
# now apply CUSUM
temps <- read.delim("~/Documents/R/GeorgiaTech/TimeSeries/temps.txt")
cusum <- function(data_mu, data, c_ratio,t_ratio) {
  # standard deviation
  data_sd <- sd(data_mu)
  mu <- mean(data_mu)
  C <- c_ratio*data_sd
  T <- t_ratio*data_sd
  # Calculate a new vector of mu - x[i] - C to detect a negative change in
  # temperature since we know temps will drop
```

```
  calc <- mu - data - C
  # Calculate a new vector called s_t using a for loop
  s_t <- c(0)
  for (i in 2:length(data)) {
    s_t[i] <- max(0,s_t[i-1]+calc[i])
  }

  for (i in 1:length(s_t)) {
    if (s_t[i] >= T) {
      result <- list("index"=i, "s"= s_t, "c"=C, "t"=T)
      return (result)
    }
  }
}
# run cusum for each year
year_result <- c(0)
start <- 1
finish <- 123
current <- 0
# the first year is 1998
for (j in 1:18) {
  begin <- start + current
  end   <- finish + current
  july <- result4$fitted[,4][begin:(begin+31)]
  current_year <- result4$fitted[,4][begin:end]
  year_result[j] <- cusum(july,current_year, 1,8)$index
  current <- current + 123
}

print (year_result)
```

```
##  [1] 91 90 87 84 84 83 83 83 81 44 43 44 44 65 59 66 78 75
```

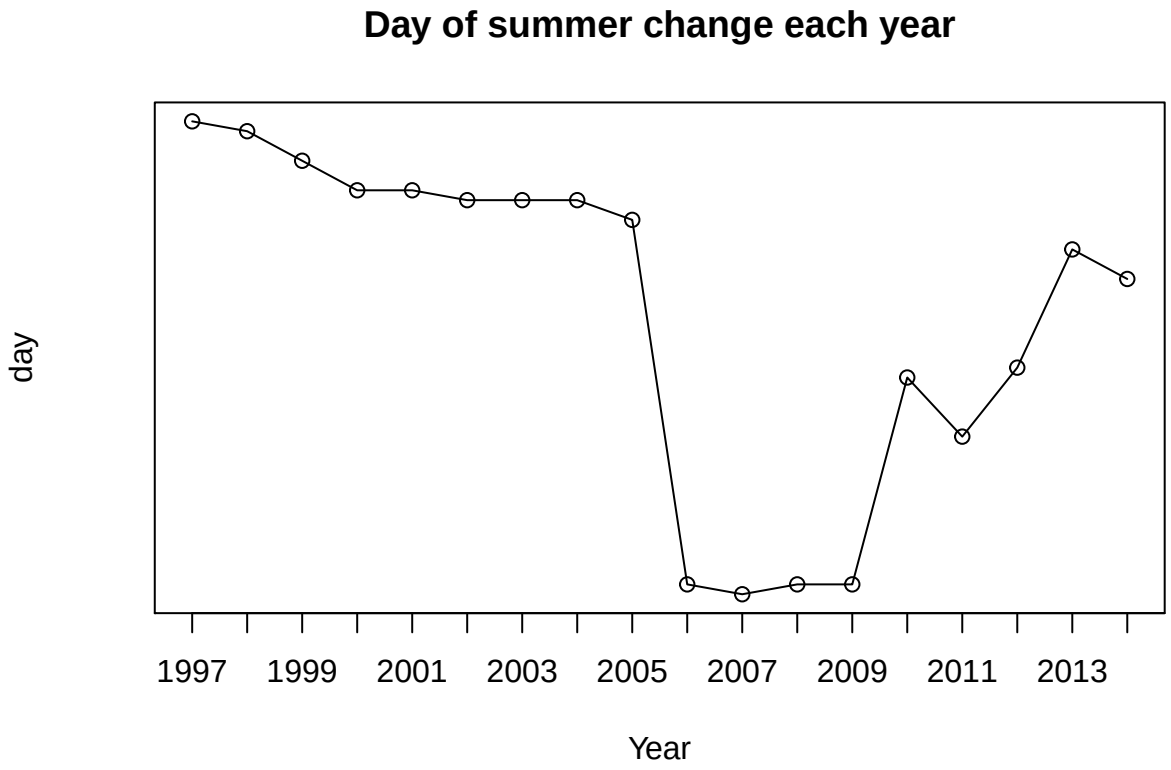
```
day_change <- mean(year_result)
print (day_change)
```

```
## [1] 71.33333
```

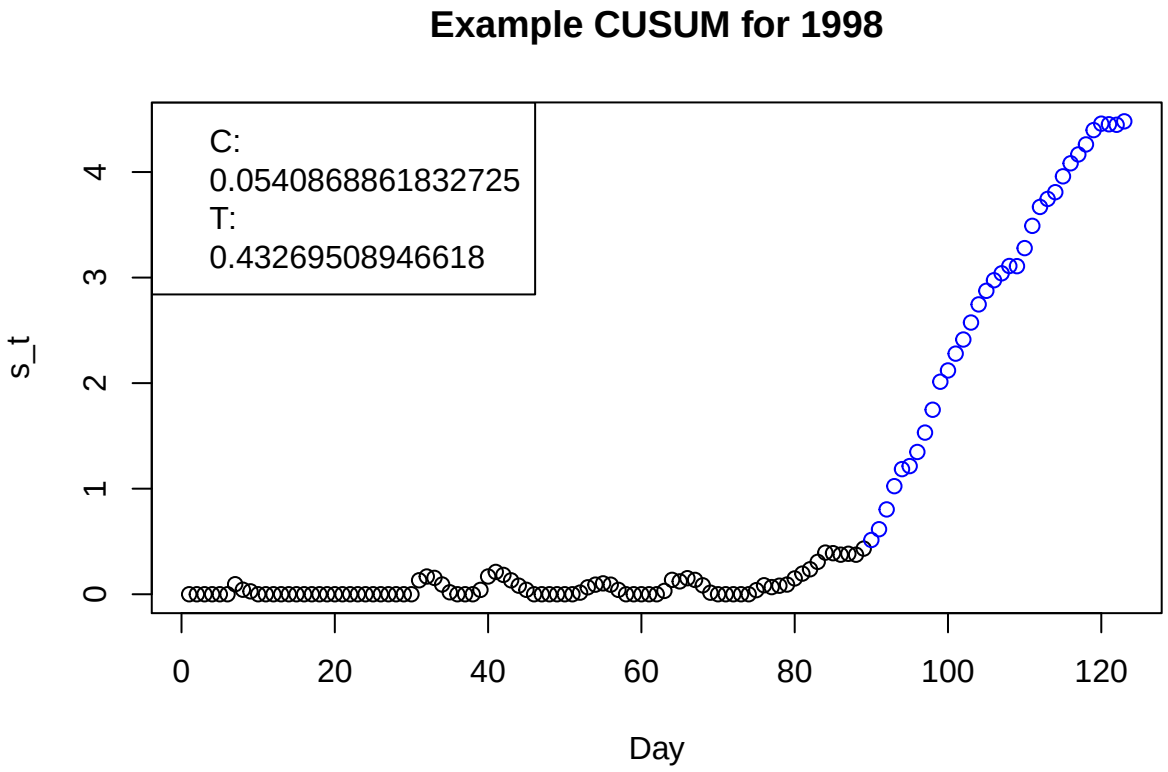
```
print (paste('The day which the summer ends for the avg year is: ',temps$DAY[day_change]))
```

```
## [1] "The day which the summer ends for the avg year is:  9-Sep"
```

```
plot(year_result, main = 'Day of summer change each year', xlab = 'Year', ylab =
'day',at=1:19, labels=as.vector(c(1997:2015)))
lines(year_result)
```




```
# just for demostration this is how a cusum individual year is shown
july <- result4$fitted[,4][1:31]
current_year <- result4$fitted[,4][124:246]
individual_result <- cusum(july,current_year, 1,8)
plot(individual_result$s, main = 'Example CUSUM for 1998', xlab = 'Day', ylab ='s_t', col = ifelse(indi
legend('topleft',legend = c('C: ',individual_result$c,'T:',individual_result$t))
```




```
# the blue points are the ones that have s_t[i] > T
```

Continue Analysis 7.2

For this part I used the seasonal model made by the smoothing exponential model to test if the summer ending day has changed through the years in Atlanta. Seasonality can be additive or mutiplicative. The difference between thes is that the additive represents a linear behaivor where changes over time are consistently made by the same amount, like a linear trend. On the other hand, the multiplicative can be exponential or quadratic an is represented by a curve. For this case, the best seasonality is the multiplicative because it can't be assured that the temperature changes would be constant through the 20 years of data.

The multiplicative curve is represented by the curve `y(t) = Level * Trend * Seasonality * Noise`. The result of this model won't include the first year (1996) because the model needs a complete season's data to determine initial estimates.

To correctly test the cusum model it was important to divide the data for each 123 data points, because here the data is not divided by years so it had to be divided manually to test each year. For the μ (expected value) for the cusum model I used only July because it is expected that the summer won't finish in that month.

The resulting plot says that the summer is ending on `9-sept` on average. The plot with the average ending summer for each year shows a significant decrease in the `2004-2005` period. That means that the summer is ending earlier after that period but then it starts to normalize after the year `2008`. Also like homework 3, the result is sensible to the C and T values.

I would say there is no strong evidence to conclude without a doubt that the summer is ending later. To have strong evidence more years should be analyzed to see what actually the trend is.

To conclude, here I wrote some tests changing the C and T of the cusum

```
matrix <- cbind(c(1,1,2,2,4,1), c(2,5,10, 15, 20, 40), c('5-Aug', '27-Aug', '1-Oct', '7-Oct', '4-Oct',
colnames(matrix) <- c("C", "T", "Date")
```

```r
library(knitr) # table library
table <- kable(matrix)
print (table)
```

```
##
##
## C    T    Date
## ---  ---  -------
## 1    2    5-Aug
## 1    5    27-Aug
## 2    10   1-Oct
## 2    15   7-Oct
## 4    20   4-Oct
## 1    40   4-Oct
```