Exponential Smoothing to Analyze End of Summer Trends

Alena Fedash

2022-09-19

Contents

Summary of Results

Since exponential smoothing is a time series forecasting method, I began my analysis by converting the data to a **time series object**. As frequency, I used the number of days which we had for each year in the data set (July through October), and the start value was the first year we had, 1996. This way, I generated a times series object with 20 'seasons', each season being defined by temperatures from July 1 to October 31.

After that, I performed visual inspection of the times series, and noticed that, according to the plot, the data is more likely to have an **additive seasonality factor**, rather than multiplicative, since there was no obvious funnel-like shape on the graph.

Nevertheless, I decomposed the time series data trying both methods, multiplicative and additive seasonality. Both showed similar results, the only difference being that in multiplicative version random and seasonal components were displayed as coefficient rather than additive values. **Decomposition also showed that the trend factor is rather insignificant**, suggesting there is no year-to-year trend in temperature change. This is an important signal that there is no general 'warming' in our data that is becoming more obvious every year - the temperatures are rather stable. There was also a **significant random component**, which had a wider value range compared to seasonality (and trend). That is a sign that it would be hard to build accurate predictions (if needed) with such data, as there is too much 'noise' present. This is logical, since temperature prognosis is usually done using winds, cyclones and rather data rather than last years' temperatures.

After decomposition, I performed **Single, Double and Triple (additive and multiplicative) Exponential smoothing**, and got the results, which can be found in **Step 5** of the solution (comparative table of 4 ES models).

Single ES did not consider trend and seasonality, and double ES considered trend, but did not take seasonality into account (which was obviously present, according to decomposition plot). In all cases, **alpha (base values) was closer to 1 rather than 0, suggesting that the model found it optimal to put more weight on recent observations rather than past ones and did not detect too much randomness in the system**. Baseline estimate ranged from 63 to 73 degrees. On the other hand, **beta (trend) values suggested that the trend slope of the data is not much dependent on recent trend slopes**. Final trend estimate was always almost 0, suggesting that **there is no significant trend in temperature change over 20 years**. Finally, **gamma values (seasonal component) for the triple ES suggested putting more weight on recent seasonalities**. For the triple ES models I also got seasonal factor estimates for each of the 123 days.

The final choice was between **Additive and Multiplicative Triple ES Models**. I fitted both models, plotted their decomposition and fit on the data. The difference was very little and rather hard to tell visually.

However, with **Additive seasonality** the **Sum of Squared Errors** was lower (66244 compared to 68904 with multiplicative seasonality), so **additive seasonality method was chosen**.

The logic for CUSUM was the following: use the known seasonal factors for each day of each year from the exponential smoothing model to build a CUSUM model for **each year** and detect violations. For each year, explore **which day** the seasonal factor **drops below the set margin** (lower violations). Then, **compare the dates of first violation** year-to-year to see if the dates are **getting later**. This would mean that the fall starts later each year, hence the summers are becoming longer. To do that, I put extracted from the model seasonal factors for each day of each year in a new data frame and used it for CUSUM model building.

To define the **No Change Period** for the model, based on which I would calculated means and standard deviations of seasonal factors for each year, I calculated **average seasonal factor value** for each day using all years' data and visualized it on a plot to see the general pattern of the factors over a season. I chose the first **51 days** as the no-change period (July 1 - Aug 20). A shorter period could be riskier in terms of false positives, as the factors for July only are in general high, and a longer period, on the other hand, could make the model miss real summer end dates due to variability in year-to-year seasonal factors (some extreme values could be included in the mean). 51 days period seemed like the perfect range of dates with a 'normal' factor values. With this period, the 19 (19 years, as 1996 is used as the first observation in exponential smoothing) calculated mean values of seasonal factors ranged from 4.5 to 5.8, and standard deviations from 3.5 to 4.6.

As for **C and T**, I decided to start with most common values (1 or 2 standard deviations for C, 4 or 5 standard deviations for T). I chose **C=1sd and T=4sd** (slightly lowered the violation margins to 4sd, making the model a bit more sensitive, since sd was not that little (3.5-4.6) compared to the range of average seasonal factor values (they go from 0 to 13 and from 0 to -17)). Since the last time unofficial summer end I found was around 20th September, I assumed that a good combination of C and T would produce a similar result $\pm$ 1 week.

The chosen C and T values turned out to be a **great combination**. As a result, I got unofficial fall start dates ranging from 20 to 24 September.

The answer to the question is negative, because there is no evidence that the summers have become longer - fall start dates vary year-to-year by one or two days, and there is no trend in such change. In fact, in 1997-1999 summer was a bit longer (fall starts on the 09.24) compared to 2014-2015 (09.22 and 09.21).

I went further and **compared the results with my previous CUSUM results**, where I calculated start of fall for each year **without exponential smoothing** using similar C and T values (C=1sd, T=5sd). The results have changed drastically - before exponential smoothing fall start dates range by almmost 2 months, from August to October, which is why it was harder to answer the question whether the summers have gotten longer. Exponential smoothing helped to get accurate results without taking the excess 'white noise' and randomness into consideration.

Below is a step-by-step solution with more reasoning and explanation of each choice I made throughout this exercise.

Solution in R

Step 0: Load the libraries

```
library(dplyr)
library(tidyverse)
library(dslabs)
library(data.table)
library(ggplot2)
library(plotly)
```

```r
library(outliers)
library(qcc)
```

Step 1: Load the dataset & do basic exploration

Here, I load the data set and rename the columns to remove the 'X' symbols near each year.

```r
data <- read.table("temps.txt",
                   header = TRUE,
                   stringsAsFactors = FALSE,
                   sep = "",
                   dec = ".")

#rename to exclude X from att names
data <- data %>% rename_at(vars(starts_with("X")), funs(str_replace(., "X", "")))
```

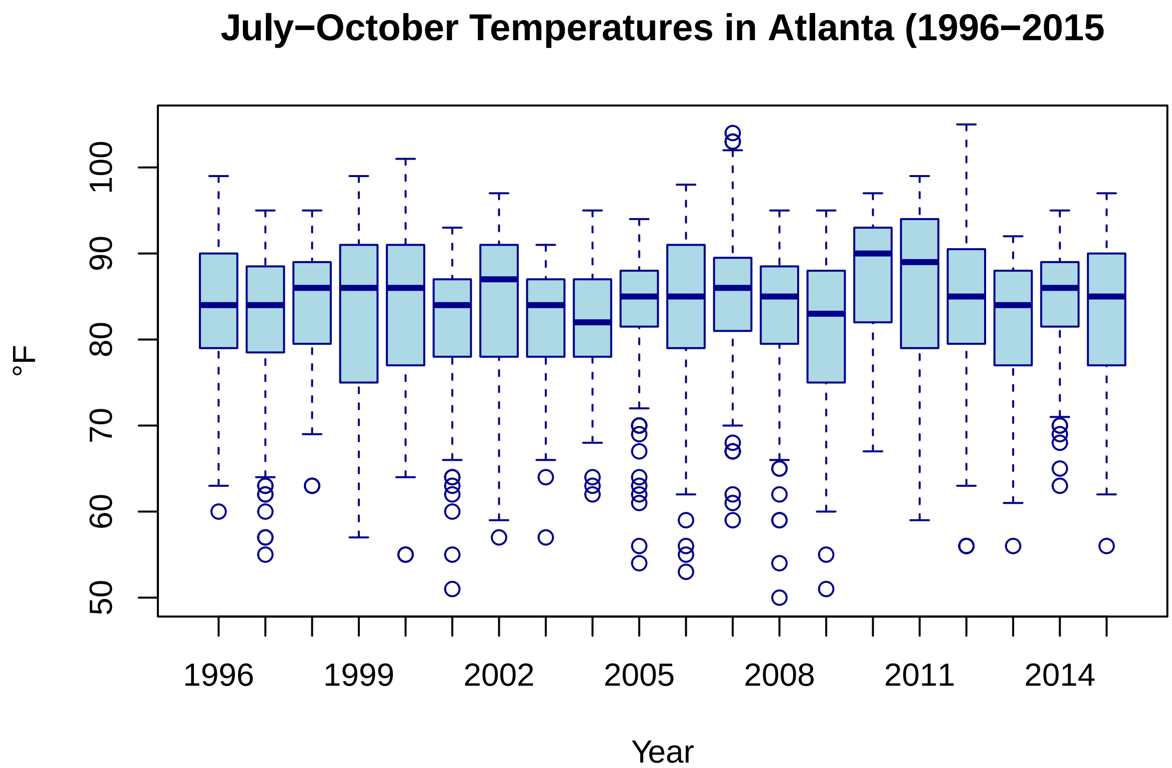
```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```r
data[,"DAY"] <- as.Date(data[,"DAY"], "%d-%B")
#leave only month and day
data[,"DAY"] <- format(data[,"DAY"],"%m.%d")
head(data)
```

```
##     DAY 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009
## 1 07.01   98   86   91   84   89   84   90   73   82   91   93   95   85   95
## 2 07.02   97   90   88   82   91   87   90   81   81   89   93   85   87   90
## 3 07.03   97   93   91   87   93   87   87   87   86   86   93   82   91   89
## 4 07.04   90   91   91   88   95   84   89   86   88   86   91   86   90   91
## 5 07.05   89   84   91   90   96   86   93   80   90   89   90   88   88   80
## 6 07.06   93   84   89   91   96   87   93   84   90   82   81   87   82   87
##   2010 2011 2012 2013 2014 2015
## 1   87   92  105   82   90   85
## 2   84   94   93   85   93   87
## 3   83   95   99   76   87   79
## 4   85   92   98   77   84   85
## 5   88   90  100   83   86   84
## 6   89   90   98   83   87   84
```

I will also perform some basic data exploration to learn more about the data.

```r
boxplot(data[-1],
        xlab="Year",
        ylab="°F",
        col="lightblue",
        border="darkblue")
title(main="July-October Temperatures in Atlanta (1996-2015")
```



There is no obvious trend of a changing median temperature over the years, according to the boxplot. It looks like temperatures in general were a bit higher in 2010 and 2011, but the median goes back down to previous values in 2013. Also, some years seem to be more 'stable' in temperature, and sometimes it varies more - there are years with short whiskers, as well as years with many outlying values. In general, I would suspect that the temperatures have not been increasing over the last few years. However, let's do a thorough analysis with exponential smoothing to draw supported conclusions.

Step 2: Convert Data to Time Series

Since exponential smoothing is a time series forecasting method, we need to convert our data to time series.

The ts() function requires **two critical inputs - frequency and start**. Since we have years' data from 1996 to 2015, **1996 would be our start**. And, because **we have 123 days' temperature known for each year, 123 would be our frequency** - that is the number of rows in out data frame.

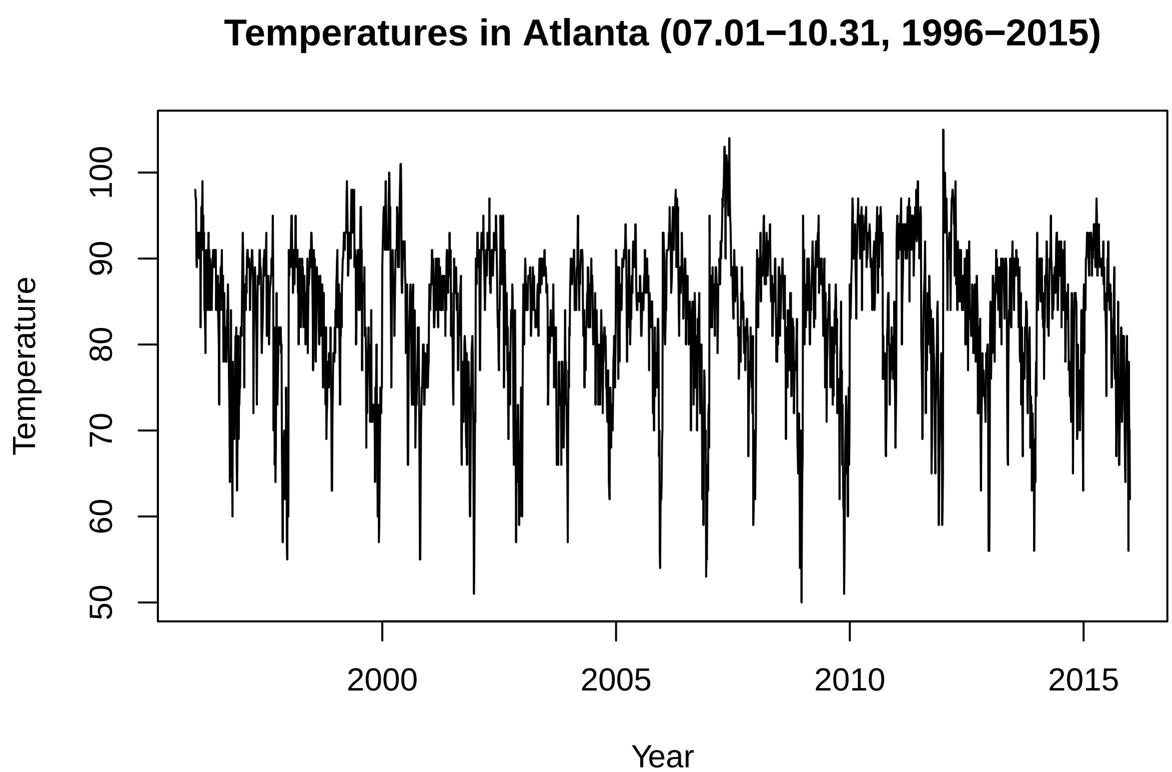
First, we need to merge all temperatures in a vector, and then generate a time-series object using the start and frequency values from above.

```
#There are two critical inputs we must give the function - frequency and start.

#Turn col data into one vector
data_vector <- as.vector(unlist(data[,2:ncol(data)]))

#Turn data to ts object
#we have 123 days' temp for each year - we will have 'seasons' from 1996 each lasting 123 days

datats <- ts(data_vector,start=1996,frequency=nrow(data))
head(datats)
```

```
## [1] 98 97 97 90 89 93
```

Visualize our time series object:

```r
plot(datats, main="Temperatures in Atlanta (07.01-10.31, 1996-2015)", xlab="Year", ylab="Temperature")
```



Looking at the graph here, although there could be some factors influencing our data (for example, randomness), we can already see that there is no obvious trend - the 'arches' (our 'seasons') stay on the same level, and there is no distinctive upwards or downwards trend in the temperatures. Also, there are some 'anomalities' in each season - some very low or very high temperatures, but there is no obvious pattern connecting them.

Step 3: Decomposition

Let us try decomposition for our data to check how significantly factors like trend, seasonality and randomness affect our temperatures.

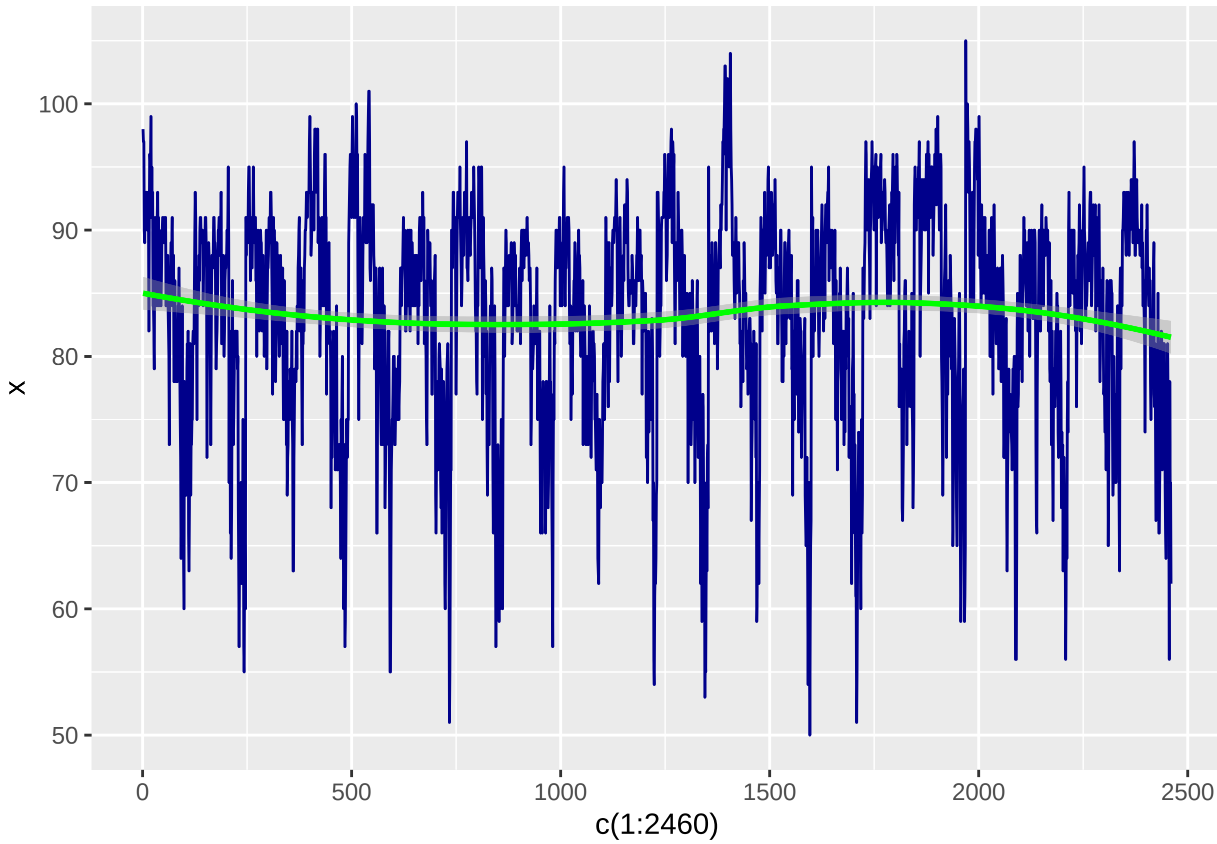
There are two types of decomposition we could perform - with **additive or multiplicative** seasonality.

Before trying both, let's think about what each seasonality type would mean for our data: -**Additive seasonality** would mean that our seasons remain similar in widths and heights, while there is a linear trend year-to year. So, our average temperature would be increasing/decreasing year to year, but the amplitude of temperature variations would stay the same. -**Multiplicative seasonality**, on the other hand, would mean that our temperatures are becoming more extreme in their range year to year - for example, each year for the part of the season with warmer temperatures, the temperatures would become even more warmer, and for the part with lower temperatures of the season, unlike with additive seasonality, where they would become warmer as well (to keep the temperature spread similar to last year), they would become even colder.

Multiplicative seasonality would most probably make our data look like a funnel, so let's check the graph again to see if that is the case. I will also add a smoothing line to see the direction of temperature changes

```r
ggplot(as.data.frame(datats),aes(x=c(1:2460),y=x))+
  geom_line(aes(x=c(1:2460),y=x),color="darkblue")+
  geom_smooth(method="loess",color="green")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting to continuous.
## `geom_smooth()` using formula = 'y ~ x'
```

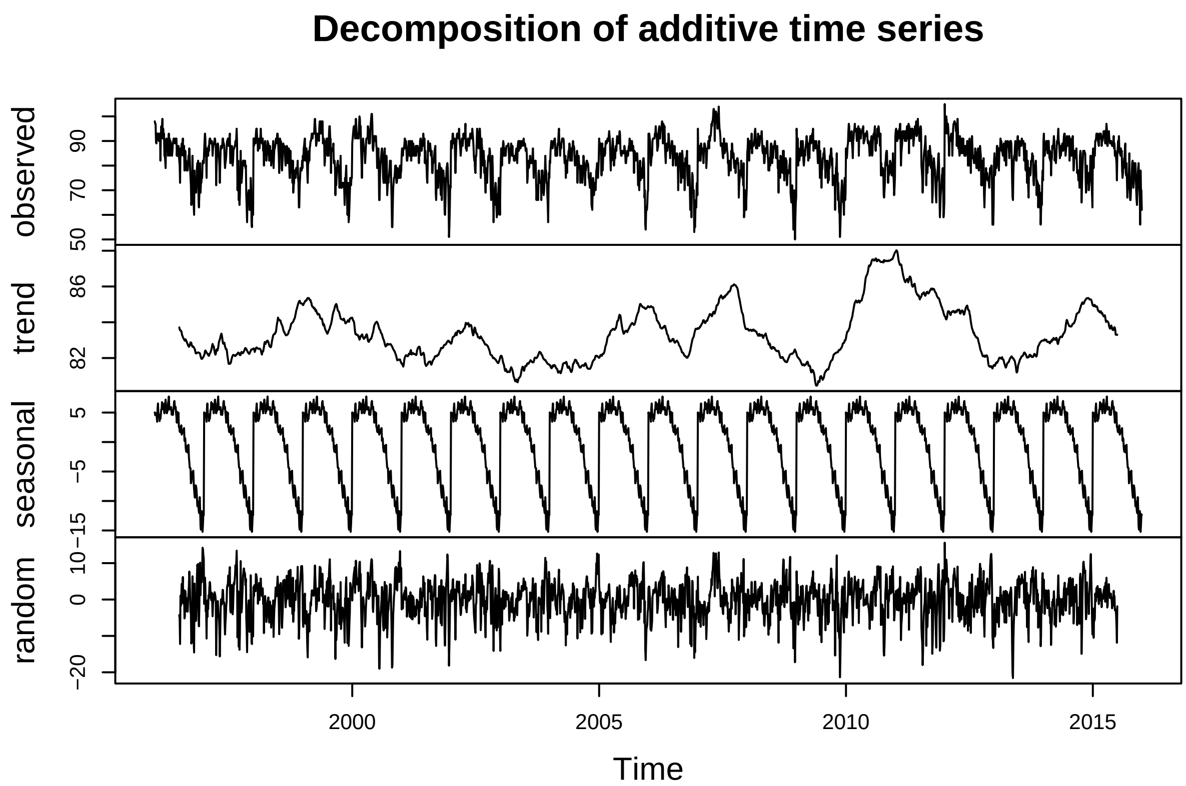


As we can see, our data does not look like a funnel. The seasonality is most probably additive. And, what is more important, it looks like there is no trend at all.

Keeping in mind that our seasonality factor would likely be additive, I will still decompose the data with both seasonality types to compare the results.

Decomposition (additive seasonality):

```r
components_dts <- decompose(datats)
plot(components_dts)
```



On the graph above besides the observed temperatures we see 3 other components of the data set: -**Trend**: shows the long-term trends in the data. **We can see that it barely ranges in values from 82 to 86, so as we suspected before, there is likely no significant long-term trend in the data**, hence we are unlikely to get proof that the summers are getting warmer or colder -**Seasonal**: repeated seasonal component that makes data vary withing the seasons. **We have a significant seasonality factor - we can see how in the beginning of each season (summer) the temperatures are higher, and how they lower towards its end (October) -Random**: Components that are left over, and are not expected from seasonality or trends. **Our random component is quite significant - its range is bigger than the range of the seasonal factor or the trend**. Higher significance of randomness compared to seasonality and trend in general would make the prediction for the future period less accurate. That is logical - with almost no trend and this much randomness we would be unlikely to predict temperatures for each day in July-October 2016 using just past years' values. After all, there are lots of factors that influence weather, and it rather depends on the ongoing forces (like cyclones) than on the last years' data.

I will output the range of all factors to compare them to decomposition with multiplicative seasonality:

```
#trend
min(na.omit(components_dts$trend))
```

```
## [1] 80.45528
```

```
max(na.omit(components_dts$trend))
```

```
## [1] 88.02439
```

```
#seasonal
min(na.omit(components_dts$seasonal))
```

```
## [1] -15.23864
```

```
max(na.omit(components_dts$seasonal))
```

```
## [1] 7.728413
```

```
#random
min(na.omit(components_dts$random))
```

```
## [1] -21.59362
```

```
max(na.omit(components_dts$random))
```

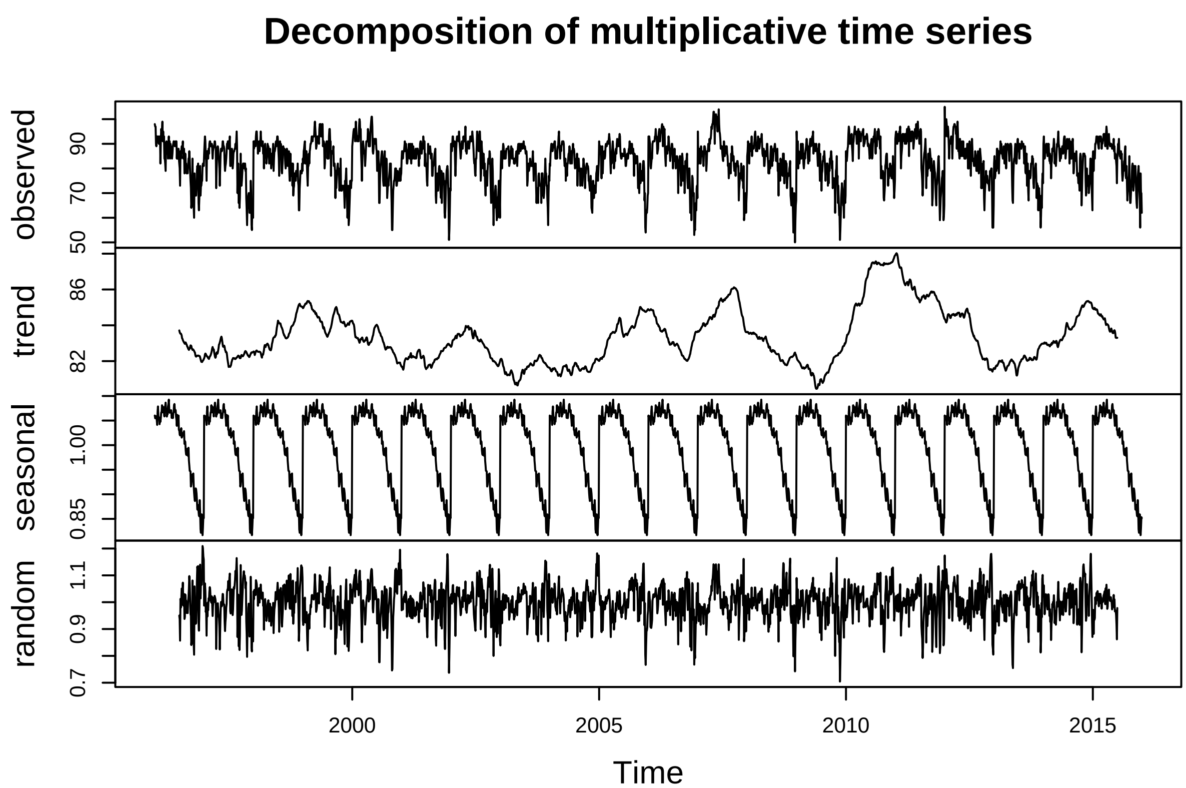
```
## [1] 15.61819
```

Indeed, the random component has the widest scale, while the trend is insignificant (we are unlikely to fin change in temperatures for the last years).

Decomposition (multiplicative seasonality):

Let us do the same with multiplicative seasonality:

```
#Decomposition (multiplicative)
components_dts_mult <- decompose(datats, type="mult")
plot(components_dts_mult)
```



The difference is that the random and seasonal components are now multiplicative instead of additive. In general, We still have a trend with low significance, good seasonality and lots of randomness.

Let's check the scale of each component:

```
#trend
min(na.omit(components_dts_mult$trend))
```

```
## [1] 80.45528
```

```r
max(na.omit(components_dts_mult$trend))
```

```
## [1] 88.02439
```

```r
#seasonal
min(na.omit(components_dts_mult$seasonal))
```

```
## [1] 0.8168494
```

```r
max(na.omit(components_dts_mult$seasonal))
```

```
## [1] 1.092631
```

```r
#random
min(na.omit(components_dts_mult$random))
```

```
## [1] 0.7041936
```

```r
max(na.omit(components_dts_mult$random))
```

```
## [1] 1.209286
```

We have the same values for the trend component, and the Seasonal and Random factors are expressed differently (with coefficients).

Conclusion for step 3:

We now have found that it is unlikely for us to detect a trend in temperatures to answer the main question, whether the summers have become longer/warmer. It seems from the graphs that any change would more likely be due to change. Moreover, there is a seasonality component with good significance, so we would need to consider it in our final model with exponential smoothing. Finally, from visual inspection of the data it appear that we would rather have additive seasonality than multiplicative.

Now, let's move on to exponential smoothing.

Step 4: Exponential smoothing

Based on the findings in the previous step, we can say that the trend for our data is minimal, and temperature changes would probably be caused by randomness.

Now, let's apply exponential smoothing to our data. We will try different models using HoltWinters function and compare them.

We will then **extract the seasonal factors** for each day of the year and use them to perform CUSUM Using these factors, we would be able to get rid of 'white noise' or randomness that could affect the accuracy of CUSUM.

There are **3 types of exponential smoothing** that we will try: -Single ES - simple ES that would use a weighted moving avg of our temperatures without trend and seasonality components -Double ES - a more reliable method that considers trends in the data, but no seasonality -Triple ES - a method for data with both trend and seasonality

From our previous findings, I would suspect that the model most suitable for our data would be **triple ES**, as we have both trend and seasonality components.

I will try all the models and compare the optimal values that they produce. I will also compare models with additive and multiplicative seasonality to make the final choice for the type of this component.

Single ES

```r
#single - no trend no season
HW1 <- HoltWinters(datats,
                     beta=FALSE, #model without trend
                     gamma=FALSE) #not considering seasonality

HW1
```

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

We have a baseline estimate of 63.309, and the optimal alpha as estimated by the model is **0.8388**. That means that there is not much randomness in the data, and we put more trust in the most recent observations (temperature in the most recent years).

Double ES

```r
HW2 <- HoltWinters(datats,
                   gamma=FALSE) #now we have trend value in the M but still no season
HW2
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = datats, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.8445729
##  beta : 0.003720884
##  gamma: FALSE
##
## Coefficients:
##         [,1]
## a 63.2530022
## b -0.0729933
```

With an ES that includes trend as a component, we get a similar baseline estimate of 63, and a **slightly higher alpha of 0.8445** - so the amount of randomness is again found to be rather low, and the recent datapoints can be trusted.

The **trend value, beta** is close to zero, which means that **the trend slope of our temperatures is less dependent on the trend slopes for the recent temperatures**. In other words, recent temperature observations do not add to the trend value that much, and that resulted in a **final trend estimate close to 0, -0.07**. This ones again is a signal that no significant trend is observed in our temperature observations overtimes, moreover, this small trend is negative, so the temperatures have not become warmer over the years.

Triple ES - Additive seasonality

First, let's try additive seasonality component gamma and check how it changes the results:

```r
HW3_a <- HoltWinters(datats)
HW3_a
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = datats)
##
## 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
```

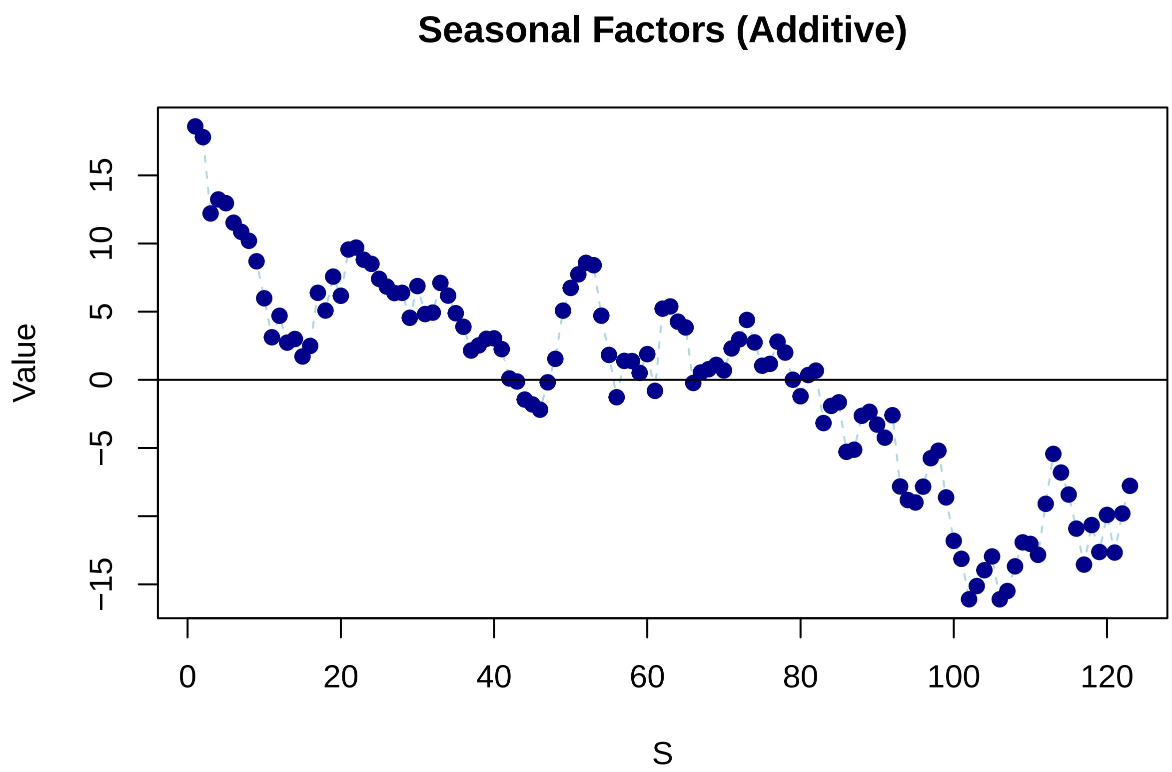
```
## s35    4.886891317
## s36    3.890547248
## s37    2.148316257
## s38    2.524866001
## s39    3.008098232
## s40    3.041663870
## s41    2.251741386
## s42    0.101091985
## s43   -0.123337548
## s44   -1.445675315
## s45   -1.802768181
## s46   -2.192036338
## s47   -0.180954242
## s48    1.538987281
## 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
```

Once we added a seasonality component, gamma, our **alpha lowered to 0.66, signalizing that now the model considers to be more randomness than before**, though the value is still closer to 1 rather than 0, so we rely more on the recent observations than on the past ones. This is not unexpected, since we saw that there was a random component when decomposing the data. The **baseline estimate is now 71** degrees, **beta is 0**, so the most recent trend slopes are not taken into account for the **final trend estimate of -0.004**. Ocne again, there is no trend in temperatures change in our data. Finally, we now have a **gamma, the seasonal component, of 0.62**, it is closer to 1 but not far from 0.5, so the weight of seasonal cycles in more or less even with a bit more accent on the recent seasonal cyclicities. We also have **123 seasonal faactors**, containing final estimates of the seasonal component for each day from July 1st to October 31st.

```
plot(HW3_a$coefficients[3:length(HW3_a$coefficients)],
     type="o",
     col="lightblue",
     lty=2,
     lwd=1,
     main="Seasonal Factors (Additive)",
     xlab="S",
     ylab="Value")
points(HW3_a$coefficients[3:length(HW3_a$coefficients)],
```

```
        col="darkblue",
        pch=19, cex=1)
abline(h=0)
```



From the plot of the seasonal factor estimates we can see that around the 80th day of a season (~20th August) are negative-only. So after that period the temperatures are estimated to be below the baseline with an application of a corresponding seasonal factor.

Triple ES - Multiplicative seasonality

Let's check if we get a different result using multiplicative seasonality:

```
HW3_m <- HoltWinters(datats, seasonal = "multiplicative")
HW3_m
```

```
## Holt-Winters exponential smoothing with trend and multiplicative seasonal component.
##
## Call:
## HoltWinters(x = datats, 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
```

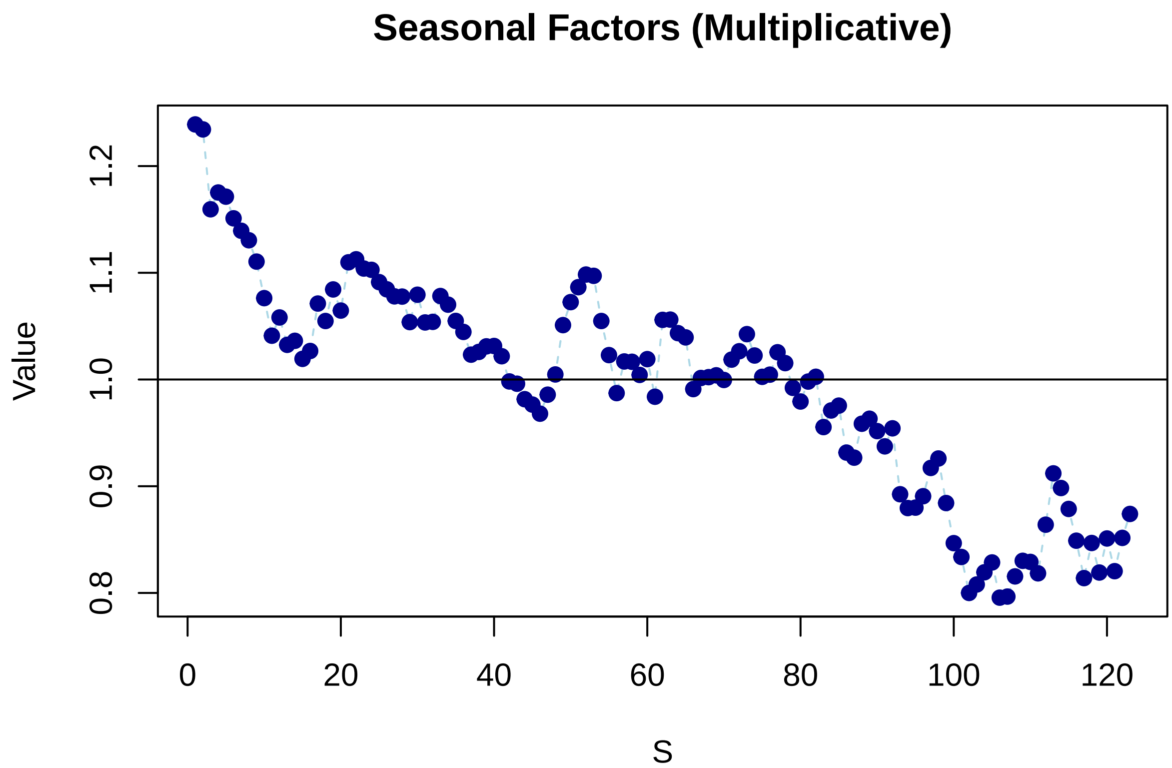
```
## s54   1.054827180
## s55   1.022866587
## s56   0.987259326
## s57   1.016923524
## s58   1.016604903
## s59   1.004320951
## s60   1.019102781
## s61   0.983848662
## s62   1.055888360
## s63   1.056122844
## s64   1.043478958
## s65   1.039475693
## s66   0.991019224
## s67   1.001437488
## 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
```

Now, we have a slightly **lower alpha and gamma, and the same beta=0**, so less weight is put on recent observations. **Baseline estimate is at 73.4**, and the trend estimate is again very insignificant and negative, **-0.004**.

Lets visualize seasonal factors for better understand of the difference of the two methods:

```r
plot(HW3_m$coefficients[3:length(HW3_m$coefficients)],
     type="o",
     col="lightblue",
     lty=2,
     lwd=1,
     main="Seasonal Factors (Multiplicative)",
     xlab="S",
     ylab="Value")
points(HW3_m$coefficients[3:length(HW3_m$coefficients)],
       col="darkblue",
       pch=19, cex=1)
abline(h=1)
```



Similar picture to the previous method - after 80th day index seasonality factors drag the temperature below the baseline.

Step 5: Choosing the model

Let's sum up the parameters of our models:

```
es_models <- data.frame(
  alpha=c(HW1$alpha,HW2$alpha,HW3_a$alpha,HW3_m$alpha),
  beta=c(NA,HW2$beta,HW3_a$beta,HW3_m$beta),
  gamma=c(NA,NA,HW3_a$gamma,HW3_m$gamma),
  Baseline_Estimate=c(HW1$coefficients[1],HW2$coefficients[1],HW3_a$coefficients[1],HW3_m$coefficients[
  Trend_Estimate=c(HW1$coefficients[2],HW2$coefficients[2],HW3_a$coefficients[2],HW3_m$coefficients[2]))

rownames(es_models) <- c('Single',
                         'Double',
                         'Triple (Additive)',
                         'Triple (Multiplicative)')

es_models
```

```
##                              alpha        beta     gamma Baseline_Estimate
## Single                   0.8388021          NA        NA          63.30952
## Double                   0.8445729 0.003720884        NA          63.25300
## Triple (Additive)        0.6610618 0.000000000 0.6248076          71.47724
## Triple (Multiplicative) 0.6150030 0.000000000 0.5495256          73.67952
##                         Trend_Estimate
```

```
## Single                              NA
## Double                    -0.072993299
## Triple (Additive)         -0.004362918
## Triple (Multiplicative)   -0.004362918
```

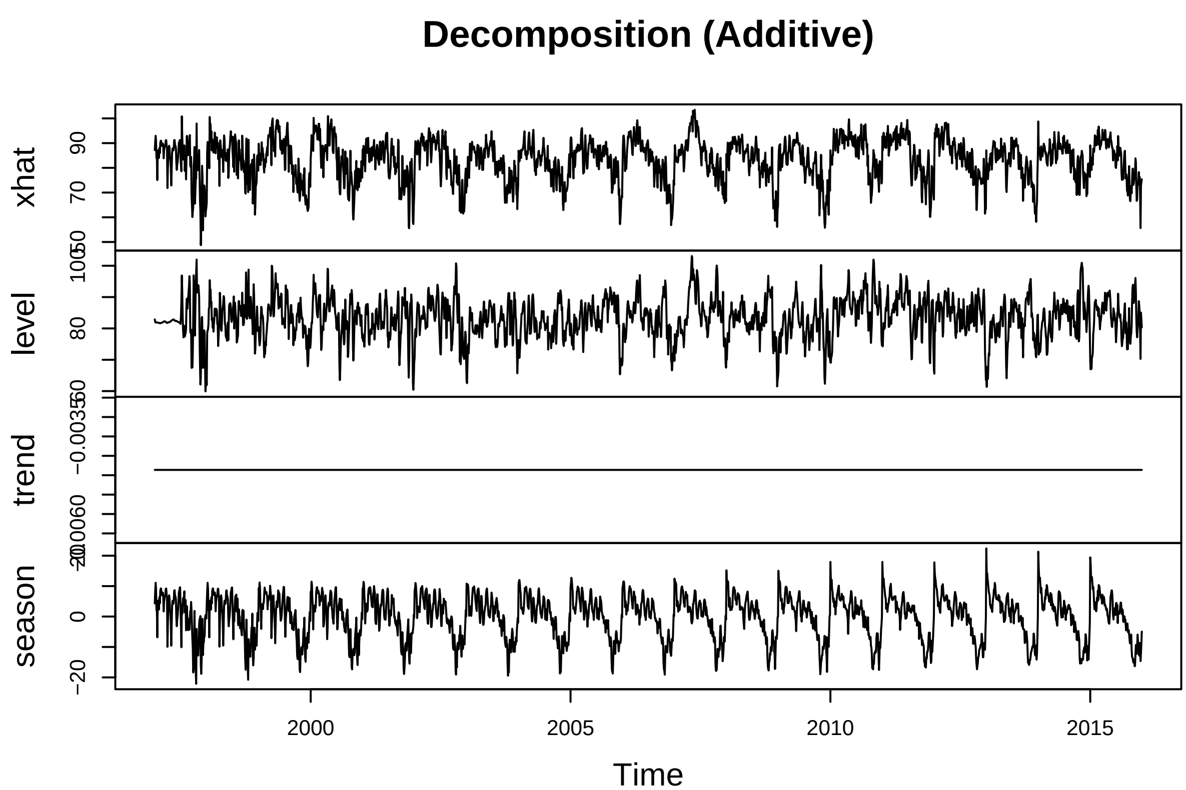
As we can see, once we introduce seasonality to the model, the alpha becomes lower and not as much weight is put on the most recent observations, as with the previous two models.

The trend estimate is close to zero and negative for all models where it is taken into account. **This signals that there is no change in our temperatures throughout the years**, so the answer to the main question is likely to be negative. Finally, multiplicative model put a bit less weight on most recent seasonal cycles and provides a slightly higher baseline estimate.

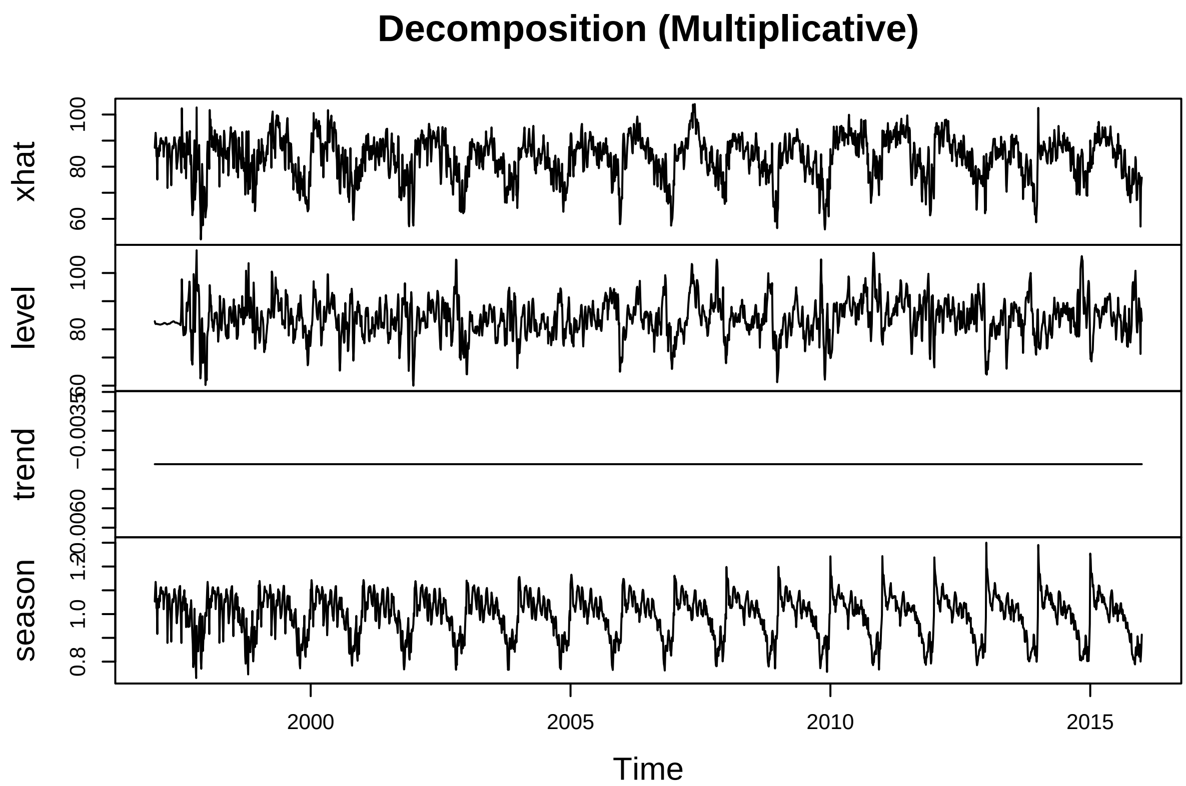
We should be choosing between additive and multiplicative triple ES models, as they take into account both trend and seasonality, and the latter is very significant for our temperatures data.

First, let's compare the decomposition plots of both models:

```r
plot(fitted(HW3_a), main="Decomposition (Additive)")
```




```r
plot(fitted(HW3_m), main="Decomposition (Multiplicative)")
```

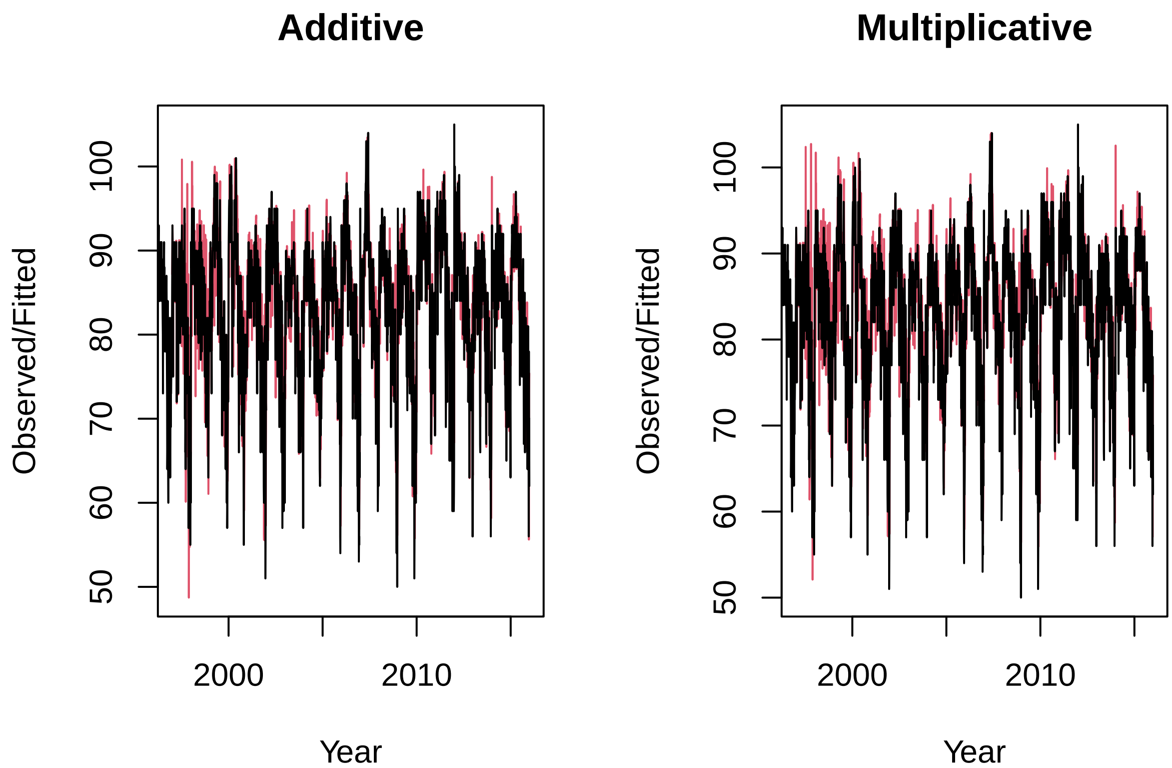


The results are very similar - both models estimate a trend close to zero, and seasonalitys have year-to-year repetitive patterns, except for the last few years in the multiplicative model, where the range of the seasonal factor becomes slightly wider (due to high seasonality factors at the beginning of the season).

This is not enough to make the final choice, so we keep exploring both models.

Let's try and visualize how both models fit the data:

```
par(mfrow=c(1,2))
plot(HW3_a, main="Additive", xlab="Year", ylab="Observed/Fitted")

plot(HW3_m, main="Multiplicative", xlab="Year", ylab="Observed/Fitted")
```



The first year is not present for the fitted values, as they are used as a first observation in ES and are not fitted according to the ES equation. Both models have a very similar fit, the only difference that is noticable visually are the slightly higher fitted values for the first few years in the multiplicative model.

Finally, **let's compare the Sum of Squared Errors for each model**

```
print('SSE Additive')
```

```
## [1] "SSE Additive"
```

```
HW3_a$SSE
```

```
## [1] 66244.25
```

```
print('SSE Multiplicative')
```

```
## [1] "SSE Multiplicative"
```

```
HW3_m$SSE
```

```
## [1] 68904.57
```

SSE is smaller for the Additive model. Hence, we should move on with the Additive ES - our initial assumption (based on visual inspection) that additive seasonality suits the data better was correct.

Step 6: Preparation for CUSUM

Although we have found the trend estimate to be around 0, we should still check if there is any change in the seasonal factor values for the past few years. On the decomposition plot we have seen that the seasonality factor does not significantly change for the latest years - the length of seasons does not increase (the 'arches' on the graph are not getting wider year to year), and the duration of warm parts of each season (the width of the higher part of the arch) appear to be the same for each year. However, it is worth checking with CUSUM for any possible change in seasonal factors, as they may be hard to detect using visual inspection.

The logic for our CUSUM model will be the following: Since we know seasonal factors for each day of each year from our ES model with additive seasonality, we can use them to detect whether the unofficial end of summer has become later. We will run CUSUM on seasonal factors for each data point for each year. By doing this we can see where in each year (on which day) a change was detected by CUSUM, and then we will compare those dates to see if the change happens later over time.

Let's have a look at our fitted model output:

```r
head(HW3_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
```

We need the 'season' column for the CUSUM - it contains the needed season factors for each day in the data set.

Now let's transform those factors to a matrix and rename columns and rows so that we can use it for CUSUM analysis.

```r
#put seasonal factors in the matrix
season_factors <- matrix(HW3_a$fitted[,4],nrow=123)
dim(season_factors)
```

```
## [1] 123  19
```

```r
head(season_factors)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,]  4.303159  4.054077  9.349191  8.130519  9.261932  8.532675 10.833750
## [2,]  8.238119  8.168393  8.457426  7.810603  8.686298  9.197265  9.837370
## [3,] 11.091777 11.100059 11.213419 11.470326 11.416575 11.012491 10.210645
## [4,]  9.042997  9.057058  9.529053 10.185524 10.863856 10.209557 10.532285
## [5,]  2.067387  2.067912  3.708913  5.588420  7.004562  8.024549  9.444664
## [6,]  2.116168  2.106939  2.232254  3.394700  4.340175  5.462823  6.487578
##           [,8]      [,9]     [,10]     [,11]    [,12]     [,13]     [,14]
## [1,] 10.596553  9.559008  9.526730 12.454249 15.18901 15.052357 17.963213
## [2,] 11.663073 10.874704 10.162519 11.021049 10.13469 11.583196 12.246533
## [3,] 12.021948 12.738520 11.468041 11.483482 10.45074 11.722773 11.707197
## [4,] 10.867244 11.779117 11.552601 11.117308 11.69283 11.650082 12.084663
## [5,]  8.518818  9.749682 10.738920 10.552850 11.29192 10.939712  8.908885
## [6,]  7.648008  8.250531  7.421824  6.156207  7.12694  6.619883  8.329696
##          [,15]     [,16]     [,17]    [,18]    [,19]
## [1,] 17.946705 17.796582 22.385105 21.31072 19.45438
## [2,] 12.817176 14.277046 14.037278 16.07719 17.19256
## [3,] 11.803979 12.726046 14.244787 12.98723 12.74993
## [4,] 12.461996 12.000781 12.458283 12.62307 11.98537
## [5,] 10.345553 10.214813 11.172561 12.77225 12.94898
## [6,]  9.151988  9.361362  9.444097 10.35325 11.13812
```

We have 19 year (1997 to 2015, as 1996 is the first observation and is not included) and 123 days in each.

Renaming columns and rows to make data look as the initial data set for convenience:

```r
#rename
colnames(season_factors) <- c(1997:2015)
rownames(season_factors) <- data$DAY
#check
head(season_factors)
```

```
##            1997      1998      1999      2000      2001      2002      2003
## 07.01  4.303159  4.054077  9.349191  8.130519  9.261932  8.532675 10.833750
## 07.02  8.238119  8.168393  8.457426  7.810603  8.686298  9.197265  9.837370
## 07.03 11.091777 11.100059 11.213419 11.470326 11.416575 11.012491 10.210645
## 07.04  9.042997  9.057058  9.529053 10.185524 10.863856 10.209557 10.532285
## 07.05  2.067387  2.067912  3.708913  5.588420  7.004562  8.024549  9.444664
## 07.06  2.116168  2.106939  2.232254  3.394700  4.340175  5.462823  6.487578
##            2004      2005      2006      2007     2008      2009      2010
## 07.01 10.596553  9.559008  9.526730 12.454249 15.18901 15.052357 17.963213
## 07.02 11.663073 10.874704 10.162519 11.021049 10.13469 11.583196 12.246533
## 07.03 12.021948 12.738520 11.468041 11.483482 10.45074 11.722773 11.707197
## 07.04 10.867244 11.779117 11.552601 11.117308 11.69283 11.650082 12.084663
## 07.05  8.518818  9.749682 10.738920 10.552850 11.29192 10.939712  8.908885
## 07.06  7.648008  8.250531  7.421824  6.156207  7.12694  6.619883  8.329696
##            2011      2012      2013     2014     2015
## 07.01 17.946705 17.796582 22.385105 21.31072 19.45438
## 07.02 12.817176 14.277046 14.037278 16.07719 17.19256
## 07.03 11.803979 12.726046 14.244787 12.98723 12.74993
## 07.04 12.461996 12.000781 12.458283 12.62307 11.98537
## 07.05 10.345553 10.214813 11.172561 12.77225 12.94898
## 07.06  9.151988  9.361362  9.444097 10.35325 11.13812
```

```r
tail(season_factors)
```

```
##             1997      1998       1999       2000       2001        2002
## 10.26 -6.4935885 -6.890605 -5.6558405 -4.4347630 -5.3063651 -7.98863484
## 10.27 -6.4854584 -9.585617 -8.5954582 -6.2875108 -5.5543340 -9.16304674
## 10.28 -0.5342389 -3.267917 -4.2692969 -4.8258201 -4.4623961 -5.06875912
## 10.29  0.4332408  1.208673 -0.2895929 -1.5318538 -3.1641748 -1.94952674
## 10.30  0.4088506  1.101306  1.4642322  1.3080103  0.5778192  2.10392807
## 10.31 -0.6480600 -1.459240 -1.4283706 -0.8678262 -0.4418413  0.08050285
##             2003       2004       2005        2006       2007        2008
## 10.26 -6.7774517 -8.4837323 -8.1417726 -7.33985781 -6.7622484 -6.67570860
## 10.27 -8.2911287 -9.3950601 -8.6616976 -7.85532651 -6.7023824 -6.89657593
## 10.28 -4.3687338 -7.0550231 -7.9364006 -8.02753487 -8.2346789 -7.33976472
## 10.29 -2.3719010 -0.9513094 -3.1770223 -3.79134413 -2.8517916 -5.38165712
## 10.30 -0.3791102  1.1636893  0.5976518  0.64984731  0.2404763 -0.21206523
## 10.31 -2.0258362 -1.5768118 -0.3402956  0.08872082 -0.9891323 -0.03411151
##             2009        2010        2011        2012         2013       2014
## 10.26 -6.7927592  -8.4238940  -7.9411253  -8.7304960  -9.50699113  -9.707349
## 10.27 -9.2180348 -10.3152173 -11.6560776 -10.7124392 -12.67269269 -10.375104
## 10.28 -9.9377581  -7.8268114  -9.2308714 -10.4825572 -14.15951727 -14.123842
## 10.29 -4.7700225  -4.3009108  -6.7931879 -10.9093072 -12.06427234 -10.165486
## 10.30  0.1720216  -2.6205780  -3.1849145  -4.9196868  -6.57865384  -6.460541
## 10.31  0.4828511   0.1067451   0.1859101  -0.2679054   0.09155862  -2.127128
##             2015
## 10.26  -8.684994
## 10.27  -9.886253
```

```
## 10.28 -14.645752
## 10.29 -12.026808
## 10.30  -8.933947
## 10.31  -4.941084
```

Everything looks correct, now we can do our CUSUM Analysis.

Step 7: CUSUM

As usual, before we perform CUSUM, we need to define our **no change period**.

Last time I used a period from 1st to 31st July.

Let's plot average seasonal factor values for each day to check if we can use the same period:

```
sf_avg<-data.frame(rowMeans(season_factors, n = 19))
colnames(sf_avg)<-'Avg_seasonal_factor'
head(sf_avg)
```

```
##       Avg_seasonal_factor
## 07.01           12.826311
## 07.02           11.183289
## 07.03           11.769472
## 07.04           11.141773
## 07.05            8.740071
## 07.06            6.691713
```

```
#ggplot(sf_avg, aes(x=1:123, y=Avg_seasonal_factor))+
  #geom_line()+
  #geom_point()

plot_ly(data=sf_avg, x=1:123, y=~Avg_seasonal_factor, type='scatter', mode='lines+markers', fill='tozer
```

Based on the graph, we can see that until day 40 (Aug 9), there were no negative values for seasonal factors. On day 44 factor values go back up, but are not as high as before.

Since the values would vary for each year, I would try to choose a safe range of first 51 days (1 July - 20 August). If we choose values up to day 60, for example, we could get more negative seasonal factor values, which might lower our mean but also increase standard deviation for some years. By increasing the period we risk adding 'summer end' values, if there are years when summer ended in the last week of August. Period of 51 days includes enough variability of the factors (they are not as stable in growth over the 123 days as temperatures were the last time) to calculate the optimal 'normal' values for each years' model. Moreover, from the last HW3 I remember that summer never ended before the last week of August (and more often in fall), so we are not including any values for fall here.

Moving on to the CUSUM, let's create vectors to store CUSUM models and violations for each year.

```
models_cusum <- vector(mode="list", length=ncol(season_factors))
violat_cusum <- vector(mode="list", length=ncol(season_factors))
```

Since we chose July as the no change period, we need to find the mean and standard deviation values for each year's seasonal factors:

```
#mean and stdev of seasonal factors for each year
#use July temps
mean_seasons <- vector(mode="list", length=0)
stdev_seasons <- vector(mode="list", length=0)
for (i in 1:ncol(season_factors)){
  mean_seasons  <- append(mean_seasons,mean(season_factors[1:51,i]))
```

```r
  stdev_seasons <- append(stdev_seasons,sd(season_factors[1:51,i]))

}
unlist(mean_seasons)
```

```
##  [1] 4.561409 4.559721 4.651009 4.692767 4.808380 4.833226 4.939809 5.058915
##  [9] 5.095669 5.114876 5.205778 5.321256 5.392601 5.482570 5.582841 5.654167
## [17] 5.727204 5.796219 5.816636
```

```r
unlist(stdev_seasons)
```

```
##  [1] 4.543606 4.534205 4.290408 4.145315 4.076333 3.901140 3.916727 3.839413
##  [9] 3.796743 3.502302 3.587251 3.619594 3.711506 3.802659 3.898234 3.990965
## [17] 4.367116 4.562904 4.477043
```

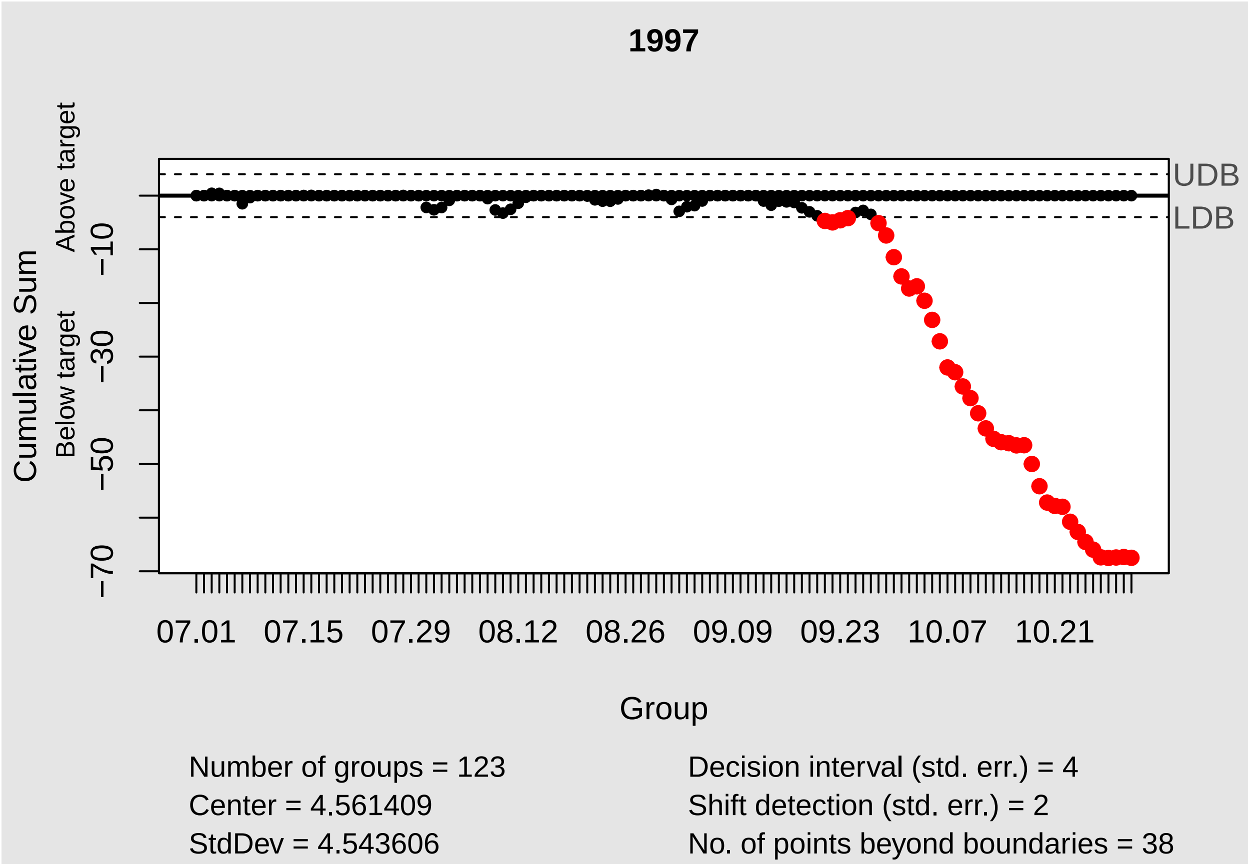
We have mean values ranging from 4.5 to 5.8 and st.dev. from 3.5 to 4.6 These values do not change so much year to year, which is good for our model and suggests that we chose the no change period right.

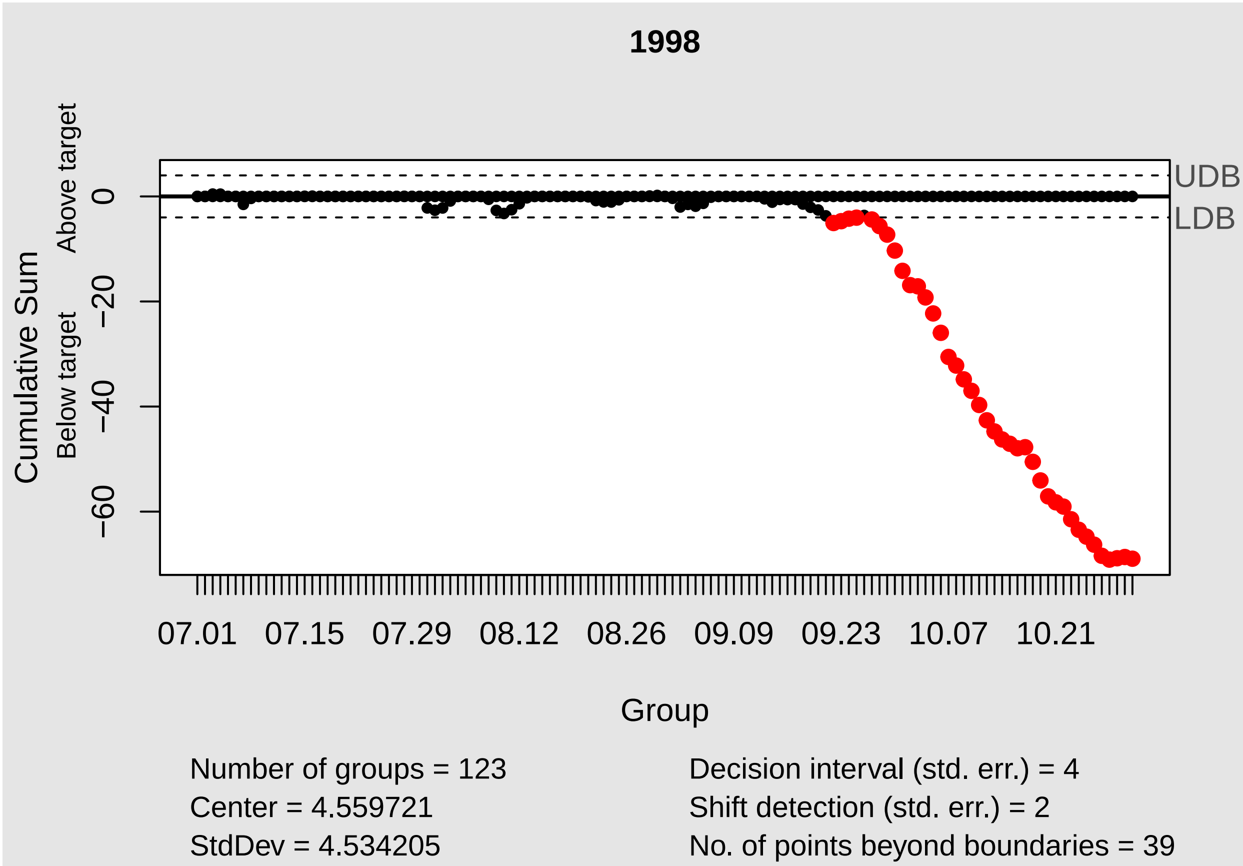
I will be using the cusum() function for each year. To start with, I will set C and T to standard values. C is usually set to 1 or 2 standard deviation, and T to 4 or 5 st.dev. I will set C to 1sd and T to 4 sd (since our st. devs are between 3.5 and 4.6, which is not that small considering the scale of our season factors, which go from 0 to 13 and 0 to -17). Then base on the results I will adjust those values to balance the sensitivity of the model.

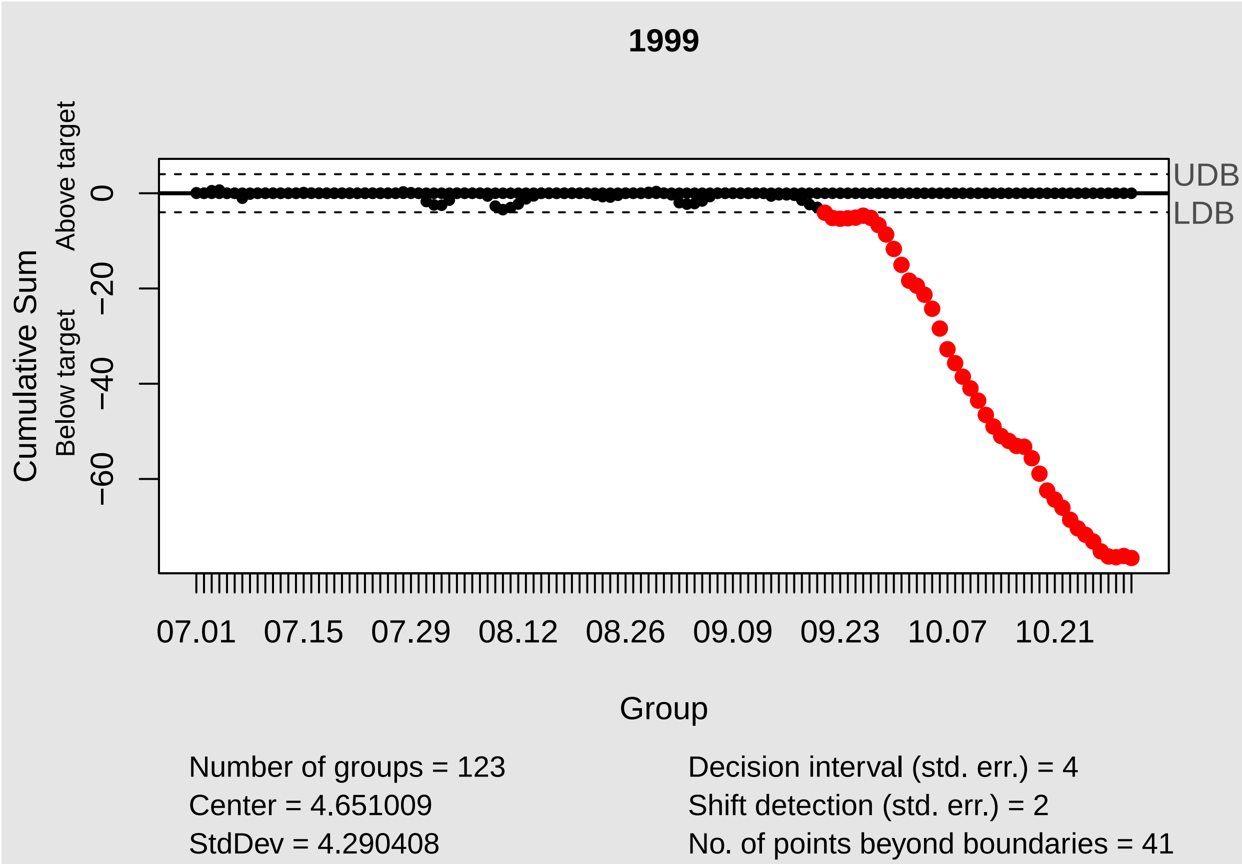
```r
dec_int <- 4 #4 st deviations for T
se_shift <- 2 #2 instead of 1, since shift = 2C. SO 1st deviation for C
```

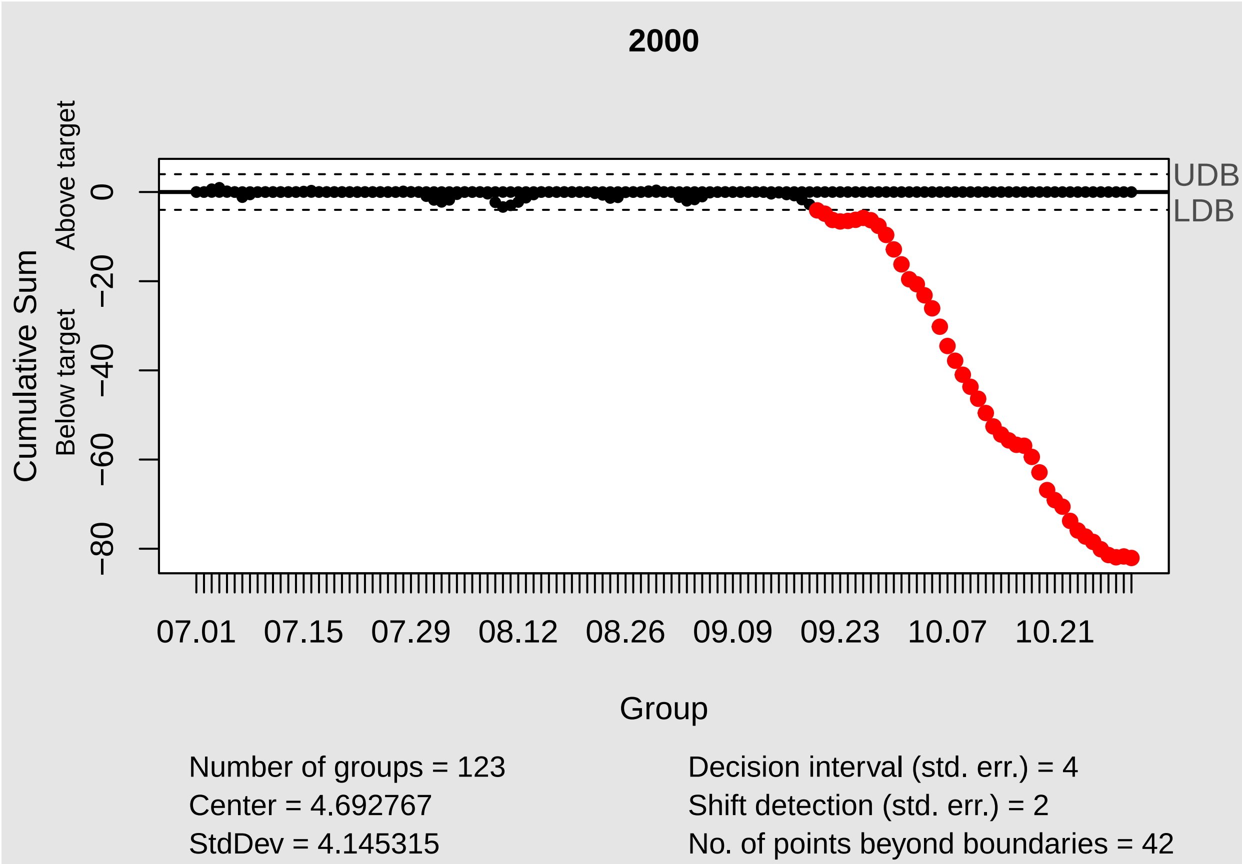
Finally, let's run CUSUM and see if we detect any violations

```r
for (y in 1:ncol(season_factors)) {
  #cusum model for each year
  models_cusum[[y]] <- cusum(season_factors[,y], #column of the year in loop
                             center = unlist(mean_seasons)[y],
                             std.dev = unlist(stdev_seasons)[y], #use st dev for the particular year
                             decision.interval = dec_int, #T value, 5 st dev-s
                             se.shift = se_shift, #shift value, 2C, 2
                             plot = TRUE,
                             title = colnames(season_factors)[y]) #add plot for each year's cusum
  #cusum violation for each year - moment when fall came
  #use the model we built above
  #$use lower violation - the first day that violated temp - the 1st day of falL!
  violat_cusum[[y]] <- models_cusum[[y]]$violations$lower}
```

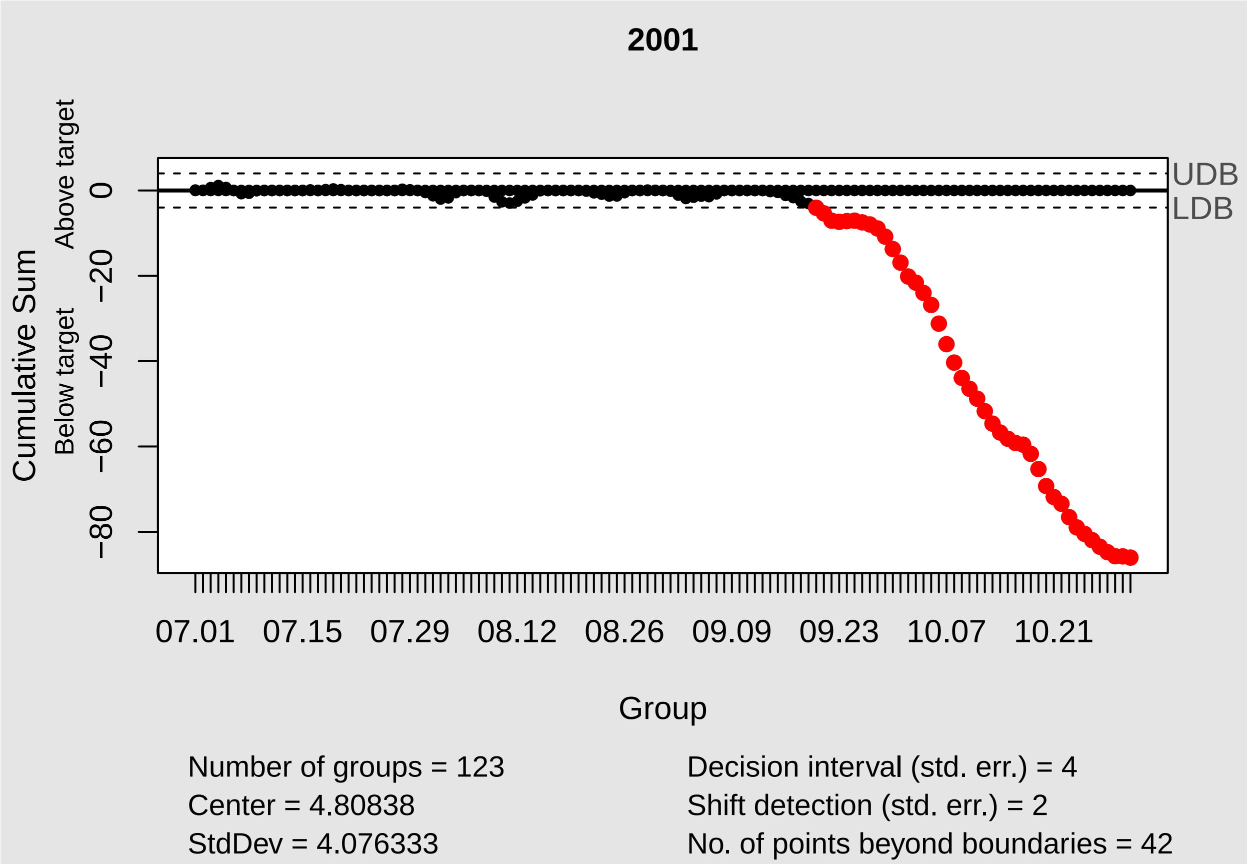

1997
Cumulative Sum
Above target
Below target
−10
−30
−50
−70
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 4.561409
StdDev = 4.543606
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 38

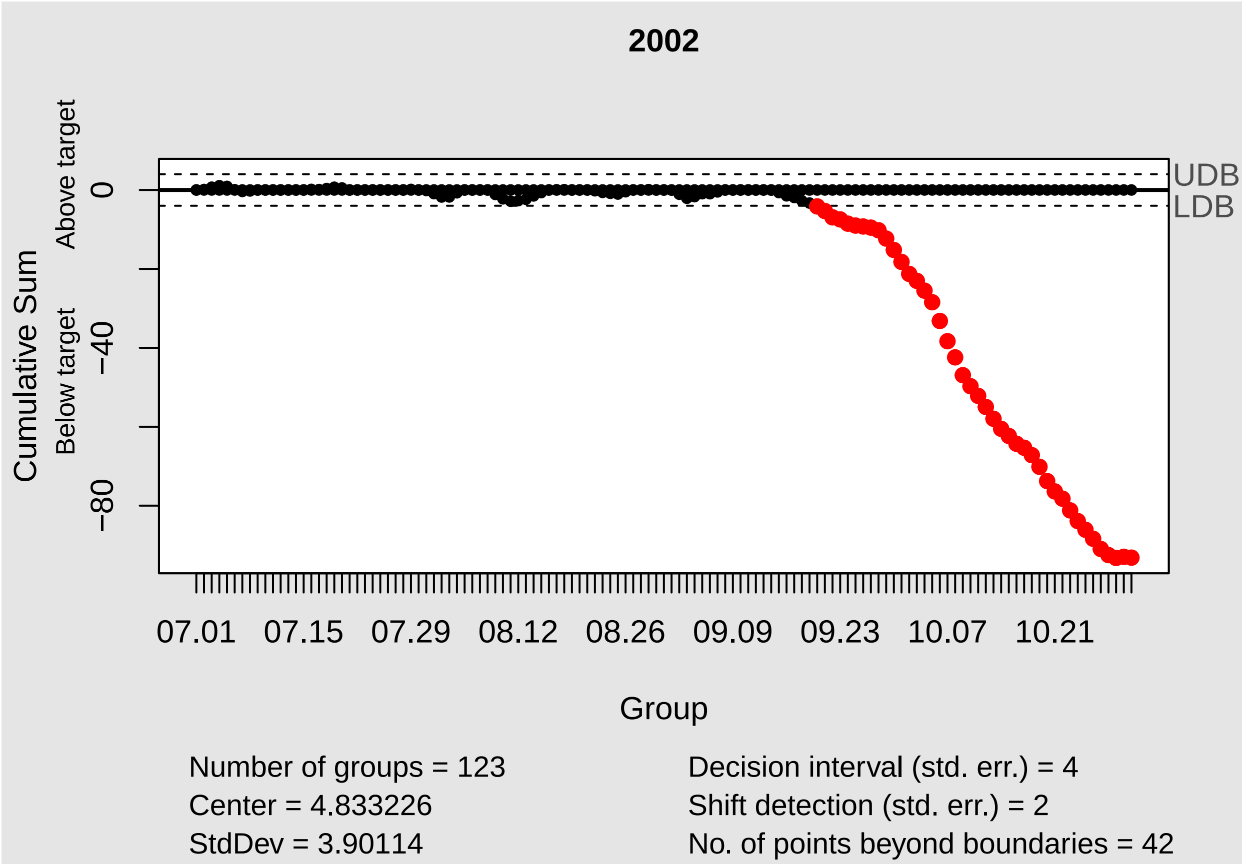

1998
Cumulative Sum
Above target
Below target
0
−20
−40
−60
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 4.559721
StdDev = 4.534205
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 39

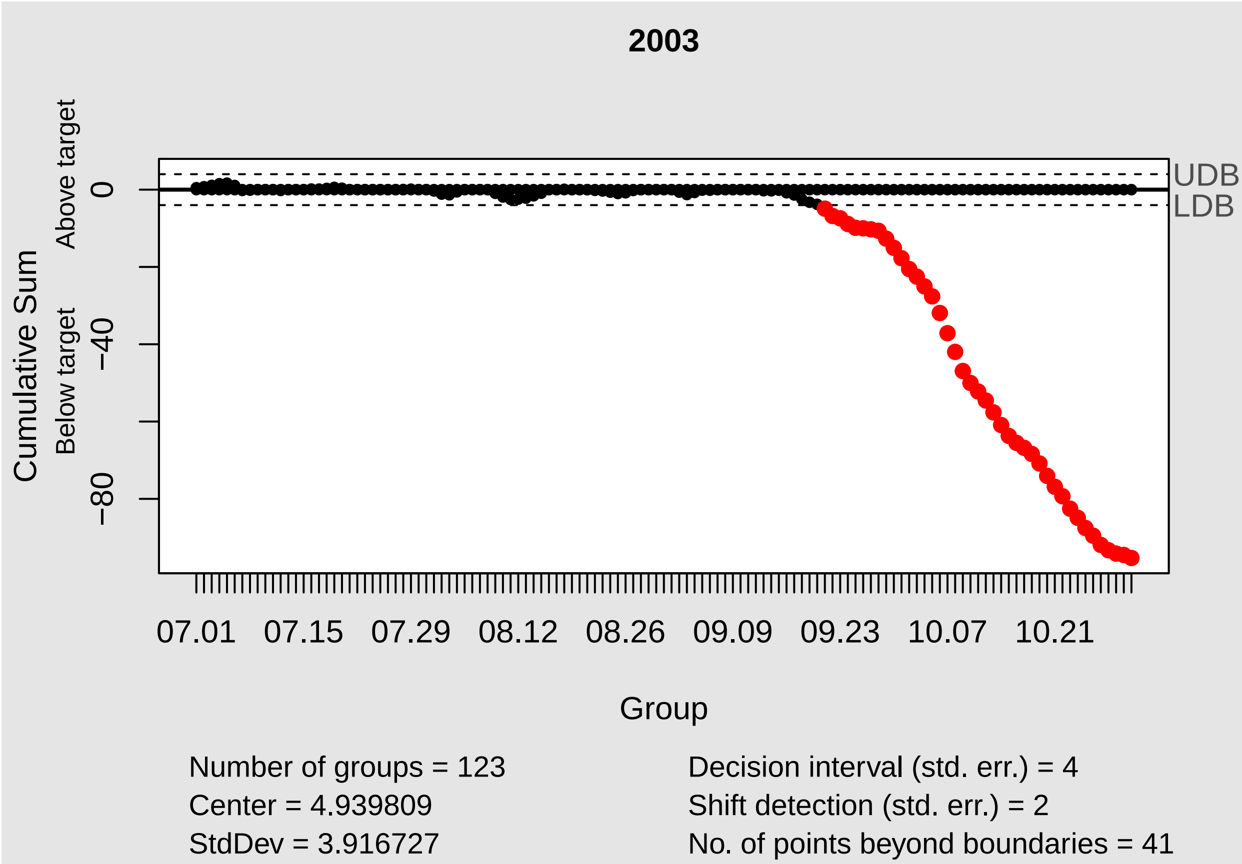

1999
Cumulative Sum
Above target
Below target
0
−20
−40
−60
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 4.651009
StdDev = 4.290408
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 41

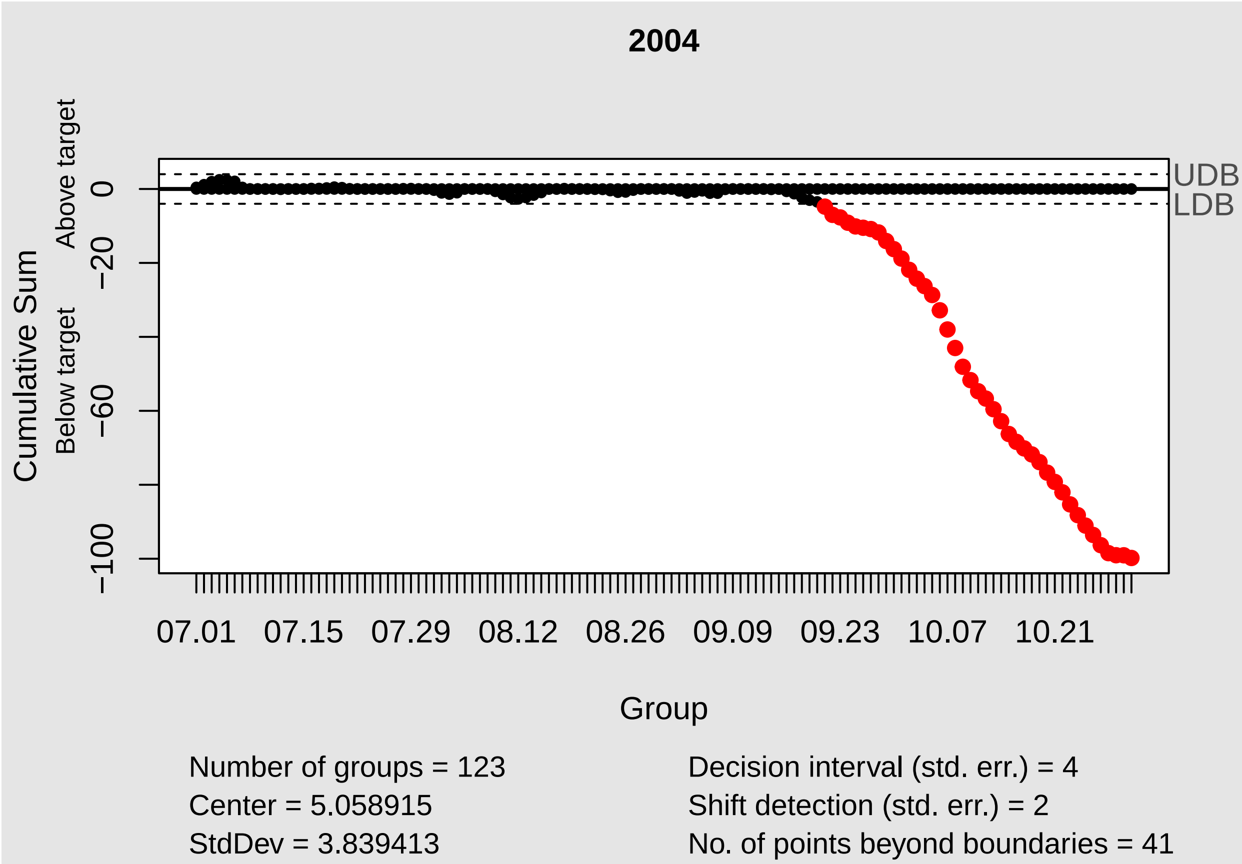


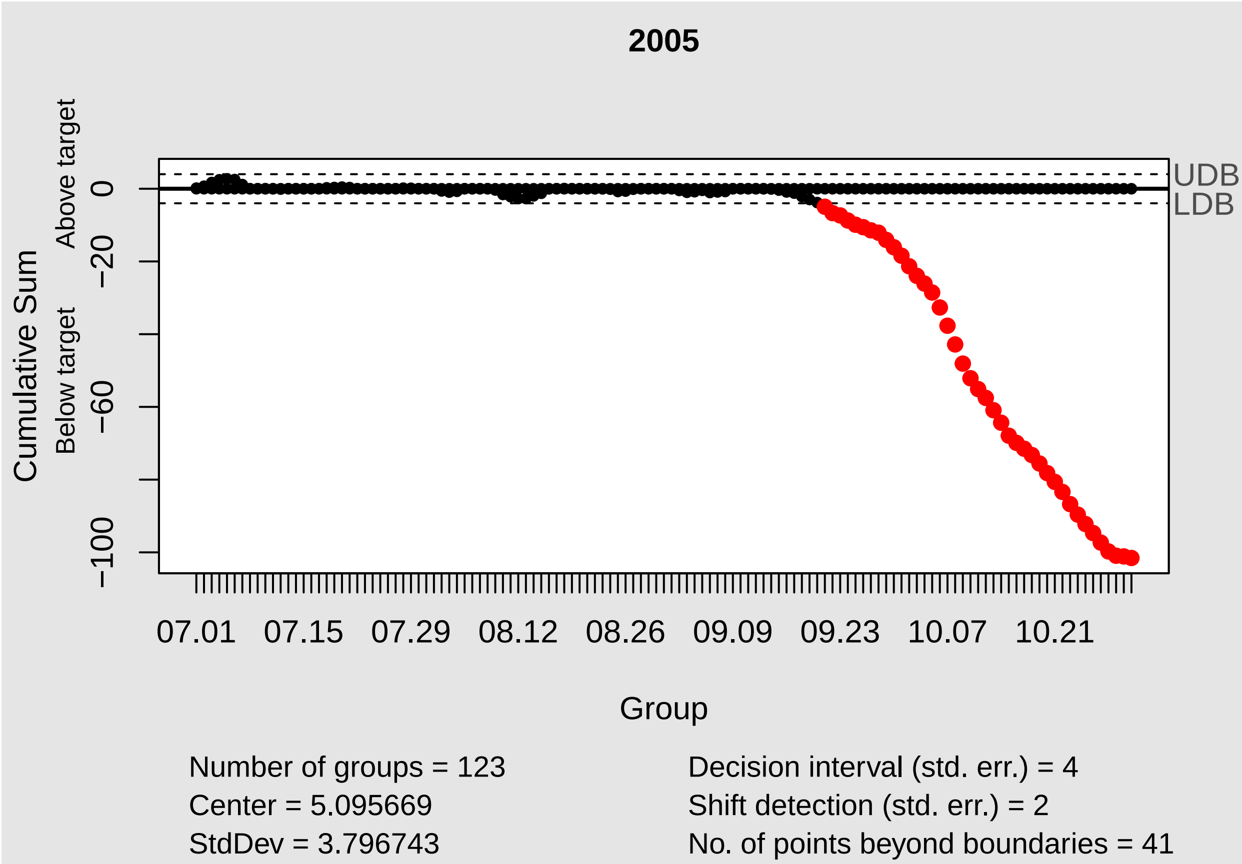
2000
Cumulative Sum
Above target
Below target
0
−20
−40
−60
−80
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 4.692767
StdDev = 4.145315
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 42

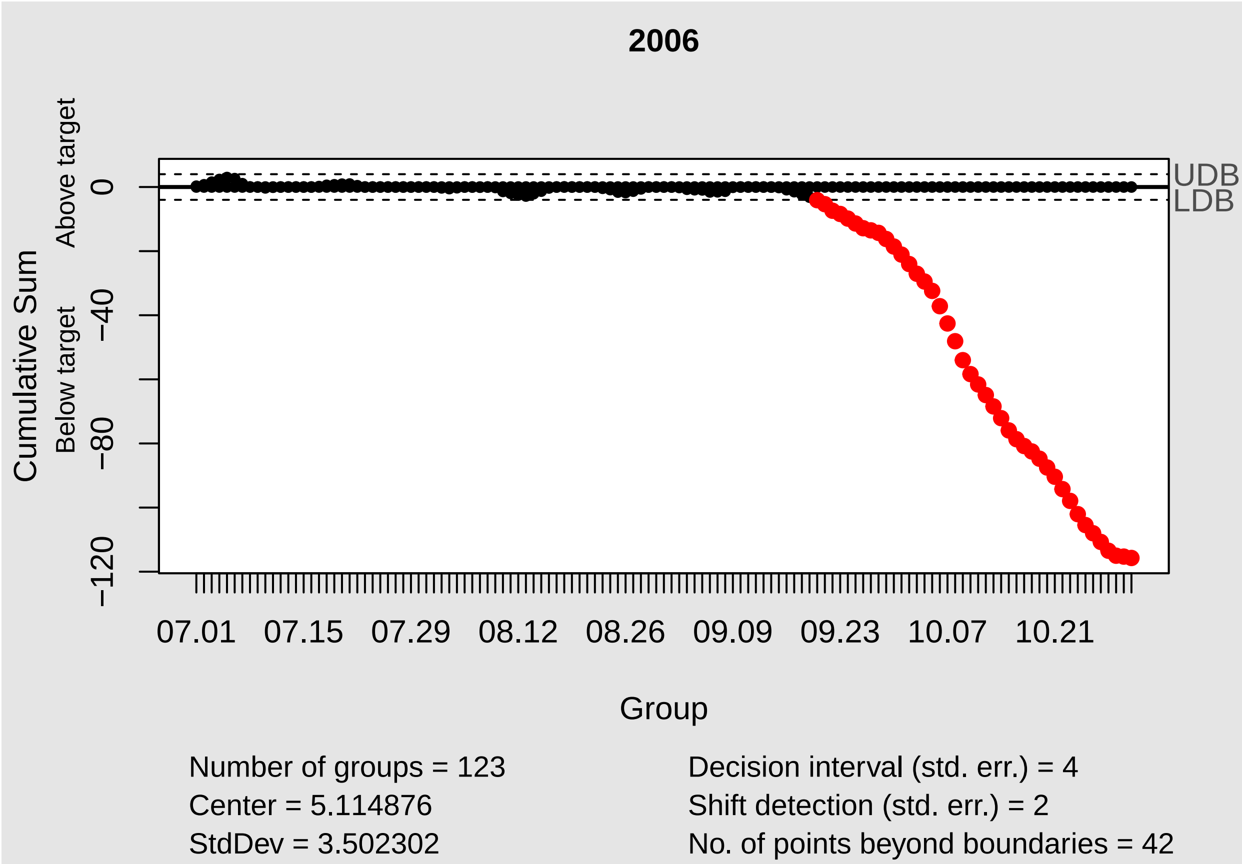

2001
Cumulative Sum
Above target
Below target
0
−20
−40
−60
−80
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 4.80838
StdDev = 4.076333
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 42

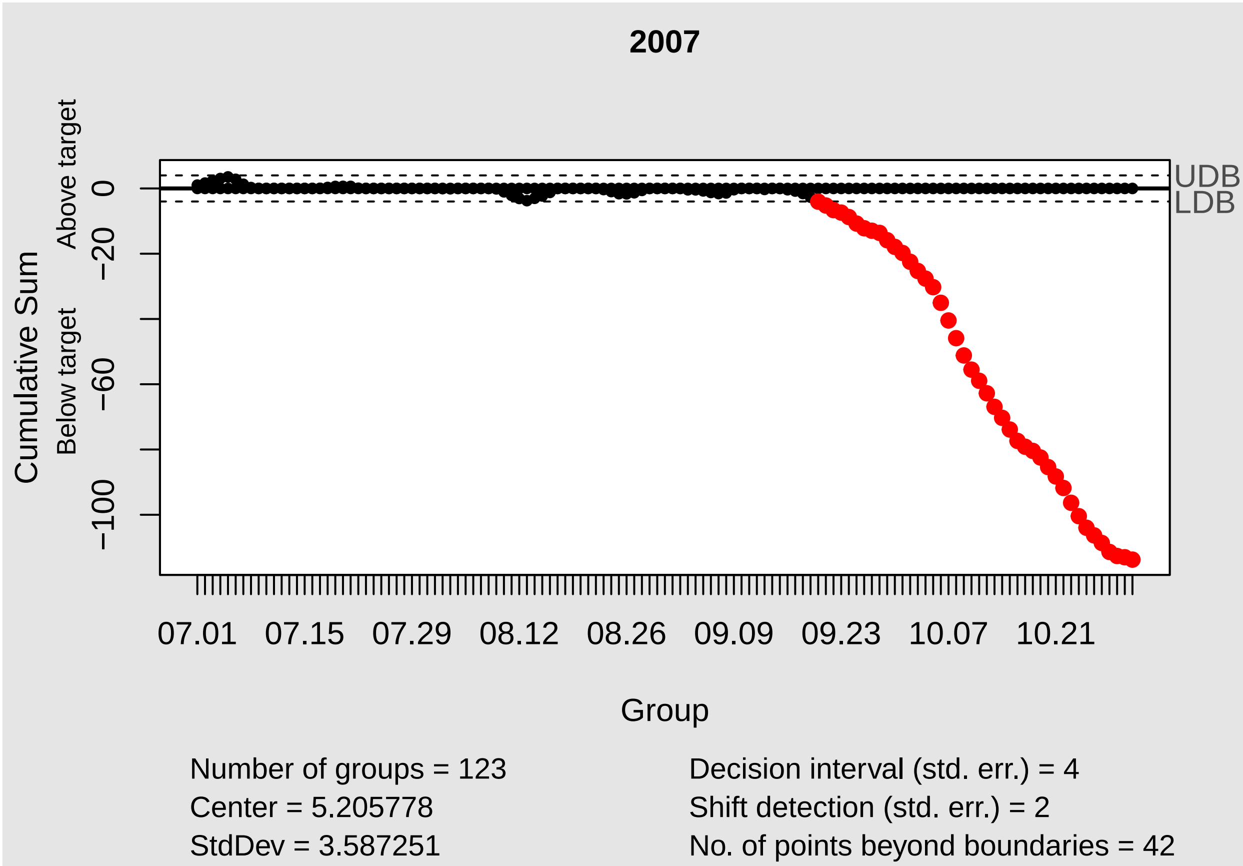

2002
Cumulative Sum
Above target
Below target
0
−40
−80
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 4.833226
StdDev = 3.90114
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 42

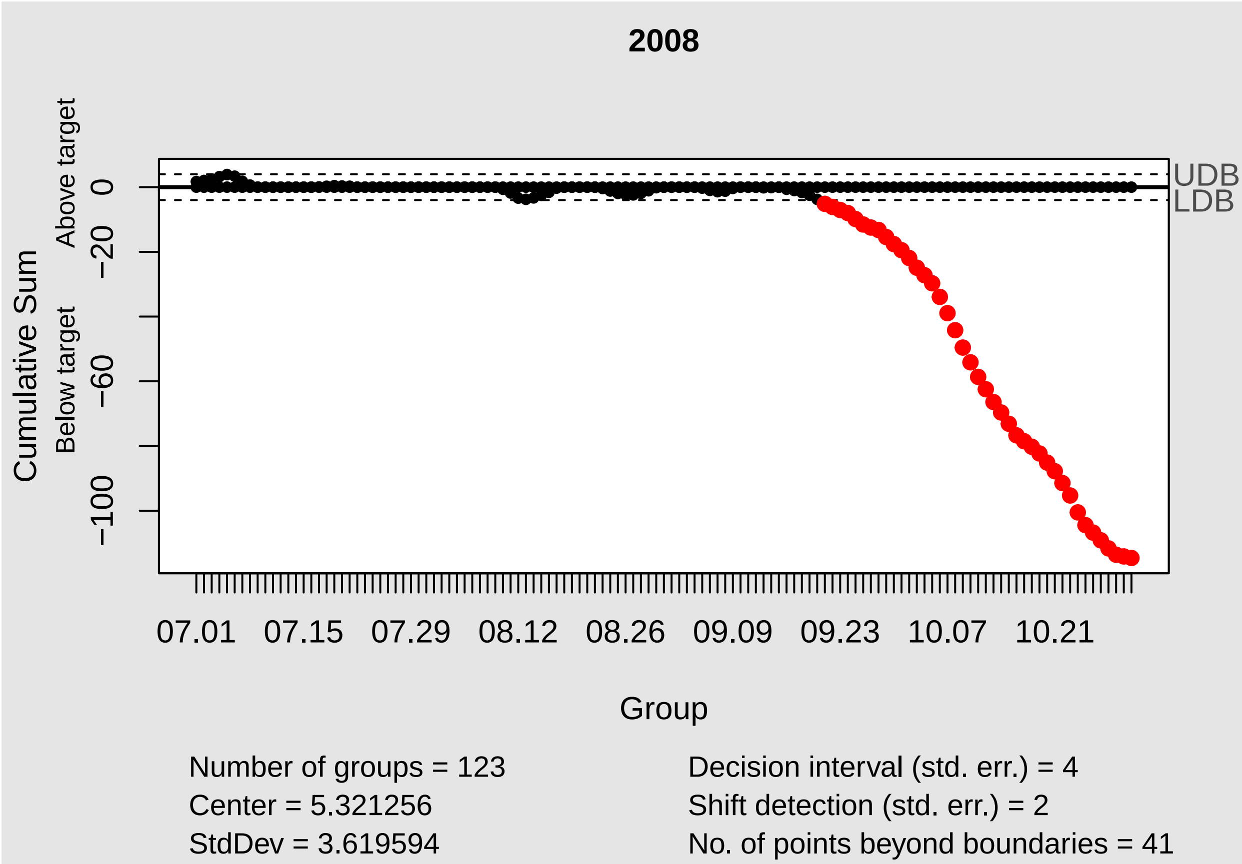

2003
Cumulative Sum
Above target
Below target
0
−40
−80
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 4.939809
StdDev = 3.916727
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 41

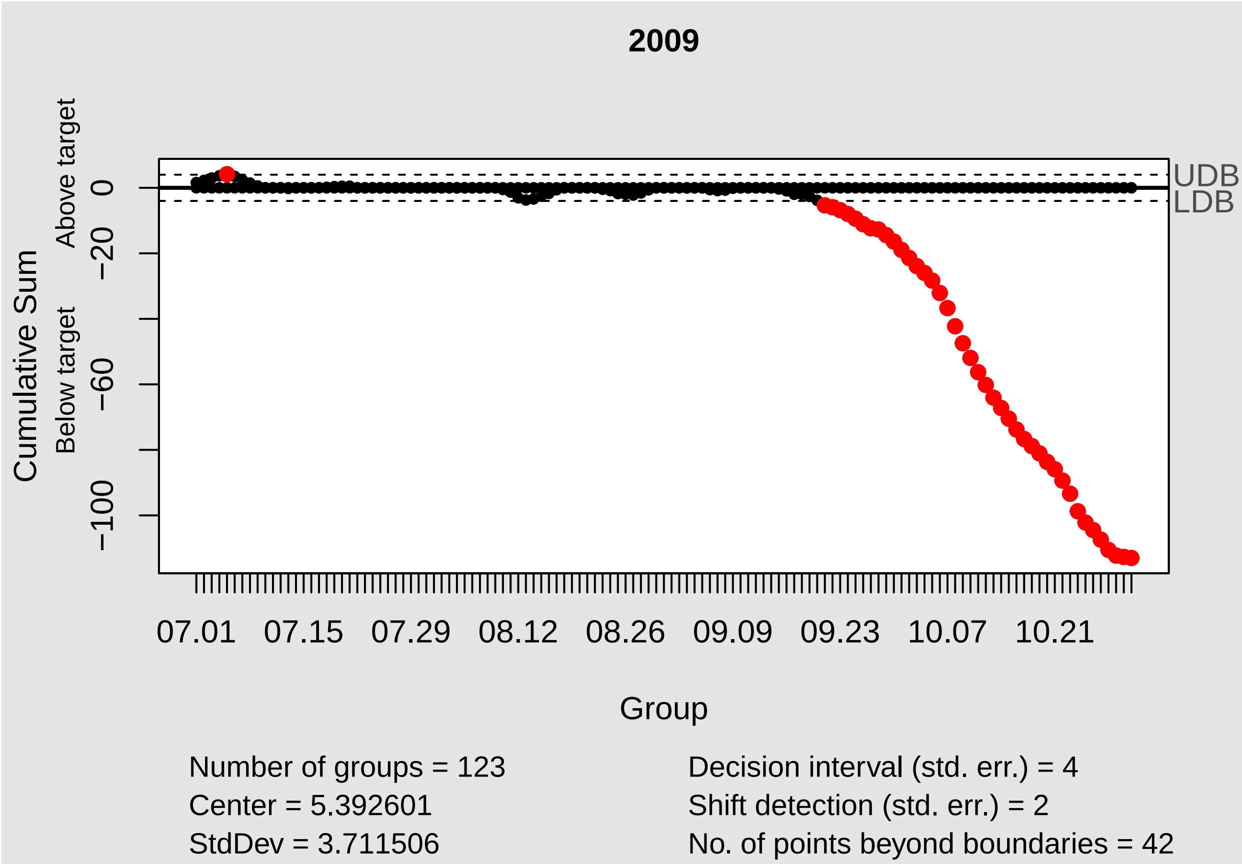

2004
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.058915
StdDev = 3.839413
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 41


2005
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.095669
StdDev = 3.796743
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 41

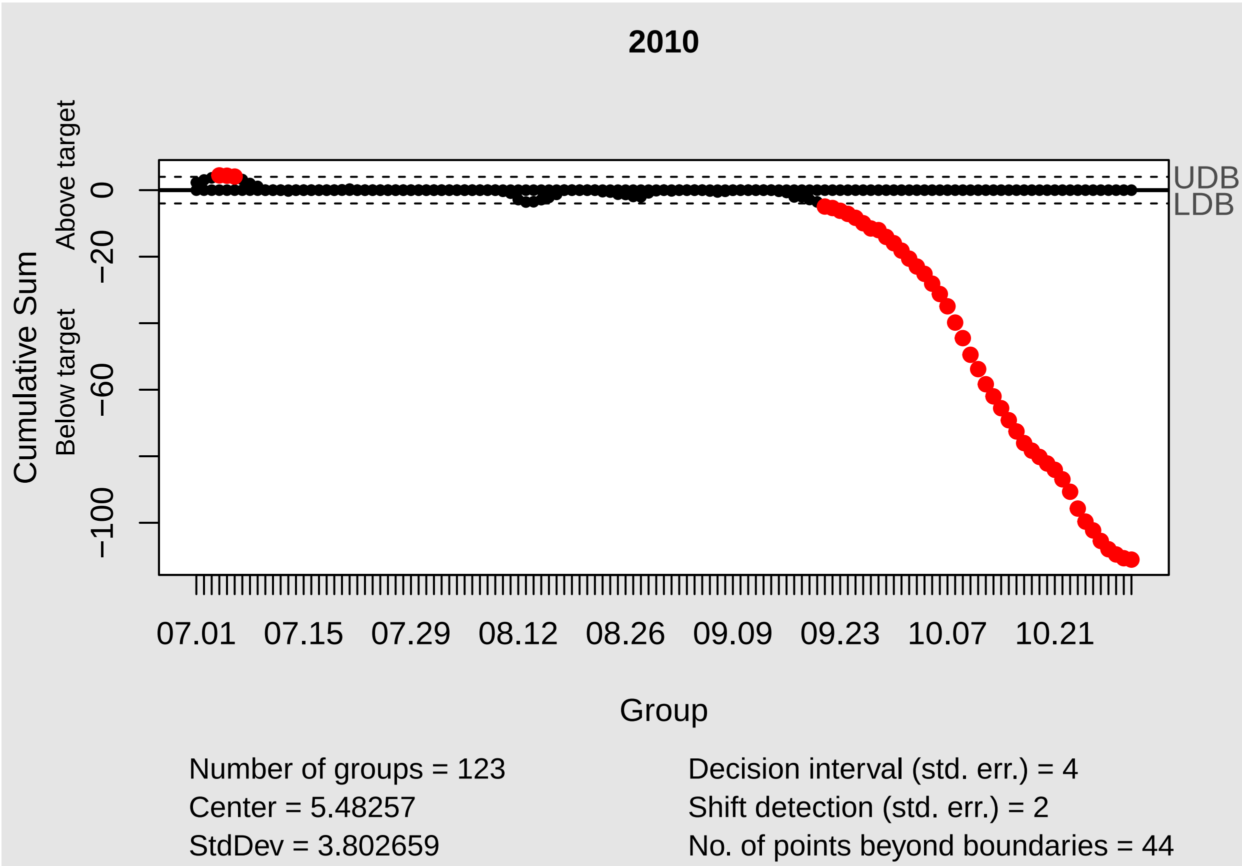

2006
Cumulative Sum
Above target
Below target
0
−40
−80
−120
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.114876
StdDev = 3.502302
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 42

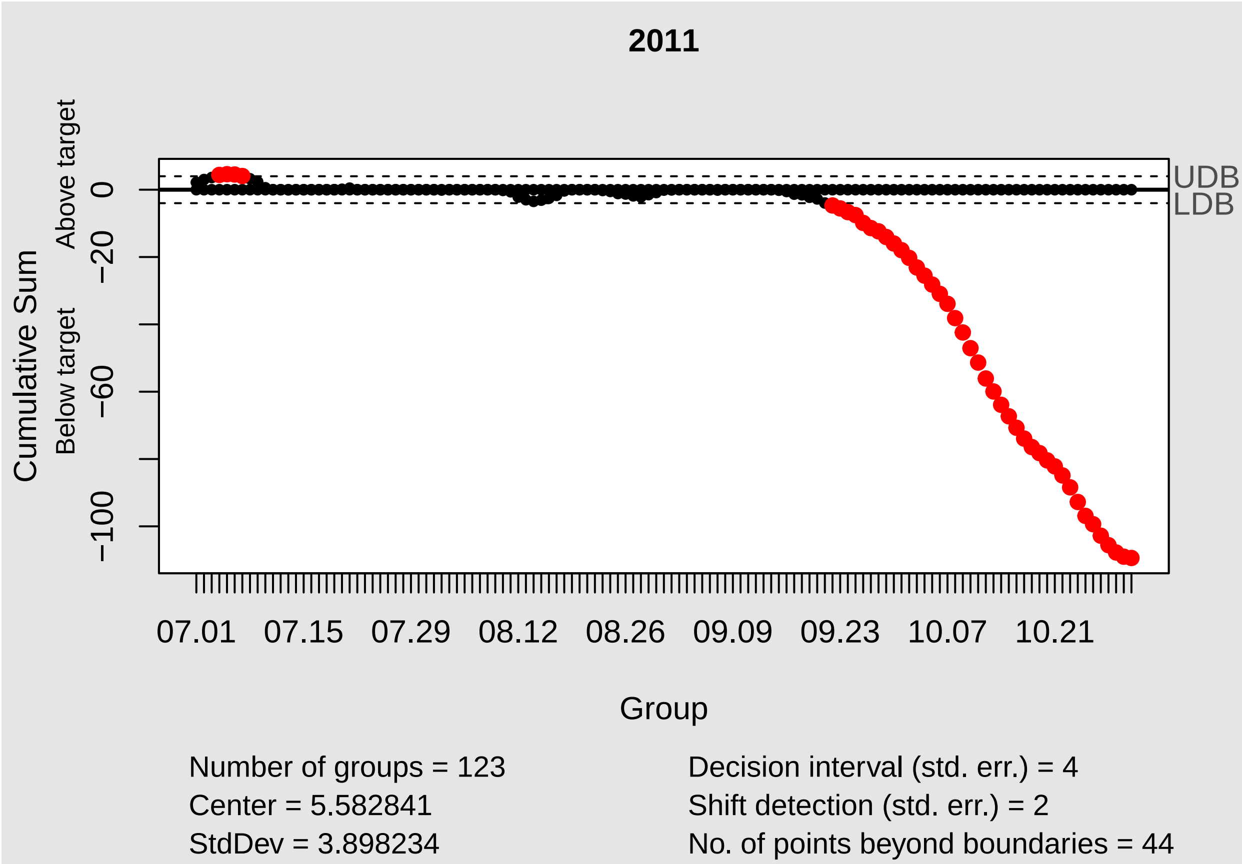

2007
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.205778
StdDev = 3.587251
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 42


2008
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.321256
StdDev = 3.619594
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 41

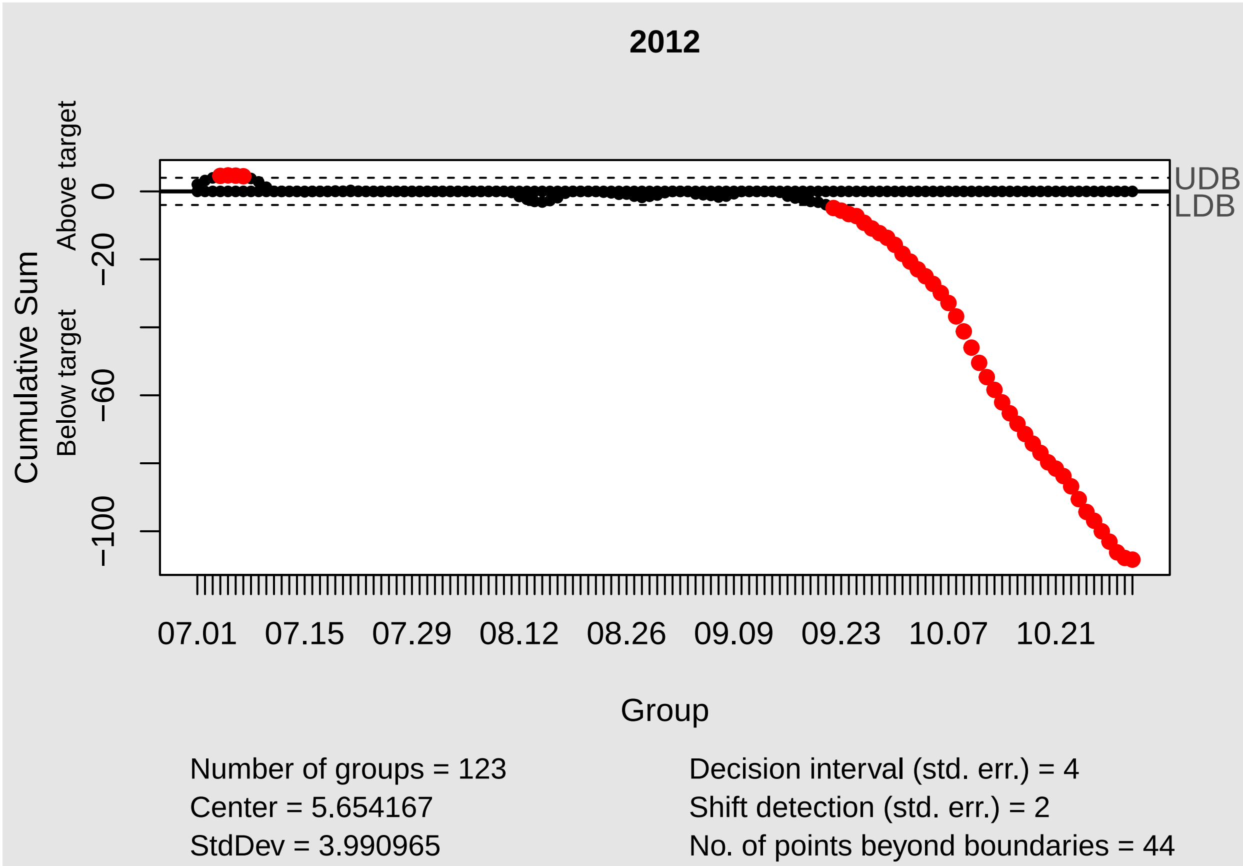


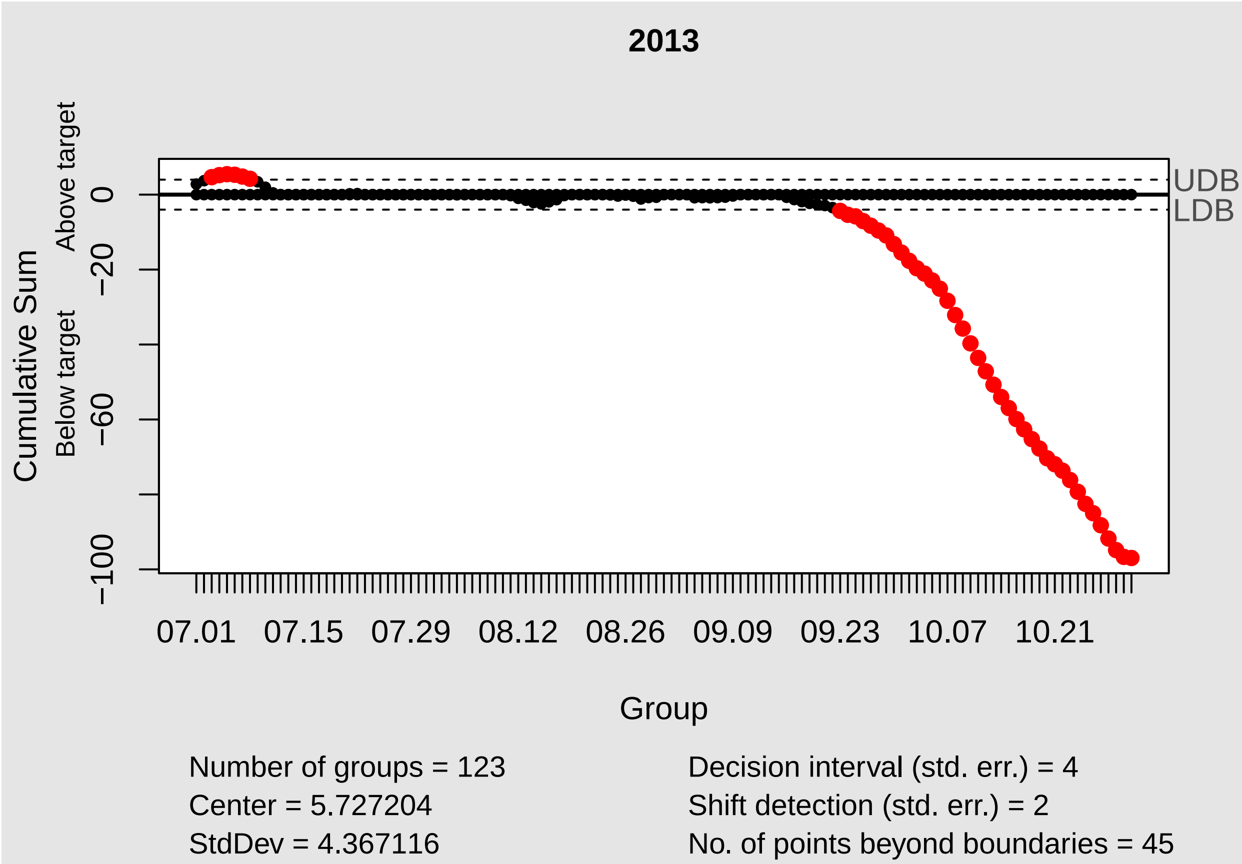
2009
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.392601
StdDev = 3.711506
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 42

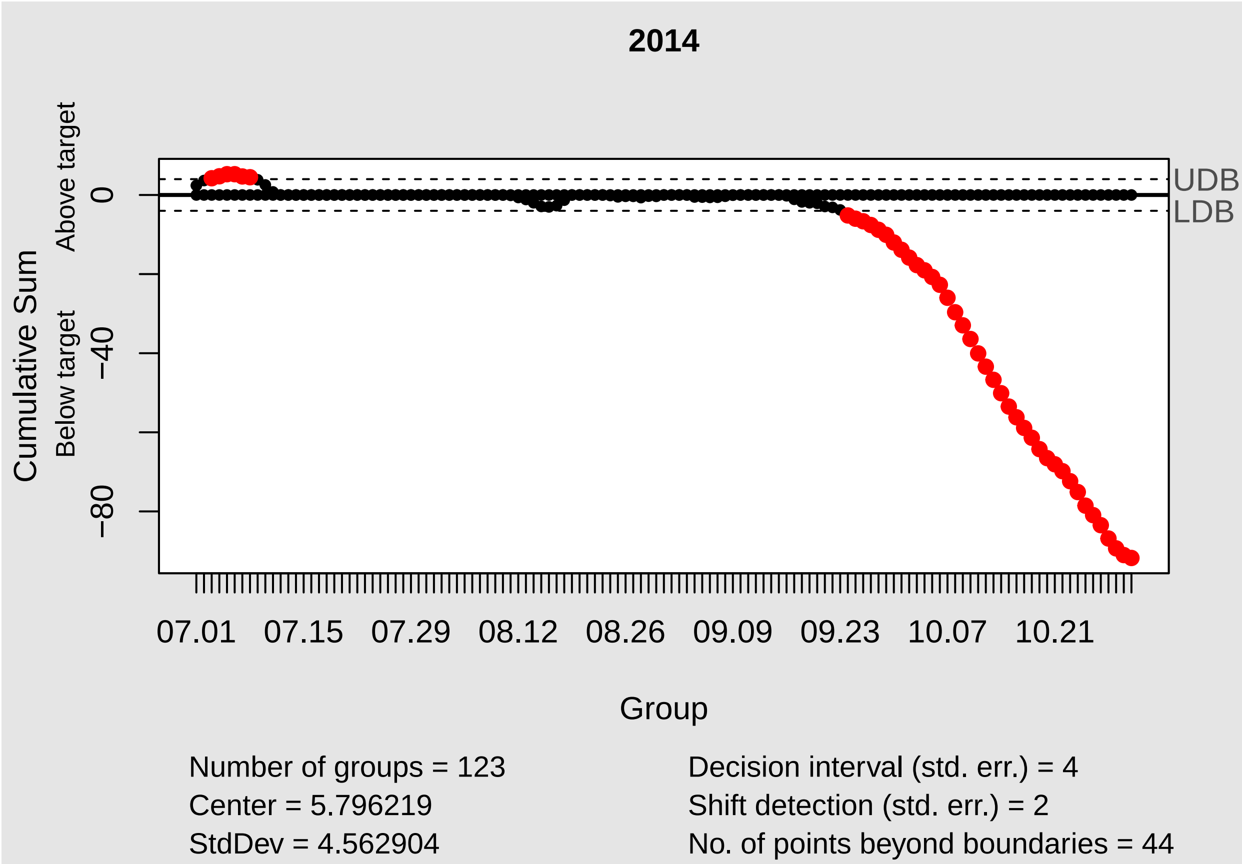

2010
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.48257
StdDev = 3.802659
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 44

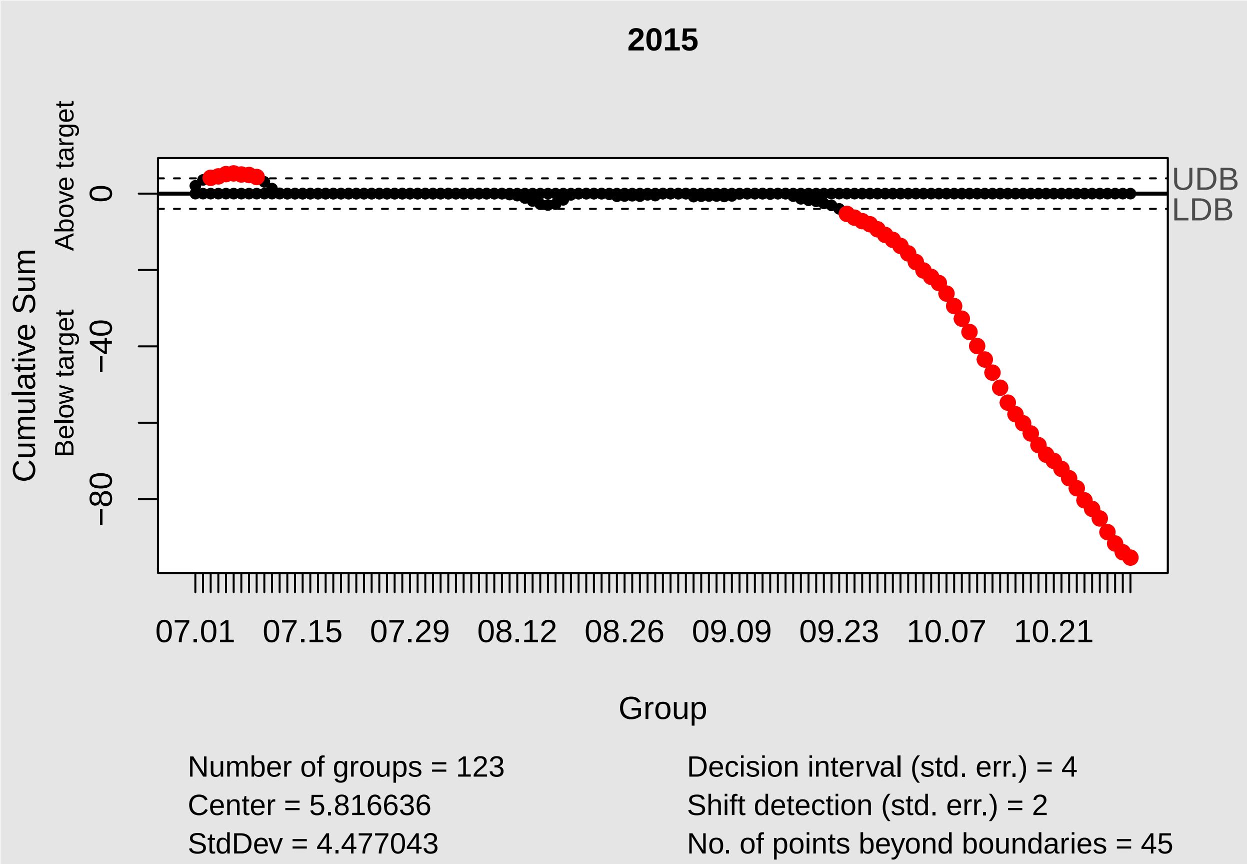


2011
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.582841
StdDev = 3.898234
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 44


2012
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.654167
StdDev = 3.990965
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 44


2013
Cumulative Sum
Above target
Below target
0
−20
−60
−100
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.727204
StdDev = 4.367116
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 45


2014
Cumulative Sum
Above target
Below target
0
−40
−80
UDB
LDB
07.01
07.15
07.29
08.12
08.26
09.09
09.23
10.07
10.21
Group
Number of groups = 123
Center = 5.796219
StdDev = 4.562904
Decision interval (std. err.) = 4
Shift detection (std. err.) = 2
No. of points beyond boundaries = 44



It seems like the choice for C and T was good. **We are looking for the lower violations on the graphs** - it is obvious that the change is usually detected after **the 3rd weeek of September** - a similar result compared to the one I got in the CUSUM project.

Visually, it **does not appear that the summer is ending later**. But let's see the exact dates for each year:

```
#vector to store first days of fall
first_days <- vector(mode="list", length=ncol(season_factors))

#add to that vector first days
for (y in 1:ncol(season_factors)){
  first_days[y] <- min(violat_cusum[[y]])
}
#day index of first day of fall
first_days <- unlist(first_days)

#turn indexes to days
fall_dates<- vector(mode="list", length=0)
for (x in 1:19){
  i=0
  d<-first_days[x]
  fall_dates <- append(fall_dates, data[d,1], after=i)
  i=i+1
  }

fall_dates<-unlist(fall_dates)
```

```
fall_dates
```

```
##  [1] "09.24" "09.24" "09.23" "09.22" "09.22" "09.21" "09.21" "09.21" "09.20"
## [10] "09.20" "09.21" "09.21" "09.21" "09.20" "09.20" "09.20" "09.21" "09.22"
## [19] "09.21"
```

We can see that the date for first violation does not vary significantly, and there is no increasing trend. To see that clearly, let's visualize the results.

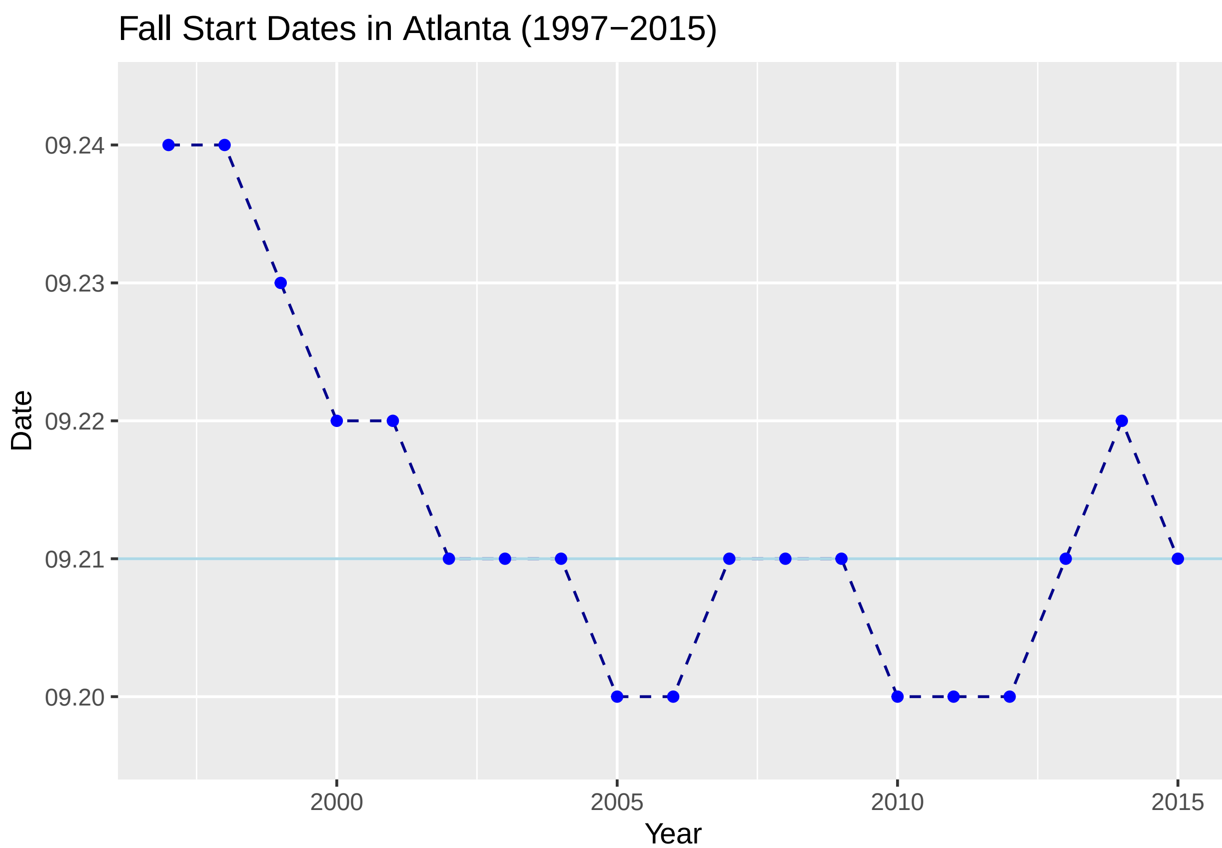
Data frame of unofficial fall start dates:

```
year_list <- 1997:2015
year_list <- as.list(year_list)
date_year <- do.call(rbind, Map(data.frame, Year=year_list, FallStart=fall_dates))
date_year
```

```
##    Year FallStart
## 1  1997     09.24
## 2  1998     09.24
## 3  1999     09.23
## 4  2000     09.22
## 5  2001     09.22
## 6  2002     09.21
## 7  2003     09.21
## 8  2004     09.21
## 9  2005     09.20
## 10 2006     09.20
## 11 2007     09.21
## 12 2008     09.21
## 13 2009     09.21
## 14 2010     09.20
## 15 2011     09.20
## 16 2012     09.20
## 17 2013     09.21
## 18 2014     09.22
## 19 2015     09.21
```

Plot:

```
results_plot<- ggplot(data = date_year, aes(x = Year, y = FallStart, group=1)) +
  geom_line(linetype="dashed", color="darkblue") +
  geom_hline(yintercept = 2, color="lightblue")+ #compare to 09.21 for 2015
  geom_point(color="blue")+
  labs(title="Fall Start Dates in Atlanta (1997-2015)", y="Date")

results_plot
```



As we can see, **there is no evidence that the summers have become longer**. Compared to 1997-1999, it even seems that summers have become slightly shorter. Hence, **the answer to our main question is No.**

Step 8: Extra - comparing results with and without smoothing

I would like to compare the results to my CUSUM project where I defined fall start for each year without exponential smoothing. I will not repeat CUSUM here, and just copy the start dates for each year. These are result for C=1sd and T=5sd.

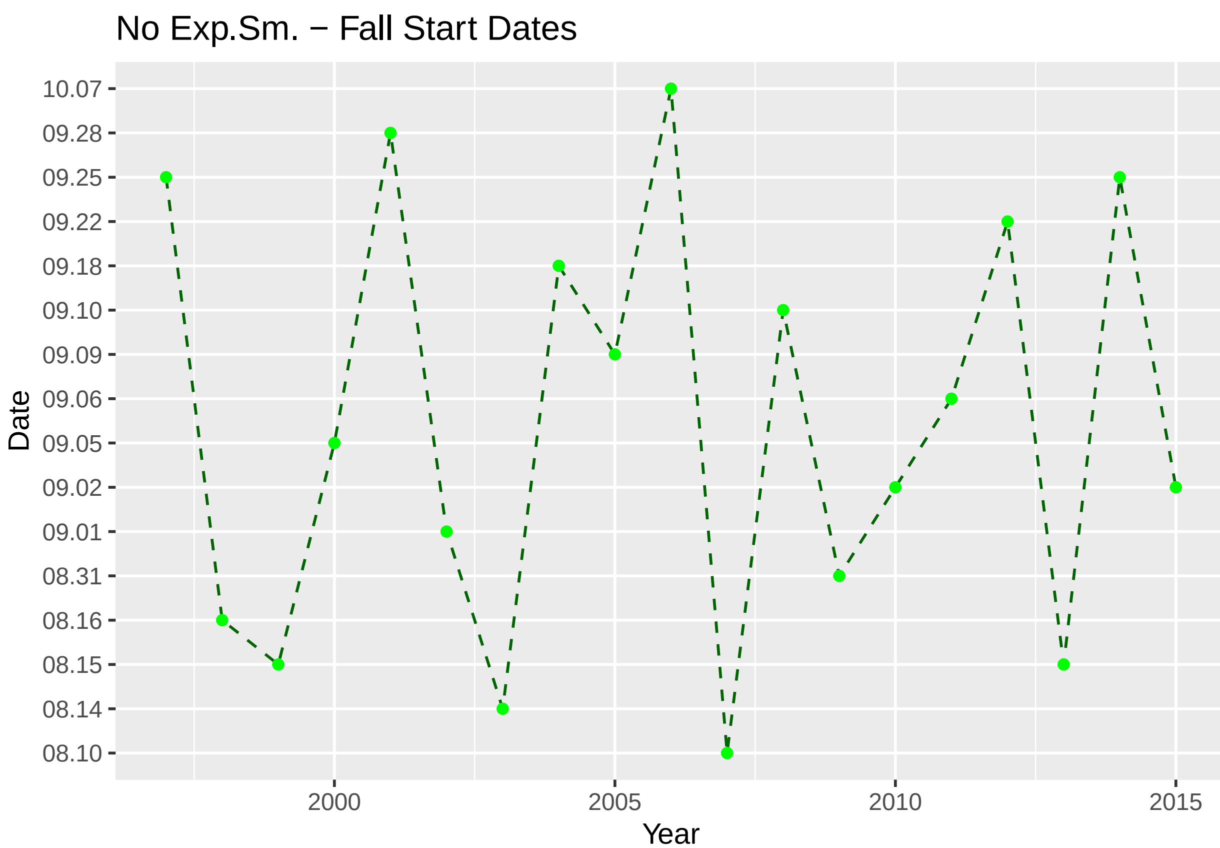
```
fall_dates_hw3 <- c('09.25','08.16','08.15','09.05','09.28','09.01','08.14','09.18','09.09','10.07','08
year_list1 <- 1997:2015
year_list1 <- as.list(year_list)
date_year1 <- do.call(rbind, Map(data.frame, Year=year_list, FallStart=fall_dates_hw3))
date_year1
```

```
##    Year FallStart
## 1  1997     09.25
## 2  1998     08.16
## 3  1999     08.15
## 4  2000     09.05
## 5  2001     09.28
## 6  2002     09.01
## 7  2003     08.14
## 8  2004     09.18
## 9  2005     09.09
## 10 2006     10.07
```

```
## 11 2007      08.10
## 12 2008      09.10
## 13 2009      08.31
## 14 2010      09.02
## 15 2011      09.06
## 16 2012      09.22
## 17 2013      08.15
## 18 2014      09.25
## 19 2015      09.02
```

Plot previous results and compare with new ones:

```r
previous<-ggplot(data = date_year1, aes(x = Year, y = FallStart, group=1)) +
  geom_line(linetype="dashed", color="darkgreen") +
  geom_point(color="green")+
  labs(title="No Exp.Sm. - Fall Start Dates", y="Date")
previous
```



The effect of exponential smoothing is indeed drastic. Before it, fall start dates varied from Aug 10 to Oct 07 - almost 2 months' difference compared to just 4 days (09.20-09.24) with exponential smoothing.

To conlcude, Exponential smoothing has helped us get rid of the excess 'noise' and randomness in data, showed that there is no trend in temperature changes year to year, and proved that the summers are not getting longer.