Forecasting Weather Changes

Eric Cai

2023-09-21

Forecasting Weather Changes Using Exponential Smoothing

Using the 20 years of daily high temperature data for Atlanta (July through October) (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.

Preparing Our Data

We load in our file as usual and inspect it.

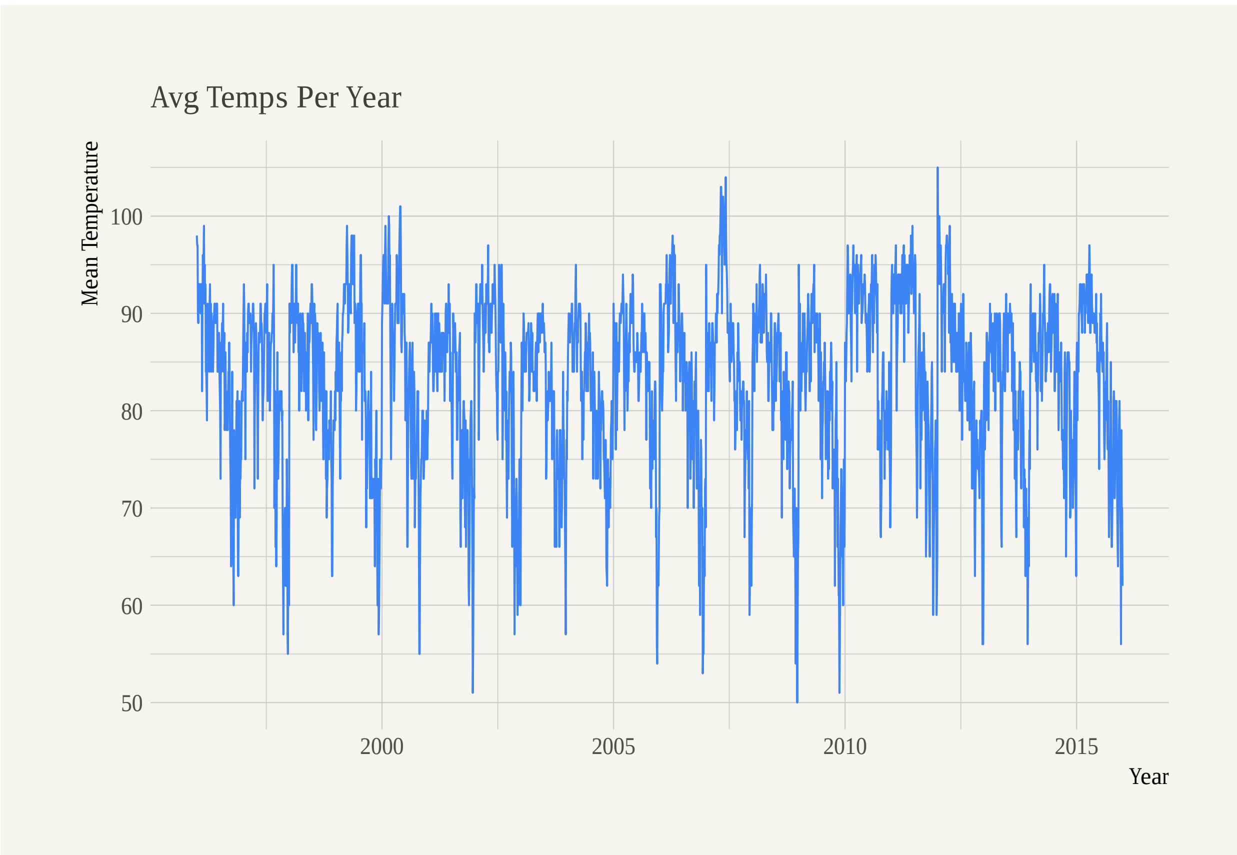
```
file <- 'temps.txt'
temps <- read.table(file,header=T)
col_names <- seq(1996,2015,by=1)
colnames(temps)[-1] <- col_names
head(temps)
```

```
##     DAY 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009
## 1 1-Jul   98   86   91   84   89   84   90   73   82   91   93   95   85   95
## 2 2-Jul   97   90   88   82   91   87   90   81   81   89   93   85   87   90
## 3 3-Jul   97   93   91   87   93   87   87   87   86   86   93   82   91   89
## 4 4-Jul   90   91   91   88   95   84   89   86   88   86   91   86   90   91
## 5 5-Jul   89   84   91   90   96   86   93   80   90   89   90   88   88   80
## 6 6-Jul   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
```

In order to build an exponential smoothing model, we need to first convert our data into a time-series object. If we take a look at `HoltWinters`, it takes in a ts object. But if we inspect the `ts()` function, we'll notice that it only takes in a vector or a matrix of our time-series values. It's like peeling back one layer of onion only for us to have to peel back another layer!

So, first, we convert our temps data into a vector and then convert it into a time-series object. This is optional, but I chose to plot how our data looks like as a time-series object.

```
temps_vect <- as.vector(unlist(temps[,2:21]))
temps.ts <- ts(temps_vect, start=colnames(temps)[2],
               frequency=nrow(temps))
ggplot(data.frame(Year = index(temps.ts), Temperature = coredata(temps.ts)),
       aes(x = Year, y = Temperature)) +
  geom_line(color = '#3D85F7', linewidth=0.35) +
  labs(x = "Year", y = "Mean Temperature", title = "Avg Temps Per Year") +
  theme_ipsum() +
  theme(
    plot.title = element_text(size = 12,
                              face = "bold",
                              color = "grey25"),
    axis.text.x = element_text(size = 9),
    axis.text.y = element_text(size = 9),
    plot.background = element_rect(fill = "#F5F4EF", color = NA)
  )
```



The Holt-Winters' Method

Looks gnarly, but the plot actually helps us identify what kind of smoothing model we need for our data. If we examine the plot, we see some familiar patterns that wax and wane for the same 4-month time period across 20ish years.

This indicates that we may be dealing with data that has seasonal trends (obviously, weather patterns are rather predictable). As a result, we'll seek to build a model that incorporates the level smoothing equation, plus the trend, and seasonal component, since we're trying to conclusively determine if there is a trend (apart from just looking at the plot).

This is otherwise known as the Holt-Winters' (HW) seasonal method, a generalization of exponential smoothing. There are two "flavors" of the HW method: *additive* and *multiplicative*.

The additive method is preferred when the 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. I'm not sure which one to choose, so I'm going to try both.

But before we do that, we need to determine our best parameters for our Holt-Winters' model. If we don't supply parameters for the model, the function will determine the best values on its own. Before I have the function do it for me, I want to determine the α, β, γ values on our own.

Choosing our α, β, γ Values

```
## we initialize a huge dataframe that consists of numerous alpha,
## beta, gamma values incremented by 0.1
parameter_grid <- as.data.frame(expand.grid(alpha = seq(0.1, 1, by = 0.1),
                                            beta = seq(0, 1, by = 0.1),
                                            gamma = seq(0.1, 1, by = 0.1),
                                            seasonal=c('additive',
                                                       'multiplicative')))
## here's an example of what the dataframe looks like
head(parameter_grid[1:6,])
```

```
##   alpha beta gamma seasonal
## 1   0.1    0   0.1 additive
## 2   0.2    0   0.1 additive
## 3   0.3    0   0.1 additive
## 4   0.4    0   0.1 additive
## 5   0.5    0   0.1 additive
## 6   0.6    0   0.1 additive
```

```
head(parameter_grid[1101:1106,])
```

```
##      alpha beta gamma       seasonal
## 1101   0.1    0   0.1 multiplicative
## 1102   0.2    0   0.1 multiplicative
## 1103   0.3    0   0.1 multiplicative
## 1104   0.4    0   0.1 multiplicative
## 1105   0.5    0   0.1 multiplicative
## 1106   0.6    0   0.1 multiplicative
```

```
## initialize empty variables so we can store them in our loops
best_mse_additive <- Inf
best_params_additive <- NULL
best_mse_multiplicative <- Inf
best_params_multiplicative <- NULL
## this will loop through our seasonal factors
for(type in c('additive','multiplicative')){
        ## this will loop through the length of parameter_grid
```

```r
        for(i in 1:nrow(parameter_grid)){
                ## we set each row as params
                params <- parameter_grid[i,]
                ## then we access the seasonal column and check if it matches
                ## type, if it does, the following happens
                if(params$seasonal==type){
                        ## we supply our model with the values of each params row
                        hw_model <- HoltWinters(temps.ts,
                                                alpha = params$alpha,
                                                beta = params$beta,
                                                gamma = params$gamma,
                                                seasonal=type)
                }
                        ## calculate the mean squared error (MSE)
                        mse <- hw_model$SSE/nrow(hw_model$fitted)
        ## then moving out of the inner loop, we check our type
        if(type=='additive'){
                ## if it meets the condition above, we pass it through another test
                if(mse < best_mse_additive) {
                        ## if our current MSE is less than the best,
                        ## then we update it as our best and repeat until it
                        ## reaches its highest value
                        best_mse_additive <- mse
                        best_params_additive <- params
                }
        } else{
                ## if type != 'additive', then this happens
                if(mse < best_mse_multiplicative){
                        ## we repeat the same process for multiplicative
                        best_mse_multiplicative <- mse
                        best_params_multiplicative <- params
                        }
                }
        }
}
best_params <- rbind(best_params_additive, best_params_multiplicative)
best_params$RMSE <- c(sqrt(best_mse_additive),sqrt(best_mse_multiplicative))
colnames(best_params) <- c("Alpha", "Beta", "Gamma", "Seasonality", "RMSE")
kable(
        best_params,
        align='c',
        caption='Best Values'
)
```

Table 1: Best Values

	Alpha	Beta	Gamma	Seasonality	RMSE
667	0.7	0	0.7	additive	5.326015
1546	0.6	0	0.5	multiplicative	5.431907

What do our α, β, γ values mean in the context of our data set? As we learned, α controls the smoothing of the level component. It determines how much weight should be given to the most recent observation when updating the estimated level. This is our level equation:

$$S_t = \alpha x_t + (1-\alpha)S_{t-1} \quad 0 < \alpha \leq 1$$

The higher α is the more weight we give to the most recent observation. Conversely, the lower α is the more we rely on past observations. An α value of 0.7 suggests that the model places relatively high weight on the most recent observation when updating the level component.

β controls the smoothing of the trend component, determining how much weight should be given the most recent change in the level when updating the estimated trend.

$$T_t = \beta(S_t - S_{t-1}) + (1-\beta)T_{t-1}$$

Like α, β determines how much weight should be given to the most recent change in the level. A higher β places more weight on recent changes, while a lower β places more weight on historical changes in the level equation. A β of 0.1 suggests that the model places relatively low weight on recent changes, implying that our trend is relatively stable and is less responsive to recent changes.

Finally, γ controls the smoothing of the seasonal component. What if our data shows trends and seasonality? In this case, double smoothing will not work, and we will need to include a seasonal component, which includes the γ parameter. γ determines how much weight should be given to seasonal patterns when updating the estimated season component, given as:

$$C_t = \gamma(\frac{x_t}{S_t}) + (1-\gamma)C_{t-1}$$

As our equation suggests, the higher γ is the more weight we place on seasonal patterns, where previous seasonal patterns matter less as the second term gets smaller and smaller. On the other hand, the smaller γ is the less weight we place on seasonality. A γ of 0.6 means that our model places relatively higher weight on seasonal patterns, making it more sensitive to changes in seasonality.

Put together, we get this nasty-looking equation:

$$S_t = \alpha(\frac{x_t}{C_{t-L}}) + (1-\alpha)(S_{t-1} + Tt - 1)$$

Another thing that we see in our table is that that our additive model performed slightly better than the multiplicative model with slightly smaller RMSE. I'm inclined to run with additive seasonals, but just for kicks, let's test to see how they both look on graphs compared to our original data.

```
hw.add <- HoltWinters(temps.ts, seasonal='additive')
hw.mult <- HoltWinters(temps.ts, seasonal='multiplicative')
```

What the Holt-Winters' model does is it uses our data from the year 1996 as its input data to predict the temperatures of the subsequent years. This is why our temps.ts data differs in the number of rows (2460) from `hw.add` and `hw.mult`, which contains 2337, missing 123 rows from 1996.

I want to create a chart that fits the models to our original data. To do that, I'm going to prepare the data and put it into a data frame.

```
## we subset our temps.ts vector starting with 1997
temps_subset <- window(temps.ts, start = c(1997, 1))
## we grab the fitted values from the HW additive method
hw.add_fitted <- hw.add$fitted[, "xhat"]
## then we grab the fitted values from the HW multiplicative method
hw.mult_fitted <- hw.mult$fitted[, "xhat"]
dates <- index(temps_subset)
## creating a data frame for plotting
```

```r
plot_data <- data.frame(
  Date = dates,
  Actual = as.vector(temps_subset),
  Additive_Forecast = as.vector(hw.add_fitted),
  Multiplicative_Forecast = as.vector(hw.mult_fitted)
)
kable(head(plot_data),
      col.names = c('Date', 'Actual Temperature',
                    'Additive Forecast',
                    'Multiplicative Forecast'),
      align = 'c',
      caption = 'Comparing Models with Original Data'
)
```

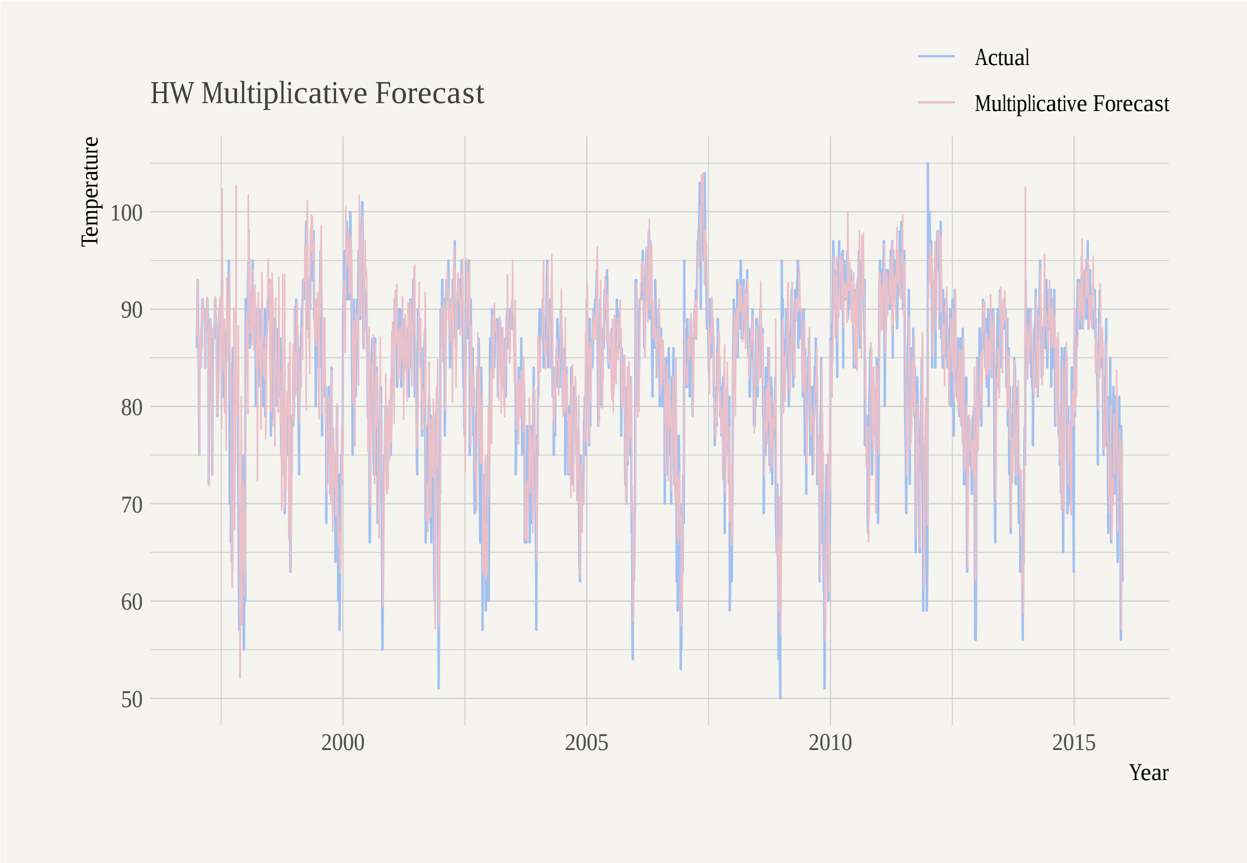
Table 2: Comparing Models with Original Data

Date	Actual Temperature	Additive Forecast	Multiplicative Forecast
1997.000	86	87.17619	87.23653
1997.008	90	90.32925	90.42182
1997.016	93	92.96089	92.99734
1997.024	91	90.93360	90.94030
1997.033	84	83.99752	83.99917
1997.041	84	84.04358	84.04496

Now that we've created our data frame, let's compare them to the original data. I'll plot the multiplicative model first, then the additive.

Multiplicative Model Plot

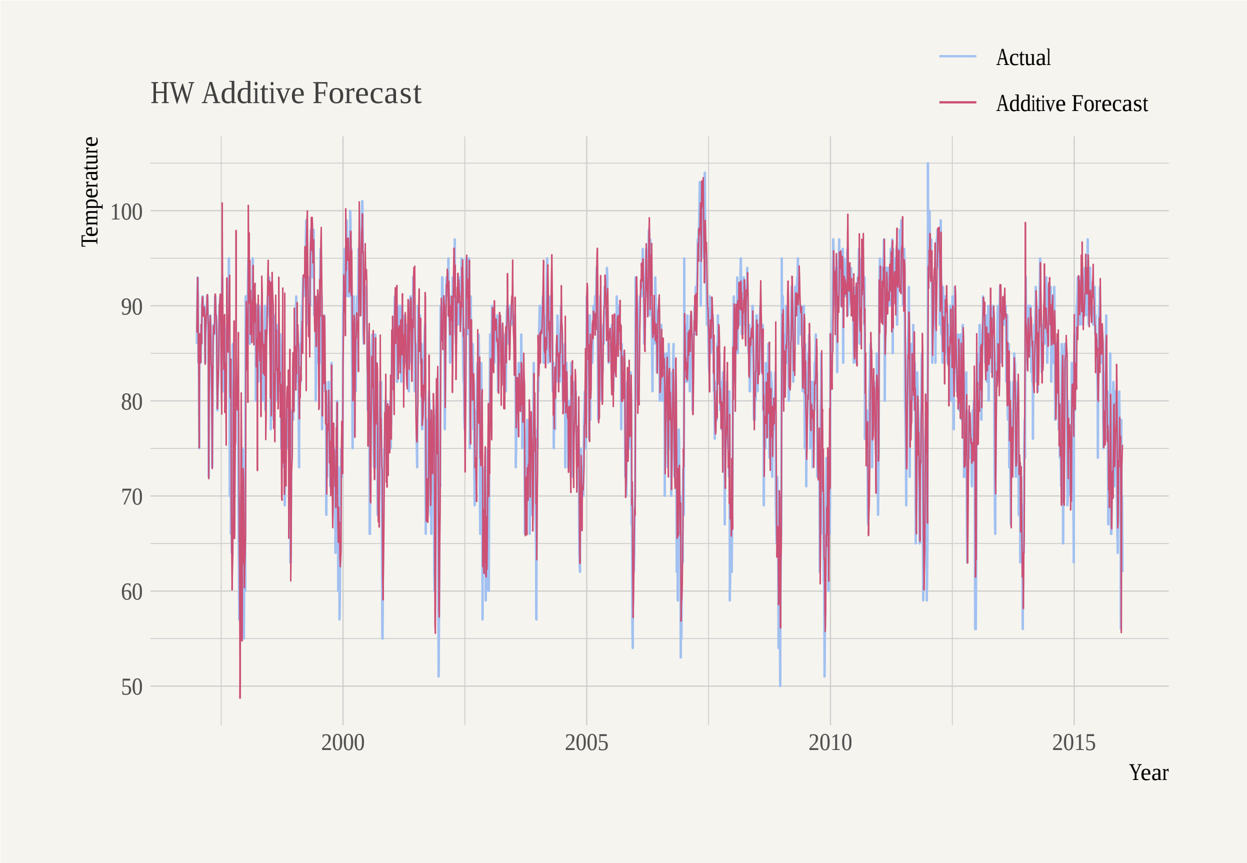
```r
## we plot the multiplicative method first
ggplot(plot_data, aes(x = Date)) +
  geom_line(aes(y = Actual, color='Actual'), linewidth = .4) +
  geom_line(aes(y = Multiplicative_Forecast, color = "Multiplicative Forecast"),
            linewidth = .25) +
  xlab("Year") +
  ylab("Temperature") +
  ggtitle("HW Multiplicative Forecast") +
  theme_ipsum() +
  theme(
    plot.title = element_text(size = 12,
                              face = "bold",
                              color = "grey25"),
    axis.text.x = element_text(size = 9),
    axis.text.y = element_text(size = 9),
    plot.background = element_rect(fill = "#F5F4EF", color = NA),
    legend.title=element_blank(),
    legend.direction = 'vertical',
    legend.pos = c(0.875, 1.11)
  ) +
  scale_color_manual(values = c("Actual" = "#A2C1F1",
                                "Multiplicative Forecast" = "#EBC0C8"))
```



It's hard to tell, but the forecast seems to fit best in more recent data points, than older data points.

Additive Model Plot

```
ggplot(plot_data, aes(x = Date)) +
  geom_line(aes(y = Actual, color='Actual'), linewidth = .4) +
  geom_line(aes(y = Additive_Forecast, color = "Additive Forecast"), linewidth = .25) +
  xlab("Year") +
  ylab("Temperature") +
  ggtitle("HW Additive Forecast") +
  theme_ipsum() +
  theme(
    plot.title = element_text(size = 12,
                              face = "bold",
                              color = "grey25"),
    axis.text.x = element_text(size = 9),
    axis.text.y = element_text(size = 9),
    plot.background = element_rect(fill = "#F5F4EF", color = NA),
    legend.title=element_blank(),
    legend.direction = 'vertical',
    legend.pos = c(0.875, 1.11)
  ) +
  scale_color_manual(values = c("Actual" = "#A2C1F1",
                                "Additive Forecast" = "#CC5175"))
```



Honestly, it looks about the same in the additive forecast, so let's get the root mean squared error of both and see how they both compare.

```
add.RMSE <- sqrt(hw.add$SSE/nrow(hw.add$fitted))
mult.RMSE <- sqrt(hw.mult$SSE/nrow(hw.mult$fitted))
cat('HW Additive RMSE:', add.RMSE, '\nHW Multiplicative RMSE:', mult.RMSE)
```

```
## HW Additive RMSE: 5.324082
## HW Multiplicative RMSE: 5.429935
```

Earlier, we chose our own α, β, γ values. Interestingly, the function produced nearly identical values. Let's see what they are:

```
add.values <- data.frame(
                  Alpha = hw.add$alpha,
                  Beta = hw.add$beta,
                  Gamma = hw.add$gamma
)
mult.values<- data.frame(
                  Alpha = hw.mult$alpha,
                  Beta = hw.mult$beta,
                  Gamma = hw.mult$gamma
)
cbnd <- rbind(add.values, mult.values)
rownames(cbnd) <- NULL
```

```r
kable(cbnd,
      align='c',
      caption='Holt-Winters\'-Produced Values')
```

Table 3: Holt-Winters'-Produced Values

Alpha	Beta	Gamma
0.6610618	0	0.6248076
0.6150030	0	0.5495256

Now that we know the function can produce very reliable values, with slightly better performance, we'll just use the `HoltWinters` function. Additionally, the additive forecast fares slightly better with a lower RMSE. What the lower RMSE of the additive forecast suggests is that the seasonal variations of the temperatures are roughly constant throughout the series.

Now that we've chosen our model and smoothed our data, the next step is to determine whether the summers get later. We do this by applying the CUSUM method to our smoothed data from 1997 to 2015.

A CUSUM Approach to the Holt-Winters' Model

We first convert our model into a data frame, so that it's prepped for CUSUM.

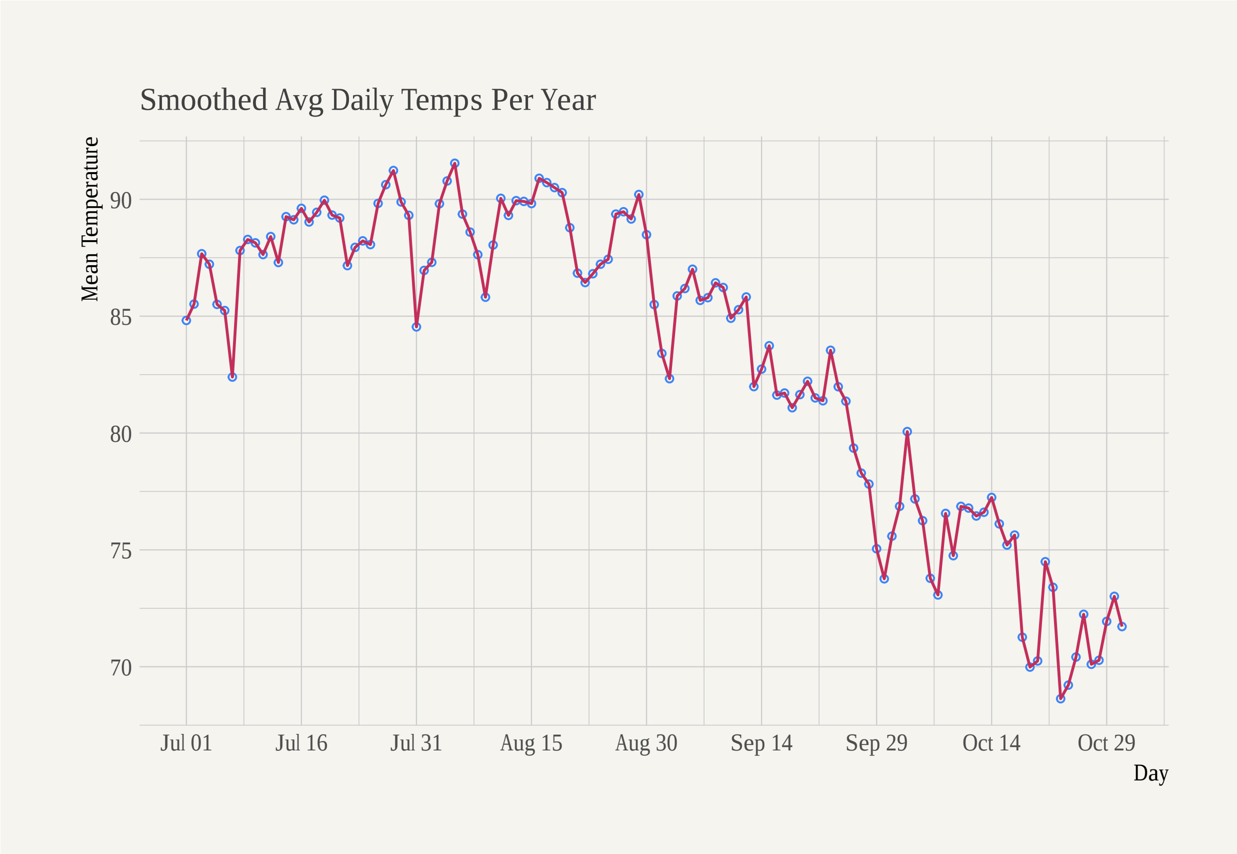
```r
## this almost looks like our temps file
temp_hw <- as.data.frame(matrix(hw.add$fitted[,'xhat'], nrow(temps)))
## assign the same funky column values
colnames(temp_hw) <- colnames(temps)[c(3:21)]
## almost there...
temp_hw <- temp_hw %>%
        mutate(DAY=temps$DAY) %>%
        relocate(DAY, .before=1)
head(temp_hw)
```

```
##     DAY     1997     1998     1999     2000     2001     2002     2003     2004
## 1 1-Jul 87.17619 65.99606 89.75466 83.65738 87.44361 79.13415 74.12006 86.89937
## 2 2-Jul 90.32925 86.63516 85.05435 86.86490 84.58717 86.97737 72.37889 84.72274
## 3 3-Jul 92.96089 90.46471 85.78686 93.25382 88.90811 90.78638 78.44689 82.61629
## 4 4-Jul 90.93360 88.77120 84.90009 91.79686 87.08965 87.47605 84.41830 83.69406
## 5 5-Jul 83.99752 83.25107 81.12482 89.31287 81.18354 86.29410 84.37192 84.18777
## 6 6-Jul 84.04358 88.40825 85.51084 91.53539 81.69877 88.16103 78.52036 87.15484
##       2005     2006     2007     2008     2009     2010     2011     2012
## 1 91.15242 79.17602 82.08626 85.64527 81.25471 87.07795 92.70889 83.33264
## 2 92.36299 88.94595 89.18548 80.16002 86.86767 81.30538 87.10638 94.13220
## 3 91.99930 92.92709 86.87669 84.99336 89.07355 82.54299 90.64593 91.82838
## 4 87.06963 93.05548 83.28236 90.20184 88.94787 83.21820 94.17789 95.83964
## 5 84.32874 90.87864 84.51007 89.66314 89.58972 81.21594 90.61737 95.47744
## 6 85.91322 86.97635 82.41612 84.39436 78.92613 85.11707 89.01132 97.60932
##       2013     2014     2015
## 1 87.07333 98.76579 89.08086
## 2 75.36736 87.73317 84.11698
## 3 81.93827 88.12055 81.57585
## 4 76.22184 87.01128 79.10413
```

```
## 5 75.44617 85.16544 83.96091
## 6 78.70689 83.29378 82.17153
```

In the following code block, I'm going to average the rows and get the daily averages through the 20ish years. Then I'm going to plot it to determine where we should take the average of the weeks *prior to any significant change*.

```r
## forgot the rows weren't numeric lol
num_rows <- sapply(temp_hw, is.numeric)
## take the average
daily_mean <- rowMeans(temp_hw[,num_rows])
## put the results in a dataframe
df <- data.frame(Day = as.Date(temp_hw$DAY, format='%d-%b'),
                 Mean_Temperature=daily_mean)
## cleaning up the column values
breaks <- df$Day[c(seq(1,length(df$Day), by=15))]
dates <- ymd(breaks)
custom_labels <- format(dates, format='%B %d')
## then plot!
df %>%
        ggplot( aes(x=Day, y=Mean_Temperature, group=1)) +
        geom_point(shape=21, color="#3D85F7", fill="#F6F5E9", size=1) +
        geom_line(color = "#C32E5A") +
        theme_minimal(base_family = "Fira Sans Compressed") +
        ggtitle("Smoothed Avg Daily Temps Per Year") +
        labs(x = "Day", y = "Mean Temperature") +
        theme_ipsum() +
        theme(
                plot.title = element_text(
                        size = 12,
                        face = "bold",
                        vjust = 0,
                        color = "grey25"
    ),
                axis.text.x = element_text(size = 9),
                axis.text.y = element_text(size = 9),
                plot.background = element_rect(fill = "#F5F4EF", color = NA)
        ) +
        scale_x_date(
                breaks = breaks,
                labels = custom_labels,
                date_labels = "%b %d"
  )
```



We see a sizable shift downward around August 30. Then it progressively gets cooler. I'm going to set our pre-shift temperature between July 1 to August 30.

CUSUM on 1997

Now, the question asks whether the summers get later and later. We're going to be using 1997 as our baseline. We will perform CUSUM on 1997 first and then do it for the rest of the years.

```
## taking the average of the temperatures from Jul 1 to Aug 30
mu.1997 <- mean(temp_hw$'1997'[1:61])
## then the standard deviation
sd.1997 <- sd(temp_hw$'1997'[1:61])
## i'm setting our threshold to 5*sigma
t.1997 <- 5 * sd.1997
summary <- as.data.frame(t(matrix(summary(temp_hw$'2010'[1:61]))))
kable(summary,
      col.names = c('Min.', '1st Qu.', 'Median', 'Mean', '3rd Qu.', 'Max'),
      align='c',
      caption='Temperatures Stats in 1997 Between Jul 1 to Aug 30'
)
```

Table 4: Temperatures Stats in 1997 Between Jul 1 to Aug 30

Min.	1st Qu.	Median	Mean	3rd Qu.	Max
81.21594	89.31165	91.67993	91.04611	93.15245	99.63876

Let's build a CUSUM function to produce our results.

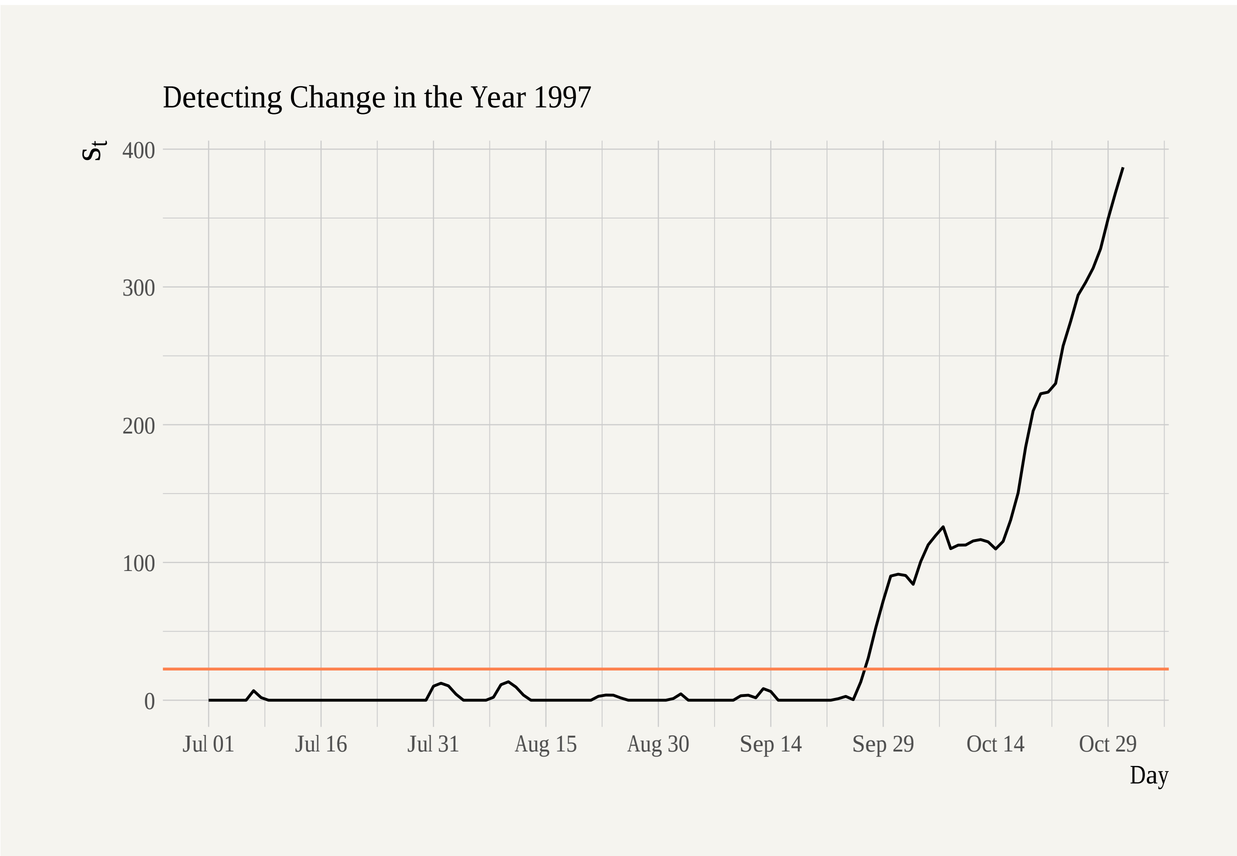
```
cusum_metric <- function(x, sigma, mu, c_val, temp, opposite = F){
        C <- c_val * sigma
        x$Year <- temp
        x$s_t <- 0
        ## if opposite is FALSE, then this runs the lower bound
        if(!opposite){
                for(i in 2:nrow(x)){
                        x[i, 's_t'] <- max(0,
                                           (x$s_t[i-1]
                                            + mu
                                            - x[i, temp]
                                            - C))
                }
                return(x)
        } else{ ## if opposite is TRUE, then this runs the upper bound
                for(i in 2:nrow(x)){
                        x[i, 's_t'] <- max(0,
                                           (x$s_t[i-1]
                                            + x[i, temp]
                                            - mu
                                            - C))
                }
                return(x)
        }
}
```

Then we apply the function to the year 1997. I initially ran it with C set to σ, but it flagged a date in early July, which is like peak summer. Atlantans would get mad at me. So I dampened the sensitivity of C, decreasing the number of false alarms.

```
## set c to 1*sigma to dampen sensitivity
cusum.1997 <- cusum_metric(temp_hw[,1:2], sigma=sd.1997,
                           mu=mu.1997, c_val=1,
                           temp='1997', opposite=F)
cusum.1997$DAY <- as.Date(cusum.1997$DAY, format = "%d-%b")
```

```
cusum.1997 %>%
        ggplot( aes(x=DAY, y=s_t)) +
        geom_line() +
        geom_hline(yintercept = t.1997, color='#FD814E')+
        scale_x_date(
                breaks = breaks,
                labels = custom_labels,
                date_labels = "%b %d"
        ) +
```

```
        theme_minimal() +
        ggtitle("Detecting Change in the Year 1997") +
        labs(x = "Day", y = expression(s[t])) +
        theme_ipsum() +
        theme(
                plot.title = element_text(size=12),
                axis.title.y = element_text(size=14),
                axis.title.x = element_text(size=10),
                axis.text.x = element_text(size = 9),
                axis.text.y = element_text(size = 9),
                plot.background = element_rect(fill = "#F5F4EF", color = NA)
        )
```




```
cusum.1997$DAY <- format(cusum.1997$DAY, format='%b %d')
```

If I had set a lower C, the peaks in July would have triggered a change. With s_t no longer triggering false alarms, we see that CUSUM detects its first significant change at the end of September.

```
kable(cusum.1997[cusum.1997$s_t>=t.1997,][1,],
      col.names = c('Changed Detected', 'Daily High', 'Year',
                    paste('$s_t$', collapse='')),
      align='c',
      caption=sprintf("Significant Temperatarure Detection in 1997 ($T$ is %s)",
                      round(t.1997,2)
```

```
    )
)
```

Table 5: Significant Temperatarure Detection in 1997 (T is 22.62)

	Changed Detected	Daily High	Year	s_t
89	Sep 27	64.64756	1997	30.60108

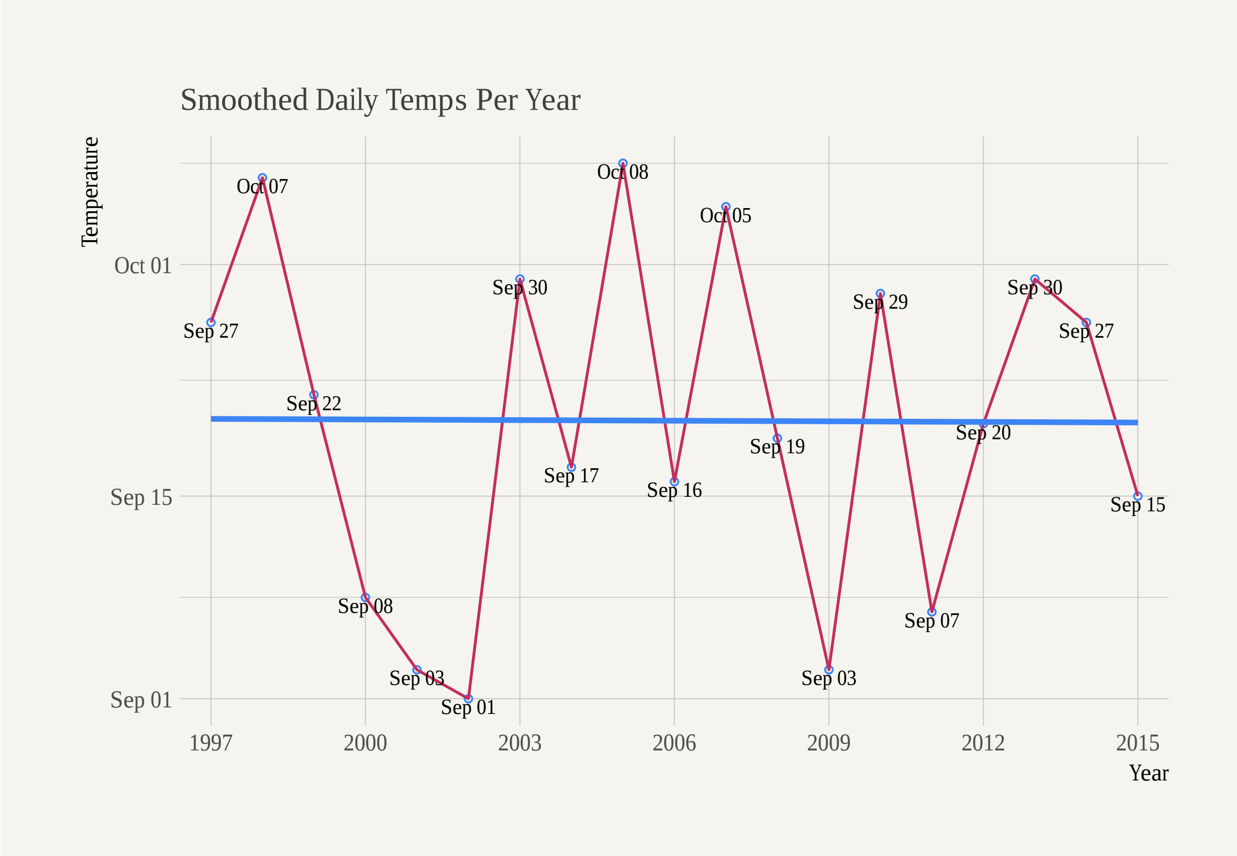
Now we repeat this process for the remaining years. I'm going to create an empty data frame to store our results. Again, this only cycles through the years 1997-2015.

CUSUM on 1997-2015

```
detection_df <- data.frame()
## this looks ugly, so i'll explain line by line
## this skips the DAY column
for(i in 2:(ncol(temp_hw))){
        ## we're going to use col_name in our function call, so we create it
        col_name <- names(temp_hw)[i]
        ## we apply col_name to our average
        mu <- mean(temp_hw[, col_name][1:61])
        ## to our sigma
        sigma <- sd(temp_hw[, col_name][1:61])
        ## threshold
        t <- 5*sigma
        ## our cusum function
        ## as you can see, i set c_val to 1.25, i'll explain why later
        cusum <- cusum_metric(temp_hw[,c('DAY',col_name)],
                              sigma=sigma,
                              mu=mu, c_val=1.25,
                              temp=col_name, opposite=F)
        ## this gets the first change detected (it's a row)
        change_detected <- cusum[which(cusum$s_t>=t),][1,]
        ## then we store it in a sub dataframe
        change_detected_df <- data.frame(Day=change_detected$DAY,
                                         Year=change_detected$Year,
                                         Temperature=change_detected[,col_name],
                                         s_t=change_detected$s_t,
                                         T_value=t)
        ## then append it to our empty data frame above
        detection_df <- bind_rows(detection_df, change_detected_df)
}
## formatting stuff
detection_df$Day <- as.Date(detection_df$Day, format = "%d-%b")
## then we plot
## this includes a regression line to see if there's any trend upward or downward
detection_df %>%
        ggplot( aes(x=Year, y=Day, group=1)) +
        geom_point(shape=21, color="#3D85F7", fill="#F6F5E9", size=1) +
        geom_line(color = "#C32E5A") +
```

```r
    theme_minimal(base_family = "Fira Sans Compressed") +
    geom_text(
            aes(label=format(Day, format='%b %d')),
            vjust=1,
            family=font_an,
            size=2.7
            ) +
    geom_smooth(method = "lm", formula = y ~ x,
                se = FALSE, color='#3D85F7') +
    ggtitle("Smoothed Daily Temps Per Year") +
    labs(x = "Year", y = "Temperature") +
    theme_ipsum() +
    theme(
            plot.title = element_text(
                    size = 12,
                    face = "bold",
                    vjust = 0,
                    color = "grey25"
  ),
            axis.text.x = element_text(size = 9),
            axis.text.y = element_text(size = 9),
            plot.background = element_rect(fill = "#F5F4EF", color = NA)
    ) +
    scale_x_discrete(
            breaks = detection_df$Year[c(seq(1,
                                            length(detection_df$Year),
                                            by = 3))],

)
```



Conclusion

After raising our C value slightly higher (1.25σ), we eliminate a false alarm on July 6 in 2010. After making the adjustment, it appears that our regression line doesn't depict any meaningful trend. It might even ever-so-slightly trend earlier rather than later.

```
detection_df$Day <- format(detection_df$Day, format='%b %d')
kable(
        detection_df,
        col.names = c('Changed Detected', 'Year','Temperature',
                    paste('$s_t$', collapse=''),paste("$T$", collapse='')),
        align='c',
        caption='Detecting Later Summers'
)
```

Table 6: Detecting Later Summers

Changed Detected	Year	Temperature	s_t	T
Sep 27	1997	64.64756	27.92920	22.61673
Oct 07	1998	70.16692	34.92750	28.42806
Sep 22	1999	76.00382	27.84704	27.80667
Sep 08	2000	69.42415	39.12205	26.61398
Sep 03	2001	75.71133	23.20604	18.25804

Changed Detected	Year	Temperature	s_t	T
Sep 01	2002	72.50946	21.67308	18.90497
Sep 30	2003	65.81322	24.91062	22.42139
Sep 17	2004	74.15518	24.01879	19.75072
Oct 08	2005	70.75830	27.93089	22.66432
Sep 16	2006	78.38509	28.55007	22.88555
Oct 05	2007	78.85634	35.45695	32.70436
Sep 19	2008	77.87367	20.80316	19.04224
Sep 03	2009	75.30371	22.35021	19.68699
Sep 29	2010	73.17106	19.81733	19.10935
Sep 07	2011	76.03326	24.36615	15.29316
Sep 20	2012	82.24143	24.09020	23.05707
Sep 30	2013	71.98010	26.29229	23.69135
Sep 27	2014	74.23674	23.82140	17.67912
Sep 15	2015	77.94594	22.22694	19.76547

We see the values a little better in this table, but our detected changes hover between early summer to early October. As a result, our CUSUM method does not assure us that summer is actually ending later.