GT Introduction to Analytics Modeling - Week 2 HW

Robert Phillips

May 24, 2017

The following contains the original questions along with answers and the R code that was used.

Question 1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

Answer Given a data set, we can use clustering to discover groups of data. This is especially useful when our data does not already have a label. For example, given survey data, we may be able to find groups of people that may share simliar interests or sentiment. When can then use this data to perform a targeted action plan.

Question 2

The iris data set contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris).

We load the data with the following snippet.

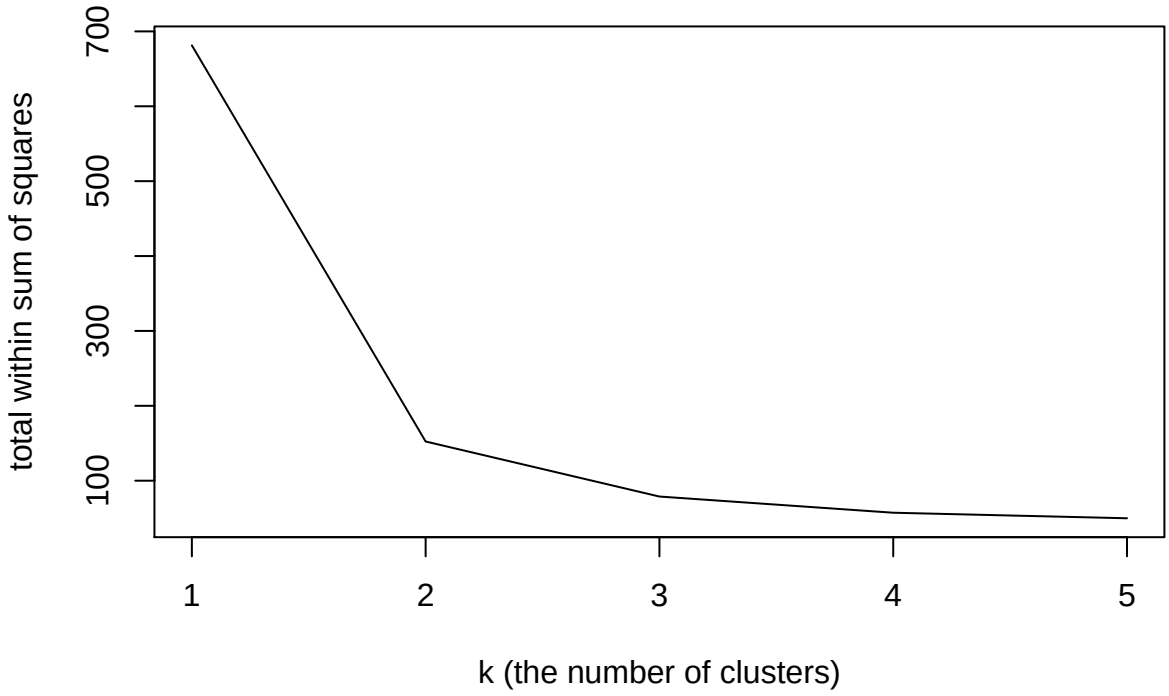
```
require(datasets)
features = iris[1:4]
target = iris[5]
```

Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

```
set.seed(1)

kmax = 5
models = list()
ss = c()

for (k in 1:kmax) {
  #store the model
  models[[k]] = kmeans(features, centers=k)
  #store the total within sum of squares
  ss = c(ss, models[[k]]$tot.withinss)
  #add the cluster center assigment to the flower types
  target[paste("k.",k, sep="")] = models[[k]]$cluster
}
```

In the previous code snippet, we create `5` models using the features “‘r colnames(features). We can now graph the total-within-sum-of-squares to visualize the potential effectiveness of the cluster number.

```
plot(x=1:kmax, y=ss, type="l", xlab="k (the number of clusters)", ylab="total within sum of squares")
```



The plot shows that a k of value 2, 3 or 4 may be suitible given the reduction in sum-of-squares value. We can use the table function to view the flower type proportions assigned to each cluster.

```
#for k = 2
table(target$Species, target$k.2)
```

```
##
##                1  2
##   setosa      50  0
##   versicolor   3 47
##   virginica    0 50
```

```
#for k = 3
table(target$Species, target$k.3)
```

```
##
##                1  2  3
##   setosa       0 50  0
##   versicolor   2  0 48
##   virginica   36  0 14
```

```
#for k = 4
table(target$Species, target$k.4)
```

```
##
##                1  2  3  4
##   setosa       0  0  0 50
##   versicolor   0 27 23  0
##   virginica   32  1 17  0
```

As expected, a value with k = 3 yields the least number of errors. The flower of type **setosa** is assigned to a single cluster. The flower of type **versicolor** is mostly assigned to different cluster, with only 2% assigned elsewhere. The flow of type **virginica** has 72% assigned to a cluster seperate from the other 2 flower types and 28% assigned to a cluster demonstrating signficant overlap with the flower of type **versicolor**.

To further improve these predictions, we could alter the feature set by removing a feature or by combining 2 or more features in some way.

Question 3

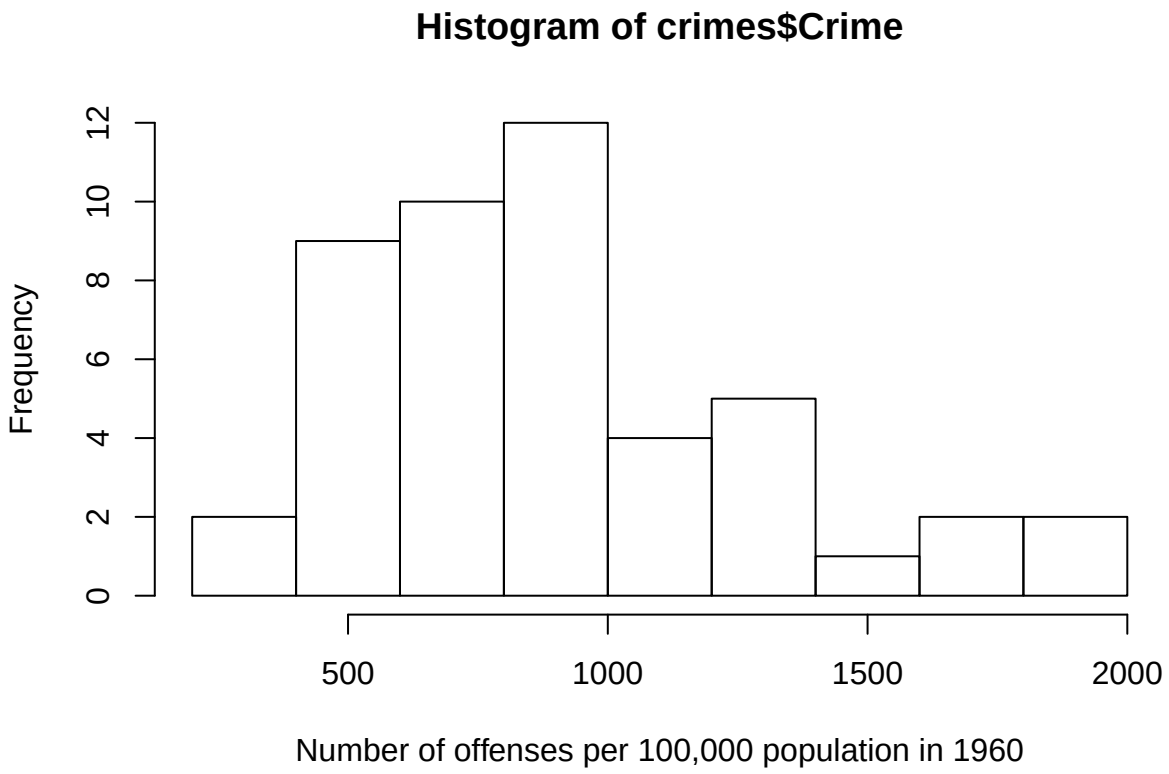
Using crime data from http://www.statsci.org/data/general/uscrime.txt (description at http://www.statsci.org/data/general/uscrime.html), test to see whether there is an outlier in the last column (number of crimes per 100,000 people). Is the lowest-crime city an outlier? Is the highest-crime city an outlier? Use the grubbs.test function in the outliers package in R.

We first load the crime data.

```
crimes = read.csv('uscrime.txt', header=T, sep='\t')
```

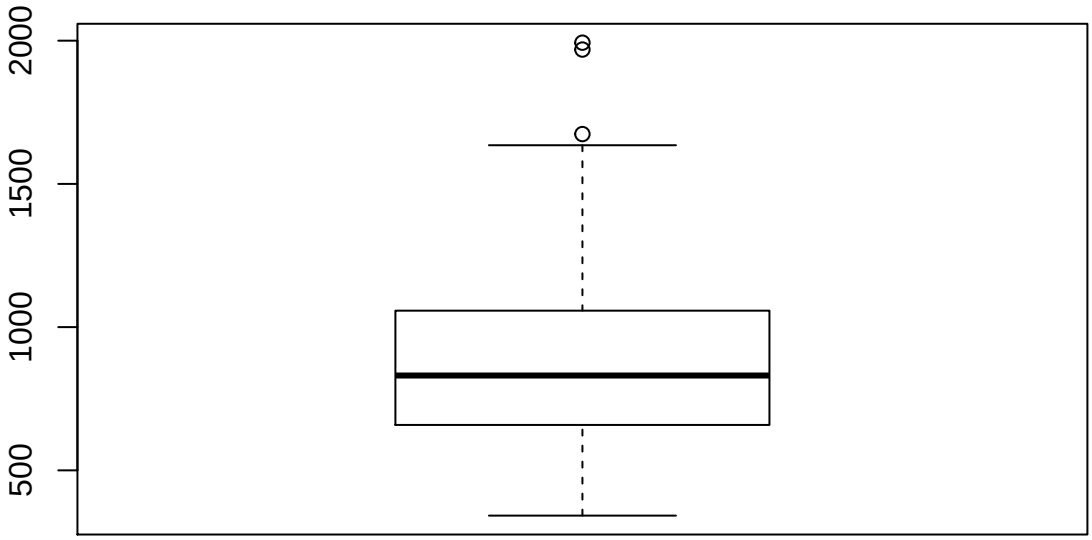
We can visual the data usig a histogram and box plot.

```
hist(crimes$Crime, xlab="Number of offenses per 100,000 population in 1960")
```



Outliers are not immediately visible on the histogram. The wikipedia page (https://en.wikipedia.org/wiki/Grubbs%27_test_for_outliers) indicates the grubbs test assumes normality of the data. The histogram shows that the data is slightly skewed. This would need futher consideration in a more detailed analysis exercise.

```
boxplot(crimes$Crime)
```



The box plot indcates there may be outliers on the high end of the range. Let's try the grubbs test as requested in the assignment. We'll use test type '11' to test the high end and low end.

```
require(outliers)

#test the high side
high = grubbs.test(crimes$Crime, opposite = F)
low = grubbs.test(crimes$Crime, opposite = T)
```

The alternative hypothesis for the high end is '`highest value 1993 is an outlier`' with p-value '`0.0788749`'. The alternative hypothesis the low end is '`lowest value 342 is an outlier`' with p-value '`1`'. Both results are above the commonly used .05 p-value, therefore we may consider the values to be within the expected range.

Question 4

Describe a situation or problem from your job, everyday life, current events, etc., for which a Change Detection model would be appropriate. Applying the CUSUM technique, how would you choose the critical value and the threshold?

My team is responsible for a nightly run of numerous financial calculations that are required to complete in a certain amount of time. We also maintain real time processing during the day that also have response time requirements. We could employ a CUMSUM algorithm to monitor these processes and proactively notify us of potential problems. The choice of a threshold would be largely driven by business requirements. We would likely allow for a larger threshold to avoid nightime calls for less urgent issues. The daytime processes would likely have a lower threshold due to the critical nature of system performance.

Question 5.1

Using July through October daily-high-temperature data for Atlanta for 1996 through 2015, use a CUSUM approach to identify when unofficial summer ends (i.e., when the weather starts cooling off) each year. That involves finding a good critical value and threshold to use across all years.

We first load the data provided with the assignment.

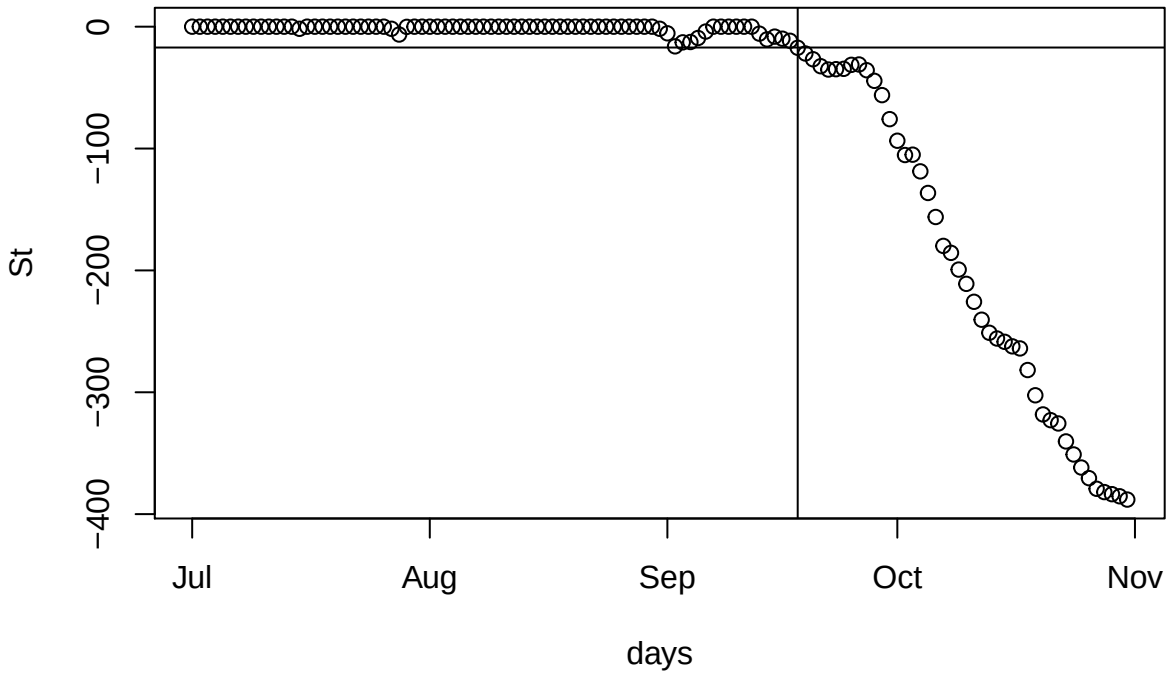
```
temps = read.csv('temps.txt', header = T, sep = '\t')
years = as.numeric(substring(colnames(temps[-1]), 2))
```

We then create a CUSUM function that can provide the values for a given set of observations, threshold and critical value.

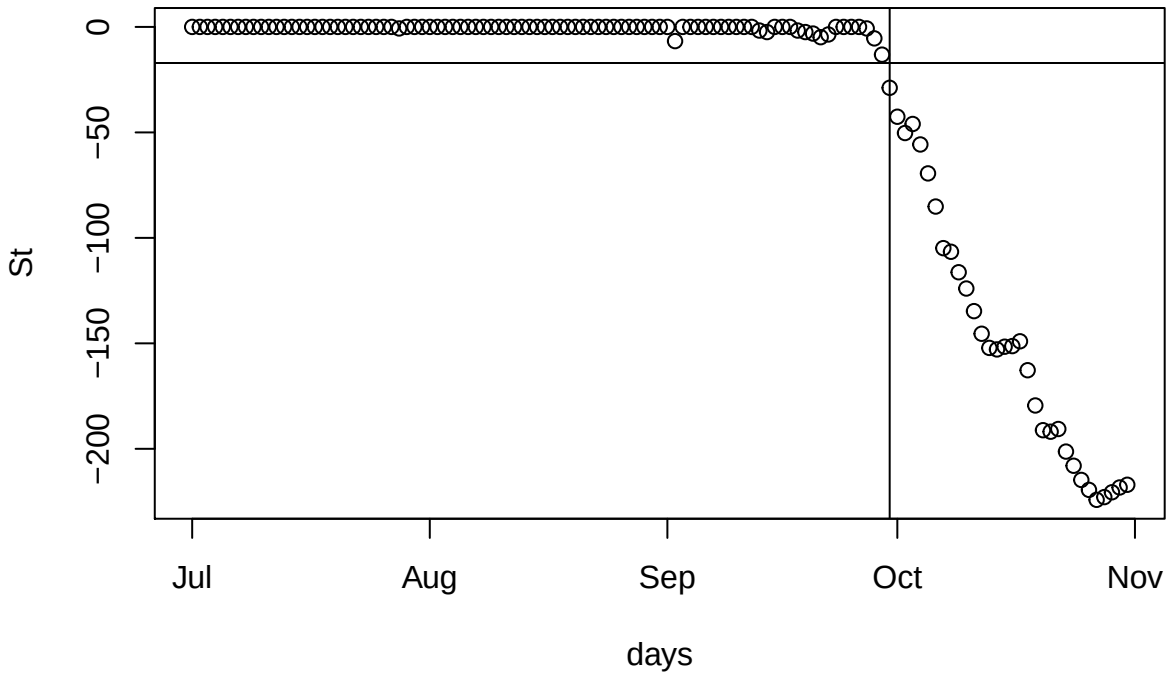
```
my.cusum1 = function(data, C, T) {
  st = 0
  sts = c()
  mu.data = mean(data)
  st.day = 0

  for(i in 1:length(data)){
    st = max(0, st + (mu.data - data[i] - C))
    sts = c(sts,st)
    if (st.day == 0 && st >= T){
      st.day = i
    }
  }
  list(sts, st.day)
}
```

We can test this function on the 1996 data and plot the results to visual how CUSUM behaves. We somewhat arbitraily choose a T that is 2 standard devations from the mean. The first choose a C of 0 to observe the default behavior, and then choose a C value intended to reduce the likelyhood of a false positive. note that we plot the negative values to visualize the behavior as a decrease.

```
t.1996 = temps$X1996
days = as.Date(paste("1996", temps$DAY, sep="-"), "%Y-%d-%b")

#choose a threshold of 2 times the std. dev.
T = 2*sd(t.1996)
C = 0

cusum1 = my.cusum1(t.1996, C, T)
plot(days, -cusum1[[1]], xlab="days", ylab="St")
abline(-T, 0)
day1 = days[cusum1[[2]]]
abline(v=day1)
```




```
C = 4

cusum2 = my.cusum1(t.1996, C, T)
plot(days, -cusum2[[1]], xlab="days", ylab="St")
abline(-T, 0)
day2 = days[cusum2[[2]]]
abline(v=day2)
```



In the first graph, the detected change is on `1996-09-18`. In the second graph, the detected change is on `1996-09-30`. As expected, this day is later since we provided a positive value for C to reduce the likelyhood of a false positive.

We can now run this function over all years to determine different values of T and C with the goal of detecting a change on about the same day for all years.

```
#updated function that only returns the day
my.cusum2 = function(data, C, T) {
  st = 0
  mu.data = mean(data)
  st.day = 0

  for(i in 1:length(data)){
    st = max(0, st + (mu.data - data[i] - C))
    if (st.day == 0 && st >= T){
      st.day = i
    }
  }
  st.day
}

detections = data.frame(row.names=colnames(temps)[-1])
cnames = c()

#compute for different values of C and T
#note that T is k*std.dev
for (C in seq(0, 4, by=.5)){
  for (k in seq(1.5, 2.5, by=.1)){
    detect = apply(temps[-1], 2, function(x){T = k*sd(x); my.cusum2(x, C, T)})
    cnames = c(cnames, paste("C.",C,".k.",k, sep=''))
    detections = cbind(detections, detect)
  }
}
colnames(detections) = cnames

#which column has minimum std. dev.
detections.min = unname(which.min(apply(detections, 2, sd)))

#what are the parameter value and selected days
params = cnames[detections.min]
days.indexes = detections[,detections.min]
```

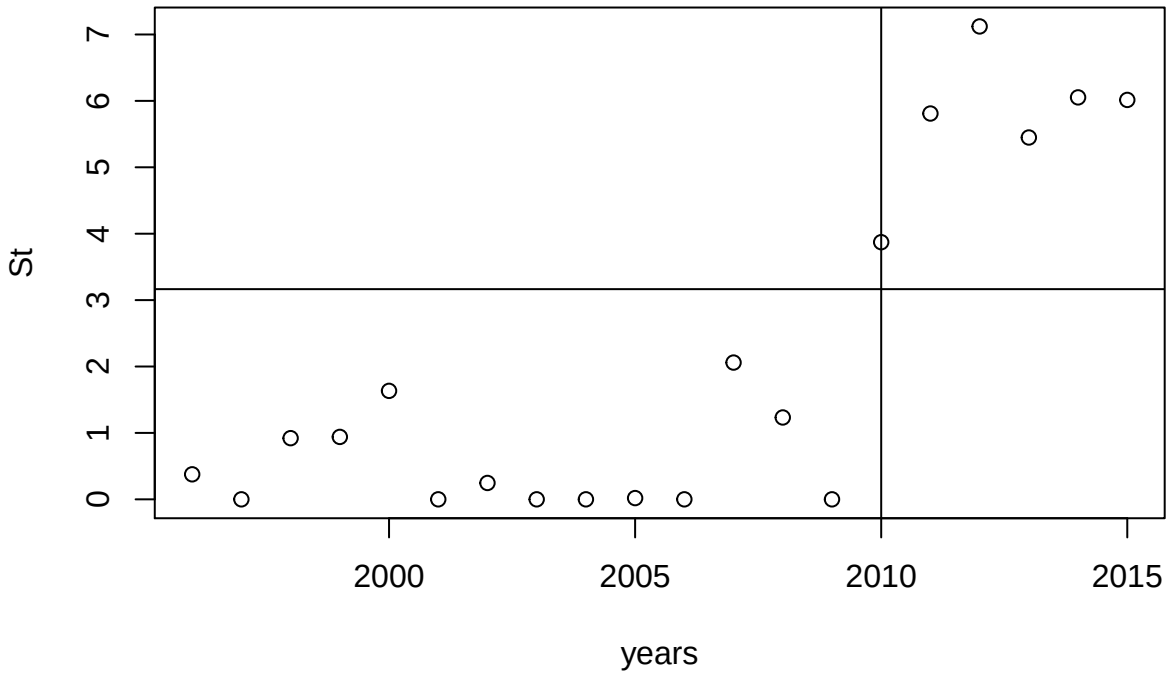
This approach yields parameters with values `C.0.k.2.5` (C = 0, k = 2.5) that result in the selection of the days `19-Sep, 25-Sep, 29-Sep, 20-Sep, 6-Sep, 25-Sep, 25-Sep, 11-Sep, 15-Sep, 7-Oct, 20-Sep, 17-Sep, 20-Sep, 19-Sep, 28-Sep, 7-Sep, 20-Sep, 16-Aug, 24-Sep, 15-Sep`. There's obviously some variation in these values, but most of the values are latter part of September. Perhaps we could further refine this by using something other than the smallest standard deviation to pick the 'best' model. However, we shouldn't expect a zero-error model since the official end of summer is simply a selected day of the year and not based data such as temperature.

Question 5.2

Use a CUSUM approach to make a judgment of whether Atlanta's summer climate has gotten warmer in that time (and if so, when).

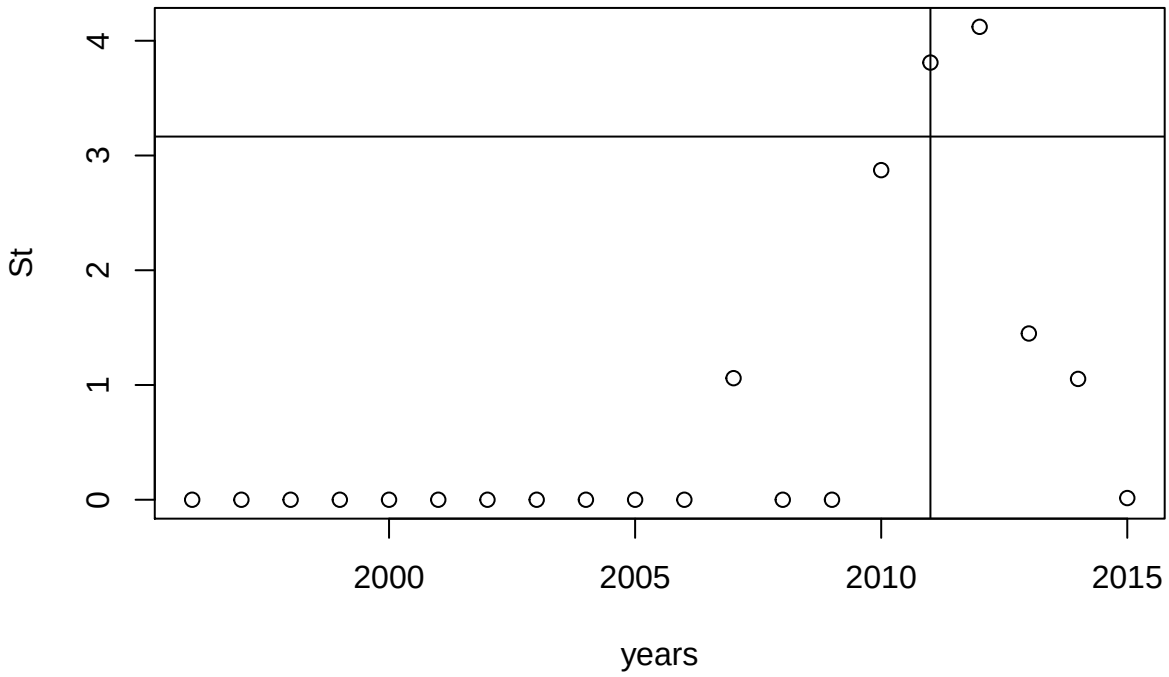
We'll assume 'warmer' means 'warmer on average.' Therefore we'll approach this question by computing the average temperature for each year and then determine if we detect an increase in average temperature.

```r
#function for increase
my.cusum3 = function(data, C, T) {
  st = 0
  sts = c()
  mu.data = mean(data)
  year = 0

  for(i in 1:length(data)){
    st = max(0, st + (data[i] - mu.data - C))
    sts = c(sts,st)
    if (year == 0 && st >= T){
      year = years[i]
    }
  }
  list(sts, year)
}

#average temp for each year
temps.avg = unname(colMeans(temps[-1]))

T = 2*sd(temps.avg)
C1 = 0

cusum1 = my.cusum3(temps.avg, C1, T)
plot(years, cusum1[[1]], xlab="years", ylab="St")
abline(T, 0)
year1 = cusum1[[2]]
abline(v=year1)
```

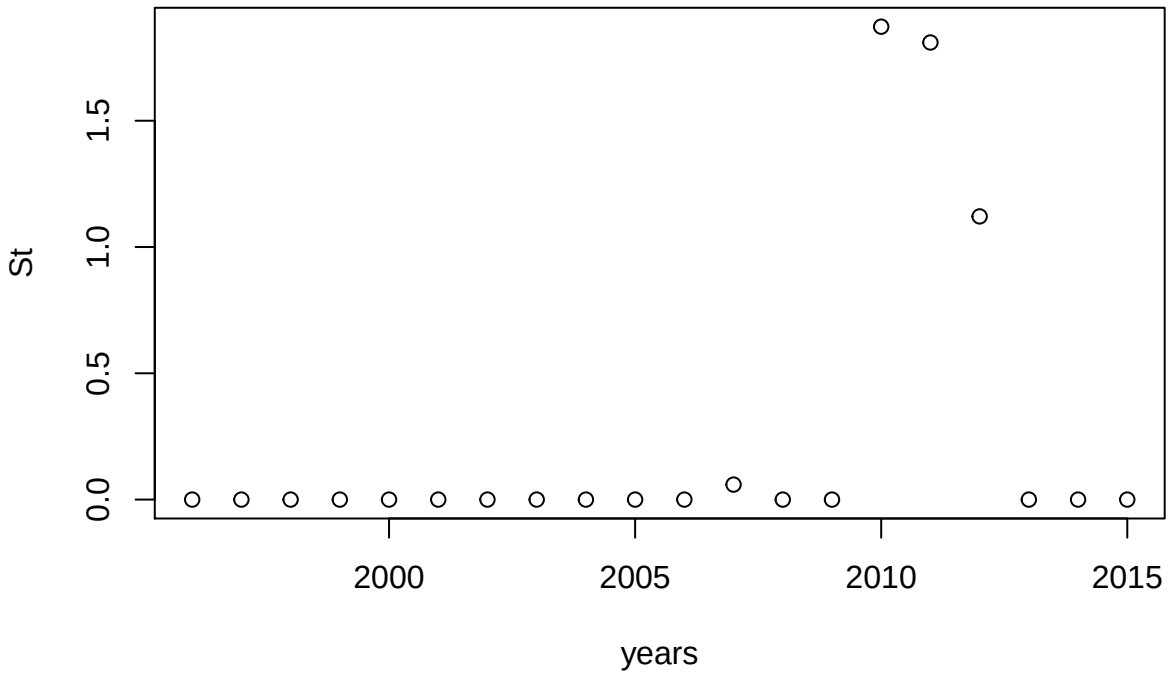



```r
C2 = 1

cusum2 = my.cusum3(temps.avg, C2, T)
```

```
plot(years, cusum2[[1]], xlab="years", ylab="St")
abline(T, 0)
year2 = cusum2[[2]]
abline(v=year2)
```




```
C3 = 2

cusum3 = my.cusum3(temps.avg, C3, T)
plot(years, cusum3[[1]], xlab="years", ylab="St")
abline(T, 0)
year3 = cusum3[[2]]
abline(v=year3)
```



All three were computed using a threshold value `3.164914`. The first chart (based on a C value of `0`) shows that we detect a change on year `2010`. The second chart (based on a C value of `1`) shows that we detect a change on year `2011`. This is slightly later as expected since we increased C. The third chart (based on a C value of `2`) does not detect a change. This may be correct since we see that the increase does not continue.

It would be prudent to test over a larger number of years. Or, given some domain knowledge, there may be better ways to quantify the temperature of these summer months which could then be input to the CUSUM algorithm.