HW3-ISYE6501x-Summer2018-OA

6/3/2018

Objectives

The focus of Week3 was Time Series and Regression models, showing how both types of models can be used for descriptive and predictive analysis. For Time Series Models, areas such as exponential smoothing (basic, trending parameters) as well as general models such as ARIMA, and GARCH were studied. For the Regression module, discussion around the linear and multi-parameter regression models were studied, and discussions around model transformations, causation vs corelation, and model quality were discussed.

Question 7.1

Describe a situation or problem from your job, everyday life, current events, etc., for which exponential smoothing would be appropriate. What data would you need? Would you expect the value of alpha (the first smoothing parameter) to be closer to 0 or 1, and why?

I (and everyone else in corporate america) is pummelled with email. The rate is never-ending; some days it spikes vs others. I would like to have a smoothing function to explain the trend, seasonality and general error in the incoming email fluctuation that comes into my inbox.

More specifically , i'm interested in some categories of email

- email from all VPs, including my boss
- email address To: me instead of Cc:
- for the sake of filtering , emails coming to some aliases that i'm subscribed to (not important but clogging my inbox nonetheless)

This would help me which day I can expect how much email from which party. What the patterns of specific VPs is when they send emails, so I can time my email communications during time pockets when they are most active (to elicit a response), who the top sender of emails are to me etc.

The data would just be my INBOX. In order to analyze it i'd need to set up an automated script to copy away this inbox every night for R&D purposes.

With respect to the smoothing parameter, alpha tends to zero when there is a lot of randomness, but tends to 1 when there is none. in my job there is definitely not a set pattern, but a little more stability vs random behaviour, so I'd imagine my alpha would be between 0.5 - 0.7

Question 7.2

Using the 20 years of daily high temperature data for Atlanta (July through October) from Question 6.2 [. . .], 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.[. . .]

The general approach I took for this model was to

- first apply the cusum model on the 20 year dataset , and get the predicted first "unofficial" day of winter for each of the years *this is pretty much repeating last week's homework*
- then, apply the exponential smoothing model on this dataset and visualize to observe if there is a trend

Step by step approach:

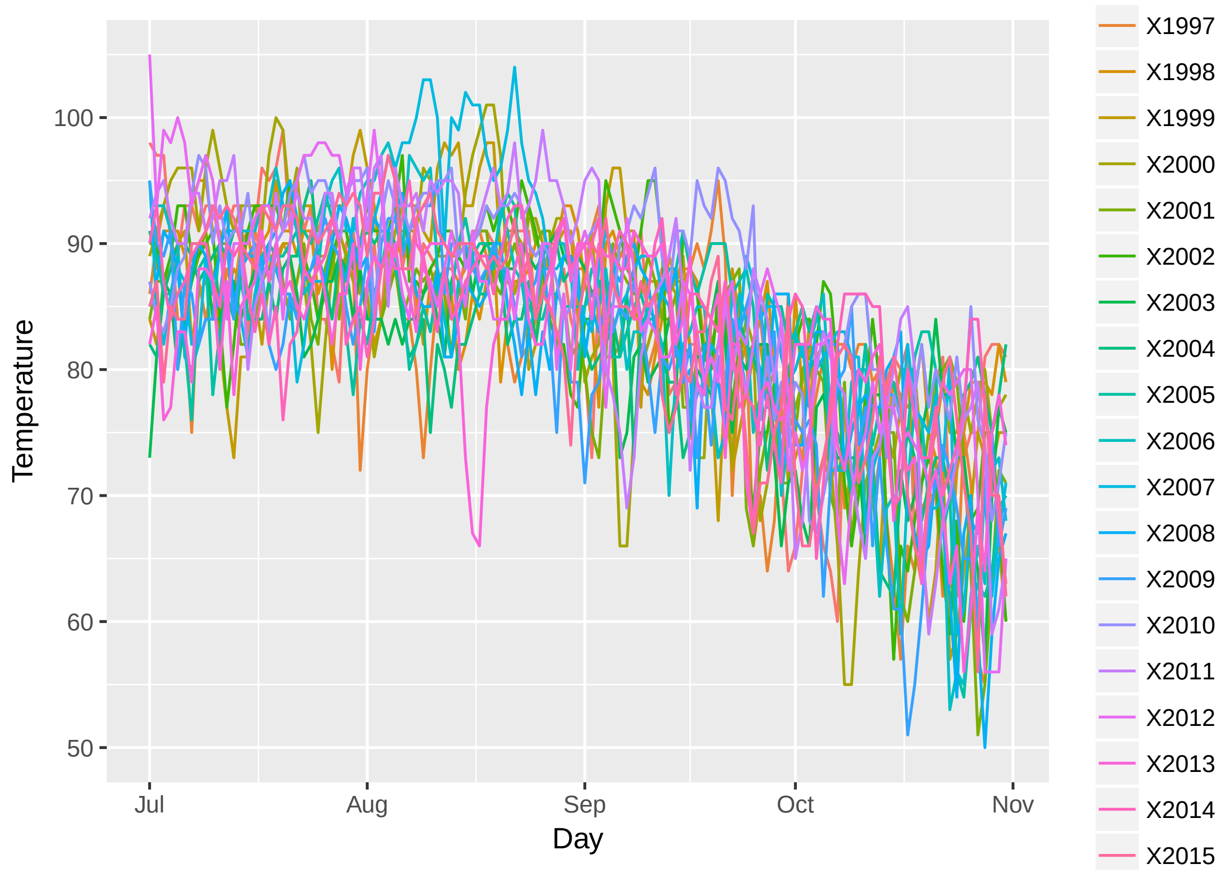
- First we load the data and clean up the date to make fully canonical

```
dataFileName <- "temps.txt"
if (!file.exists(dataFileName)) {
  url1 <- paste0(c("https://prod-edxapp.edx-cdn.org/assets/courseware/v1/592f3be3e90d2bdfe6a69f62374a12
  download.file(url1, dataFileName) }


AtlantaTemperatureData <- read.table(dataFileName, stringsAsFactors = TRUE, header = TRUE )
AtlantaTemperatureData$DAY <- as.Date(AtlantaTemperatureData$DAY, '%e-%b')
```

Exploratory Analysis:

- Prior to applying models, just trending the entire data set over the months to observe a visual pattern. Clearly there is a pattern, with some interestingly sharp outliers.

```
meltedtemps <- melt(AtlantaTemperatureData, id = 'DAY')
ggplot(meltedtemps, aes(x = DAY, y = value, colour = variable, group=variable)) +
  geom_line() +
  ylab("Temperature") +
  xlab("Day")
```



- Then we apply the CUSUM model..
- first let's create the cusum model. (I didnt use qcc::cusum because some students observed unexpected behavior while reviewing its code)
- also, let's find out the mean and standard deviation for the first 30 days of each year 1996:2016

```r
cusumLocalModel <- function(data, CCoefficient, TCoefficient){
  ans <- data.frame(S=double(), alarm=integer())
  ans[nrow(ans)+1,] <- c(0,0)
  mu <- mean(data)
  std <- sd(data)
  C <- CCoefficient * std
  thresh <- TCoefficient * std
  for (i in 2:length(data)){
    S <- max(0, ans[[i-1,1]] + (mu - data[i] - C))
    alarm <- S > thresh
    ans[nrow(ans)+1,] <- c(S, alarm)
  }
  ans
}
# map_dbl from the purr library is similar to the lapply, sapply class of functions
# but a lot easier to use! Here I use it to iterate on each of the columns (years) and create the mean
# no need to run this below anymore since cusumLocalModel calculates the AVE and SD real time for every
averageForEachYear <- map_dbl(AtlantaTemperatureData[1:30,2:21], mean)
standardDeviationForEachYear <- map_dbl(AtlantaTemperatureData[1:30,2:21], sd)
```

- threshold target T as 4 x STDEV
- and dampener C as 0.7 x STDEV
- pass T and C coefficients as well as the temperature for each of the 123 days into the cusum model

```r
CCoefficient <- 0.7
TCoefficient <- 4

firstPredictedDayOfWinter <- c()

# iterate through each year
for (cnum in 1:(ncol(AtlantaTemperatureData) - 1)){
  tempdat <- AtlantaTemperatureData[,cnum + 1]
  # pass the temperature for each of the 123 days into the cusum model
  # along with the C and T coefficient

  ans <- cusumLocalModel(tempdat, CCoefficient, TCoefficient)
  firstDayOfDropRaw <- which(ans$alarm == 1)[1]
  firstPredictedDayOfWinter <- c(firstPredictedDayOfWinter,firstDayOfDropRaw)
}
```

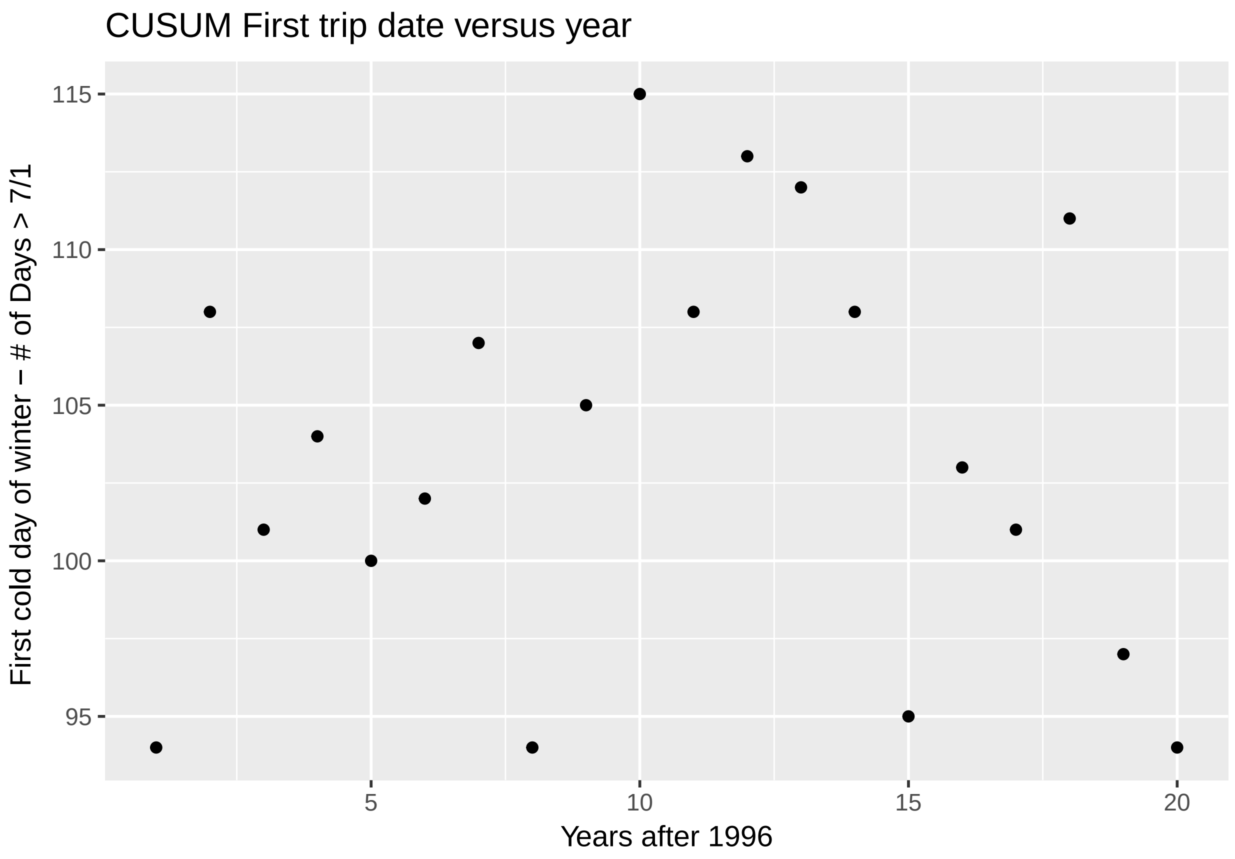
- Finding the median of the predicted first day of winter across 1996:2016:

```r
# taking the median of the first day of coldness:
MedianFirsColdtDay <- floor(median(firstPredictedDayOfWinter, na.rm = TRUE))
MeanFirstColdDay <- floor(mean(firstPredictedDayOfWinter))
# comes out as 91
veryFirstDay <- as.Date("1996-07-01") #first of july is the first day in the atlanta data set

firstPredictedDayOfWinterHumanReadable <- veryFirstDay + MeanFirstColdDay

cat(c("The first day of winter roughly and unofficially is: ",
      format(firstPredictedDayOfWinterHumanReadable, format = "%m/%d"), "\n"))
```

```
## The first day of winter roughly and unofficially is:  10/12
```

- and here is the spread of all the first unofficial days of winter

- note. 10/12 is day # 103.

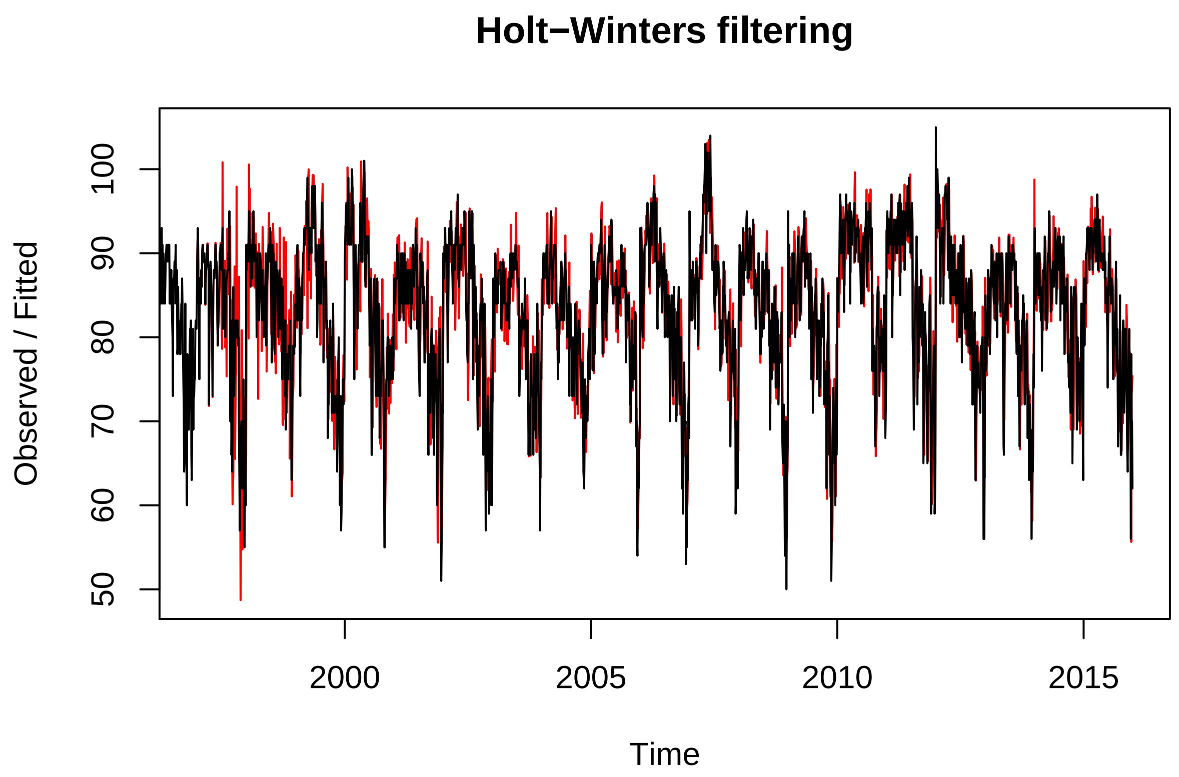


- **At this point, we have successfully used the CUSUM model to predict the first day of winter for each of the 20 years of data**
- Now we move to the real meat of this assignment: using exponential smoothing to determine if the unofficial end of summer has gotten later over the 20 years:
- To refresh our memory, exponential smoothing model of type triple uses
- S(t) = alpha * x(t) / C(t-L) + (1-alpha)(S(t-1)+T(t-1))
  - S expected value (predicted*)
  - x(t) observed value
  - C - cyclical impact (t-L means take from last observed cycle L ago)
  - T - trend impact
- alpha : for adjusting standard expected value vs observed value, low alpha for less randomness.
- beta : for adjusting for trends. low beta weights last trend higher for the new trend value
  - T(t) = beta * (S(t)-S(t-1)) + (1-beta) * T(t-1)
- gamma : for adjusting for cyclical effects. lower gamma means rely on last cycles value
  - C(t) = gamma * x(t) / S(t) + (1-gamma) * C(t-L)

So, running holt-winter model against the entire 20 year data, we see the following:

```r
unlistTempData <- as.vector(unlist(AtlantaTemperatureData[,2:ncol(AtlantaTemperatureData)]))
# convert to time series
time_series <- ts(unlistTempData, start=1996, frequency = 123)
```

```r
esModel <- HoltWinters(time_series)

plot(esModel)
```



```r
cat(c("Alpha: ", as.numeric(esModel$alpha), "\n"))
```

```
## Alpha:  0.661061754684708
```

```r
cat(c("Beta: ", as.numeric(esModel$beta), "\n"))
```

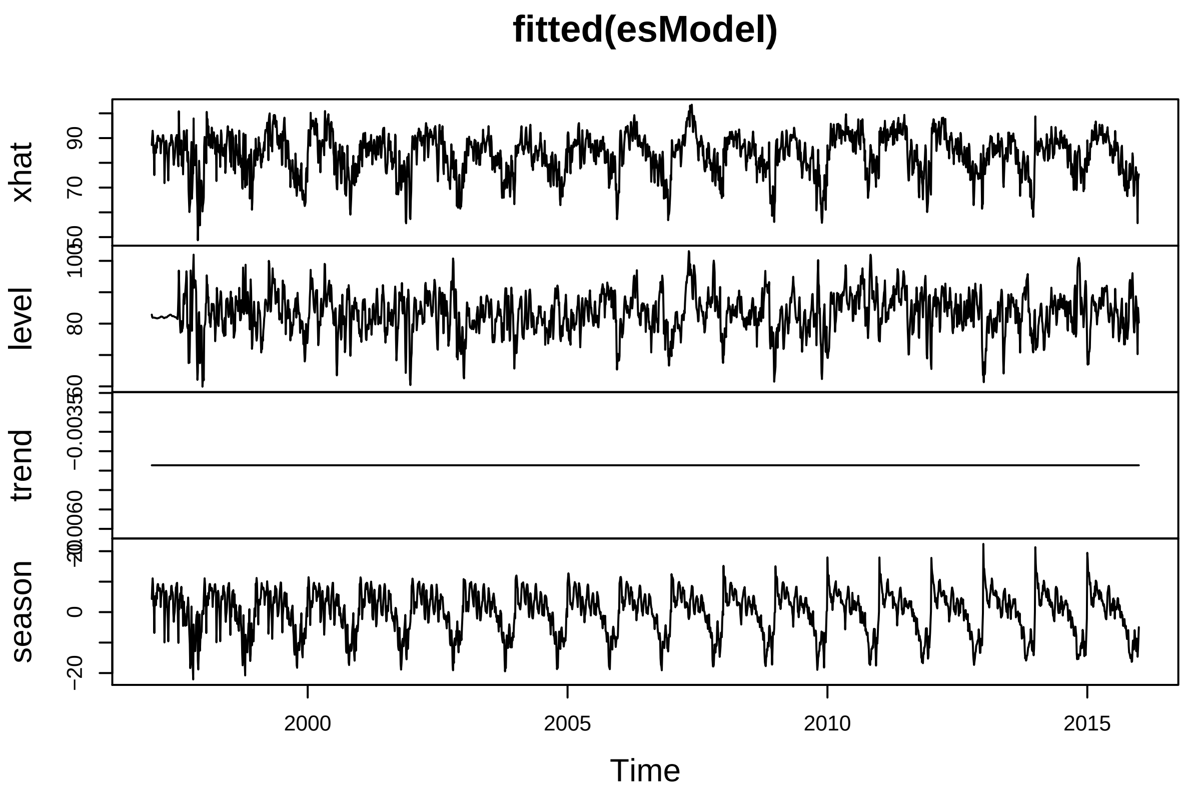
```
## Beta:  0
```

```r
cat(c("Gamma: ", as.numeric(esModel$gamma), "\n"))
```

```
## Gamma:  0.624807621487671
```

- Here's a more revealing plot of the fitted values from the model

```r
plot(fitted(esModel))
```



- Notice the season graph… the temperature is trending to be higher and higher at the start of July year over year from 1996 to 2015. Does that mean its taking longer to cool down, i.e. summers getting over later?

- From the trend graph, visually it looks like thre is no real trend here, every year pretty much mimics the prior roughly, but the amplitude is not increasing or decreasing.
- This is corroborated by beta being 0. If T(0) is set to 0 to start, having beta of 0 ensures that T(t) for all values of t remains zero, (referring to the equation above)
- The Gamma value does suggest a cyclical pattern, which is accurate.
- Let nonetheless look at the seasonal factors and run CUSUM on them to see if the unofficial summer end has gotten later over the years…
- note that the seasonalFactors is flipped, we are now looking at each of the 123 days as a column over 20 years.
- we pass each day's seasonal co-efficients for the 20 years into the cusum model, to see if there is a change detected for that co-efficient
- essentially we are asking the question, during each day (e.g. August 1st), which year did we start seeing a change in the seasonal co-efficient.
- First, let's sample the output of the fitted vector within the model:

```
head(esModel$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 see that the seasonal coefficients are in the 4th column.
- now we run the CUSUM model against the seasonal metrics
- note, I initially ran the CUSUM model with C of 0.5. Noticed that I was getting low temperature values in early July - aka spurious, however , a higher value of C, 0.6 or 0.7 started to mask out changes completely, which I knew was obviously incorrect as well, since there is definitely change happening, but was not getting detected. *Therefore I stuck with 0.5, and ignored the earlier month's spurious changes*

```
# we will use the 4th column and run it through CUSUM..
seasonalFactors <- as.data.frame(matrix(esModel$fitted[,4], ncol = 123))
didSummerComeLater <- c()

for (cnum in 1:(ncol(seasonalFactors))) {
  #seasonal <- seasonalFactors[,cnum]
  # pass the temperature for each of the 123 days into the cusum model
  # along with the C and T coefficient
  cusumOutputForTheSpecificDay <-  cusumLocalModel(seasonalFactors[,cnum], 0.5, 4)

  if (any(cusumOutputForTheSpecificDay$alarm == 1)) {
  whichYearDidTheSeasonalComponentChange <- which(grepl(1, cusumOutputForTheSpecificDay$alarm))[1] # [1
  #whatTheChangeWas <- cusumOutputForTheSpecificDay$S[]
  } else {
    whichYearDidTheSeasonalComponentChange <- 0
  }
  #ans2 <- cusumLocalModel(seasonal, CCoefficient, TCoefficient)
  #whichYearDidTheSeasonalComponentChange <- which(ans2$alarm == 1)
  #cat(c("Count: ", cnum, "\n"))
  didSummerComeLater <- c(didSummerComeLater,whichYearDidTheSeasonalComponentChange)
}
print("the following matrix shows which years each day (July through Oct) there were changes in the sea
```

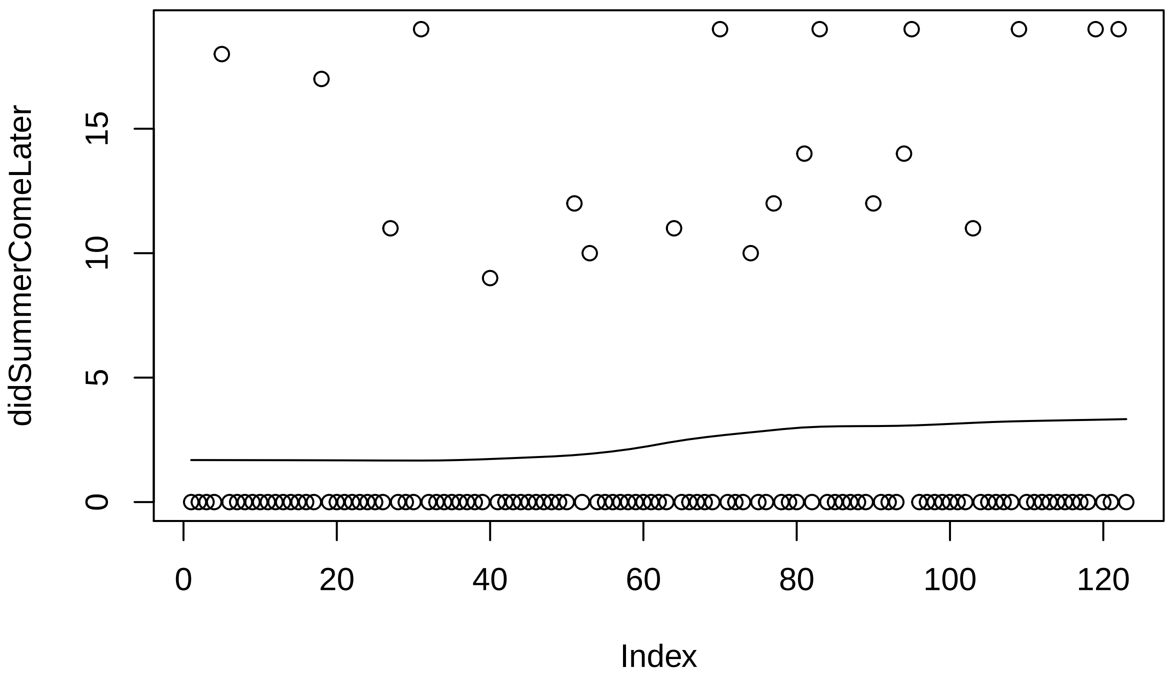
```
## [1] "the following matrix shows which years each day (July through Oct) there were changes in the sea
```

```
didSummerComeLater
```

```
##   [1]  0  0  0  0 18  0  0  0  0  0  0  0  0  0  0  0  0 17  0  0  0  0  0
##  [24]  0  0  0 11  0  0  0 19  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0
##  [47]  0  0  0  0 12  0 10  0  0  0  0  0  0  0  0  0  0 11  0  0  0  0  0
##  [70] 19  0  0  0 10  0  0 12  0  0  0 14  0 19  0  0  0  0  0  0 12  0  0
##  [93]  0 14 19  0  0  0  0  0  0  0 11  0  0  0  0  0 19  0  0  0  0  0  0
## [116]  0  0  0 19  0  0 19  0
```

- at this point, we plot this data. the graph is not pretty, but here is the legend:
- y axis is the years 1995 through 2015
- x axis are the days from 7/1 to 10/31
- the circles plotted on the graph are days when there was a change detected in the seasonal coefficients (meaning something fundamentally changed in the seasonal pattern)
- note I'm using a slightly sensitive value of C, which means its not dampening outliers as effectively. However this is on purpose, since I want to catch the subtle changes later on during the years..

```
scatter.smooth(didSummerComeLater)
```



There is a slight, almost imperceptible change as the years progress. The change in seasonal co-efficients happens later and later (but only slightly). This suggests (not proves) that summer is getting later.

Had there been a longer period of time, such as 50 years, this change may have been more definite.

Question 8.1

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

We have hundreds of disks in production, and many of them fail on a weekly basis for our storage solution. I would use a regression model to predict when a certain system would be failing in the future. I would use the following predictors:

- age of the system
- usage of that disk (capacity used)
- number of IOPS (Input/Output per sec) on that disk
- the manufacturer of the disks (Seagate vs Toshiba there is a difference)

We already have good data around which disks fail , just have to collect them in terms of these predictors to train the model.

Question 8.2

Using crime data [. . .], use regression (a useful R function is lm or glm) to predict the observed crime rate in a city with supplied data [. . .] (a) Show your model (factors used and their coefficients), (b) the software output, and (c) the quality of fit.

For this question, let's understand the data first. In 1936, a biologist named Ronald Fisher measured 50 data sets across three different species of the iris flower.

- now we calculate the value of "Crime" by multiplying the coefficients derived from the model, with the corresponding values provided in the data
- printed is the summary of the regression model, and the co-efficients

```
model <- lm (Crime ~ . , data = crimeDataTable)
summary(model)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = crimeDataTable)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

```
as.data.frame(model$coefficients)
```

```
##             model$coefficients
## (Intercept)      -5.984288e+03
## M                 8.783017e+01
## So               -3.803450e+00
## Ed                1.883243e+02
## Po1               1.928043e+02
## Po2              -1.094219e+02
## LF               -6.638261e+02
## M.F               1.740686e+01
## Pop              -7.330081e-01
## NW                4.204461e+00
## U1               -5.827103e+03
## U2                1.677997e+02
## Wealth             9.616624e-02
## Ineq               7.067210e+01
## Prob              -4.855266e+03
## Time              -3.479018e+00
```

- now we predict the value of Crime given the data and using the co-efficients:

```
# data:
M = 14.0
So = 0
Ed = 10.0
Po1 = 12.0
Po2 = 15.5
LF = 0.640
M.F = 94.0
Pop = 150
NW = 1.1
U1 = 0.120
U2 = 3.6
Wealth = 3200
Ineq = 20.1
Prob = 0.04
Time = 39.0

CrimePredictorData = c(1, M, So, Ed, Po1, Po2, LF, M.F, Pop, NW, U1, U2, Wealth, Ineq, Prob, Time) # pr
CrimePredictorCoefficients = array(data = model$coefficients) # need to fix this
PredictedValue = crossprod(CrimePredictorCoefficients, CrimePredictorData)

#this answer ,155, is confirmed by the predict function too:
# predict(model, data.frame(CrimePredictorData))

cat(c("The crime rate (number of offenses per 100,000 population): ",
      as.numeric(round(PredictedValue)), "\n"))
```

```
## The crime rate (number of offenses per 100,000 population):  155
```

- Finally, we show quality of fit for this model.
- In short, we will just look at the r.squared value of this model to see how much of the variability this model accounts for the data:

```
cat(c("This model explains ",
      round(summary(model)$r.squared, 4), " of the variability in the data\n"))
```

```
## This model explains  0.8031  of the variability in the data
```

- with an 80.31% of coverage, we can see this model has a high level of quality of fit
- though it may be because of the excessive predictors.