Using 20 years of daily high temperature data for Atlanta (July through October), building an exponential smoothing model

```
rm(list=ls())
set.seed(1)
```

```
#import data
temp <-read.table("temps.txt", stringsAsFactors = FALSE, header = TRUE)
head(temp)
```

```
##     DAY X1996 X1997 X1998 X1999 X2000 X2001 X2002 X2003 X2004 X2005 X2006 X2007
## 1 1-Jul    98    86    91    84    89    84    90    73    82    91    93    95
## 2 2-Jul    97    90    88    82    91    87    90    81    81    89    93    85
## 3 3-Jul    97    93    91    87    93    87    87    87    86    86    93    82
## 4 4-Jul    90    91    91    88    95    84    89    86    88    86    91    86
## 5 5-Jul    89    84    91    90    96    86    93    80    90    89    90    88
## 6 6-Jul    93    84    89    91    96    87    93    84    90    82    81    87
##   X2008 X2009 X2010 X2011 X2012 X2013 X2014 X2015
## 1    85    95    87    92   105    82    90    85
## 2    87    90    84    94    93    85    93    87
## 3    91    89    83    95    99    76    87    79
## 4    90    91    85    92    98    77    84    85
## 5    88    80    88    90   100    83    86    84
## 6    82    87    89    90    98    83    87    84
```

```
tail(temp)
```

```
##        DAY X1996 X1997 X1998 X1999 X2000 X2001 X2002 X2003 X2004 X2005 X2006
## 118 26-Oct    75    71    79    69    75    64    68    68    79    61    62
## 119 27-Oct    75    57    79    75    78    51    69    64    81    63    66
## 120 28-Oct    81    55    79    73    80    55    75    57    78    62    63
## 121 29-Oct    82    64    78    72    75    63    75    70    75    64    72
## 122 30-Oct    82    66    82    75    77    72    68    77    78    69    73
## 123 31-Oct    81    60    79    75    78    71    60    75    82    70    68
##     X2007 X2008 X2009 X2010 X2011 X2012 X2013 X2014 X2015
## 118    68    70    65    85    77    80    61    84    67
## 119    67    59    60    76    79    70    69    84    56
## 120    70    50    71    74    74    56    64    77    78
## 121    62    59    75    68    59    56    75    73    70
## 122    67    65    66    71    61    56    78    68    70
## 123    71    67    69    75    65    65    74    63    62
```

```
str(temp)
```

```
## 'data.frame':    123 obs. of  21 variables:
##  $ DAY  : chr  "1-Jul" "2-Jul" "3-Jul" "4-Jul" ...
##  $ X1996: int  98 97 97 90 89 93 93 91 93 93 ...
##  $ X1997: int  86 90 93 91 84 84 75 87 84 87 ...
##  $ X1998: int  91 88 91 91 91 89 93 95 95 91 ...
##  $ X1999: int  84 82 87 88 90 91 82 86 87 87 ...
##  $ X2000: int  89 91 93 95 96 96 96 91 96 99 ...
##  $ X2001: int  84 87 87 84 86 87 87 89 91 87 ...
```

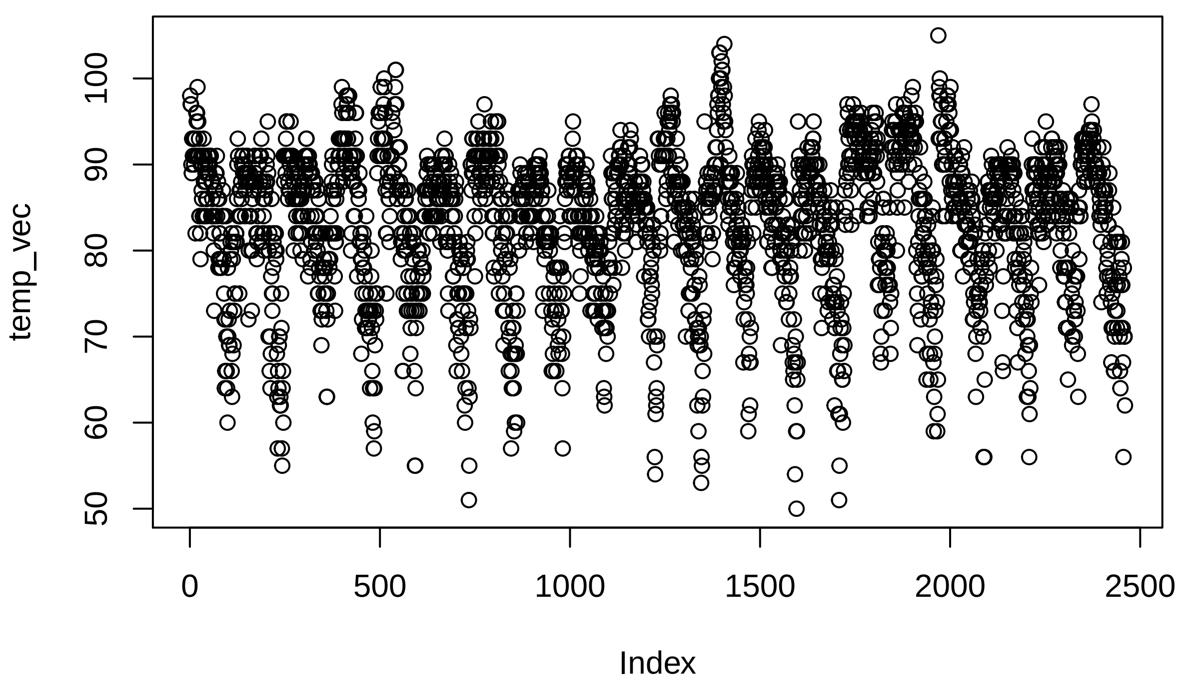
```
##  $ X2002: int  90 90 87 89 93 93 89 89 90 91 ...
##  $ X2003: int  73 81 87 86 80 84 87 90 89 84 ...
##  $ X2004: int  82 81 86 88 90 90 89 87 88 89 ...
##  $ X2005: int  91 89 86 86 89 82 76 88 89 78 ...
##  $ X2006: int  93 93 93 91 90 81 80 82 84 84 ...
##  $ X2007: int  95 85 82 86 88 87 82 82 89 86 ...
##  $ X2008: int  85 87 91 90 88 82 88 90 89 87 ...
##  $ X2009: int  95 90 89 91 80 87 86 82 84 84 ...
##  $ X2010: int  87 84 83 85 88 89 94 97 96 90 ...
##  $ X2011: int  92 94 95 92 90 90 94 94 91 92 ...
##  $ X2012: int  105 93 99 98 100 98 93 95 97 95 ...
##  $ X2013: int  82 85 76 77 83 83 79 88 88 87 ...
##  $ X2014: int  90 93 87 84 86 87 89 90 90 87 ...
##  $ X2015: int  85 87 79 85 84 84 90 90 91 93 ...
```

```
The temperature dataset contains 21 columns from year 1996 to 2015 and 123 rows with daily temperature
```

Create data into time series

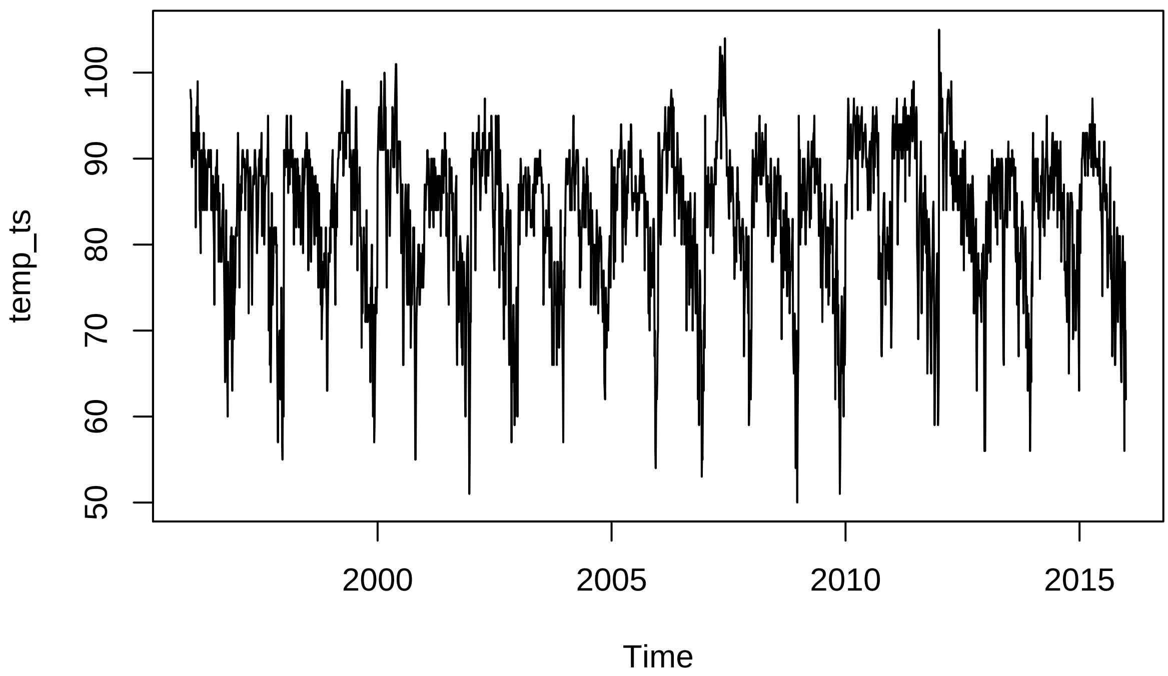
1. First changing from dataframe/matrix set up to one long list of vectors using unlist.

```r
temp_vec <- as.vector(unlist(temp[,2:21]))
#temp_vec
plot(temp_vec)
```



2. Changing the long vectors just created to time series by starting at 1996 and repeating it every 123 observation until 2015

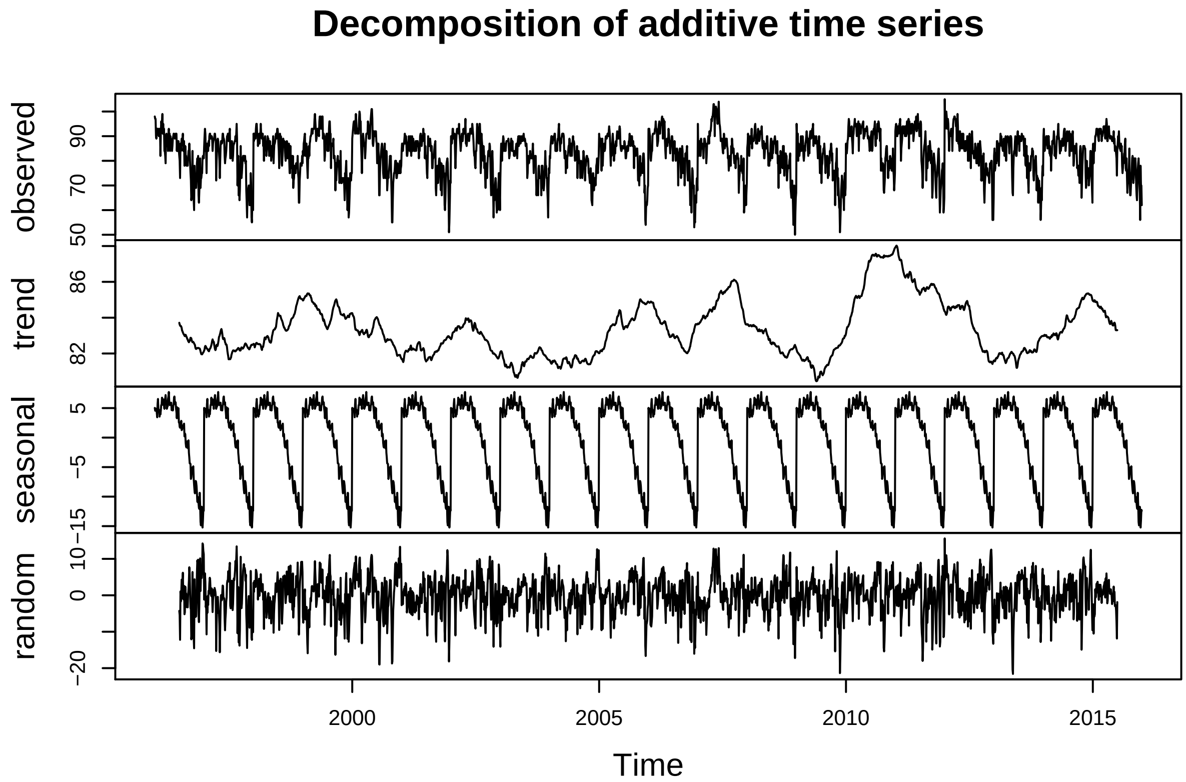
```r
#?ts

temp_ts <- ts(temp_vec, start=1996, frequency=123)
plot(temp_ts)
```




```
Plotting the timeseries data, we realize that temp is fairy consistent in the range 70-90 with avg arour
```

Now, viewing the components of the time series data separately to get a better understanding using 'decompose'

```r
#?decompose
plot(decompose(temp_ts))
```



Observed data lies on the top, so using moving average, the graph gives more information on trend, seas

Exponential smoothing will smooth out the time-series data allowing to put greater weight on recent obs

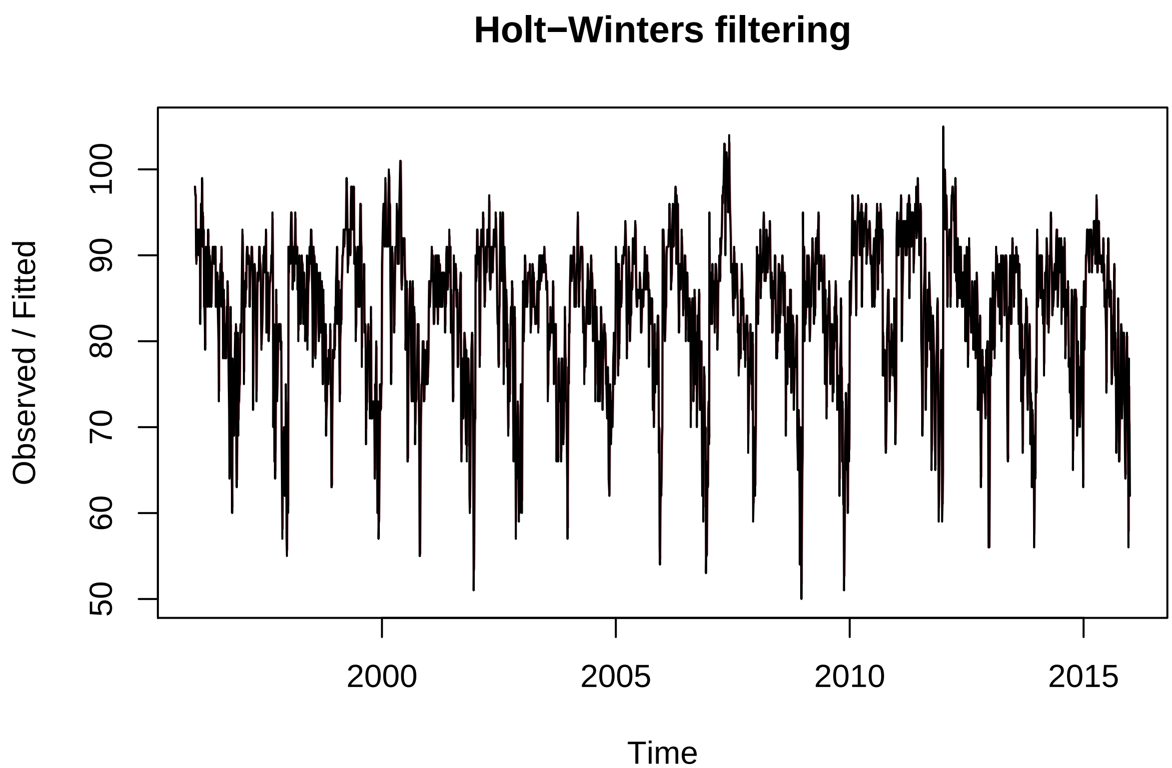
Breakdown into exponential smoothing - single (without any trend or seasonality), double and triple smoothing.

```
#1 single exponential smoothing WITH NO TREND nOR seasonality
#not including trend (beta) and seasonality (gamma) i.e., FALSE
#Leaving the alpha (level) and beta (trend) parameters as null allows the function to identify optimal p

single_expo <- HoltWinters(temp_ts,beta=FALSE,gamma=FALSE)
single_expo
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = temp_ts, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.8388021
##  beta : FALSE
##  gamma: FALSE
##
```

```
## Coefficients:
##       [,1]
## a 63.30952
```

```r
plot(single_expo)
```



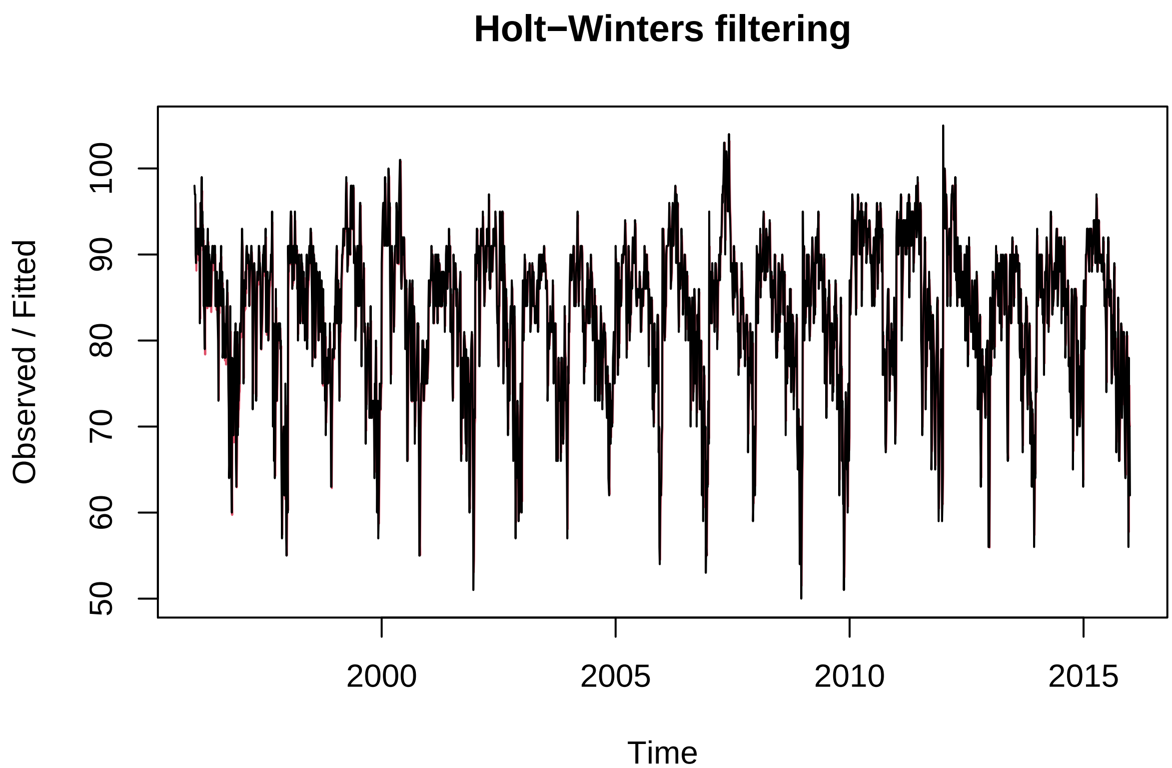
So, the baseline estimate at the end is 63.30952, and the best value of alpha found is 0.8396301. [Of course, both of those have more significant digits reported than are reasonable.]

```r
#double exponential smoothing
## Holt-Winters exponential smoothing with trend and without seasonal component.
double_expo <- HoltWinters(temp_ts,gamma=FALSE)
double_expo
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = temp_ts, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.8445729
##  beta : 0.003720884
##  gamma: FALSE
##
## Coefficients:
```

```
##          [,1]
## a 63.2530022
## b -0.0729933
```

```
plot(double_expo)
```



#beta taks about the randomness below # Notice that the final trend estimate (b) is very close to zero # (-0.004838906) and the value of beta is also very close to zero. # This suggests that there isn't really a significant trend.

```
# Triple exponential smoothing (includes both trend and seasonality)

triple_expo <- HoltWinters(temp_ts)
triple_expo
```

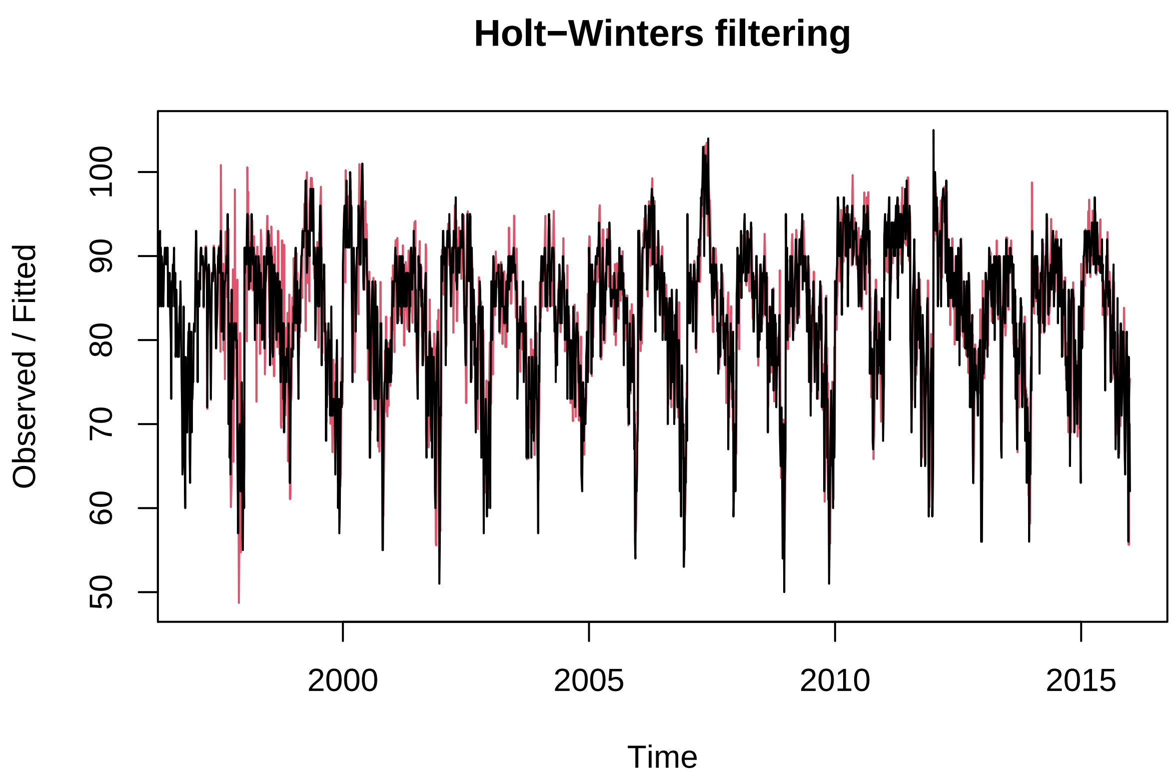
```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = temp_ts)
##
## Smoothing parameters:
##  alpha: 0.6610618
##  beta : 0
##  gamma: 0.6248076
##
## Coefficients:
```

```
##                [,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
```

```
plot(triple_expo)
```




```
#Lots of output (123 seasonal factors) but the key is that
# b and beta are again both zero or very close to it.
```

```r
# Triple exponential smoothing (multiplicative seasonality)

m3m <- HoltWinters(temp_ts,seasonal="multiplicative")
m3m
```

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

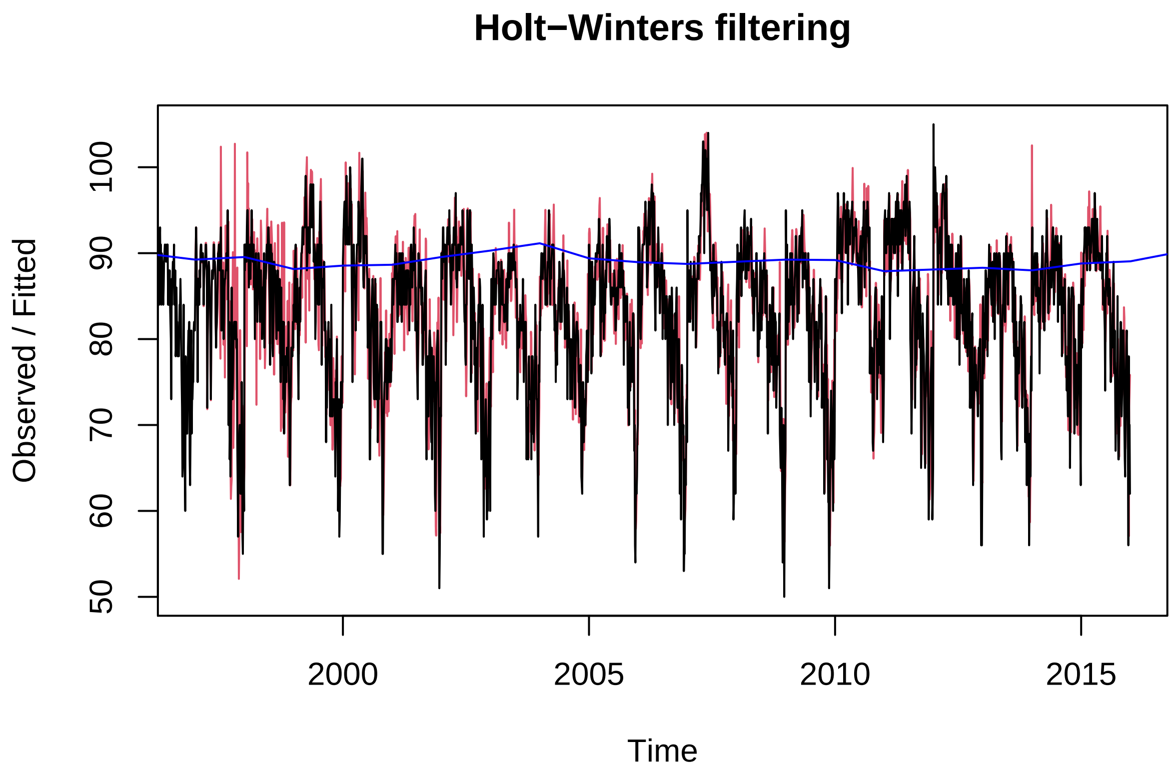
```
## s35   1.054891375
## s36   1.044587771
## s37   1.023285461
## s38   1.025836722
## s39   1.031075732
## s40   1.031419152
## s41   1.021827552
## s42   0.998177248
## s43   0.996049257
## s44   0.981570825
## s45   0.976510542
## s46   0.967977608
## s47   0.985788411
## s48   1.004748195
## 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
```

```
summary(m3m)
```

```
##              Length Class  Mode
## fitted       9348   mts    numeric
## x            2460   ts     numeric
## alpha           1   -none- numeric
## beta            1   -none- numeric
## gamma           1   -none- numeric
## coefficients  125   -none- numeric
## seasonal        1   -none- character
## SSE             1   -none- numeric
## call            3   -none- call
```

```
fulldate <- rep(temp[,1], 20)
fit <- glm(temp_ts~fulldate)
plot(m3m)
lines(fitted(fit), col="blue")
```




```r
# create fit.df to have the fitted values in a dataframe across the years

xhat <- m3m$fitted[,1] #Extracts the fitted values for the first column (level component)

fit.df <- data.frame(temp[,1]) #Creates a data frame named with the original time series data column

s <- seq(1, length(temp_ts), 123) #a sequence s starting from 1, with a step size of 123, up to the len

for (x in 1:19){
fit.df[x+1] <- xhat[s[x]:(s[x]+122)]
}

colnames(fit.df) <- c("Day", 1996:2014)
head(fit.df, 3)
```

```
##     Day     1996     1997     1998     1999     2000     2001     2002     2003
## 1 1-Jul 87.23653 65.04516 90.29613 83.39938 87.68863 78.07509 73.10059 87.27074
## 2 2-Jul 90.42182 84.87634 85.44878 86.44444 84.78855 86.02384 72.13247 85.01878
## 3 3-Jul 92.99734 89.61560 85.65942 92.85774 88.70570 90.23022 77.77739 82.68648
##         2004     2005     2006     2007     2008     2009     2010     2011
## 1 92.29714 78.50826 81.58696 84.72917 79.51855 86.74604 93.88371 82.30605
## 2 92.85614 88.18138 88.52648 80.39548 85.65722 81.47324 87.43846 92.55001
## 3 92.33884 92.43570 86.72311 84.53380 88.31357 82.29310 90.24836 91.18746
##         2012      2013     2014
## 1 84.88750 102.54643 90.07756
## 2 76.18707  89.57468 85.16854
```

```
## 3 81.46207  88.15080 82.09161
```

#Running CUSUM analysis, but on the seasonal factors.

```r
# Running CUSUM model to detect change points in the time series data stored in the fit.df data frame

my_cusum <- function(Thresh=72, C=0) {
  changed <- c() # empty list will fill with change point indices
  year <- c()
  for (col in 2:20) { # iterates over columns 2 to 20 of the fit.df data frame.
    xt <- fit.df[, col]
    mu <- mean(xt)
    s1 <- fit.df[1, col]
    for (t in 2:length(xt)) {
      st <- max(0, s1 + (xt[t] - mu - C))
      s1 <- st
      if (st <= Thresh) {
        changed <- c(changed, temp[t, 1]) #add date to list
        year <- c(year, names(fit.df[col]))
        break
      }
      else {
      }
    }
  }
result <- data.frame(year=year, day=changed) #etected change points.
result
}
cs <- my_cusum(Thresh=70, C=2)
cs
```

```
##    year    day
## 1  1996 18-Oct
## 2  1997  2-Jul
## 3  1998 16-Jul
## 4  1999 10-Oct
## 5  2000 13-Oct
## 6  2001 18-Oct
## 7  2002  2-Jul
## 8  2003  8-Oct
## 9  2004 15-Oct
## 10 2005 19-Oct
## 11 2006  6-Jul
## 12 2007 20-Oct
## 13 2008 17-Oct
## 14 2009  4-Jul
## 15 2010 14-Oct
## 16 2011 13-Oct
## 17 2012  4-Jul
## 18 2013  5-Oct
## 19 2014 11-Oct
```

Based on the output, there does not seem to be any significant change in the end of summer over the past 20 years. The end seems to fluctuate between mostly in early to mid-October with no real pattern.

Holt-Winters exponential smoothing with trend and without seasonal component.

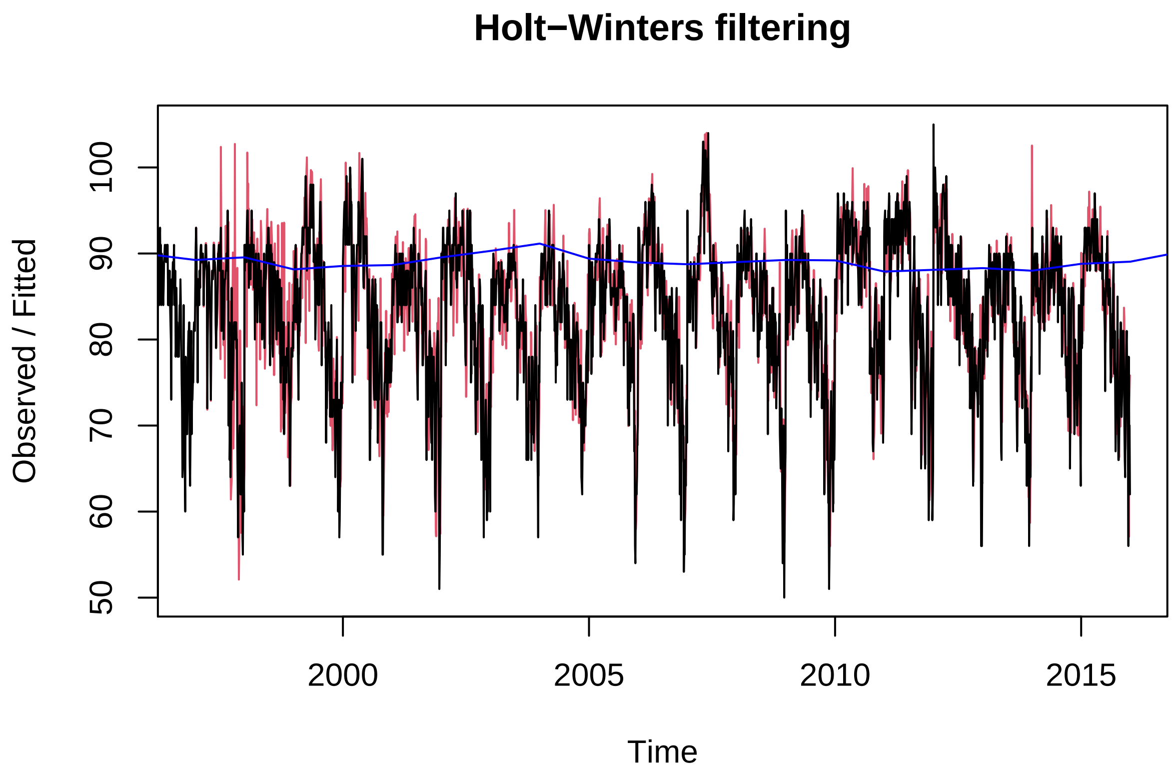
```
temp_a <- HoltWinters(temp_ts, alpha=NULL, beta=NULL, gamma=NULL, seasonal="additive")
temp_a2 <- HoltWinters(temp_ts, alpha=NULL, beta=NULL, gamma=0.5, seasonal="additive") #specifying gamm
```

#Picking the optimal values for levels, trend, and seasonaity with the seasonal parameter to be multiplicative

```
temp_m <- HoltWinters(temp_ts, alpha=NULL, beta=NULL, gamma=NULL, seasonal="multiplicative")
summary(temp_m)
```

```
##              Length Class  Mode
## fitted       9348   mts    numeric
## x            2460   ts     numeric
## alpha           1   -none- numeric
## beta            1   -none- numeric
## gamma           1   -none- numeric
## coefficients  125   -none- numeric
## seasonal        1   -none- character
## SSE             1   -none- numeric
## call            6   -none- call
```

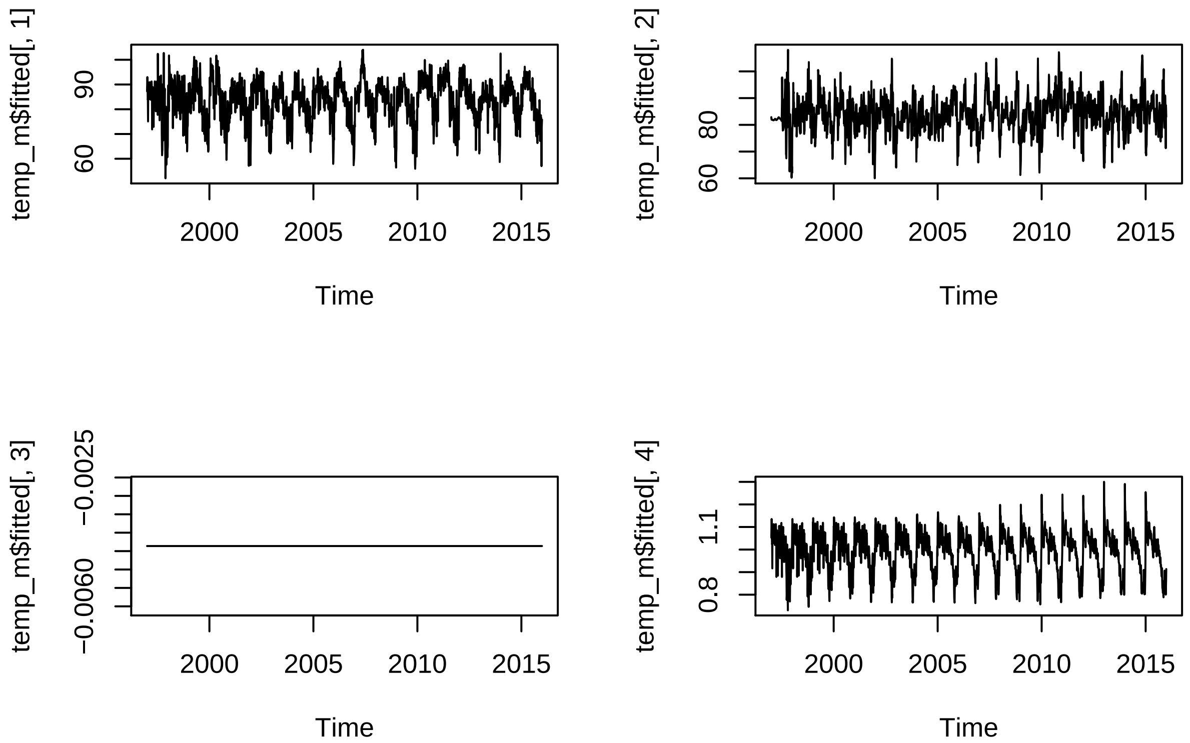
```
fulldate <- rep(temp[,1], 20)
fit <- glm(temp_ts~fulldate)
plot(temp_m)
lines(fitted(fit), col="blue")
```




```
#temp_m
```

```
Will focus on the results of the multiplicative model:
temp_m
#Smoothing parameters:
#alpha: 0.615003
#beta : 0
#gamma: 0.5495256

Smoothing parameters assign weights for historical and current data to consider. Based on the results o

The seasonality in temperatures spiking up and down each consecutive year. Adding a blue line fitted to
```

```
#temp_m$fitted
#plot(temp_m$fitted) #fitted plot. analyze component separately

par(mfrow = c(2,2))
plot(temp_m$fitted[,1]) #xhat
plot(temp_m$fitted[,2]) #level
plot(temp_m$fitted[,3]) #trend
plot(temp_m$fitted[,4]) #seasonal
```




```r
par(mfrow = c(1,1))
```

```
xhat=triple Exponential Smoothing output (which is aggregate of level, trend and seasonality). In the mo
```

```r
#temp_m$coefficients
```

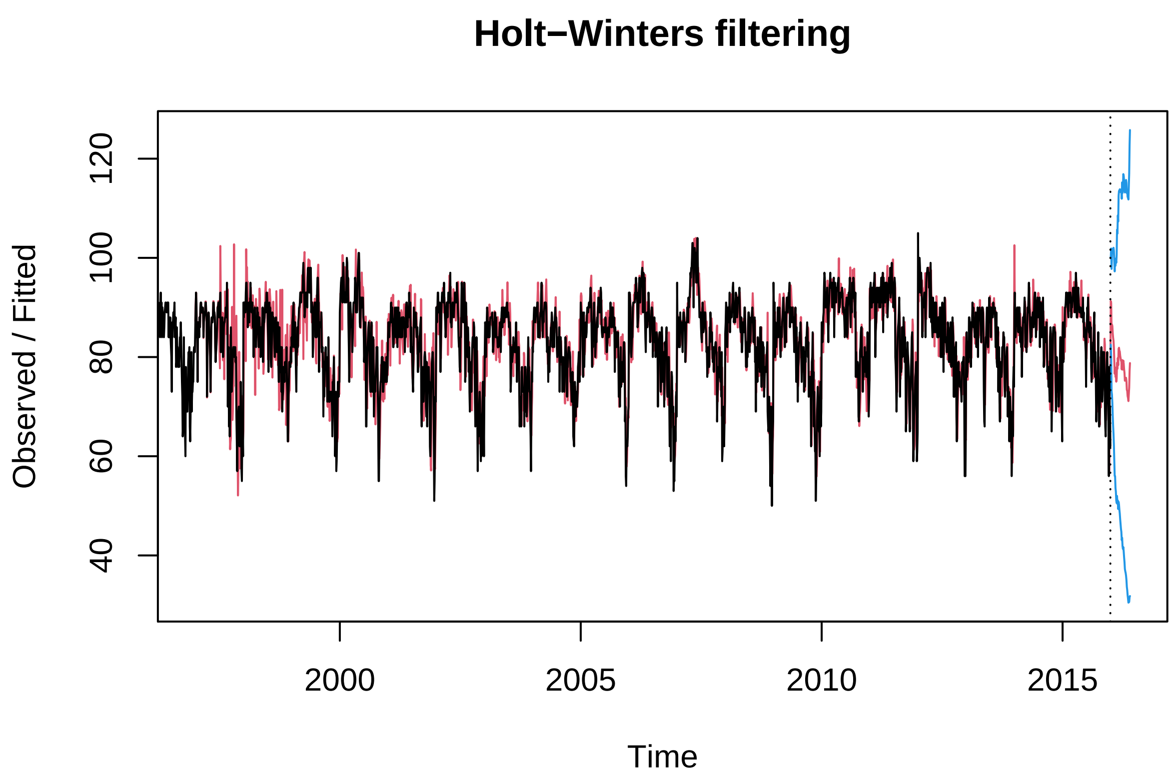
Fitted data for xhat and season

```r
#temp_m$fitted[,1]
#temp_m$fitted[,4]
c <- temp_m$fitted[,1]
predict(temp_m, h=123)
```

```
## Time Series:
## Start = c(2016, 1)
## End = c(2016, 1)
## Frequency = 123
##           fit
## [1,] 91.28516
```

```
Predicting at 91.3
```

```
#predicting on single exponential model
#predict(temp_m, 123, prediction.interval=TRUE)
#outcome_m

plot(temp_m, predict(temp_m,n.ahead=50, prediction.interval=TRUE, level=.95)) #achange thew number in a
```




```
#fitted values of the seasonal component
temp_m_sf <- matrix(temp_m$fitted[,4],nrow=123)
tail(temp_m_sf)
```

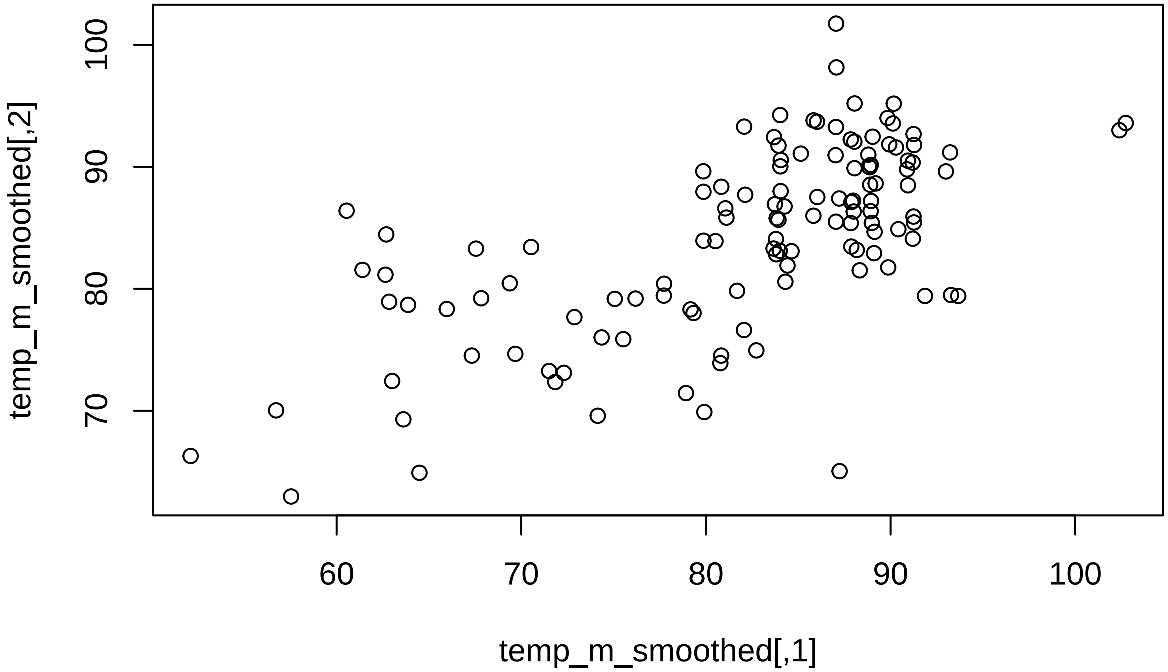
```
##              [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [118,] 0.9202878 0.9167266 0.9316249 0.9483276 0.9356822 0.8960745 0.9115292
## [119,] 0.9203791 0.8752448 0.8890989 0.9189986 0.9279178 0.8676299 0.8817807
## [120,] 0.9934182 0.9493421 0.9386713 0.9343247 0.9395149 0.9269720 0.9360466
## [121,] 1.0052839 1.0170234 0.9947869 0.9777463 0.9568681 0.9754782 0.9685817
## [122,] 1.0049851 1.0160082 1.0182545 1.0152111 1.0042247 1.0293585 0.9928965
## [123,] 0.9920411 0.9788295 0.9796735 0.9865343 0.9910570 1.0001962 0.9651261
##              [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## [118,] 0.8880224 0.8919086 0.9016390 0.9096762 0.9076538 0.9056794 0.8805521
## [119,] 0.8675880 0.8778461 0.8903499 0.9073530 0.9042270 0.8709238 0.8536790
## [120,] 0.8945719 0.8856237 0.8857793 0.8844382 0.8961658 0.8523899 0.8794887
## [121,] 0.9862772 0.9533857 0.9449564 0.9580790 0.9214616 0.9301216 0.9347985
## [122,] 1.0143191 1.0032329 1.0049513 1.0001026 0.9921573 1.0005049 0.9591606
## [123,] 0.9733806 0.9879126 0.9947735 0.9800279 0.9931484 1.0024638 0.9948792
##             [,15]     [,16]     [,17]     [,18]     [,19]
```

```
## [118,] 0.8824280 0.8730733 0.8639161 0.8614441 0.8717307
## [119,] 0.8396531 0.8495173 0.8266791 0.8544568 0.8598048
## [120,] 0.8631928 0.8499712 0.7994624 0.8024431 0.8002607
## [121,] 0.8985886 0.8409815 0.8223158 0.8457842 0.8257107
## [122,] 0.9478091 0.9185031 0.8930257 0.8944247 0.8615051
## [123,] 0.9941488 0.9852592 0.9909004 0.9581546 0.9137532
```

```
head(temp_m_sf)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 1.052653 1.049468 1.120607 1.103336 1.118390 1.108172 1.140906 1.140574
## [2,] 1.100742 1.099653 1.108025 1.098323 1.110184 1.116213 1.126827 1.154074
## [3,] 1.135413 1.135420 1.139096 1.142831 1.143201 1.138495 1.129678 1.156092
## [4,] 1.110338 1.110492 1.117079 1.125774 1.134539 1.126117 1.130758 1.137722
## [5,] 1.025231 1.025233 1.044684 1.067291 1.084725 1.097239 1.115055 1.103877
## [6,] 1.025838 1.025722 1.028169 1.042340 1.053954 1.067494 1.080203 1.094312
##          [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]    [,16]
## [1,] 1.125438 1.122063 1.161415 1.198102 1.198910 1.243012 1.243781 1.238435
## [2,] 1.142187 1.131889 1.144549 1.134661 1.153433 1.165431 1.172935 1.190735
## [3,] 1.165657 1.147982 1.149459 1.135756 1.153310 1.155197 1.157286 1.169773
## [4,] 1.150639 1.146992 1.142497 1.150162 1.151169 1.157751 1.163844 1.159343
## [5,] 1.120818 1.133733 1.132167 1.142714 1.139244 1.112909 1.132435 1.132045
## [6,] 1.102680 1.092178 1.075766 1.088547 1.082185 1.103092 1.115071 1.118575
##         [,17]    [,18]    [,19]
## [1,] 1.300204 1.290647 1.254521
## [2,] 1.191956 1.219190 1.228826
## [3,] 1.189915 1.172309 1.169045
## [4,] 1.166605 1.167993 1.158956
## [5,] 1.145230 1.168161 1.170449
## [6,] 1.121598 1.134962 1.145475
```

```
temp_m_smoothed <- matrix(temp_m$fitted[,1],nrow=123) #the smoothed values of the level component
plot(temp_m_smoothed)
```




```r
#Create a csv file with smoothened data, which can further be used for CUSUM
write.csv(temp_m_smoothed, file="tempsea2.csv", fileEncoding = "UTF-16LE")
```

Good to take seasonality into consideration. to help make a judgment of whether the unofficial end of summer has gotten later over the 20 years.