Homework 4: Exponential Smoothing

2025-02-01

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 a (the first smoothing parameter) to be closer to 0 or 1, and why?

Exponential smoothing could be useful if a brand is dropping a new clothing line and they want to predict how many pieces they'll sell over the first few weeks. Demand is unpredictable, maybe there's a huge surge at launch, then sales slow down, or maybe a celebrity wears it, and sales spike again. Exponential smoothing could help adjust their forecast in real time based on how sales are actually going.

What Data Would Help? Early sales numbers (how fast things sell in the first few hours/days) Pre-order & website traffic (how many people showed interest before launch) Marketing impact (influencer posts, ads, and hype levels) Stock & restocks (if it sells out, does demand stay high or drop off?)

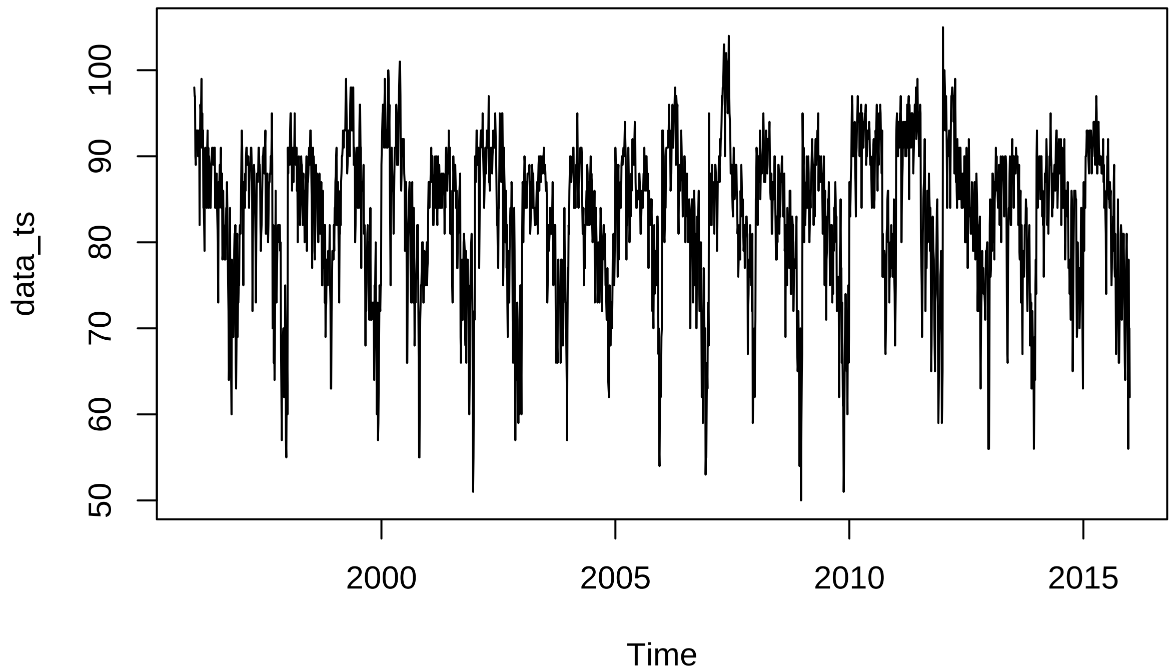
Would a Be Closer to 0 or 1? Closer to 1 (like 0.7 - 0.9) because demand can change fast. If a product is blowing up on social media, the forecast needs to adjust quickly. If sales slow down after launch, the model should pick up on that too. For something more consistent, like a basic hoodie that always sells at a steady rate, a would be lower (closer to 0) to smooth out random spikes. But for more unique or bold pieces, the model needs to react fast to sudden surges in demand.

Question 7.2

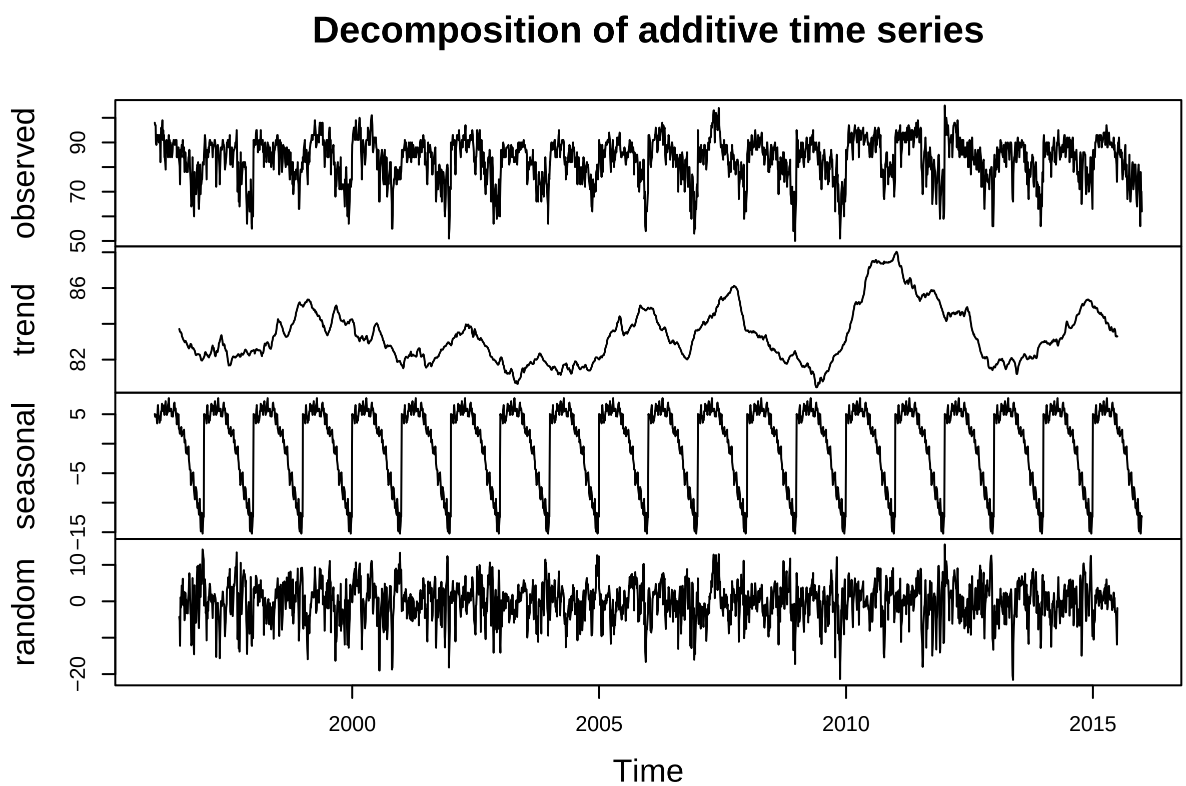
Using the 20 years of daily high temperature data for Atlanta (July through October) from Question 6.2 (file temps.txt), build and use an exponential smoothing model to help make a judgment of whether the unofficial end of summer has gotten later over the 20 years. (Part of the point of this assignment is for you to think about how you might use exponential smoothing to answer this question. Feel free to combine it with other models if you'd like to. There's certainly more than one reasonable approach.)

Note: in R, you can use either HoltWinters (simpler to use) or the smooth package's es function (harder to use, but more general). If you use es, the Holt-Winters model uses model="AAM" in the function call (the first and second constants are used "A"dditively, and the third (seasonality) is used "M"ultiplicatively; the documentation doesn't make that clear).

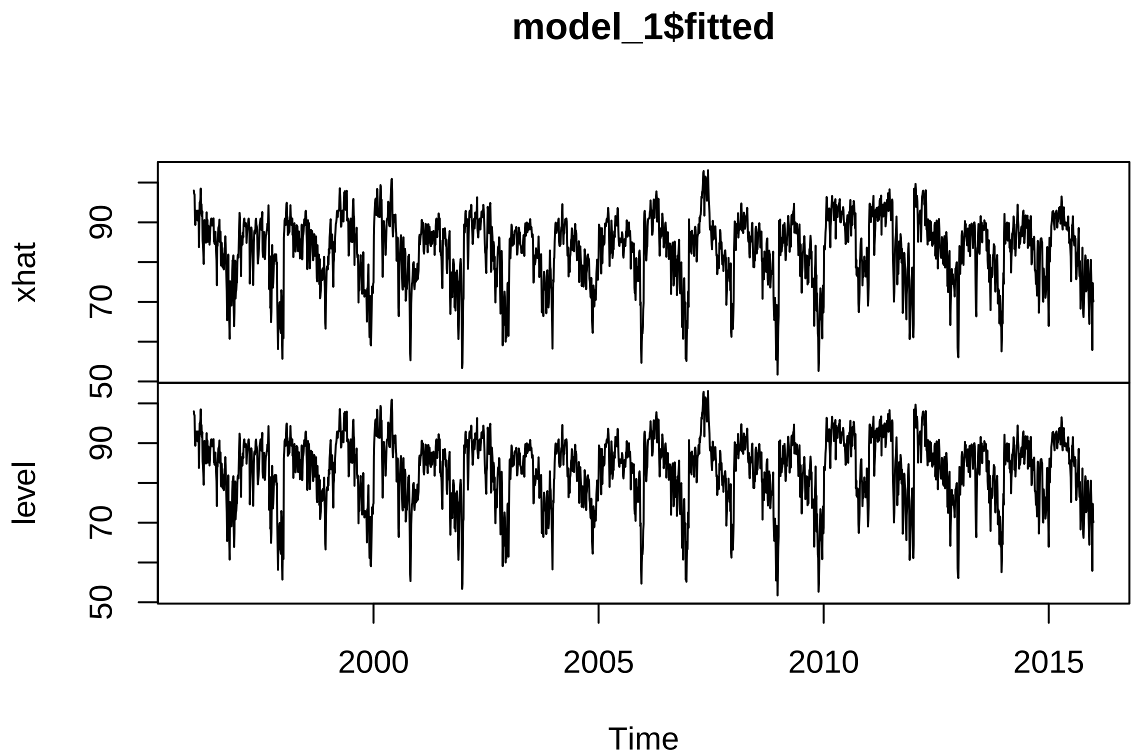
```r
#load the temps data
data <- read.table('temps.txt', stringsAsFactors = FALSE, header = TRUE)
library(ggplot2)

data_vec <- as.vector(unlist(data[,2:21]))
#convert vector data to time series data with 123 observations for 20 years.
data_ts <- ts(data_vec, start = 1996, frequency = 123)
plot(data_ts)
```




```r
#check if there is trend and seasonality via decompose function
plot(decompose(data_ts))
```




```r
#model_1 for single exponential smoothing
model_1 <- HoltWinters(data_ts, beta = FALSE, gamma = FALSE)
plot(model_1$fitted)
```



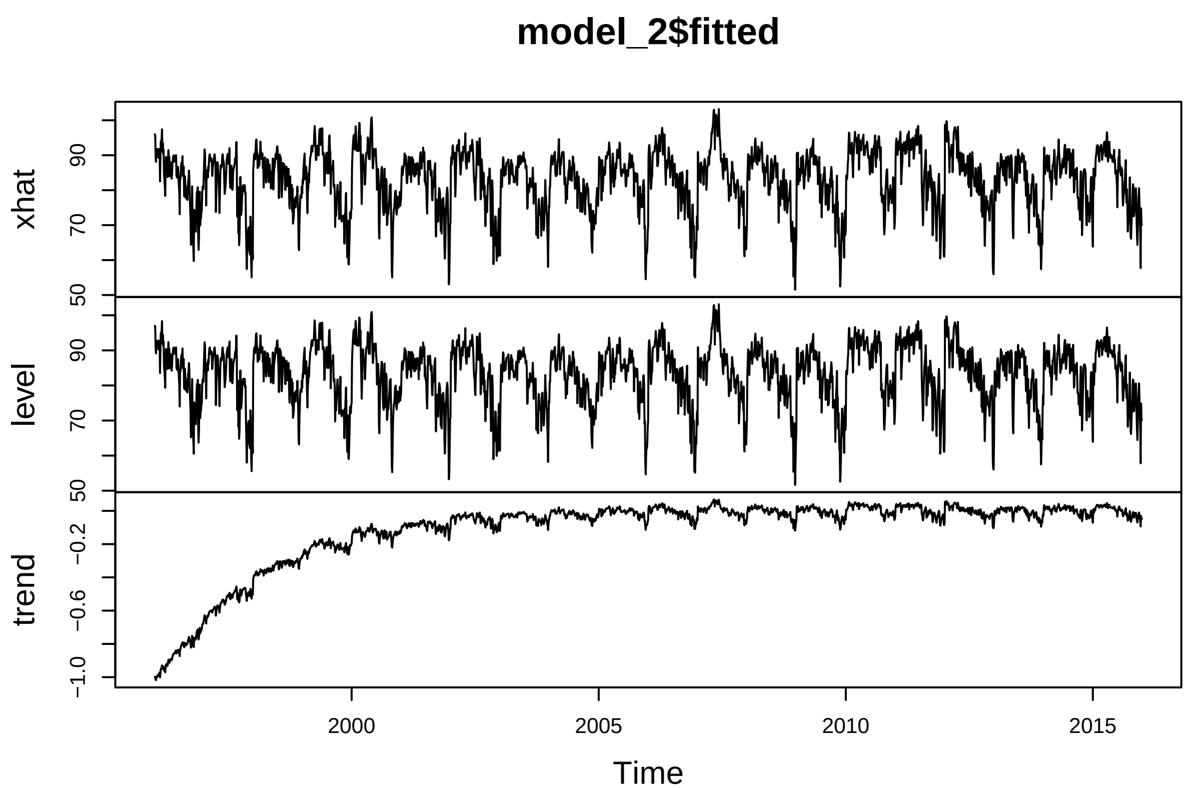

```r
#model_2 for double exponential smoothing
model_2 <- HoltWinters(data_ts, gamma = FALSE)
```

```
## Warning in HoltWinters(data_ts, gamma = FALSE): optimization difficulties:
## ERROR: ABNORMAL_TERMINATION_IN_LNSRCH
```

```r
model_2
```

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

```r
plot(model_2$fitted)
```




```r
#model_3 for triple exponential smoothing with seasonal=additive
model_3 <- HoltWinters(data_ts)
model_3
```

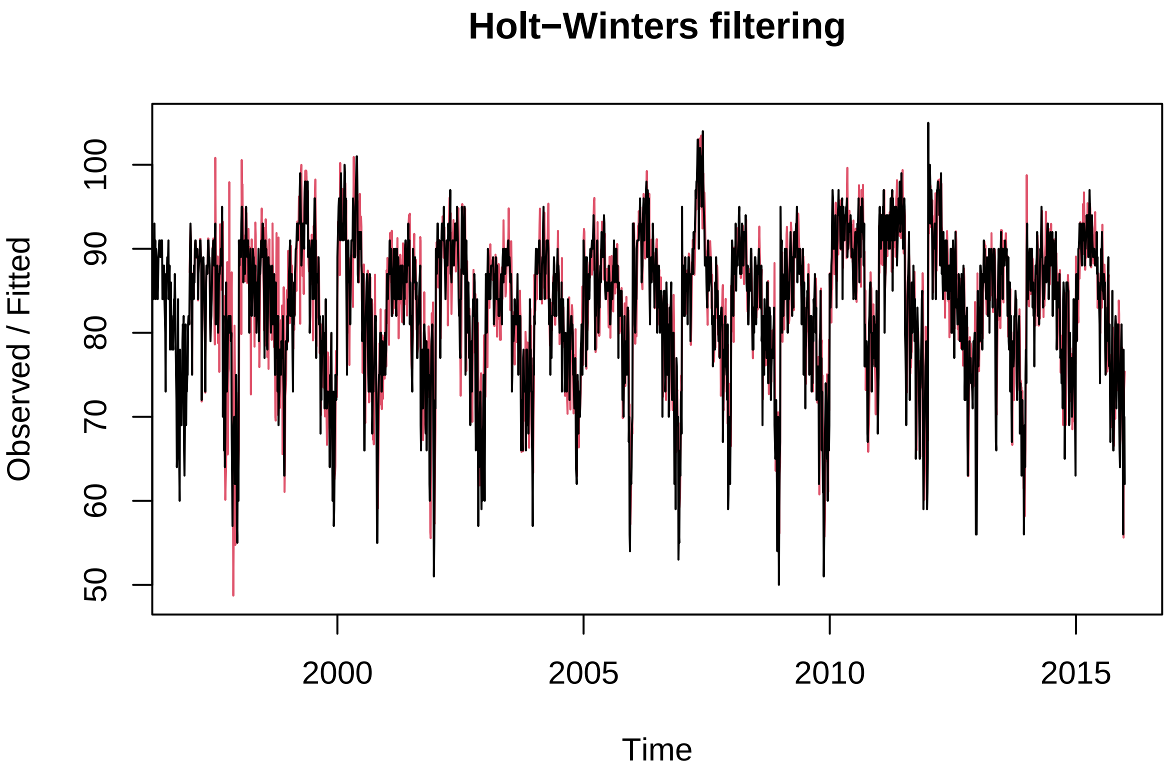
```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = data_ts)
##
## Smoothing parameters:
##  alpha: 0.6610618
##  beta : 0
##  gamma: 0.6248076
##
## Coefficients:
##              [,1]
## a     71.477236414
## b     -0.004362918
## s1    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
```

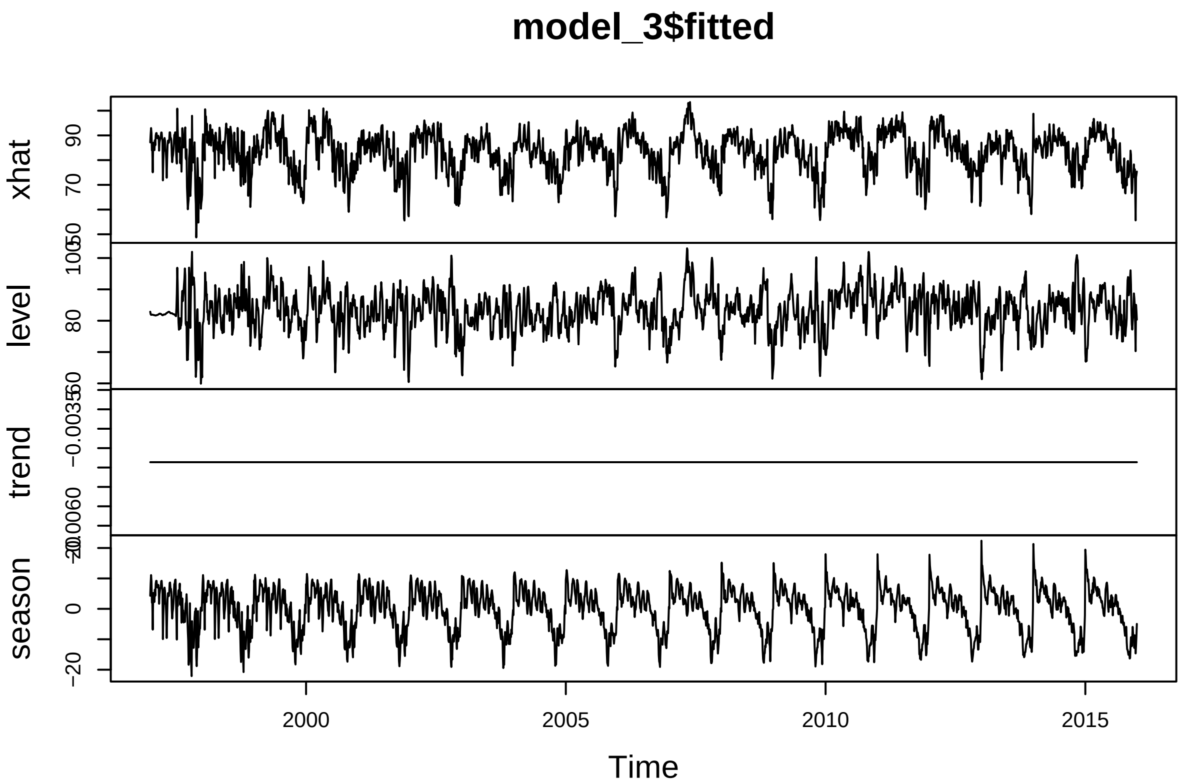
```
## s66   -0.231239928
## s67    0.542761270
## s68    0.780131779
## s69    1.096690727
## s70    0.690525998
## s71    2.301303414
## s72    2.965913580
## s73    4.393732595
## s74    2.744547070
## s75    1.035278911
## s76    1.170709479
## s77    2.796838283
## s78    2.000312540
## s79    0.007337449
## 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
```

```r
plot(model_3)
```




```r
plot(model_3$fitted)
```




```r
#model_4 for triple exponential smoothing with seasonal=multiplicative
model_4 <- HoltWinters(data_ts, seasonal = 'multiplicative')
model_4
```

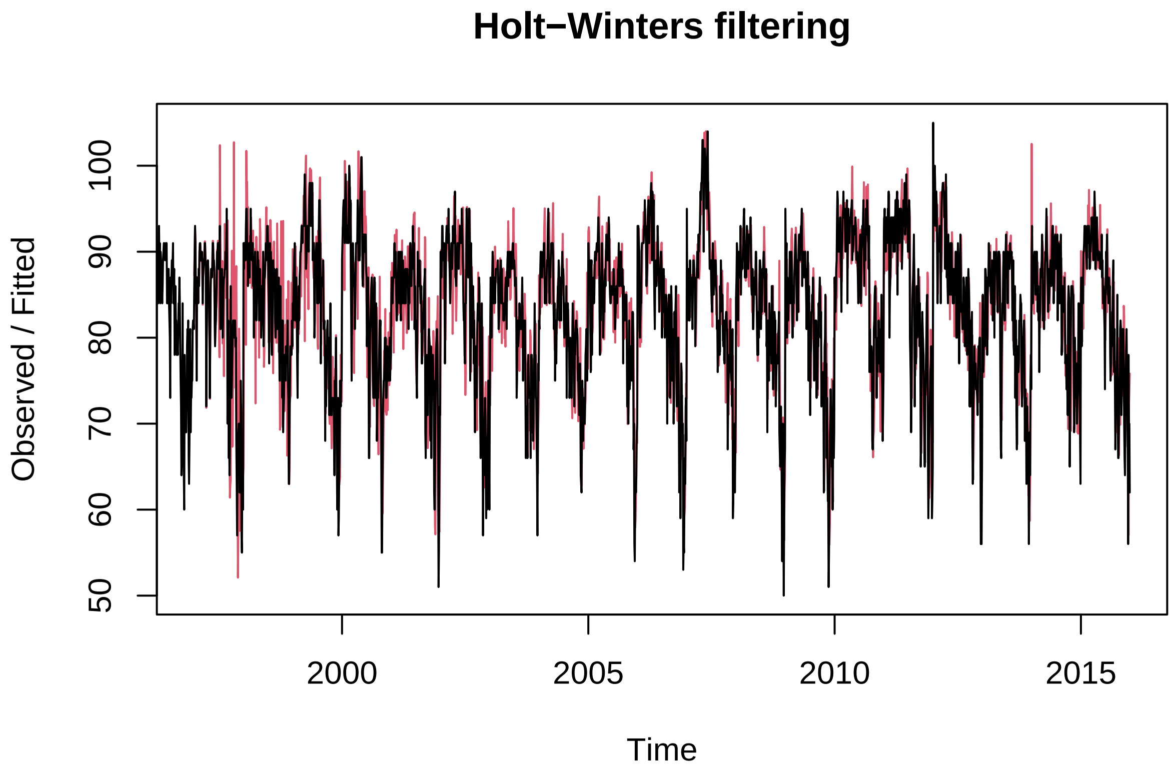
```
## Holt-Winters exponential smoothing with trend and multiplicative seasonal component.
##
## Call:
## HoltWinters(x = data_ts, seasonal = "multiplicative")
##
## Smoothing parameters:
##  alpha: 0.615003
##  beta : 0
##  gamma: 0.5495256
##
## Coefficients:
##             [,1]
## a    73.679517064
## 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
```

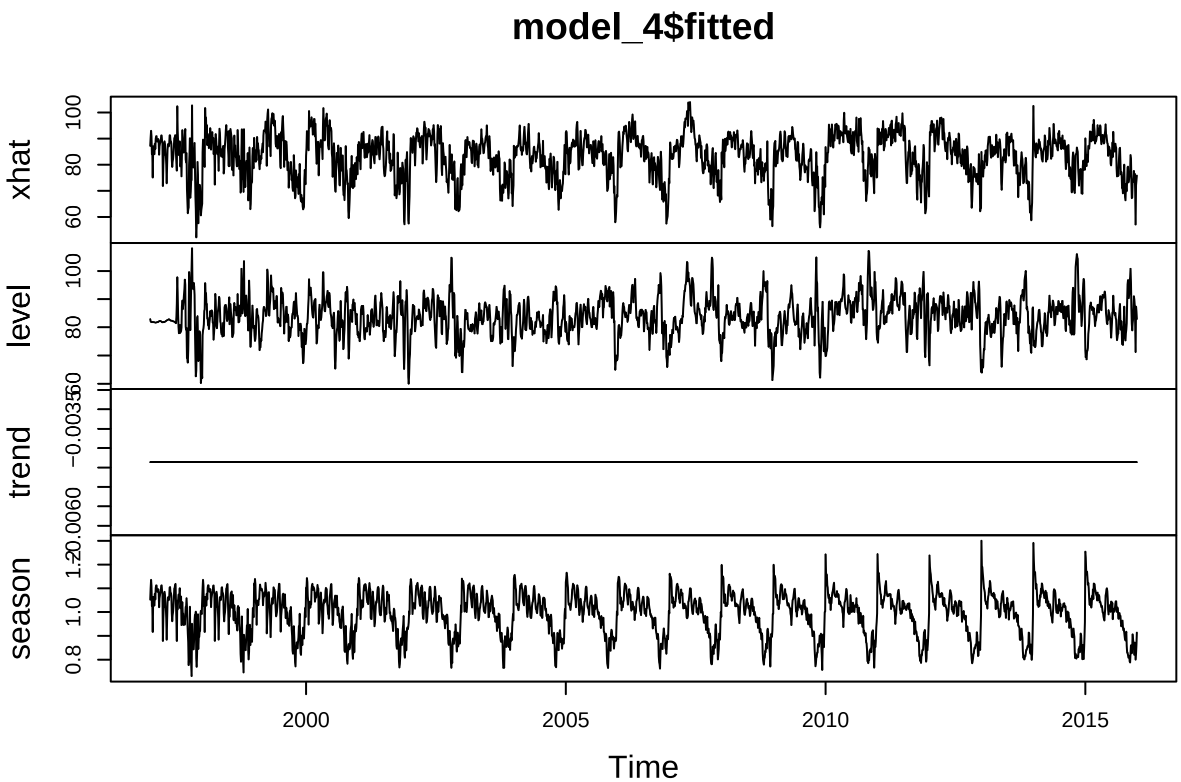
```
## s66   0.991019224
## s67   1.001437488
## s68   1.002221759
## s69   1.003949213
## s70   0.999566344
## s71   1.018636837
## s72   1.026490773
## s73   1.042507768
## s74   1.022500795
## s75   1.002503740
## s76   1.004560984
## s77   1.025536556
## s78   1.015357769
## s79   0.992176558
## 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
```

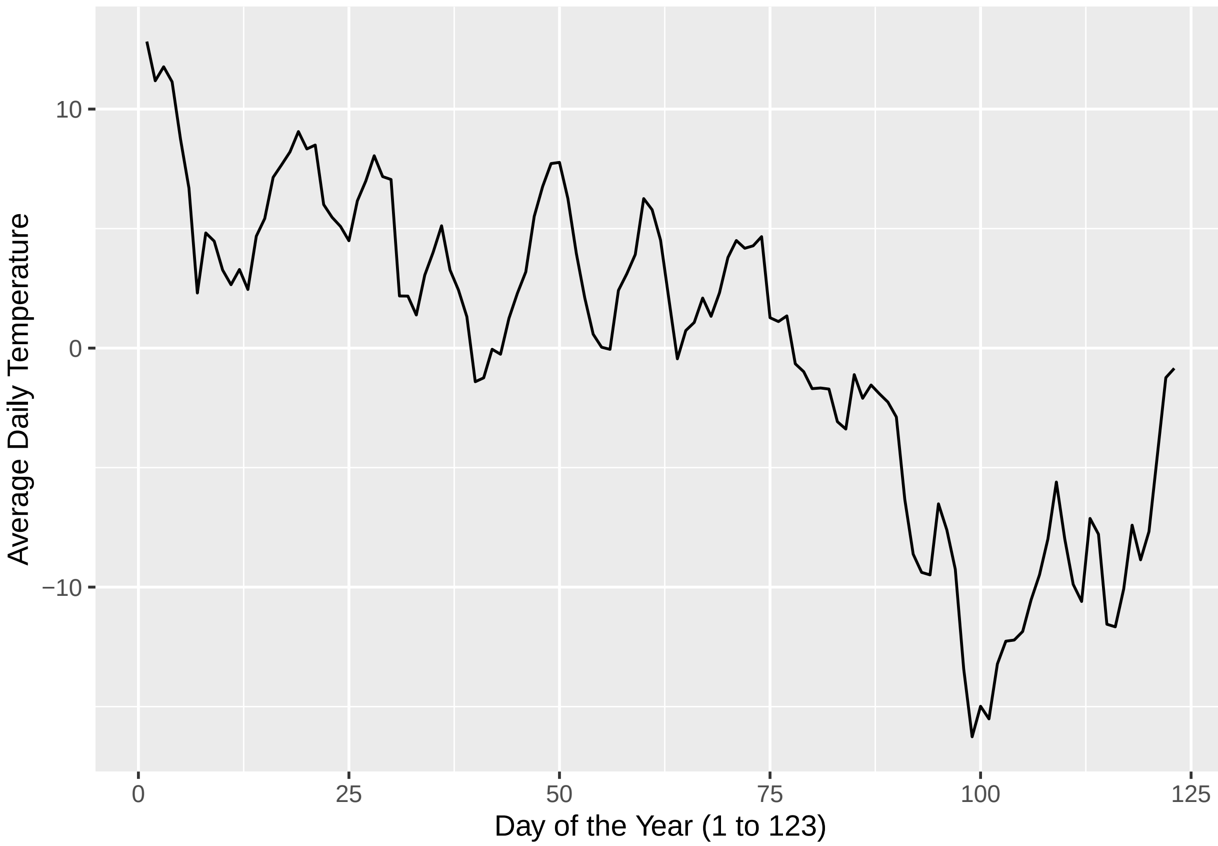
```r
plot(model_4)
```




```r
plot(model_4$fitted)
```




```r
#calculate mean of each day across the 20 years
model_3_season <- matrix(model_3$fitted[,4], nrow = 123)
avg_daily <- data.frame(rowMeans(model_3_season, n = 19))
ggplot(avg_daily, aes(x = 1:123, y = rowMeans.model_3_season..n...19.)) +
  geom_line() +
  labs(
    x = "Day of the Year (1 to 123)",    # x-axis label
    y = "Average Daily Temperature"      # y-axis label
  )
```




```
std <- sd(avg_daily$rowMeans.model_3_season..n...19.[1:62])
C <- std*0.3
T <- std*5
C
```

```
## [1] 0.9697854
```

```
T
```

```
## [1] 16.16309
```

```
s = list()
avg_daily$s <- 0
#cusum method from hw3 is applied to check change to the end of summer
cusum <- function(C, T, cl){
  for (j in cl){
    for (i in 1:nrow(avg_daily)){
      if (i-1 != 0){
        s <- max(0, avg_daily$s[i-1] + mean(avg_daily[,j]) - avg_daily[i,j] - C)
        }
      else{s <- 0}
      if (s >= T){s}
      else{s <- 0}
      avg_daily$s[[i]] <- s
      }
    }
  return(data.frame(1:123, avg_daily[,j], avg_daily$s)[78:123,])
  # set output start from base level day 78
  }
```

```r
cusum(C=0.9697854, T=16.16309, cl=1)
```

```
##      X1.123 avg_daily...j. avg_daily.s
## 78       78     -0.6562640           0
## 79       79     -0.9865849           0
## 80       80     -1.6968615           0
## 81       81     -1.6688708           0
## 82       82     -1.7134622           0
## 83       83     -3.0786163           0
## 84       84     -3.3870835           0
## 85       85     -1.1096552           0
## 86       86     -2.0995425           0
## 87       87     -1.5440073           0
## 88       88     -1.9147525           0
## 89       89     -2.2592635           0
## 90       90     -2.8837074           0
## 91       91     -6.3260493           0
## 92       92     -8.6200910           0
## 93       93     -9.3828956           0
## 94       94     -9.4880777           0
## 95       95     -6.5151863           0
## 96       96     -7.6117861           0
## 97       97     -9.2529673           0
## 98       98    -13.4282487           0
## 99       99    -16.2629184           0
## 100     100    -14.9832377           0
## 101     101    -15.5134511           0
## 102     102    -13.2152299           0
## 103     103    -12.2627214           0
## 104     104    -12.2140597           0
## 105     105    -11.8562254           0
## 106     106    -10.5370460           0
## 107     107     -9.4835355           0
## 108     108     -7.9795695           0
## 109     109     -5.6027596           0
## 110     110     -7.9832251           0
## 111     111     -9.8836170           0
## 112     112    -10.6011249           0
## 113     113     -7.1261238           0
## 114     114     -7.7851236           0
## 115     115    -11.5537389           0
## 116     116    -11.6623496           0
## 117     117    -10.0716372           0
## 118     118     -7.4072725           0
## 119     119     -8.8583903           0
## 120     120     -7.6735617           0
## 121     121     -4.4656976           0
## 122     122     -1.2369297           0
## 123     123     -0.8479661           0
```

Has Summer Been Ending Later in Atlanta? A Look at 20 Years of Temperature Data

This analysis explores whether the end of summer in Atlanta has been getting later over the past 20 years (1996–2015) by looking at daily high temperatures from July through October. Using exponential smoothing models and CUSUM (Cumulative Sum Control Chart) analysis, we tried to detect any meaningful shifts in late-summer temperatures.

Breaking Down the Models

To smooth out fluctuations and identify trends, we used different types of Holt-Winters exponential smoothing models:

- **Model 1 Single smoothing:** Only smooths the temperature data, assuming no trend or seasonality.
- **Model 2 Double smoothing:** Incorporates trend but assumes no seasonal variation.
- **Model 3 Triple smoothing (additive seasonality):** Assumes temperature increases or decreases by a fixed amount each year.
- **Triple smoothing (multiplicative seasonality):** Assumes seasonal temperature variations scale proportionally rather than remain constant.

The trend coefficient in our models came out to be close to zero (-0.0044), which suggests that the overall timing of late summer temperatures hasn't shifted much. While there are small variations, the seasonal patterns have remained fairly stable.

Checking for Changes Using CUSUM

To detect whether summer has been ending later, the CUSUM method was applied. This technique accumulates small changes in temperature patterns to identify significant deviations. The key thresholds were determined using the standard deviation of early summer temperatures:

- **Control Limit (C):** 30% of the standard deviation, used to filter out normal fluctuations. C = 0.9697854 is the threshold to account for minor noise that don't indicate a real shift.
- **Threshold (T):** 5 times the standard deviation, defining a significant shift. If the cumulative deviation exceeds T = 16.16309, it is a signal of a true shift or change in the temperature pattern. If the cumulative sum reaches or exceeds this value, it suggests that a meaningful change in the data has occurred, beyond just random noise.

The results showed no clear sign that summer has been ending later. While there were some fluctuations in later years, they weren't strong enough to be considered a real trend.

Conclusion

Based on these models, there's **no strong evidence that the unofficial end of summer in Atlanta has been shifting later** over the past two decades. The seasonal temperature patterns look pretty stable, and while there may be small changes year to year, they aren't statistically significant.