Homework 4: ISYE 6501 - Introduction to Analytics Modeling

Question 7.1

Prompt

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

Solution

Considering american football (National Football League/NFL), exponential smoothing is useful for forecasting metrics over time that have a level of randomness. Analysts can use exponential smoothing when considering models that predict a team's future performance based on previous results.

The data needed to forecast a team's performance require data from previous games. This would include historical game data including scores, data, and player statistics. In addition, one may consider additional factors that could impact performance such as injuries, weather conditions or the consistency of players on the roster.

For the smoothing parameter, α, values closer to 0 or 1 would depend on the team and performance consistency. For example, let's consider data from 2014-2023.

- The Indianapolis Colts have not won their division since 2014 and have made the playoffs only 3 times in 10 years. This has resulted in performance that has high variance and has changed quickly year over year. In this instance, α **would be closer to 1**.
- The Kansas City Chiefs have made the playoffs 9 times in 10 seasons, including winning the championship 3 times. The Chiefs performance is relatively stable over the last 10 seasons as they have been successful. In this instance, α **would be closer to 0**.

Question 7.2

Prompt

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).

Exponential Smoothing

Exponential smoothing refers to using an exponentially weighted moving average (EWMA) to smooth a time series. Exponential smoothing is defined as:

$$S_t = \alpha x_t + (1 - \alpha) S_{t-1}$$

In the equation above, S_t is the expected baseline response at time period t. The observed response is defined as x_t and the range for α is defined as $0 < \alpha < 1$. Tuning α to 0 means there is a lot of randomness in the system, while a value of 1 means there is *not* a lot of randomness in the system.

The Holt-Winters method [1] is an exponential smoothing method that expands upon Holt's method by adding a seasonal component. Holt-Winters method can be utilized when time series data has both trend and seasonality. There are two seasonal models to consider, the additive model and the multiplicative model.

The additive model is prescribed when the seasonal variations are roughly constant throughout the series. The seasonal component is added to trend and level components.

The multiplicative model is optimal when the seasonal variations change proportionally to the level of the series. Simply put, the seasonal component is multiplied with trend and level components.

Holt-Winters Method

Now that we have defined exponential smoothing, the data is explored. First the data is imported and the code is expanded step-by-step to eventually use the Holt-Winters method for the temperature dataset.

```r
# Import libraries
# No import needed, HoltWinters is in stats, pre-installed
library(ggplot2)

# Set the working directory
setwd("~/projects/ISYE6501/HW4")

# Load the temperature data into a table
data <- read.table("data/temps.txt", stringsAsFactors = FALSE, header = TRUE)

# View the imported data to check import but only first few rows
head(data, 5)
```

```
##     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
##   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
```

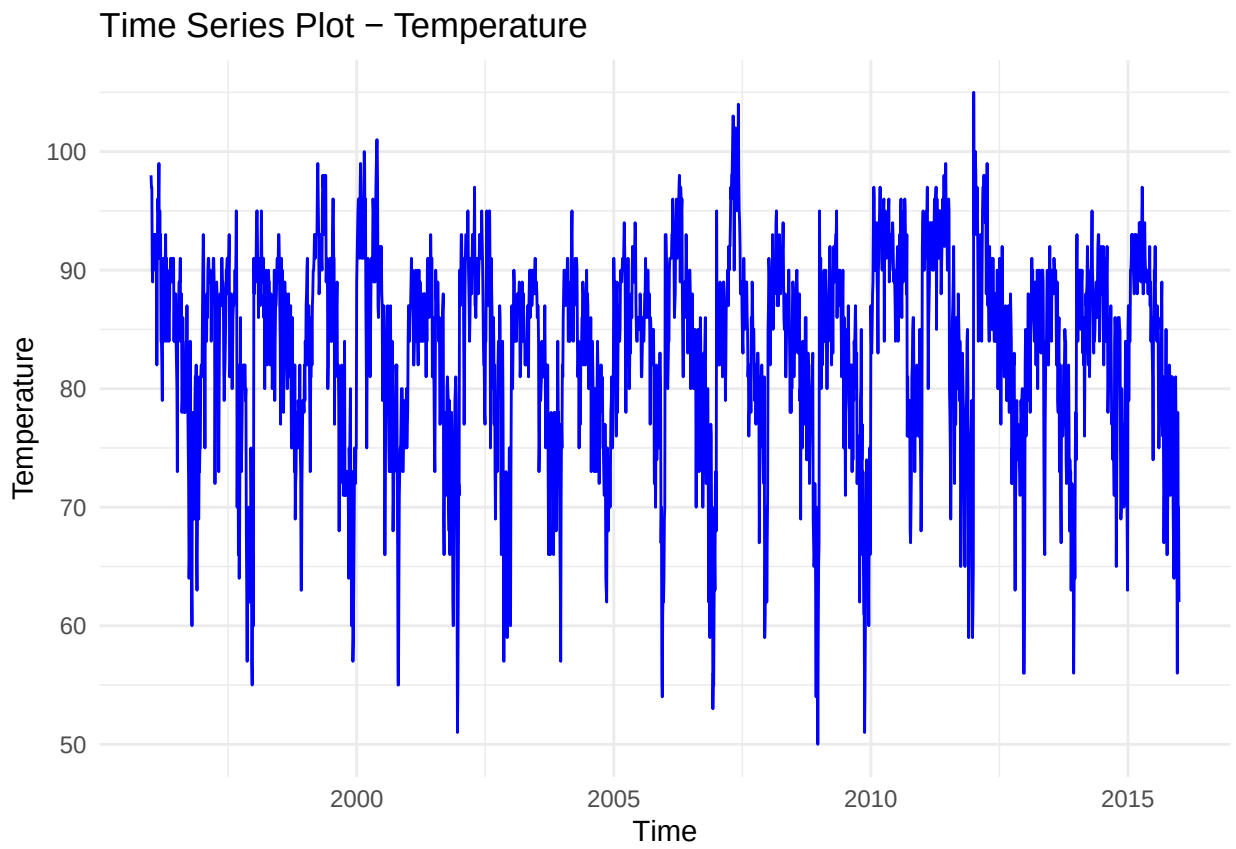
Now that we have confirmed that temperature data is imported and how it is organized, we need to do a little bit of data wrangling to be able to look at our time series data.

First, the data in R needs to be formatted as a vector to be converted into a time series object. The variable `temps` is created with all temperature data excluding the dates, hence why we use `data[, 2:21]`.

Second, the temperatures listed as a vector now can be converted into a time series using `ts()`. From observing our original data stored in `data`, we know the first period where the data is recorded starts in 1996, so in `ts()` it is defined `start = 1996`. The term frequency is also defined, which is the number of observations per unit time. This information and the function `ts()` requirements were obtained from documentation by using `?ts` in the RStudio console. The number of observations per unit time is the number of temperatures recorded per year, which is observed in the original variable `data` as 123 temperatures recorded per year, so `frequency = 123`.

Finally, some additional data parsing is conducted to store both the temperatures and time into a single dataframe so that a plot can be generated with `ggplot2`. The plot is created to display the time vs. temperature time series.

```r
# Time series objects in R  need to be formatted as a vector
# We need the values - Drop the dates column
temps <- as.vector(unlist(data[, 2:21]))

# Use ts to create a time series object from the vector "temps"
# From data we know the start time is 1996, so start = 1996
# There are 123 temperature recordings or "frequency"
temps_ts <- ts(temps, start = 1996, frequency = 123)

# Convert time series to data frame to plot with ggplot2
ts_df <- data.frame(
  Time = as.numeric(time(temps_ts)),  # Extract time as a numeric vector
  Temperature = as.numeric(temps)        # Extract values
)

# Plot the time series data to visualize stored in a dataframe
temp_plot <- ggplot(ts_df, aes(x = Time, y = Temperature)) +
  geom_line(color = "blue") +  # Line plot
  labs(title = "Time Series Plot - Temperature", x = "Time", y = "Temperature") +
  theme_minimal()

print(temp_plot)
```



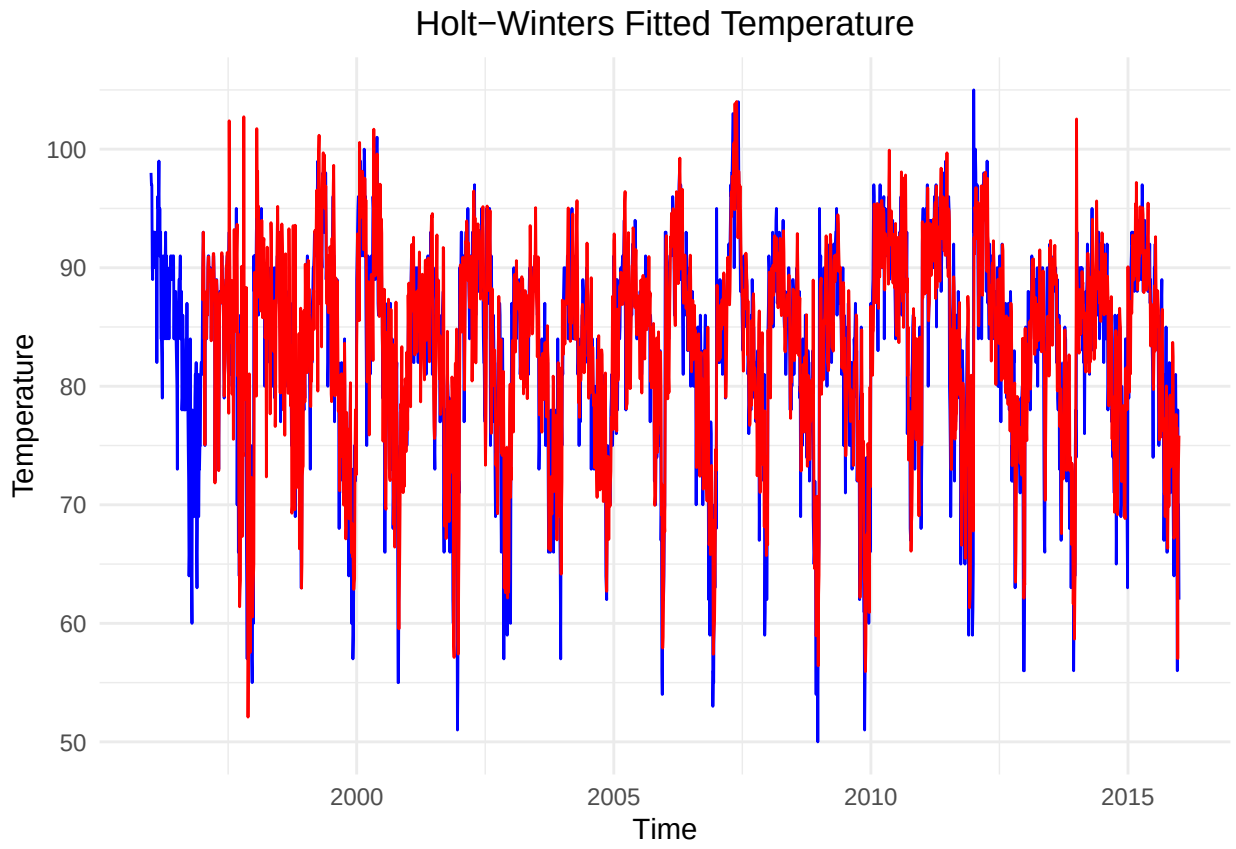
Observing the plot of the temperature time series above, we can see the variation in temperature year over year and the fluctuations present. Next, the Holt-Winters method is used with the temperature time series to smooth the data.

With the `HoltWinters` function called and stored in `temps_hw`, some additional data organization is used to plot again. A separate dataframe `fitted_df` is defined to store the fitted data from the Holt-Winters model. In order to plot this against the original time series data, the number of values needs to equivalent for both, so a technique to pad the data is applied. This means that some NA values are generated so there is an equal length of data between the original temperatures and fitted results. The NA values are not utilized in the plot, so this method is only to structure the data accordingly so it can be shown in a plot. Once the data is structured in `fitted_df` to be equivalent length as the time series dataframe, both are merged into a single dataframe and plotted.

Note that the original time series data is in blue and the Holt-Winters fitted temperatures are in red.

```
# Smooth temperature time series data with Holt-Winters
temps_hw <- HoltWinters(temps_ts, alpha=NULL, beta=NULL, gamma=NULL, seasonal = "multiplicative")

# Create a separate data frame for the fitted values
# Since the fitted values are not the same length as the original data,
# the dataframe needs to be padded with NA values (that won't be used in plot)
# This will allow for me to merge all data into a single dataframe
fitted_df <- data.frame(
  Time = as.numeric(time(temps_ts)),       # Use the same time points
  Fitted = c(rep(NA, length(temps_ts) - length(fitted(temps_hw)[, 1])), fitted(temps_hw)[, 1])
)
# Merge the original data frame with the fitted values data frame
```

```r
combined_df <- merge(ts_df, fitted_df, by = "Time")

# Plot the original and fitted values for temperature time series
temp_plot_fit <- ggplot(combined_df, aes(x = Time)) +
  geom_line(aes(y = Temperature), color = "blue", linetype = "solid") +   # Original data
  geom_line(aes(y = Fitted), color = "red", linetype = "solid") + # Fitted values
  labs(title = "Holt-Winters Fitted Temperature", x = "Time", y = "Temperature") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) # Center the title

print(temp_plot_fit)
```



Observing the plot with fitted temperatures using the Holt-Winters method, it appears that some smoothing has occurred.

Comparing Seasonal Models

Using the Holt-Winters method, the seasonal model can either be `additive` or `multiplicative` [2]. Each choice should be considered as each option has their own use cases. The additive method is optimal when the seasonal variations are roughly constant through the series. The multiplicative method is better for seasonal variations that are changing proportional to the level of the series.

To compare these two options, a model is created for both the additive method and multiplicative method. The values for `SSE` are compared, which is the final sum of squared errors achieved in optimizing. This means that `SSE` compares the sum of squared differences between the actual and predicted values. In summary,

lower SSE values indicate that the fitted model predictions are closer to the actual data points, which means the model has better accuracy in capturing the patterns in the data.

By comparing the additive and multiplicative SSE values, it is shown below that the additive method has a lower SSE compared to the multiplicative method (66244.25 v. 68904.57). This should make sense as it confirms the additive model is more optimal for temperature data since the seasonal variations are roughly constant through the series.

```
# Holt-Winters method with the additive seasonal model
temps_hw_add <- HoltWinters(temps_ts, alpha=NULL, beta=NULL, gamma=NULL, seasonal = "additive")

# Holt-Winters method with the multiplicative seasonal model
temps_hw <- HoltWinters(temps_ts, alpha=NULL, beta=NULL, gamma=NULL, seasonal = "multiplicative")

cat("SSE for Holt-Winters Additive Method", temps_hw_add$SSE, "\n")
```

```
## SSE for Holt-Winters Additive Method 66244.25
```

```
cat("SSE for Holt-Winters Multiplicative Method", temps_hw$SSE, "\n")
```

```
## SSE for Holt-Winters Multiplicative Method 68904.57
```

Determine End of Summer

Next, an exponential smooth model is utilized to determine whether the unofficial end of summer has gotten later over the years. First, we will use the Holt-Winters method with the `additive` seasonal model since we determined it is more optimal for temperature data in the section above.

Second, the data needs to be cleaned up. Using the results from the model, we will access the level component of the fitted values and store them for comparison later. The trend component is padded with `NA` for the initial periods where the Holt-Winters model has no fitted values to ensure the length of the data matches the temperature time series (`temps_ts`).

Third, *assume that the unofficial end of Summer is October 1st* to identify if the temperature is cooling down. The position or index in the data is obtained by looking for the string `1-Oct`. The index is used to pull the temperature data for September 1st from the original data and the trend determined by the Holt-Winters method. All of the data is saved in a new dataframe and the `NA` values are removed that were used for padding.

Now that all the data cleanup is finished, a quick check of the data is performed using `head()`.

```
# Holt-Winters method with the additive seasonal model
temps_hw_add <- HoltWinters(temps_ts, alpha=NULL, beta=NULL, gamma=NULL, seasonal = "additive")

# Extract the trend component by accessing the level component of the fitted values
trend_component <- temps_hw_add$fitted[, "level"]

# Pad trend_component with NAs to match the length of temps_ts
trend_component_padded <- c(rep(NA, length(temps_ts) - length(trend_component)), trend_component)

# Assuming end of summer is September 1st, find what the indices is in data to call later
indices <- which(data$DAY == "1-Oct")

# Index for September 1st in each year (row 63 for each year checking data)
```

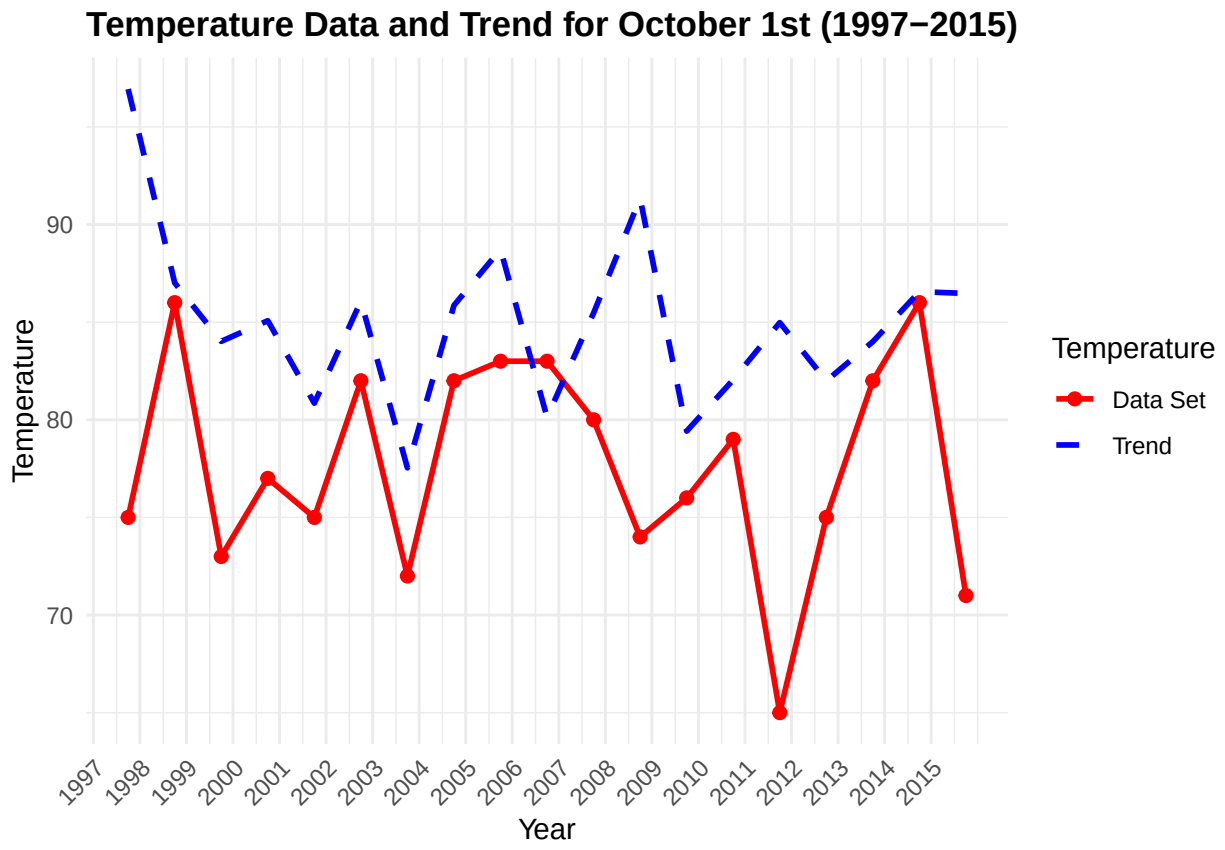
```r
october_1st_index <- seq(indices, length(temps_ts), by = 123)  # Every year's September 1st index

# Extract original temperature data for September 1st each year
october_1st_original <- temps_ts[october_1st_index]

# Extract the trend component for September 1st (corresponding trend values)
october_1st_trend <- trend_component_padded[october_1st_index]

# Create a data frame for comparison
october_1st_data <- data.frame(
  Year = time(temps_ts)[october_1st_index],
  Original = october_1st_original,
  Trend = october_1st_trend
)

# Remove NA values (if any)
october_1st_data <- na.omit(october_1st_data)
# Show the data to check
head(october_1st_data, 5)
```

```
##       Year Original    Trend
## 2 1997.748       75 96.93957
## 3 1998.748       86 87.01248
## 4 1999.748       73 84.01814
## 5 2000.748       77 85.07599
## 6 2001.748       75 80.84858
```

Finally, we will use our data from the original time series for temperature and the trend temperature to determine how temperature changes to determine the end of summer. The plot is generated below to look at the temperature for October 1st.

```r
# Plot the September 1st temperature data along with the trend
plot_october_1st <- ggplot(october_1st_data, aes(x = Year)) +
  # Plot temperature data where line color is defined farther below
  geom_line(aes(y = Original, color = "Data Set"), size = 1) +
  # Add in scatter points on the line
  geom_point(aes(y = Original, color = "Data Set"), size = 2) +
  # Plot the trend line from the fitted Holt-Winters data
  geom_line(aes(y = Trend, color = "Trend"), linetype = "dashed", size = 1) +
  # Add labels to for a title and axes
  labs(title = "Temperature Data and Trend for October 1st (1997-2015)",
       x = "Year", y = "Temperature",
       color = "Temperature") +
  # Use floor() so the years are shown as integers rounding down
  scale_x_continuous(breaks = floor(october_1st_data$Year)) +
  theme_minimal() +
  # Rotate x-axis labels so they do not overlap
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(face = "bold")) +
  # Set the colors for the legend
  scale_color_manual(values = c("Data Set" = "red", "Trend" = "blue"))
```

```r
# Display the plot
print(plot_october_1st)
```



Conclusion

From the plot above, it is clear that there is a trend of increasing temperatures each year on the 1st of October. Clarifying the unofficial end of summer is when temperatures start to cool, the end is later each year because there are still temperatures that are increasing on the trend even though we are officially in the Fall season.

References

[1] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004, doi: https://doi.org/10.1016/j.ijforecast.2003.09.015.

[2] "HoltWinters function - r documentation — rdocumentation.org." https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/HoltWinters.