Goal: Analyze and forecast time series data using exponential versus ARIMA models and provide deliverables associated with **Task 1 (see description below)** and **Task 2 (see description on page 4)**.

Task1: *Write an R script that:*

- *Uses the "gtrendsR" library to extract all monthly historical data associated with the following search queries: "TV", "data science", and "hey google"*
- *Decomposes the resulting series into their seasonal, trend, and remainder components*
- *Using the HoltWinters and auto.arima functions, fits a single, double, and triple exponential smoothing to the data as well as an ARIMA model using only data for up to December 2016 (as we saw in class).*
- *Calculates the accuracy of the predictions for all 2017 of each model via the mean absolute error.*

Methods and Results:

1. Using the "gtrendsR" library, monthly historical data was retrieved for the following search queries: "TV", "Data Science", and "Hey Google".
2. Retrieved data was stored as a ts object and decomposed for their trend, seasonal, and random components as shown in **Figures 1, 2,** and **3**.

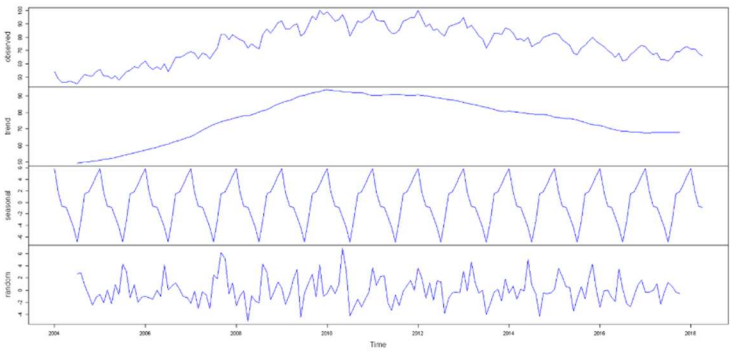

Figure 1: Decomposition of "TV" searches in monthly time series from 2004-2018

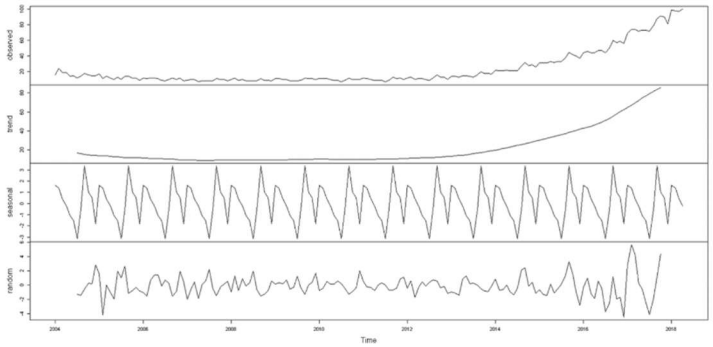

Figure 2: Decomposition of "Data Science" searches in monthly time series from 2004-2018

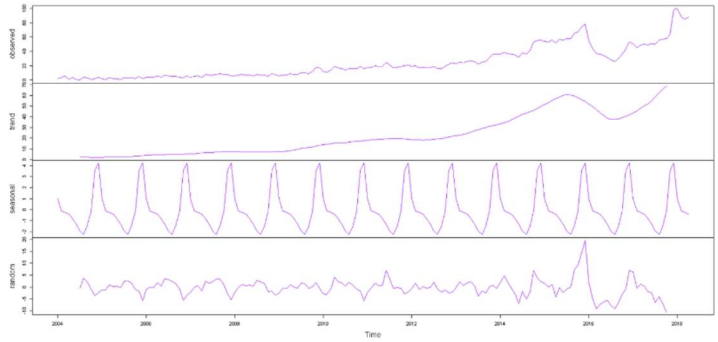

Figure 3: Decomposition of "Hey Google" searches in monthly time series from 2004-2018

3. The time series data up to 2016 for each were modeled using *HoltWinters/ets, Auto.arima, and Arima*. The "in-time" fit was evaluated for each data set and shown in Figures, 4, 5, 6.

   All the models were executed properly except, triple exponential smoothing – see example code/error below:
   TV_intime_forecasts<-HoltWinters(TVtrain).
   Error in decompose(ts(x[1L:wind], start = start(x), frequency = f), seasonal) : time series has no or less than 2 periods

I could not resolve the error. *Consequently, and as an alternative to triple exponential smoothing, I executed automated exponential smoothing using the ets function – this automatically assigns the least erroneous values to alpha, beta, and/or gamma coefficients.*

Figures, 4, 5, and 6 showing the *"in-time" fits* for each of the time series follow below:

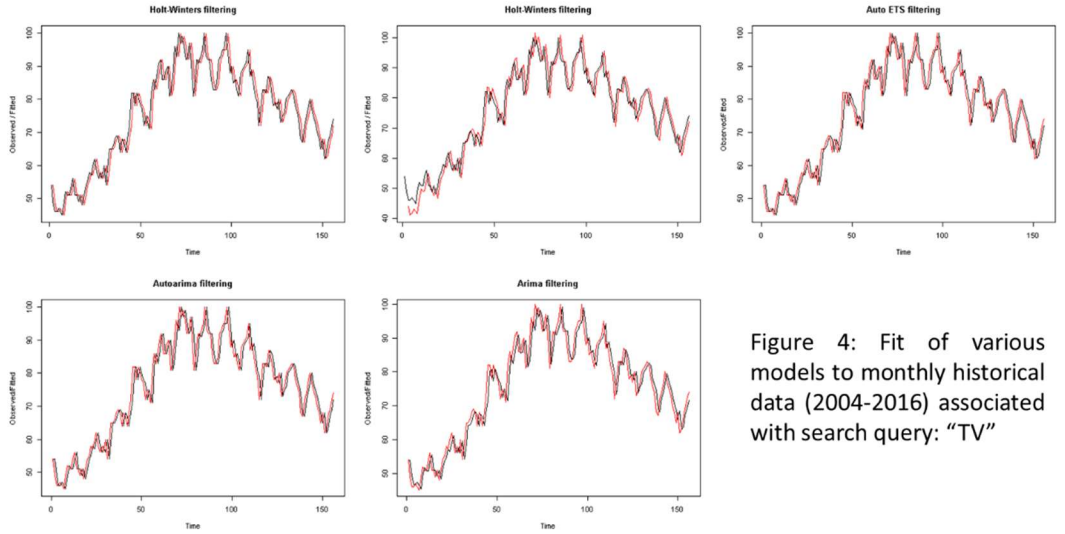

Figure 4: Fit of various models to monthly historical data (2004-2016) associated with search query: "TV"

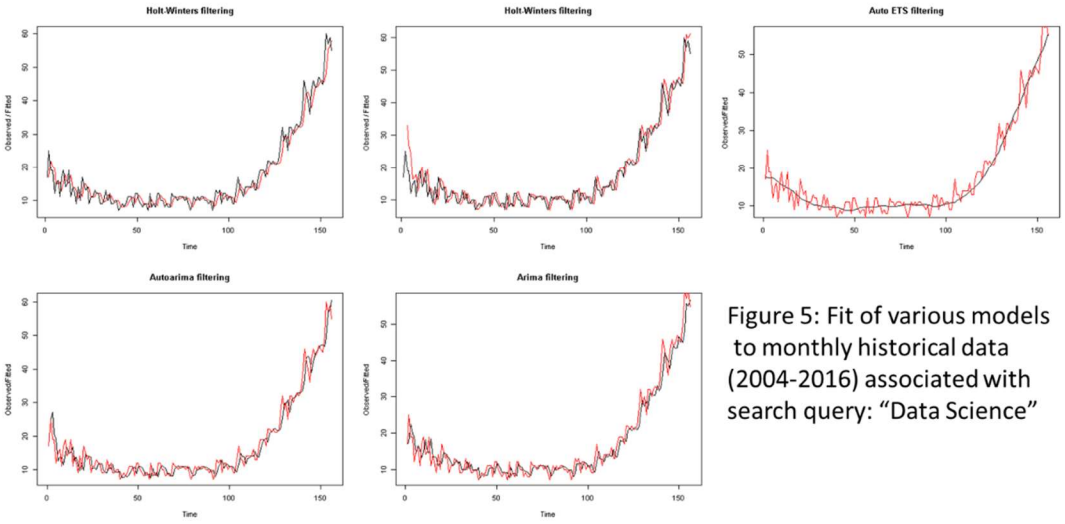

Figure 5: Fit of various models to monthly historical data (2004-2016) associated with search query: "Data Science"

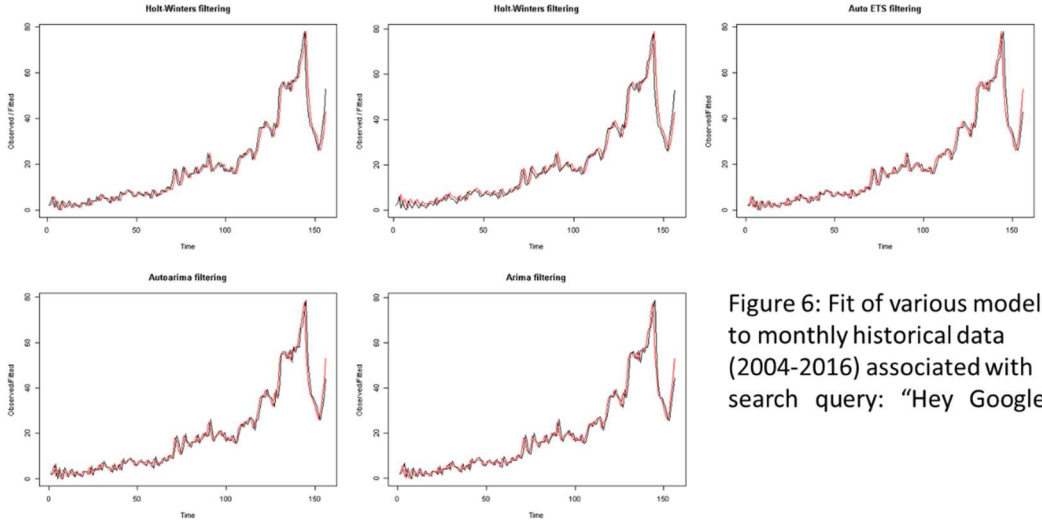

Figure 6: Fit of various models to monthly historical data (2004-2016) associated with search query: "Hey Google"

4. *Out-of-time forecasts* were generated for the next one year, and these results are shown in **Figures 7, 8,** and **9** for each of the time series. Their accuracy was evaluated by comparing "predicted" values against "actual" values for 2017; absolute errors were visualized in box plot and the mean absolute error (MAE) calculated. *The best model – with the lowest MAE - is indicated in figure legends and italicized.*

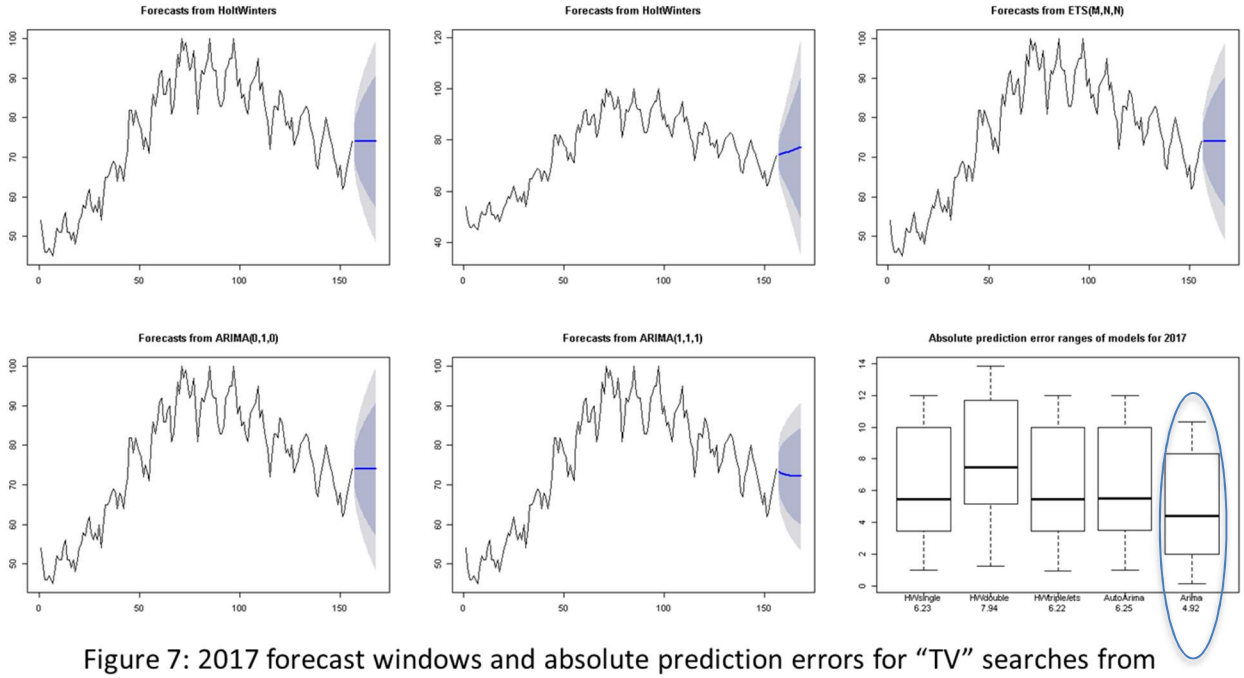

Figure 7: 2017 forecast windows and absolute prediction errors for "TV" searches from all models. Absolute prediction errors are shown as boxplots for each at bottom right panel, and the mean absolute error shown below each model. *Arima was the best model with a MAE of 4.92*

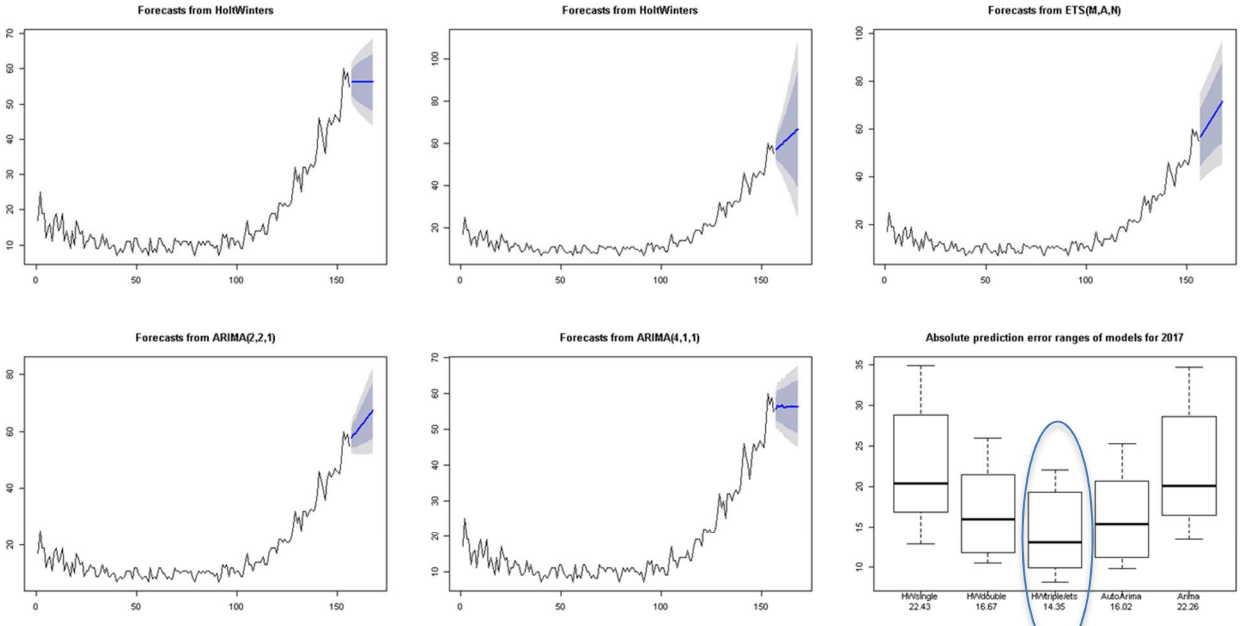

Figure 8: 2017 forecast windows and absolute prediction errors for "Data Science" searches from all models. Absolute prediction errors are shown as boxplots for each model at bottom right panel, and the mean absolute error shown below each model within boxplots image. *Auto ets was the best model with a MAE of 14.35*

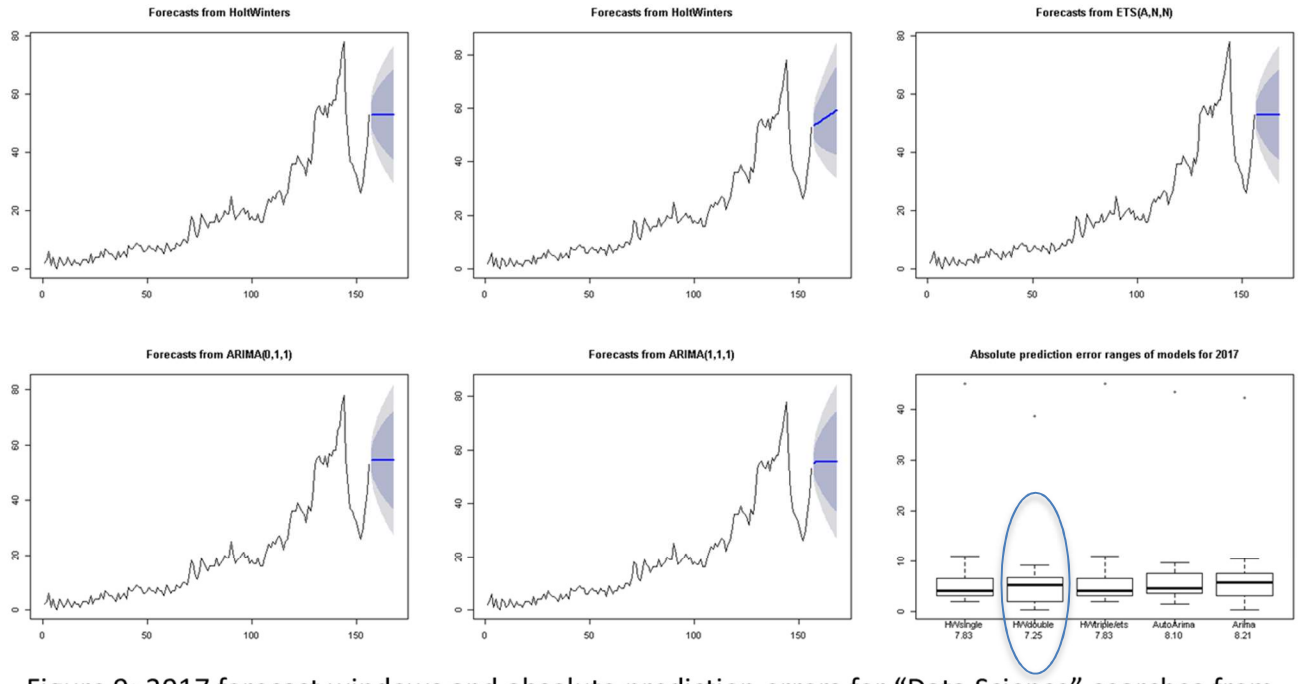

Figure 9: 2017 forecast windows and absolute prediction errors for "Data Science" searches from all models. Absolute prediction errors are shown as boxplots for each model at bottom right panel, and the mean absolute error shown below each model within boxplots image. *Holt Winters with double smoothing was the best model with a MAE of 7.25*

Task2: *Write another R script that:*

- *Using the same data of the previous script, plots the autocorrelation function for up to 20 lags for each time series. Remove the trend component if necessary to obtain useful information.*
- *Using the information from the correlograms, explain your choice of parameters for an ARIMA model.*
- *With your decision, fit an ARIMA model and calculate the mean absolute error of your model over the whole data set (that is you don't need to hold out part of the data as in the previous script)*
- *Fit an ARIMA model with auto.arima on the same data. Compare the resulting model complexity with your choice.*

Methods and Results:

The general method to identify "p", "d", and "q" values for ARIMA modeling of each dataset was as follows below:

1. Develop a stationary time series and determine number of differencings needed for stationary (order of "d").
    - For all data series, the order of differencing was set at "d"=1 based on visual check of effect of differencing and confirmation via Augmented Dickey Fuller test (adf.test) within the tseries library.
2. Generate an ACF and PACF plot of the differenced/stationary time series.
3. For selecting order of "p" (# of coefficients within auto-regressive component of ARIMA), inspect as below:
    - Sinusoidal nature of ACF plot. If not sinusoidal, set "p" = 0
    - If ACF plot is sinusoidal, check for # of highly significant spikes in PACF plot after which there is decay/inflection to non-significance. Choose this number as the order of "p"
        i. For the TV time series, using above steps, "p" was set to 1
        ii. For the DS time series, using above steps, "p" was set to 4
        iii. For the HG data series, using above steps, "p" was set to 1

4. For selecting order of "q" (# of coefficients within moving-average component of ARIMA), inspect as below:
    - Sinusoidal nature of PACF plot. If not sinusoidal, set "q" = 0
    - If PACF plot is sinusoidal, check for # of highly significant spikes in ACF plot after which there is decay/inflection to non-significance. Choose this number as the order of "q"
        i. For the TV time series, using above steps, "q" was set to 1
        ii. For the DS time series, using above steps, "q" was set to 1
        iii. For the HG time series, using above steps, "q" was set to 1

Arima modeling was performed for each of the time series (2004-2018) using above-determined values for "p", "q", and "d". The results of arima models was compared against that of an auto.arima model for the same data and the accuracy determined using *accuracy()* function within the forecast library.

Code used and **results/conclusion** for each time series are shown below for each time series; Mean Absolute Error (**MAE**) is shown in bold and underlined:

- **"TV" search queries time series modeling**

```
> TVFCAST_a<-arima(TVdataTS, order=c(1,1,1))
> (forecast::accuracy(TVFCAST_a))
                     ME     RMSE      MAE         MPE     MAPE      MASE         ACF1
Training set 0.07905014 3.404173 2.699635 0.03300783 3.627274 0.9992154 -0.01362238
>
> TVFCAST_aa<-auto.arima(TVdataTS)
> (forecast::accuracy(TVFCAST_aa))
                      ME     RMSE      MAE        MPE     MAPE     MASE         ACF1
Training set -0.2289648 2.409049 1.730098 -0.2838258 2.268403 0.318912 0.001086456
```

Conclusion: *Auto.arima outperforms arima based on lower MAE (and other measures of error)*

- **"data science" search queries time series modeling**

```
> DSFCAST_a<-arima(DSdataTS, order = c(4,1,1))
> (forecast::accuracy(DSFCAST_a))
                     ME     RMSE      MAE        MPE     MAPE      MASE         ACF1
Training set 0.4560334 3.194651 2.279403 -0.2498623 13.01355 0.9192875 -0.07928219

> DSFCAST_aa<-auto.arima(DSdataTS)
> (forecast::accuracy(DSFCAST_aa))
                     ME     RMSE      MAE        MPE     MAPE      MASE        ACF1
Training set 0.2351454 2.687672 1.895684 -0.3008715 11.63457 0.2808421 0.01674695
```

Conclusion: *Auto.arima outperforms arima based on lower MAE (and other measures of error)*

- **"hey google" search queries time series modeling**

```
> DSFCAST_a<-arima(DSdataTS, order = c(4,1,1))
> (forecast::accuracy(DSFCAST_a))
                     ME     RMSE      MAE        MPE     MAPE      MASE         ACF1
Training set 0.4560334 3.194651 2.279403 -0.2498623 13.01355 0.9192875 -0.07928219
>
> DSFCAST_aa<-auto.arima(DSdataTS)
> (forecast::accuracy(DSFCAST_aa))
                     ME     RMSE      MAE        MPE     MAPE      MASE        ACF1
Training set 0.2351454 2.687672 1.895684 -0.3008715 11.63457 0.2808421 0.01674695
```

Conclusion: *Auto.arima outperforms arima based on lower MAE (and other measures of error)*

Note: R Code for all tasks uploaded separately (see TimeSeries_Implementations_R.R)