144_project3

Krish Methi, Andrew Brown, Krithik J

3/7/2024

```
# Clear everything and load libraries
#rm(list=ls(all=TRUE))

# Load libraries
library(prophet)
```

```
## Loading required package: Rcpp

## Warning: package 'Rcpp' was built under R version 4.1.2

## Loading required package: rlang

## Warning: package 'rlang' was built under R version 4.1.2
```

```
library(strucchange)
```

```
## Warning: package 'strucchange' was built under R version 4.1.2

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.1.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.1.2

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.1.2
```

```
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.1.2
```

```
## -- Attaching packages --------------------------------------------- fpp3 0.5 --
```

```
## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.2     v tsibbledata 0.4.1
## v tidyr       1.2.1     v feasts      0.3.1
## v lubridate   1.8.0     v fable       0.3.3
## v ggplot2     3.4.4     v fabletools  0.3.4
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
## Warning: package 'tsibble' was built under R version 4.1.2
```

```
## Warning: package 'tsibbledata' was built under R version 4.1.2
```

```
## Warning: package 'feasts' was built under R version 4.1.2
```

```
## Warning: package 'fable' was built under R version 4.1.2
```

```
## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::index()     masks zoo::index()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```
library(tseries)
library(seasonal)
```

```
## Warning: package 'seasonal' was built under R version 4.1.2
```

```
##
## Attaching package: 'seasonal'
```

```
## The following object is masked from 'package:tibble':
##
##     view
```

```
library(fable)
library(stats)
require(graphics)
library(dplyr)
library(tsibble)
library(tsibbledata)
```

Part 1: Introduction

The data we are working with for this project is data on monthly imports of goods from January of 2010 to December of 2023 for Georgia. We picked Georgia as it is one of the more impactful states in the US in terms of economic productivity. The dataset came from the FRED database.

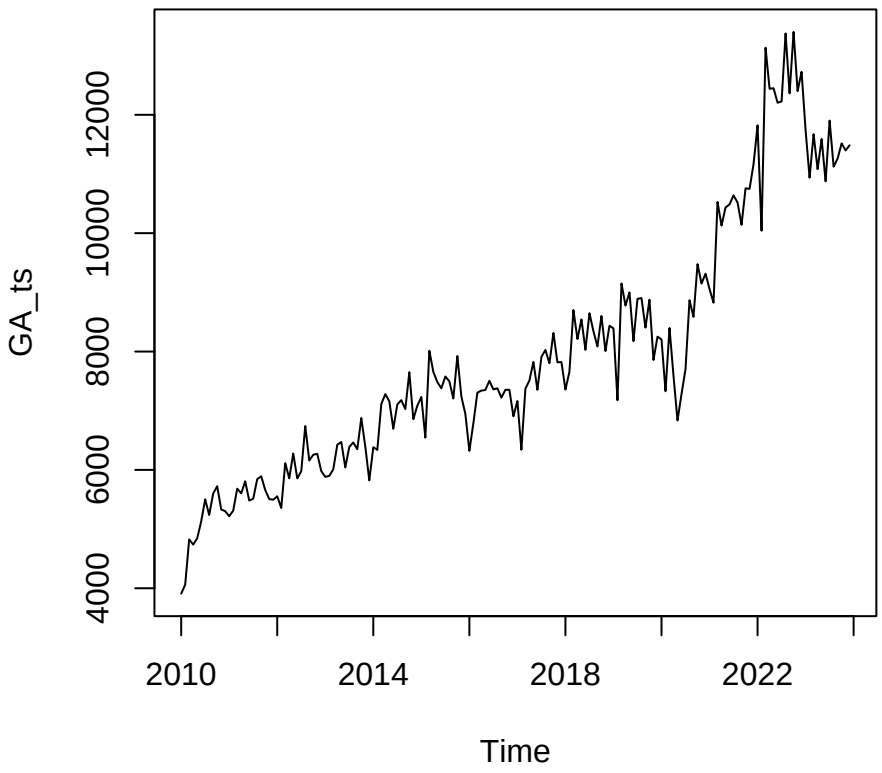
In the rest of our project, we model and forecast the Georgia monthly data with a number of different models, including an Arima model, ETS, Holt Winters, Prophet, NNETAR, and a combined model as well. Our end goal is comparing these model fits and forecasts and selecting a preferred model.

To evaluate the performance of the models, we did a training testing split of the data, and in terms of diagnostics, we utilized MAE, RMSE and MAPE

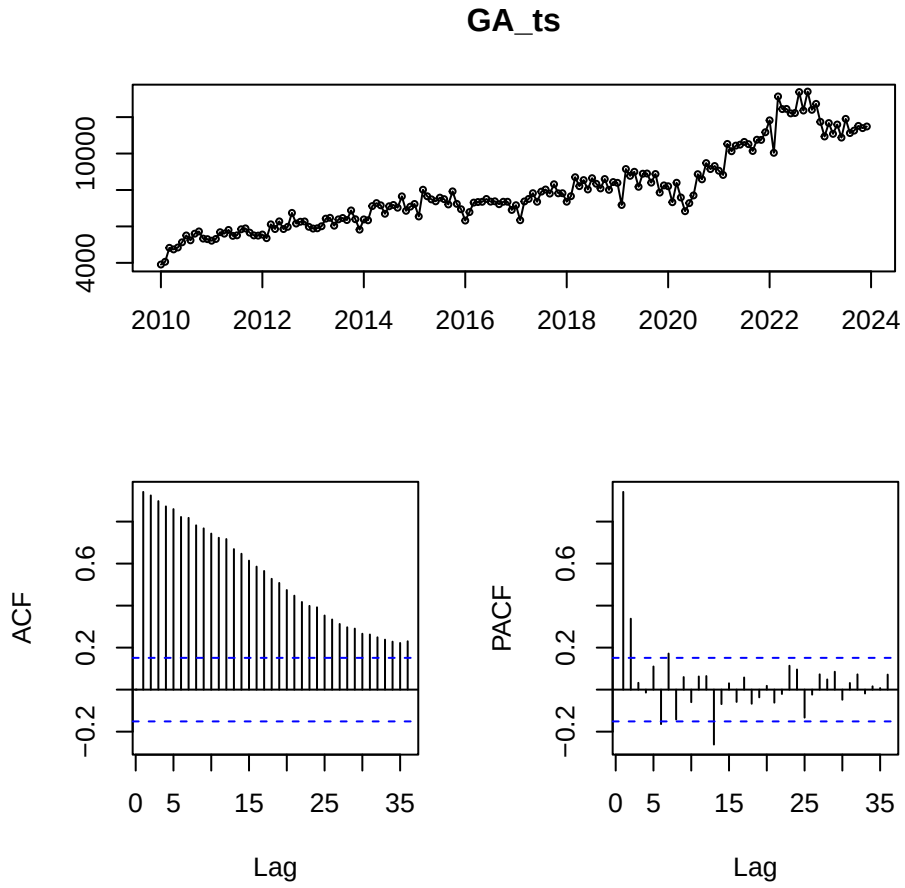
Part 2: Results

```
# read in data and create time series
data <- read.csv("/Users/kmethi2/Downloads/IMPTOTGA.csv")
GA_ts <- ts(data$IMPTOTGA, start = 2010, frequency = 12)
# Create the training and testing split
train_GA <- head(GA_ts, -60)
test_GA <- tail(GA_ts, 60)
```

```
# Look at the data with tsdisplay
plot(GA_ts)
```




```
tsdisplay(GA_ts)
```



Looking at the time series as a whole, we can see that by first looking at the plot, we see an upward trend, there seems to be some seasonality and non-constant variation, and also a potentially cyclical component as well. Now looking at the decay in the ACF and spikes in the PACF with the tsdisplay, this suggests an AR process as well for the data. There also looks to be spikes at 12 and 24, signaling a seasonal ARMA component could be present as well. In terms of an AR order, an AR(2) or an AR (13) or something like that seems likely.

Arima model

For our Arima model, we used auto.arima to automatically fit the best arima model to our data

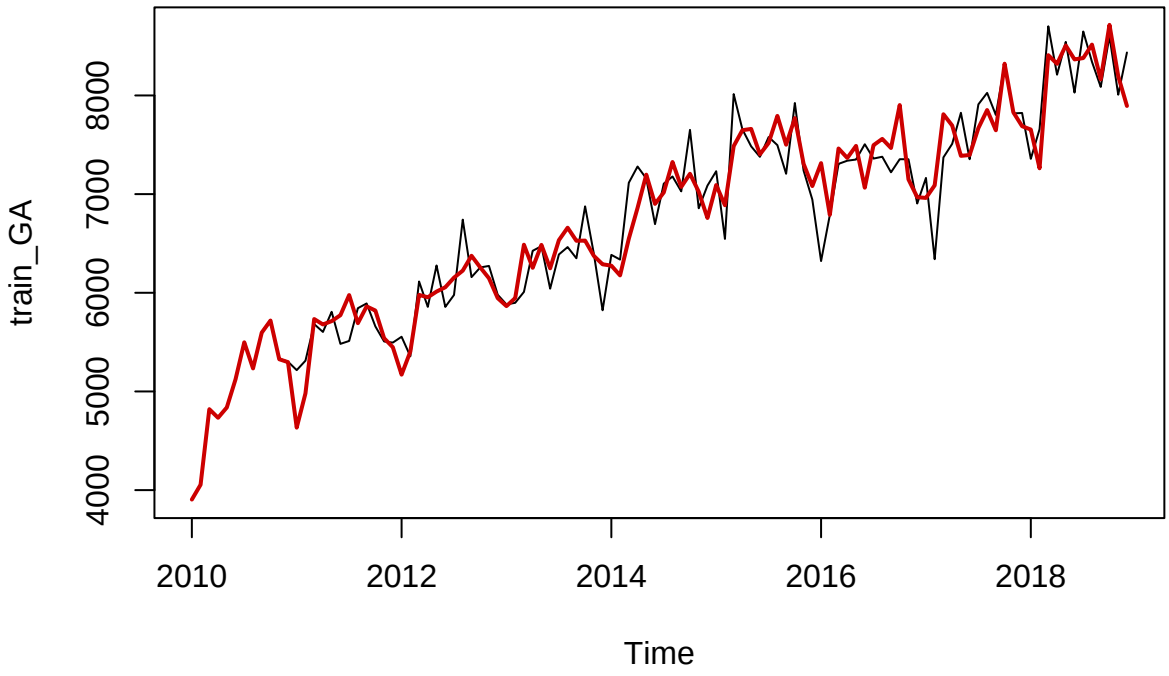
```
# Create the arima model with the training data
GA_arima <- auto.arima(train_GA)
GA_arima
```

```
## Series: train_GA
## ARIMA(3,0,0)(2,1,0)[12] with drift
##
## Coefficients:
##          ar1     ar2     ar3     sar1     sar2    drift
##       0.2310  0.2237  0.3816  -0.7112  -0.3822  34.4967
## s.e.  0.0987  0.1020  0.1023   0.1073   0.1115   7.1296
##
## sigma^2 = 83993:  log likelihood = -681.41
```

```
## AIC=1376.81   AICc=1378.08   BIC=1394.76
```

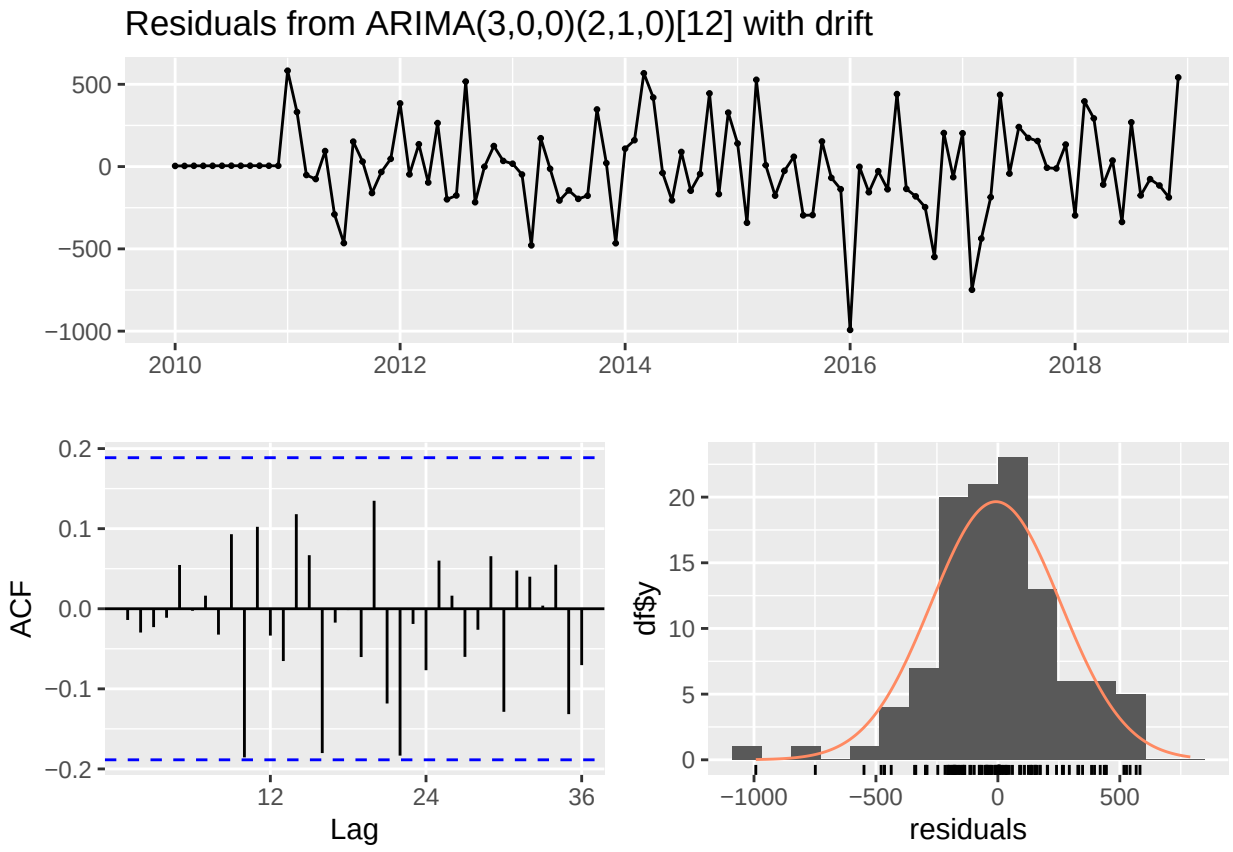
From the results, we can see that auto arima fit an ARIMA(3,0,0) to the data with a seasonal AR(2) after a differencing of order 1 with drift.

```
# Show fit of the model on the training data
plot(train_GA)
lines(GA_arima$fitted,col="red3",lwd=2)
```



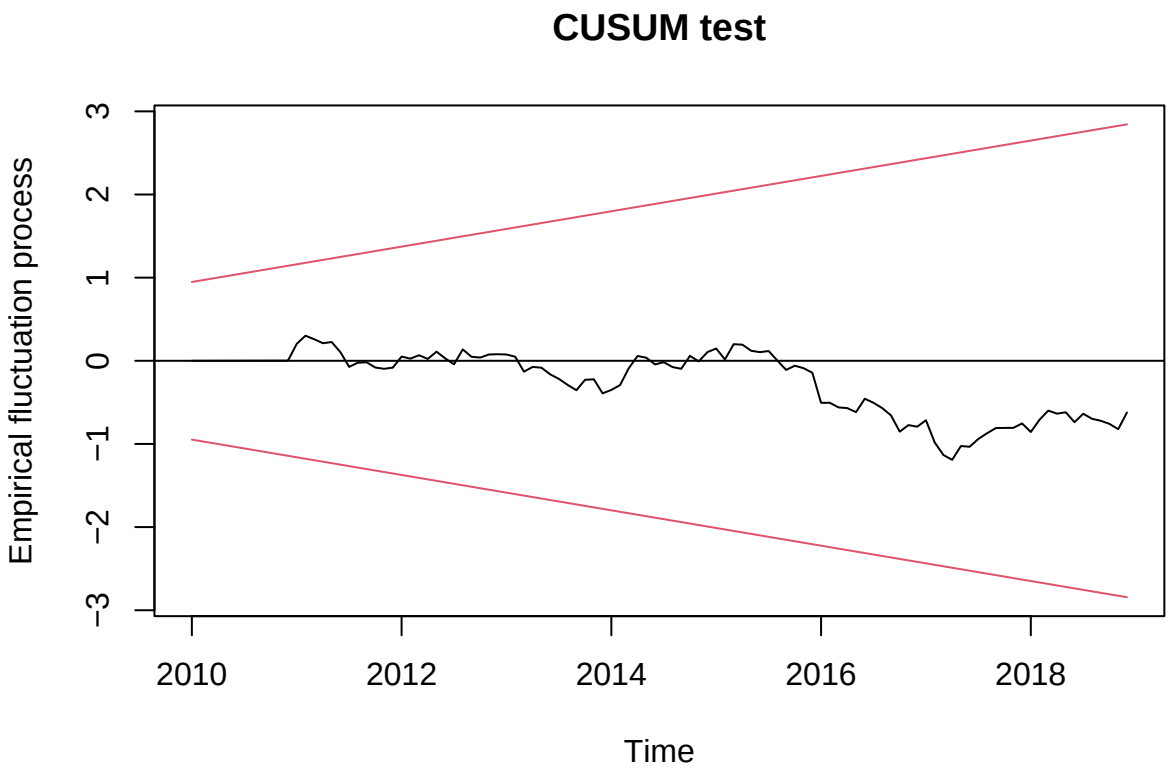
Looking at the model fit to the training data, the model seems to do OK, not too bad however there are some spikes that were not captured well.

```
# Assess the model validity of the Arima model by running check residuals, and running a cusum test on
checkresiduals(GA_arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,0)(2,1,0)[12] with drift
## Q* = 23.926, df = 17, p-value = 0.1215
##
## Model df: 5.   Total lags used: 22
```

```
plot(efp(GA_arima$residuals ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```




```
# Get the MAE, RMSE and ME values for the Arima model on the training data
cat("--- Training Dataset ---")
```

```
## --- Training Dataset ---
```

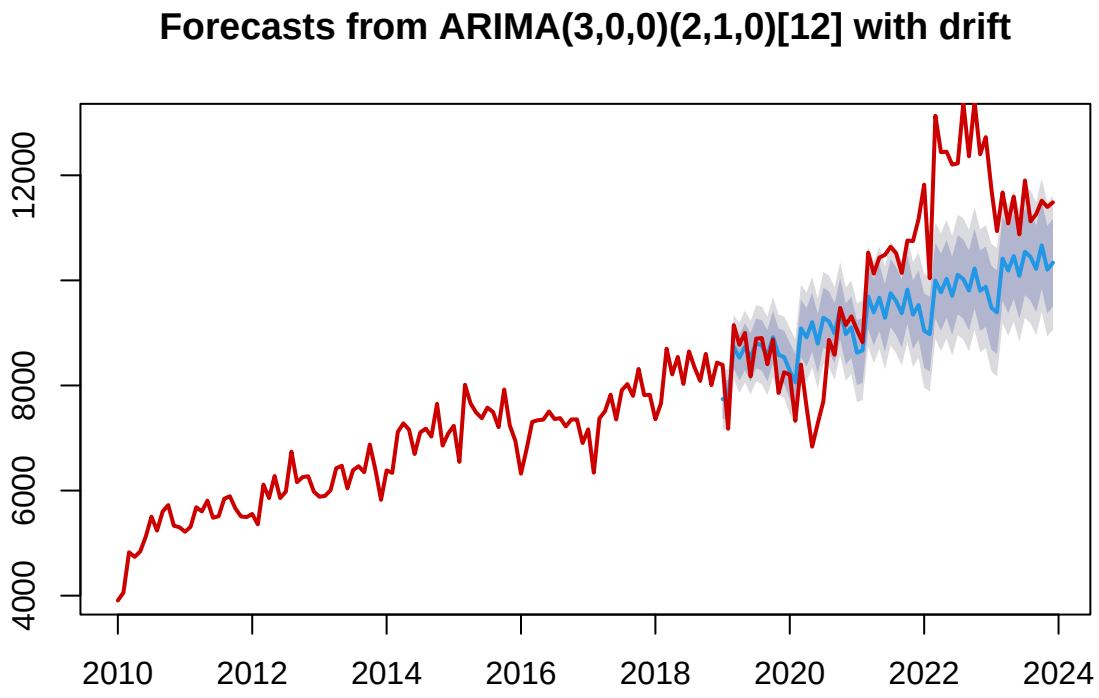
```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = GA_arima$residuals, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = GA_arima$residuals, .actual = train_GA),
      "\nME:   ", ME(.resid = GA_arima$residuals, .actual = train_GA))
```

```
##
## MAE:  190.3075
## RMSE: 264.5634
## ME:   -7.987067
```

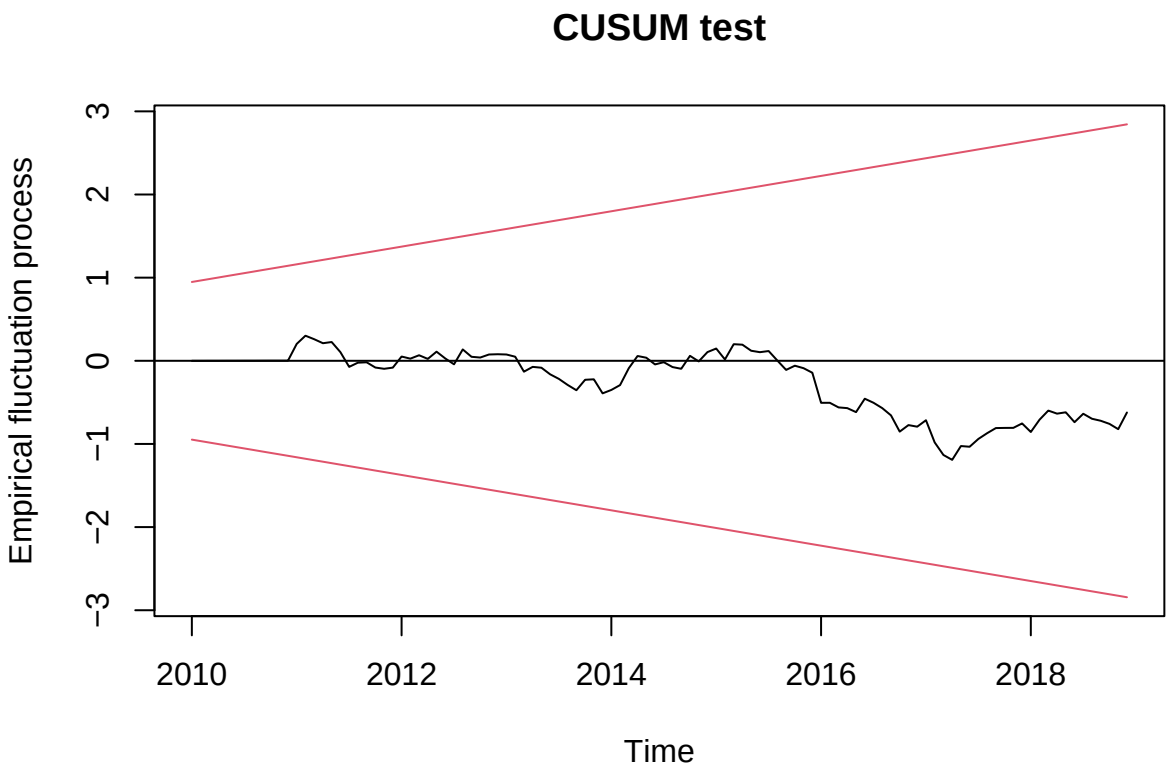
Based on the MAE and RMSE, it seems as though the arima model does reasonably well with the data, compared to the actual data, the values are relatively small. Based on the ME, we can see the model overestimated the data as a whole slightly. We anticipate this model will do adequate with the test data, but we anticipate there will be models that can do better than the arima model. Based on the Ljung Box test, the high P value means we fail to reject the null of there being no serial correlation, and therefore we can say the model is effective overall in accounting for the dynamics of the data.

Looking at the Cusum plot, we can see there are no structural breaks in the model

```r
# Forecast the arima model based on the testing data
GA_arima_forecast <- forecast(GA_arima, h = 60)
plot(GA_arima_forecast, ylim = c(4000, 13000))
lines(GA_ts,col="red3",lwd=2)
```




```r
# Plot cusum plot
plot(efp(GA_arima_forecast$residuals ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```



Based on the quick glance at the plot of the forecast with the original data overlayed, we can see that the model does not seem to do well with the test data, as it does not do well with the erratic nature that we see from years 2019-24 in the data.

Looking at the Cusum plot for the forecast, also there are no structural breaks.

```
# Extract the residuals of the forecast
arima_resid <- test_GA - GA_arima_forecast$mean
arima_forecast_resid <- ts(arima_resid, start = 2019, frequency = 12)
cat("--- Testing Dataset ---")
```

```
## --- Testing Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = arima_forecast_resid, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = arima_forecast_resid, .actual = test_GA),
      "\nME:   ", ME(.resid = arima_forecast_resid, .actual = test_GA))
```

```
##
## MAE:  1156.509
## RMSE: 1473.696
## ME:   784.1569
```

Looking at the diagnostic statistics, we can see how much more the MAE and RMSE values are compared to the training data. They skyrocket for the testing data, with an MAE of 1156 and an RMSE of 1473, showing how poorly the model performs with the forecast, although it must be acknowledged the erratic nature of the testing data compared to the relatively smooth training data.

ETS Model

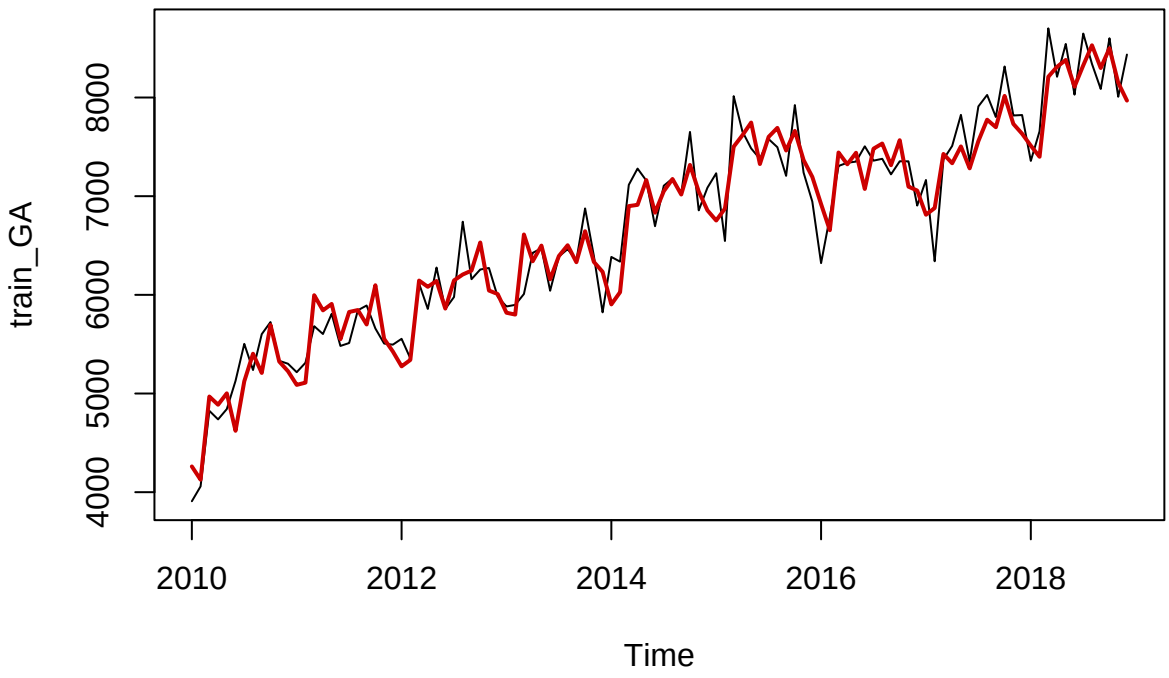
The next model we will fit to the data is an ETS model, where we will do a box cox transformation, and then the algorithm will fit the best ETS model to the data.

```
# Fit the ets model and look at the summary
ets_model <- ets(train_GA, lambda = "auto")
ets_model
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = train_GA, lambda = "auto")
##
##   Box-Cox transformation: lambda= 1.2255
##
##   Smoothing parameters:
##     alpha = 0.3532
##     beta  = 7e-04
##     gamma = 1e-04
##     phi   = 0.98
##
##   Initial states:
##     l = 25610.7073
##     b = 448.852
##     s = -1687.163 -639.5167 2379.061 328.5313 1645.878 1026.138
##            -775.7532 1801.908 1065.674 1695.176 -3669.509 -3170.423
##
##   sigma:  1974.608
##
##      AIC     AICc      BIC
## 2162.208 2169.893 2210.486
```

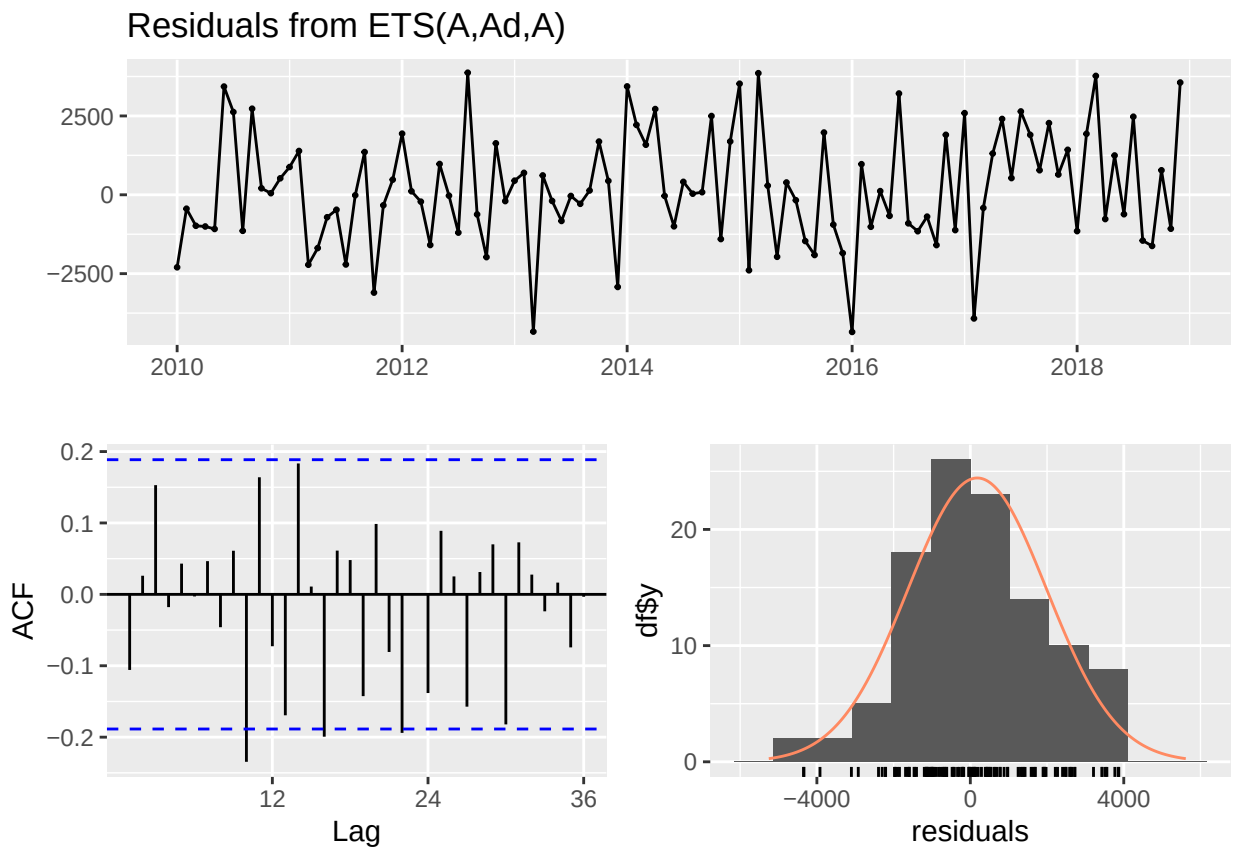
We can see that the algorithm fitted an ETS(A, Ad, A) model to the training data, with a Box Cox transformation lambda of 1.2255.

```
# Show fit of the ETS model on the training data
plot(train_GA)
lines(ets_model$fitted,col="red3",lwd=2)
```



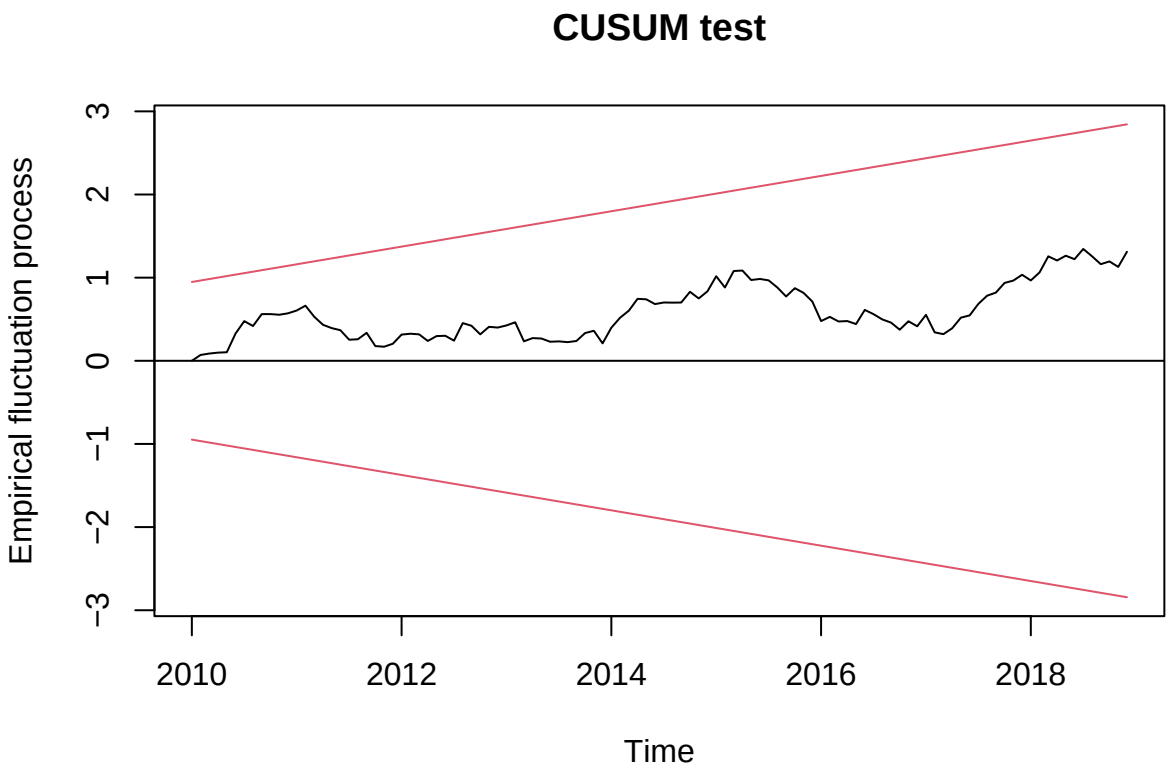
Here we can see the fit of the model on the training data, which looks pretty good and seems to do better than the arima model in terms of the spikes in the data.

```
# Assess the model validity of the ets model
checkresiduals(ets_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,A)
## Q* = 39.678, df = 22, p-value = 0.01179
##
## Model df: 0.   Total lags used: 22
```

```
plot(efp(ets_model$residuals ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```



Looking at the residuals, we can see the ACF looks fairly clean, although there are a number of lags that get close to the threshold, and the lag at around 11 clears the threshold by a little. The time plot also seems to suggest a slight pattern, slightly concerning. The ljung box test gives a p value of 0.01 and so we would reject the null at the 5% level and say that there is evidence of serial correlation in the residuals. Therefore it seems as though the ETS model doesn't do an adequate job of capturing all the dynamics.

Looking at the Cusum plot for the model, there are no structural breaks.

```
# Get the MAE, RMSE and ME values for the ETS model on the training data
cat("--- Training Dataset ---")
```

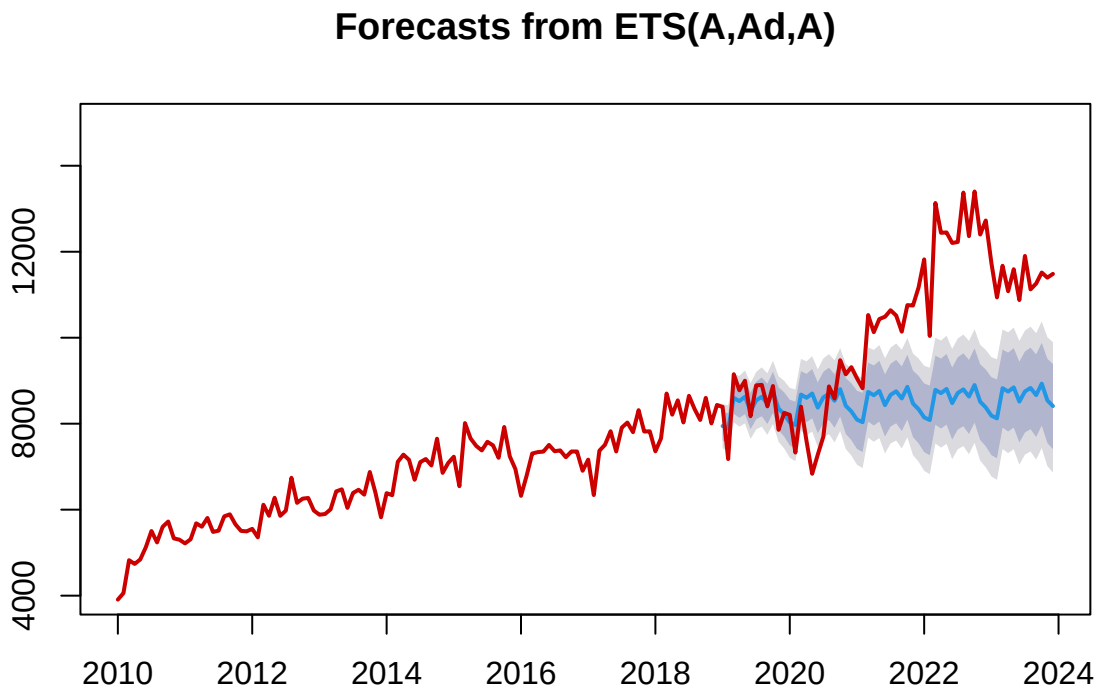
```
## --- Training Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = ets_model$residuals, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = ets_model$residuals, .actual = train_GA),
      "\nME:   ", ME(.resid = ets_model$residuals, .actual = train_GA))
```

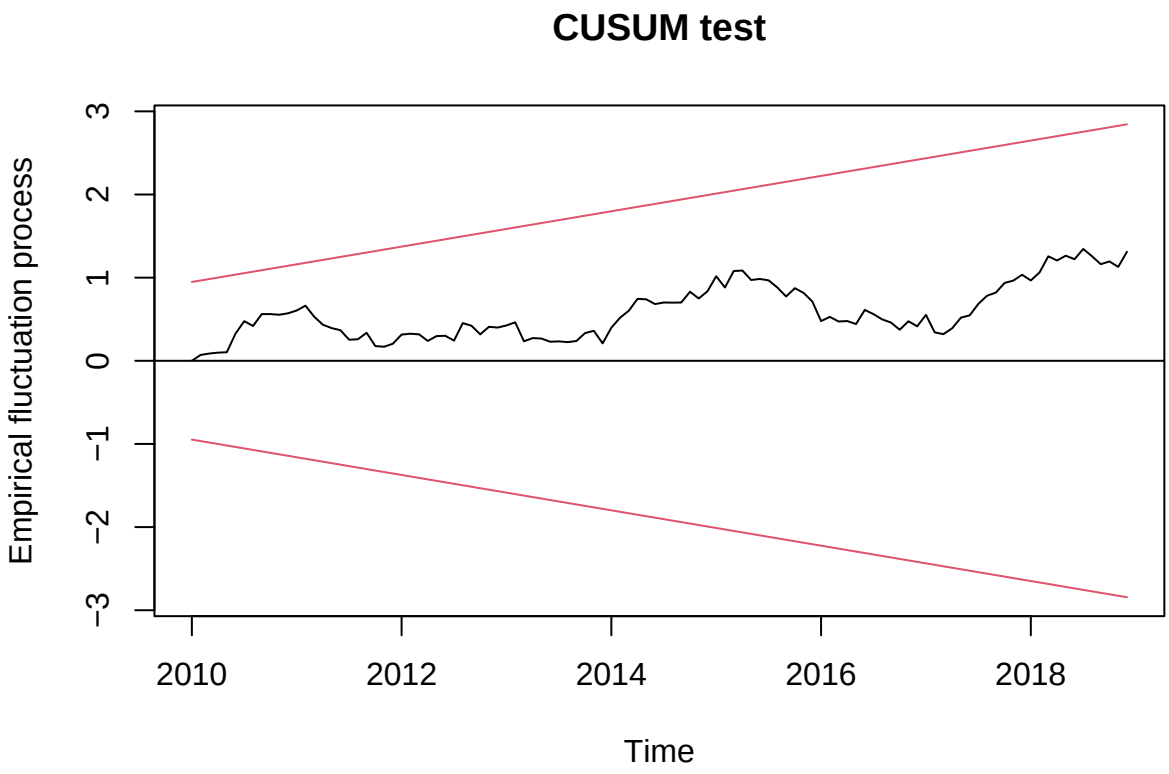
```
##
## MAE:  1436.788
## RMSE: 1812.549
## ME:   181.3225
```

Furthermore, looking at the diagnostic statistics, the MAE and RMSE are extremely high, at 1436 and 1812, and we have a mean error of 181. This further shows us that this model isn't adequate for our data, and most likely will not produce a great forecast.

```r
# Forecast the ets model based on the testing data
GA_ets_forecast <- forecast(ets_model, h = 60)
plot(GA_ets_forecast, ylim = c(4000, 15000))
lines(GA_ts,col="red3",lwd=2)
```




```r
plot(efp(GA_ets_forecast$residuals ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```



From this plot, we can see how poorly the ets model does with its forecast, even with the erratic nature of the test data.

Looking at the Cusum plot for the model, there are no structural breaks.

```
# Extract the residuals from the ets forecast on the testing data
ets_resid <- test_GA - GA_ets_forecast$mean
ets_forecast_resid <- ts(ets_resid, start = 2019, frequency = 12)
cat("--- Testing Dataset ---")
```

```
## --- Testing Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = ets_forecast_resid, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = ets_forecast_resid, .actual = test_GA),
      "\nME:   ", ME(.resid = ets_forecast_resid, .actual = test_GA))
```

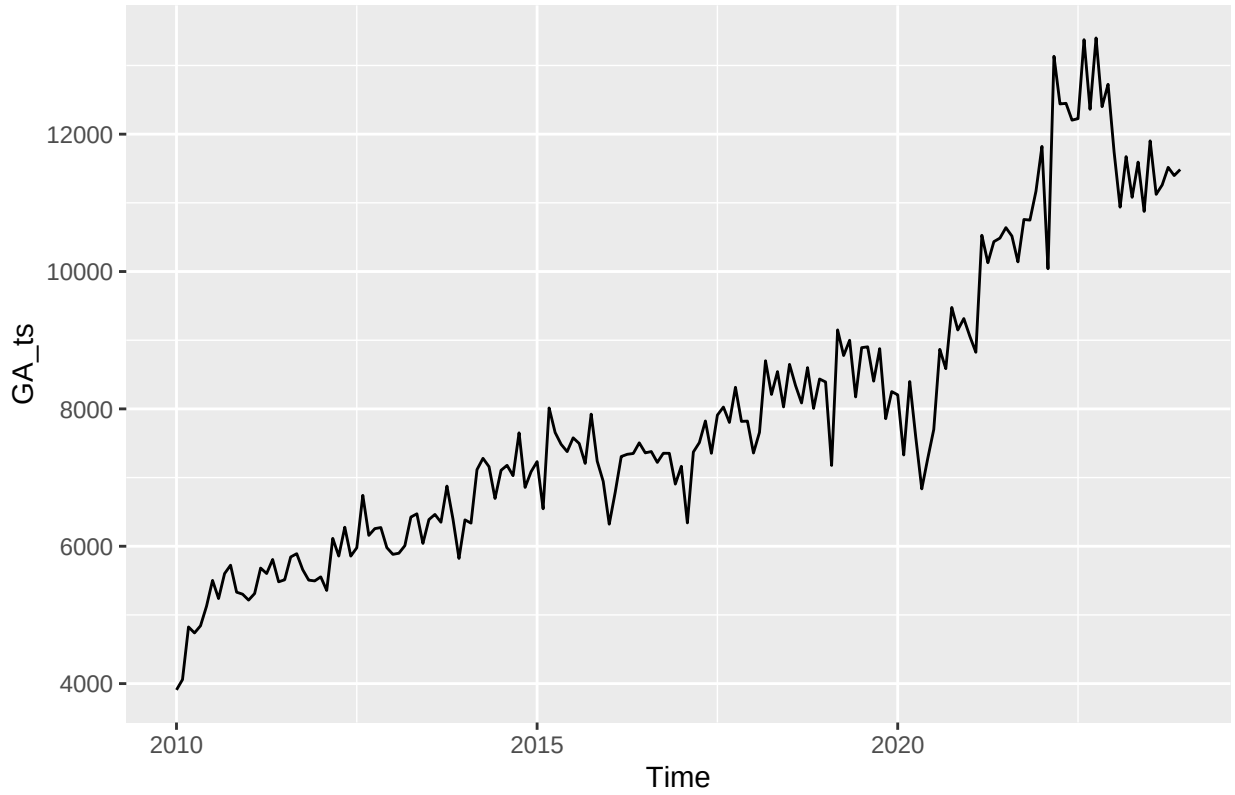
```
##
## MAE:  1872.065
## RMSE: 2320.95
## ME:   1632.264
```

Looking at the diagnostics from the forecast, we can see that similar to the testing data, the MAE and RMSE are very high for the forecast, and we see a massive increasing in the ME, from 181 to 1632 from the training to the testing data.

Holt-Winters model

The next model we will try out is the Holt-Winters model. Within the Holt-Winters model, we should start with the Holt's linear trend method (no Winters). This method allows for forecasting data with a trend.

```
autoplot(GA_ts)
```



From a quick graph, we see that there is a trend and possibly some seasonality (we will address the seasonality later).

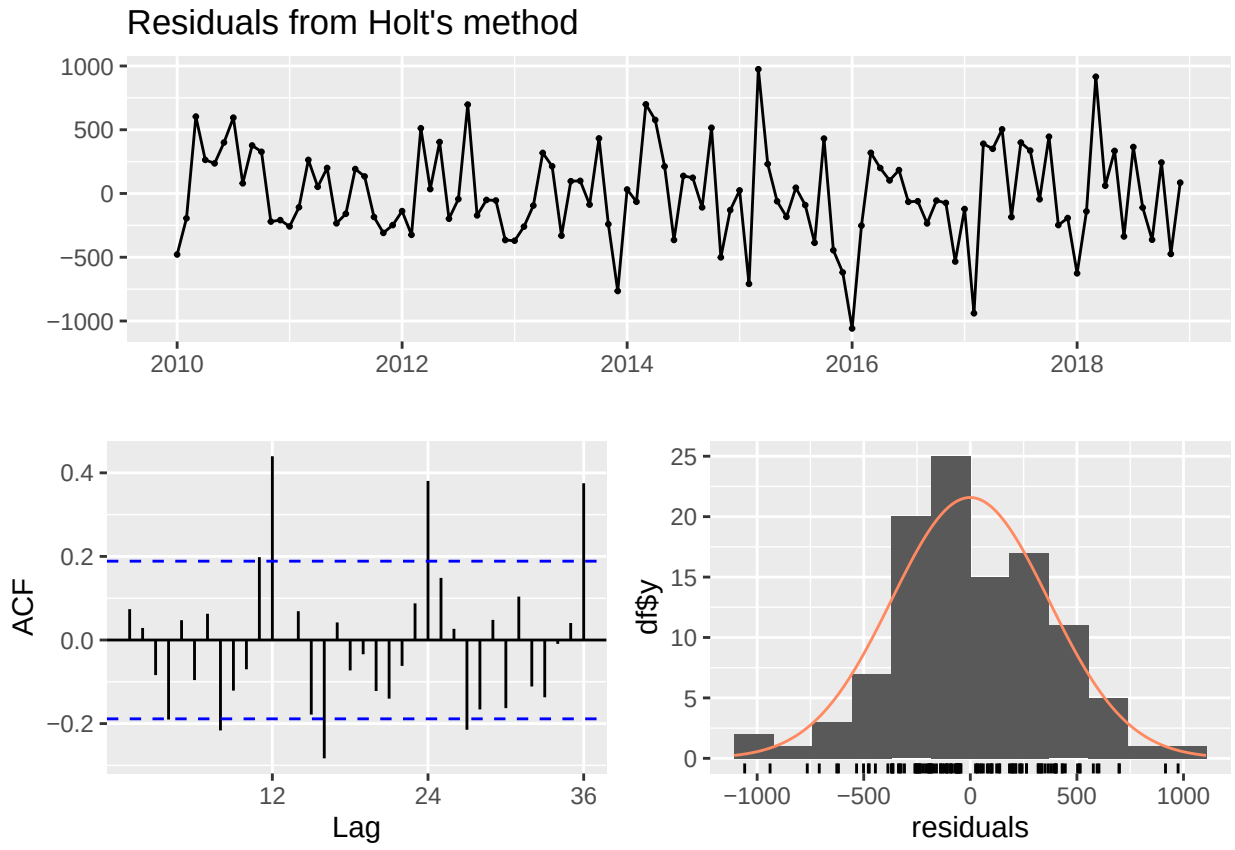
```
# fit the holt wintes model and check the residuals for model validity
ga_holt <- holt(train_GA, h = length(test_GA))
summary(ga_holt)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = train_GA, h = length(test_GA))
##
##   Smoothing parameters:
##     alpha = 0.3568
##     beta  = 1e-04
```

```
##
##   Initial states:
##     l = 4348.9347
##     b = 37.6135
##
##   sigma:  374.2423
##
##      AIC     AICc      BIC
## 1791.373 1791.962 1804.784
##
## Error measures:
##                     ME     RMSE     MAE        MPE     MAPE      MASE       ACF1
## Training set -1.0254 367.2465 293.022 -0.1805497 4.471893 0.5967083 0.07380614
##
## Forecasts:
##          Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
## Jan 2019       8418.153 7938.542  8897.763 7684.651  9151.654
## Feb 2019       8455.755 7946.513  8964.997 7676.937  9234.573
## Mar 2019       8493.357 7956.101  9030.614 7671.694  9315.021
## Apr 2019       8530.960 7967.064  9094.856 7668.556  9393.364
## May 2019       8568.562 7979.216  9157.908 7667.235  9469.889
## Jun 2019       8606.165 7992.410  9219.920 7667.507  9544.822
## Jul 2019       8643.767 8006.524  9281.010 7669.189  9618.346
## Aug 2019       8681.370 8021.462  9341.277 7672.128  9690.611
## Sep 2019       8718.972 8037.141  9400.803 7676.201  9761.743
## Oct 2019       8756.574 8053.491  9459.658 7681.301  9831.847
## Nov 2019       8794.177 8070.454  9517.900 7687.338  9901.016
## Dec 2019       8831.779 8087.978  9575.580 7694.233  9969.325
## Jan 2020       8869.382 8106.019  9632.744 7701.920 10036.844
## Feb 2020       8906.984 8124.539  9689.429 7710.338 10103.631
## Mar 2020       8944.587 8143.503  9745.670 7719.435 10169.738
## Apr 2020       8982.189 8162.881  9801.497 7729.165 10235.213
## May 2020       9019.791 8182.645  9856.938 7739.487 10300.096
## Jun 2020       9057.394 8202.772  9912.015 7750.363 10364.425
## Jul 2020       9094.996 8223.240  9966.752 7761.760 10428.232
## Aug 2020       9132.599 8244.029 10021.168 7773.649 10491.549
## Sep 2020       9170.201 8265.121 10075.281 7786.001 10554.401
## Oct 2020       9207.804 8286.500 10129.107 7798.792 10616.816
## Nov 2020       9245.406 8308.151 10182.661 7811.998 10678.814
## Dec 2020       9283.008 8330.060 10235.956 7825.600 10740.417
## Jan 2021       9320.611 8352.215 10289.006 7839.578 10801.644
## Feb 2021       9358.213 8374.605 10341.822 7853.914 10862.513
## Mar 2021       9395.816 8397.217 10394.414 7868.591 10923.040
## Apr 2021       9433.418 8420.043 10446.793 7883.595 10983.241
## May 2021       9471.021 8443.074 10498.967 7898.912 11043.129
## Jun 2021       9508.623 8466.300 10550.946 7914.528 11102.718
## Jul 2021       9546.225 8489.714 10602.736 7930.431 11162.020
## Aug 2021       9583.828 8513.309 10654.347 7946.610 11221.046
## Sep 2021       9621.430 8537.076 10705.784 7963.054 11279.806
## Oct 2021       9659.033 8561.011 10757.055 7979.753 11338.312
## Nov 2021       9696.635 8585.106 10808.164 7996.698 11396.572
## Dec 2021       9734.238 8609.356 10859.119 8013.879 11454.596
## Jan 2022       9771.840 8633.755 10909.925 8031.289 11512.391
## Feb 2022       9809.442 8658.298 10960.586 8048.920 11569.965
```

```
## Mar 2022       9847.045 8682.981 11011.108 8066.763 11627.326
## Apr 2022       9884.647 8707.799 11061.496 8084.813 11684.482
## May 2022       9922.250 8732.747 11111.753 8103.062 11741.437
## Jun 2022       9959.852 8757.821 11161.883 8121.504 11798.200
## Jul 2022       9997.454 8783.018 11211.891 8140.134 11854.775
## Aug 2022      10035.057 8808.333 11261.781 8158.945 11911.169
## Sep 2022      10072.659 8833.764 11311.555 8177.932 11967.387
## Oct 2022      10110.262 8859.306 11361.217 8197.090 12023.433
## Nov 2022      10147.864 8884.957 11410.771 8216.414 12079.314
## Dec 2022      10185.467 8910.713 11460.220 8235.900 12135.033
## Jan 2023      10223.069 8936.573 11509.566 8255.542 12190.596
## Feb 2023      10260.671 8962.531 11558.812 8275.338 12246.005
## Mar 2023      10298.274 8988.587 11607.960 8295.281 12301.267
## Apr 2023      10335.876 9014.738 11657.015 8315.370 12356.383
## May 2023      10373.479 9040.981 11705.977 8335.599 12411.358
## Jun 2023      10411.081 9067.313 11754.849 8355.966 12466.197
## Jul 2023      10448.684 9093.734 11803.634 8376.467 12520.901
## Aug 2023      10486.286 9120.239 11852.333 8397.098 12575.474
## Sep 2023      10523.888 9146.828 11900.949 8417.857 12629.920
## Oct 2023      10561.491 9173.499 11949.483 8438.740 12684.242
## Nov 2023      10599.093 9200.249 11997.938 8459.745 12738.442
## Dec 2023      10636.696 9227.076 12046.315 8480.869 12792.523
```

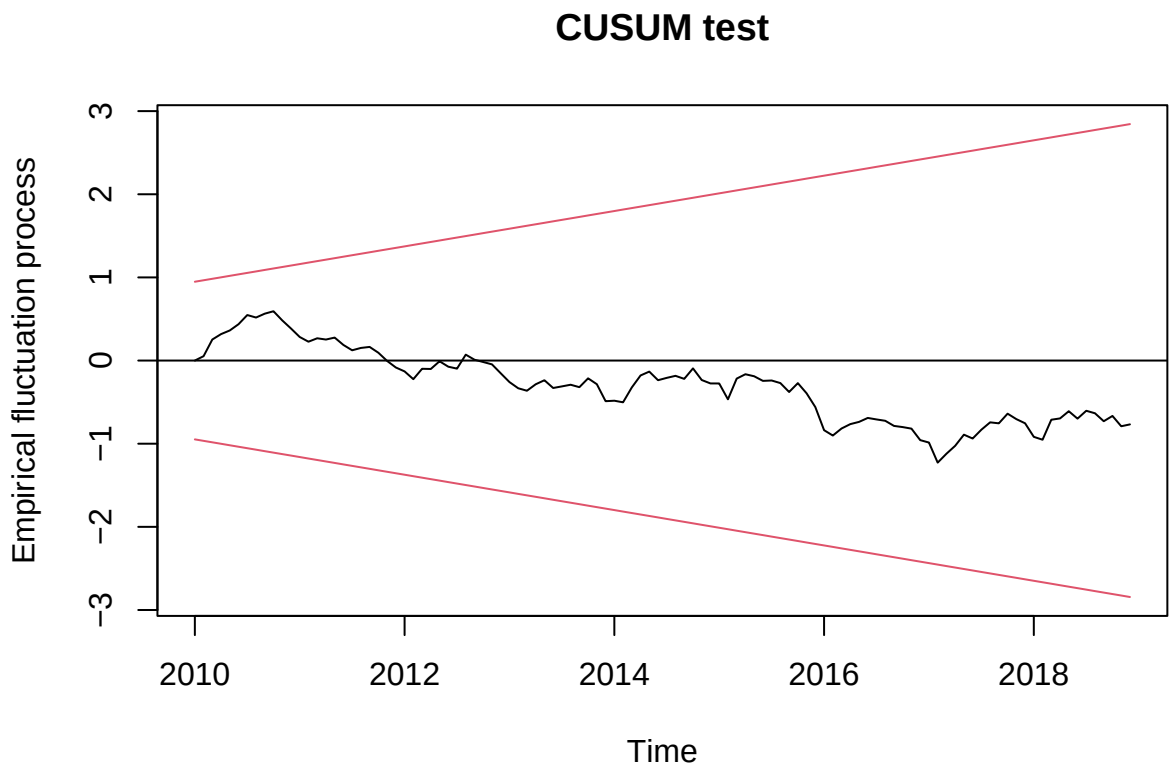
```
checkresiduals(ga_holt)
```




```
##
```

```
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 65.407, df = 22, p-value = 3.422e-06
##
## Model df: 0.   Total lags used: 22
```

```
plot(efp(ga_holt$residuals ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```



When looking at the residuals of the Holt model, we see promising results, but there is still a lot to be desired. For one, we see significant spikes at 12, 24, and 36. This implies that there is seasonality, which we will move on to later. The fact that the null hypothesis of the Ljung-Box test got rejected implies there is further work to do.

Looking at the cusum plot, there are no structural breaks in the model.

```
#Extract the residuals of the model fit on the training data and testing data as well
holt_resid_train <- train_GA - ga_holt$fitted
cat("--- Training Dataset ---")
```

```
## --- Training Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = holt_resid_train, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = holt_resid_train, .actual = train_GA),
      "\nME:   ", ME(.resid = holt_resid_train, .actual = train_GA))
```

```
##
## MAE:  293.022
## RMSE: 367.2465
## ME:   -1.0254
```

```
holt_resid <- test_GA - ga_holt$mean
holt_forecast_resid <- ts(holt_resid, start = 2019, frequency = 12)
cat("--- Testing Dataset ---")
```

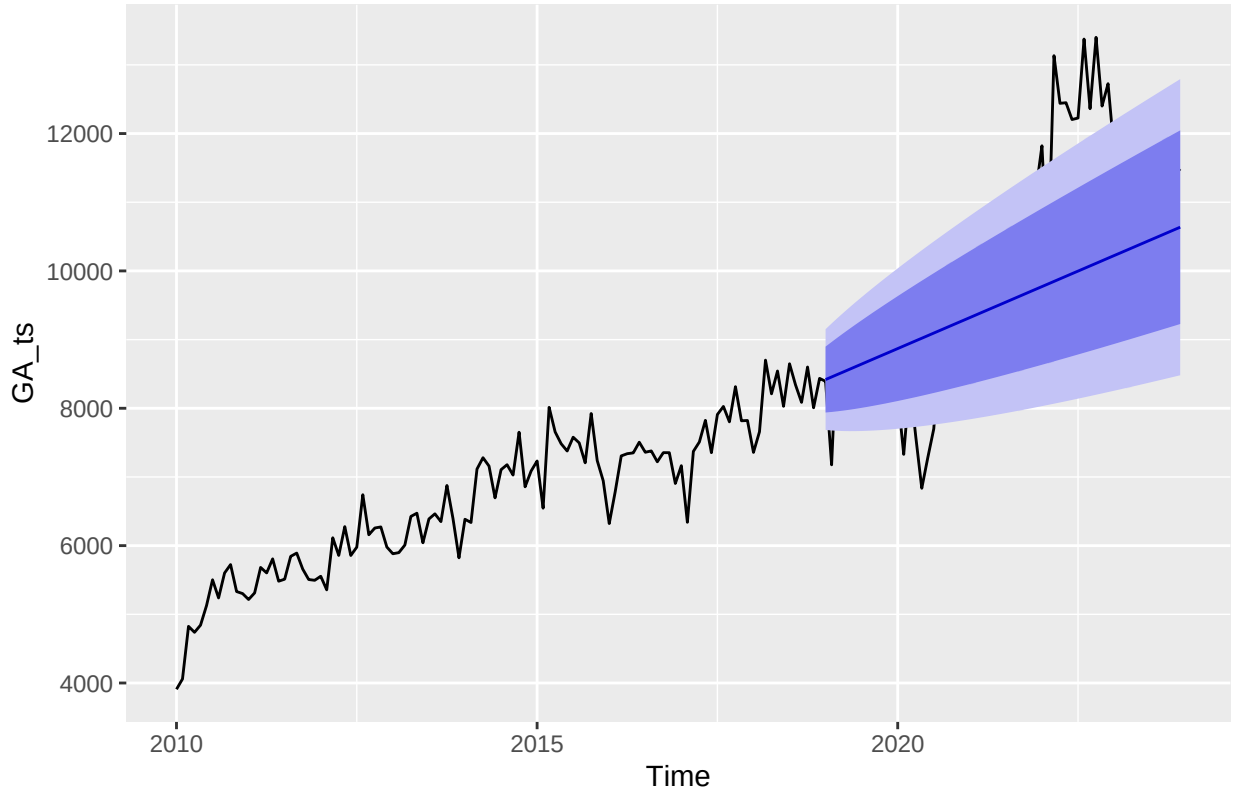
```
## --- Testing Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = holt_forecast_resid, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = holt_forecast_resid, .actual = test_GA),
      "\nME:   ", ME(.resid = holt_forecast_resid, .actual = test_GA))
```

```
##
## MAE:  1120.636
## RMSE: 1409.286
## ME:   625.5859
```

We get pretty high errors, but we can hopefully reduce this when we add in seasonality. It is important to note that the RMSE for the training is much lower than the testing, so it should be noted that if the testing data followed a similar path to the training data, we could get a decent forecast out of this. However, that is a pretty unrealistic assumption.

```
# Look at the forecast fit with the original data
autoplot(GA_ts) +
  autolayer(ga_holt)
```



When looking at the graph, the model doesn't do a great job, but at the same time, it is important to note that after the training segment cuts off, the graph in the original data does take a different path. Before we move on to seasonality, we can also try "dampening" the model. This will create a flatter line to avoid overforecasting.

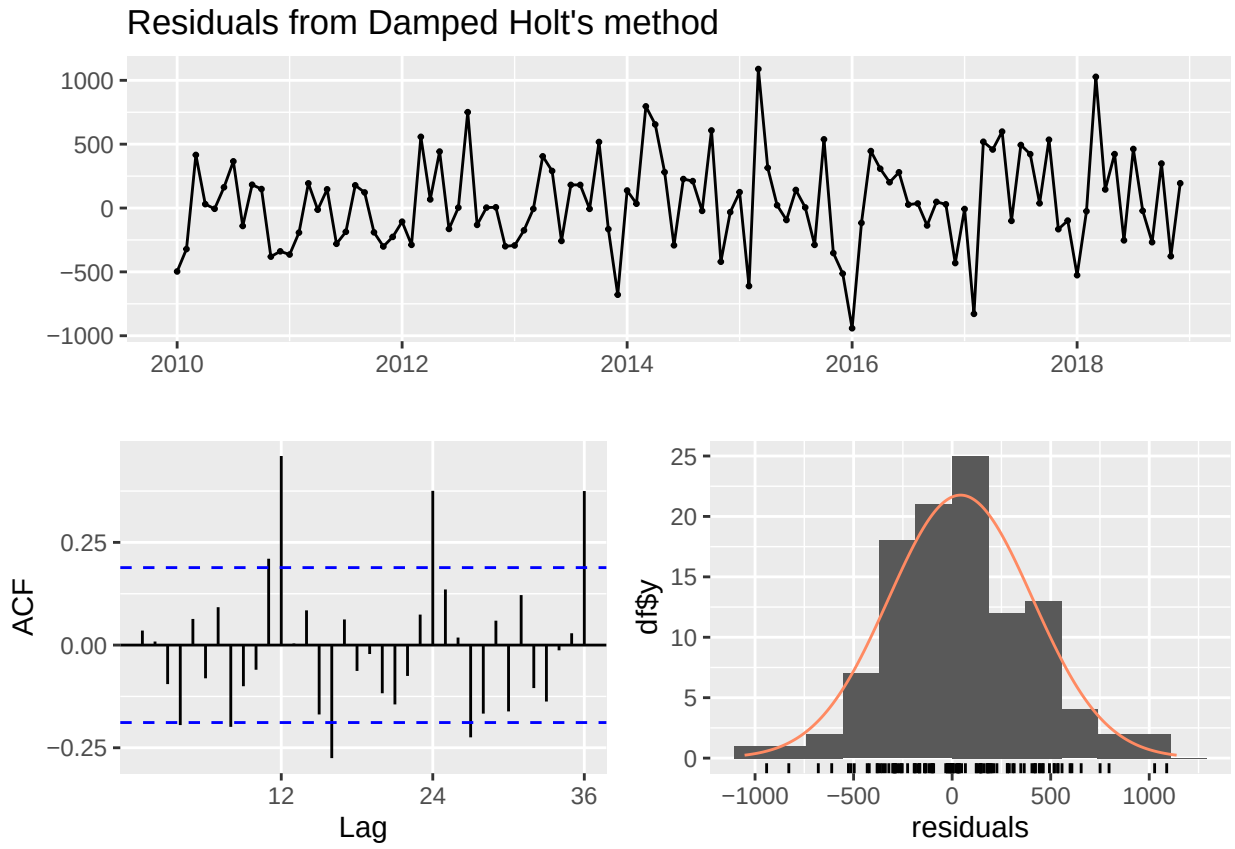
```
# Fit the damped model and extract residuals
ga_holt_damped <- holt(train_GA, damped = TRUE, h = length(test_GA))
summary(ga_holt_damped)
```

```
##
## Forecast method: Damped Holt's method
##
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = train_GA, h = length(test_GA), damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.381
##     beta  = 1e-04
##     phi   = 0.9233
##
##   Initial states:
##     l = 4227.0352
##     b = 192.6712
##
```

```
##   sigma:  374.9326
##
##      AIC     AICc      BIC
## 1792.728 1793.560 1808.821
##
## Error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 42.9139 366.1507 282.4571 0.3060909 4.248357 0.5751939 0.03531174
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
## Jan 2019       8315.412 7834.916 8795.907 7580.557  9050.266
## Feb 2019       8315.538 7801.329 8829.748 7529.123  9101.954
## Mar 2019       8315.655 7769.797 8861.514 7480.837  9150.474
## Apr 2019       8315.763 7739.980 8891.546 7435.179  9196.347
## May 2019       8315.863 7711.625 8920.100 7391.761  9239.964
## Jun 2019       8315.955 7684.534 8947.375 7350.280  9281.629
## Jul 2019       8316.040 7658.551 8973.529 7310.497  9321.582
## Aug 2019       8316.118 7633.548 8998.688 7272.218  9360.019
## Sep 2019       8316.191 7609.423 9022.958 7235.282  9397.099
## Oct 2019       8316.257 7586.087 9046.428 7199.558  9432.957
## Nov 2019       8316.319 7563.468 9069.170 7164.933  9467.705
## Dec 2019       8316.376 7541.503 9091.249 7131.311  9501.442
## Jan 2020       8316.429 7520.139 9112.719 7098.608  9534.249
## Feb 2020       8316.477 7499.327 9133.627 7066.754  9566.200
## Mar 2020       8316.522 7479.028 9154.016 7035.686  9597.358
## Apr 2020       8316.564 7459.205 9173.922 7005.347  9627.780
## May 2020       8316.602 7439.826 9193.377 6975.689  9657.514
## Jun 2020       8316.637 7420.863 9212.412 6946.668  9686.606
## Jul 2020       8316.670 7402.288 9231.051 6918.244  9715.095
## Aug 2020       8316.700 7384.081 9249.319 6890.382  9743.017
## Sep 2020       8316.728 7366.219 9267.236 6863.050  9770.405
## Oct 2020       8316.753 7348.684 9284.822 6836.219  9797.287
## Nov 2020       8316.777 7331.458 9302.095 6809.862  9823.692
## Dec 2020       8316.799 7314.526 9319.071 6783.955  9849.642
## Jan 2021       8316.819 7297.873 9335.765 6758.476  9875.162
## Feb 2021       8316.838 7281.486 9352.189 6733.404  9900.271
## Mar 2021       8316.855 7265.352 9368.357 6708.720  9924.989
## Apr 2021       8316.871 7249.460 9384.281 6684.407  9949.334
## May 2021       8316.885 7233.800 9399.971 6660.449  9973.321
## Jun 2021       8316.899 7218.362 9415.436 6636.831  9996.967
## Jul 2021       8316.911 7203.136 9430.687 6613.538 10020.284
## Aug 2021       8316.923 7188.114 9445.732 6590.558 10043.287
## Sep 2021       8316.934 7173.288 9460.579 6567.879 10065.988
## Oct 2021       8316.943 7158.651 9475.236 6545.488 10088.399
## Nov 2021       8316.952 7144.196 9489.709 6523.376 10110.529
## Dec 2021       8316.961 7129.915 9504.006 6501.532 10132.390
## Jan 2022       8316.969 7115.804 9518.133 6479.946 10153.991
## Feb 2022       8316.976 7101.856 9532.096 6458.610 10175.342
## Mar 2022       8316.982 7088.065 9545.900 6437.515 10196.449
## Apr 2022       8316.988 7074.427 9559.550 6416.654 10217.323
## May 2022       8316.994 7060.936 9573.053 6396.018 10237.970
## Jun 2022       8316.999 7047.587 9586.411 6375.601 10258.397
## Jul 2022       8317.004 7034.378 9599.630 6355.396 10278.612
```

```
## Aug 2022       8317.009 7021.302 9612.715 6335.397 10298.620
## Sep 2022       8317.013 7008.357 9625.668 6315.596 10318.429
## Oct 2022       8317.016 6995.538 9638.495 6295.989 10338.044
## Nov 2022       8317.020 6982.841 9651.198 6276.570 10357.470
## Dec 2022       8317.023 6970.265 9663.782 6257.334 10376.712
## Jan 2023       8317.026 6957.804 9676.248 6238.275 10395.777
## Feb 2023       8317.029 6945.456 9688.602 6219.389 10414.669
## Mar 2023       8317.031 6933.218 9700.845 6200.671 10433.392
## Apr 2023       8317.034 6921.087 9712.981 6182.117 10451.950
## May 2023       8317.036 6909.060 9725.012 6163.722 10470.349
## Jun 2023       8317.038 6897.135 9736.941 6145.484 10488.592
## Jul 2023       8317.040 6885.309 9748.770 6127.396 10506.683
## Aug 2023       8317.041 6873.580 9760.503 6109.457 10524.626
## Sep 2023       8317.043 6861.945 9772.141 6091.662 10542.424
## Oct 2023       8317.044 6850.402 9783.687 6074.008 10560.081
## Nov 2023       8317.046 6838.949 9795.142 6056.492 10577.600
## Dec 2023       8317.047 6827.584 9806.510 6039.110 10594.984
```

```
checkresiduals(ga_holt_damped)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt's method
## Q* = 66.607, df = 22, p-value = 2.237e-06
```

```
##
## Model df: 0.   Total lags used: 22
```

```
cat("--- Training Dataset ---")
```

```
## --- Training Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = ga_holt_damped$residuals, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = ga_holt_damped$residuals, .actual = train_GA),
      "\nME:   ", ME(.resid = ga_holt_damped$residuals, .actual = train_GA))
```

```
##
## MAE:  282.4571
## RMSE: 366.1507
## ME:   42.9139
```

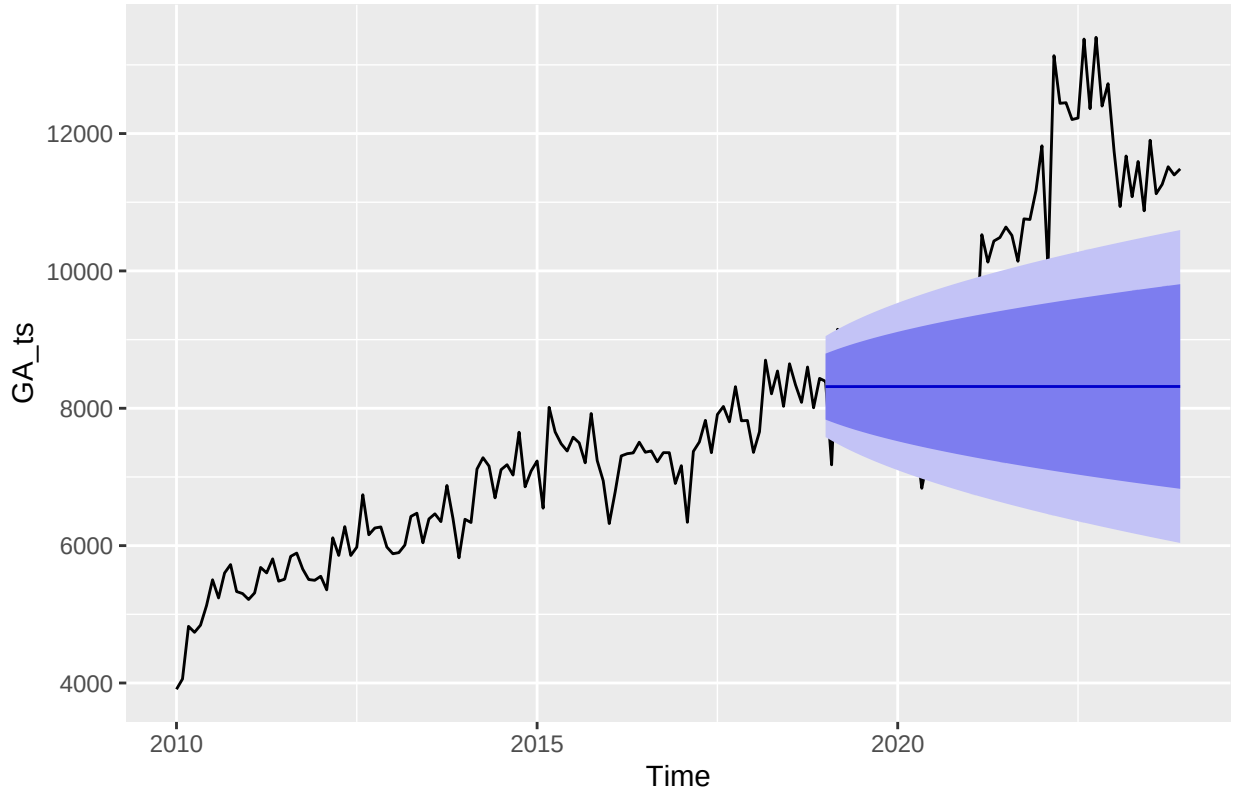
```
holt_damped_resid <- test_GA - ga_holt$mean
holt_damped_forecast_resid <- ts(holt_damped_resid, start = 2019, frequency = 12)
cat("--- Testing Dataset ---")
```

```
## --- Testing Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = holt_damped_forecast_resid, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = holt_damped_forecast_resid, .actual = test_GA),
      "\nME:   ", ME(.resid = holt_damped_forecast_resid, .actual = test_GA))
```

```
##
## MAE:  1120.636
## RMSE: 1409.286
## ME:   625.5859
```

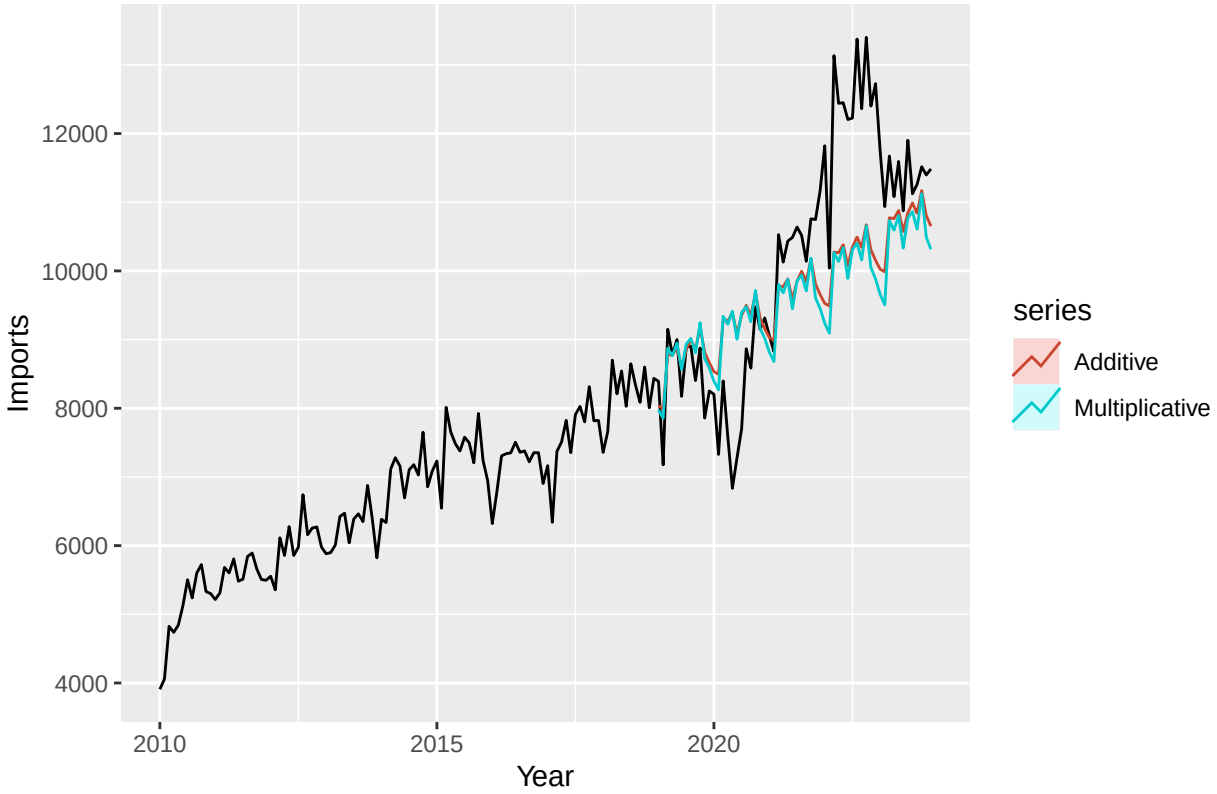
```
autoplot(GA_ts) +
  autolayer(ga_holt_damped)
```



In this instance, damping it made it much worse. This was to be expected. In the original data, once we passed the training set, there was a much bigger increase, but the purpose of dampening the model is that we hope the increase fizzles out and it goes back to being flat, which is the exact opposite of what we needed to do here.

Now that we have an idea of how the Holt model works, we should try out the Holt-Winters method, which will add a seasonal component to our model. Within this, we can use an additive or multiplicative model. We will try both.

```
# Fit new holt winters additive and multiplicative models
hw_add <- hw(train_GA, h=length(test_GA), seasonal="additive")
hw_mult <- hw(train_GA, h=length(test_GA), seasonal = "multiplicative")

autoplot(GA_ts) +
  autolayer(hw_add, series = "Additive", PI=FALSE) +
  autolayer(hw_mult, series = "Multiplicative", PI=FALSE) +
  xlab("Year") +
  ylab("Imports")
```



Additive and Multiplicative are both pretty similar in this case, but just by looking at the graph, mutliplicative seems to be a little bit better. We can test this out through comparing their residuals and finding the RMSE.

```
# Extract training and testing residuals for both additive and multiplicative models
cat("--- Additive Training Dataset ---")
```

```
## --- Additive Training Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = hw_add$residuals, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = hw_add$residuals, .actual = train_GA),
      "\nME:   ", ME(.resid = hw_add$residuals, .actual = train_GA))
```

```
##
## MAE:  201.9228
## RMSE: 253.9978
## ME:   -24.00756
```

```
cat("--- Multiplicative Training Dataset ---")
```

```
## --- Multiplicative Training Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = hw_mult$residuals, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = hw_mult$residuals, .actual = train_GA),
      "\nME:   ", ME(.resid = hw_mult$residuals, .actual = train_GA))
```

```
##
## MAE:  0.02997471
## RMSE: 0.04013408
## ME:   -0.003806911
```

```
hw_add_resid <- test_GA - hw_add$mean
cat("--- Additive Testing Dataset ---")
```

```
## --- Additive Testing Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = hw_add_resid, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = hw_add_resid, .actual = test_GA),
      "\nME:   ", ME(.resid = hw_add_resid, .actual = test_GA))
```

```
##
## MAE:  970.5785
## RMSE: 1271.149
## ME:   455.9686
```

```
hw_mult_resid <- test_GA - hw_mult$mean
cat("--- Mutliplicative Testing Dataset ---")
```

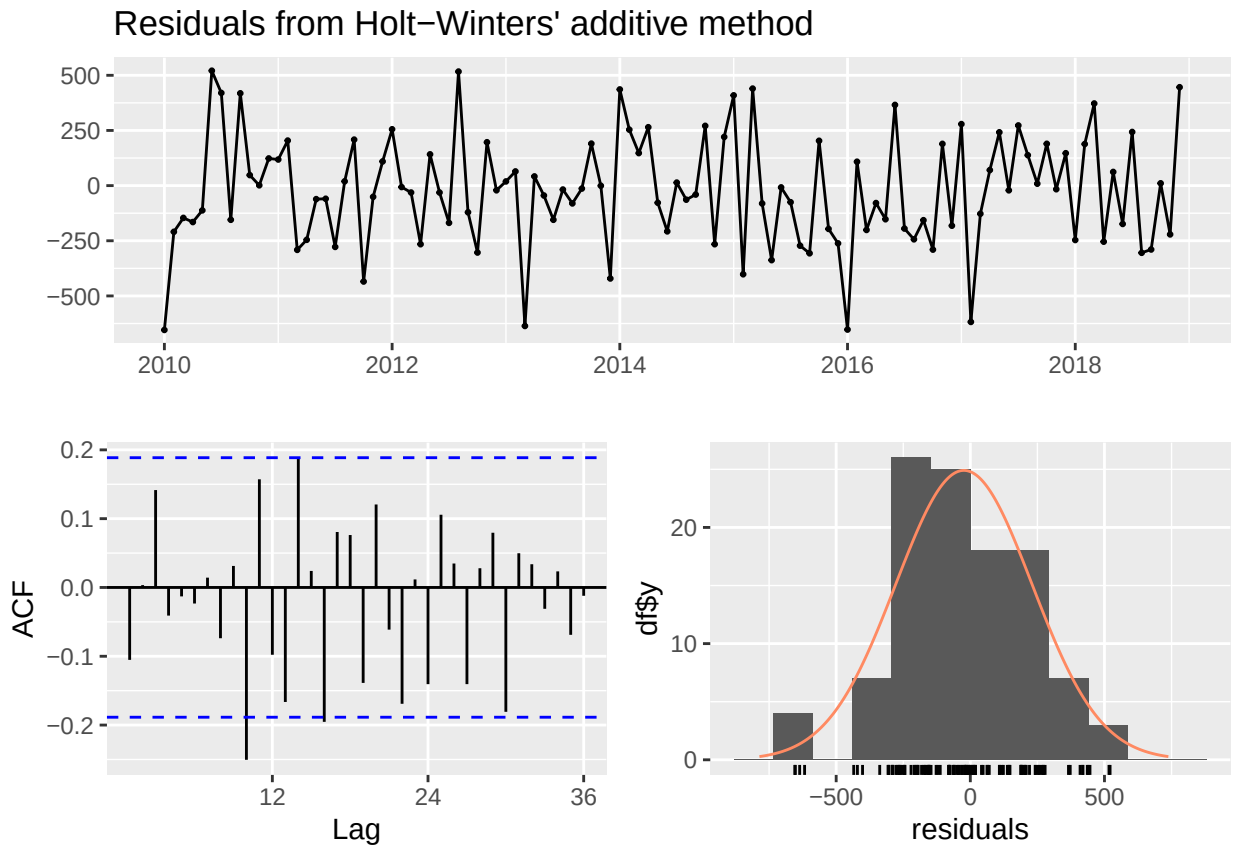
```
## --- Mutliplicative Testing Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = hw_mult_resid, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = hw_mult_resid, .actual = test_GA),
      "\nME:   ", ME(.resid = hw_mult_resid, .actual = test_GA))
```

```
##
## MAE:  1046.886
## RMSE: 1353.048
## ME:   565.4003
```

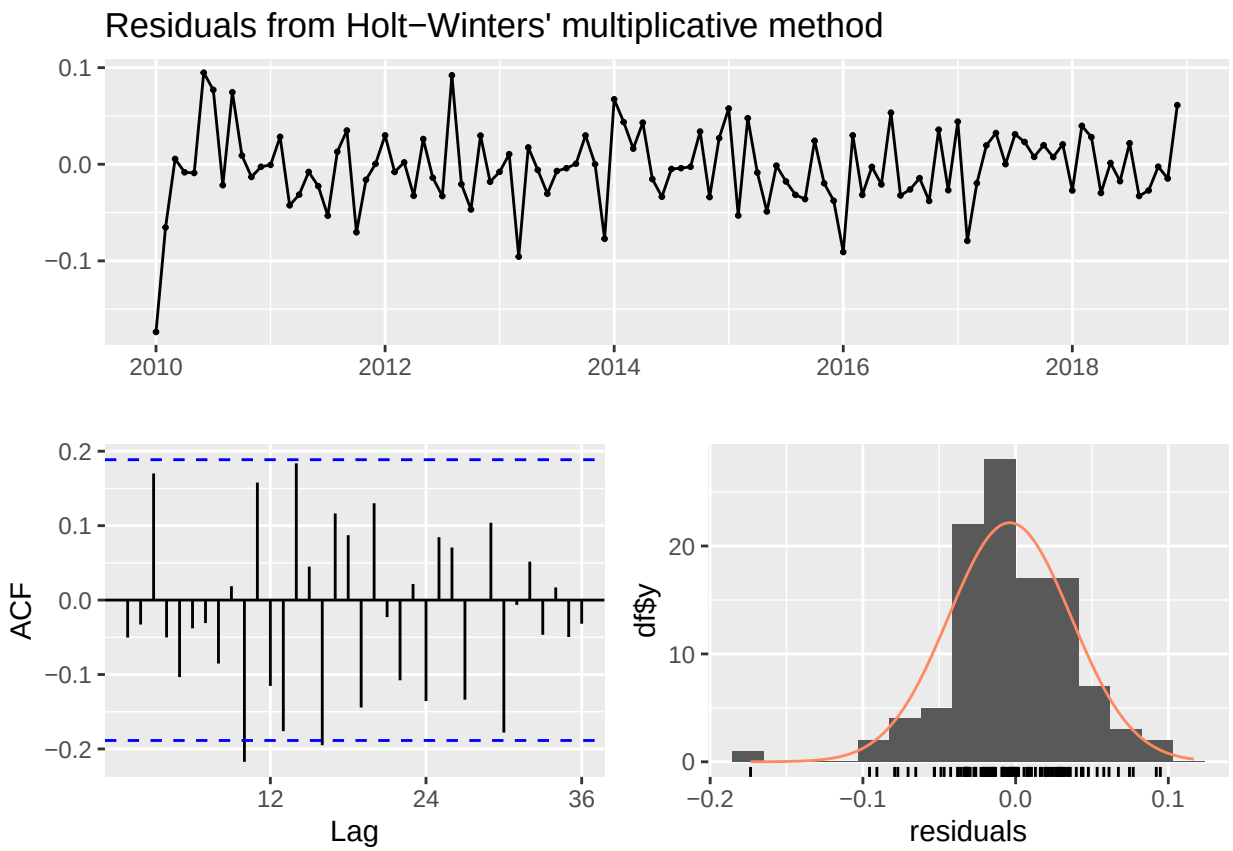
While the graph doesn't show it as well, the Multiplicative version does a MUCH better job as we get a RMSE that is less than 1 in the training set. In the Testing set, additive is actually better, but not by much.

```
# Check the residuals of both models
checkresiduals(hw_add)
```



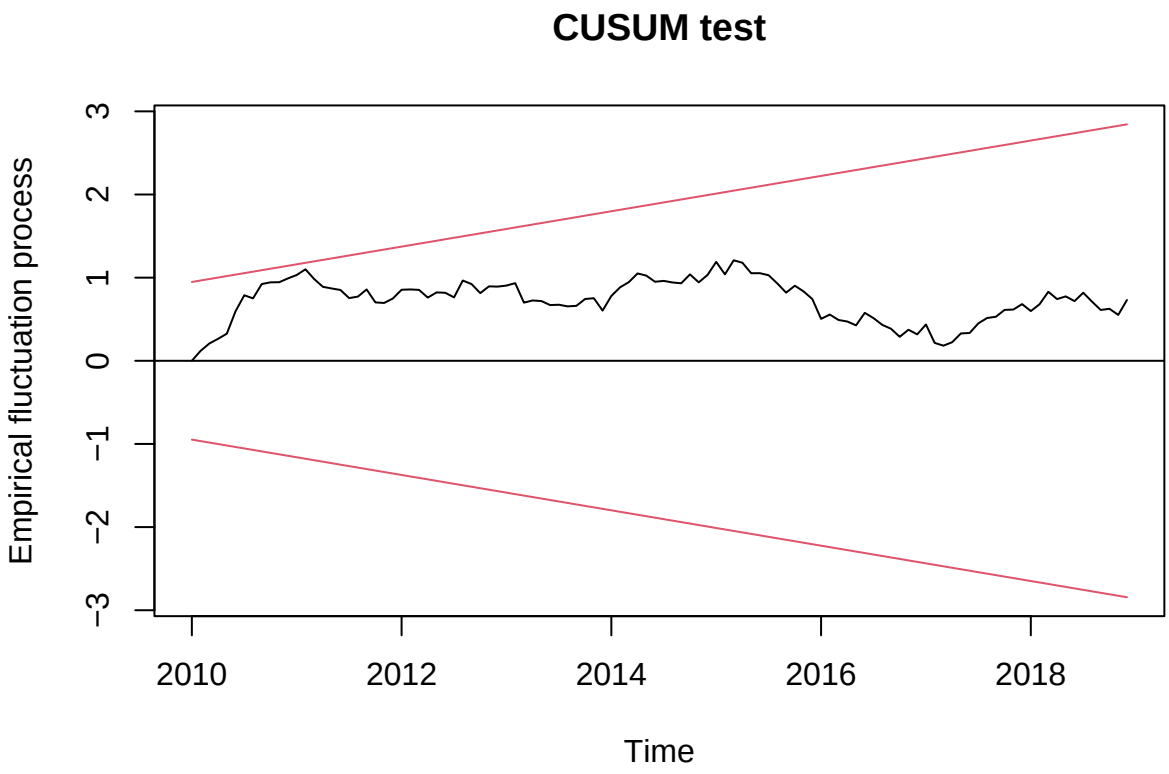

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' additive method
## Q* = 39.938, df = 22, p-value = 0.01099
##
## Model df: 0.   Total lags used: 22
```

```
checkresiduals(hw_mult)
```

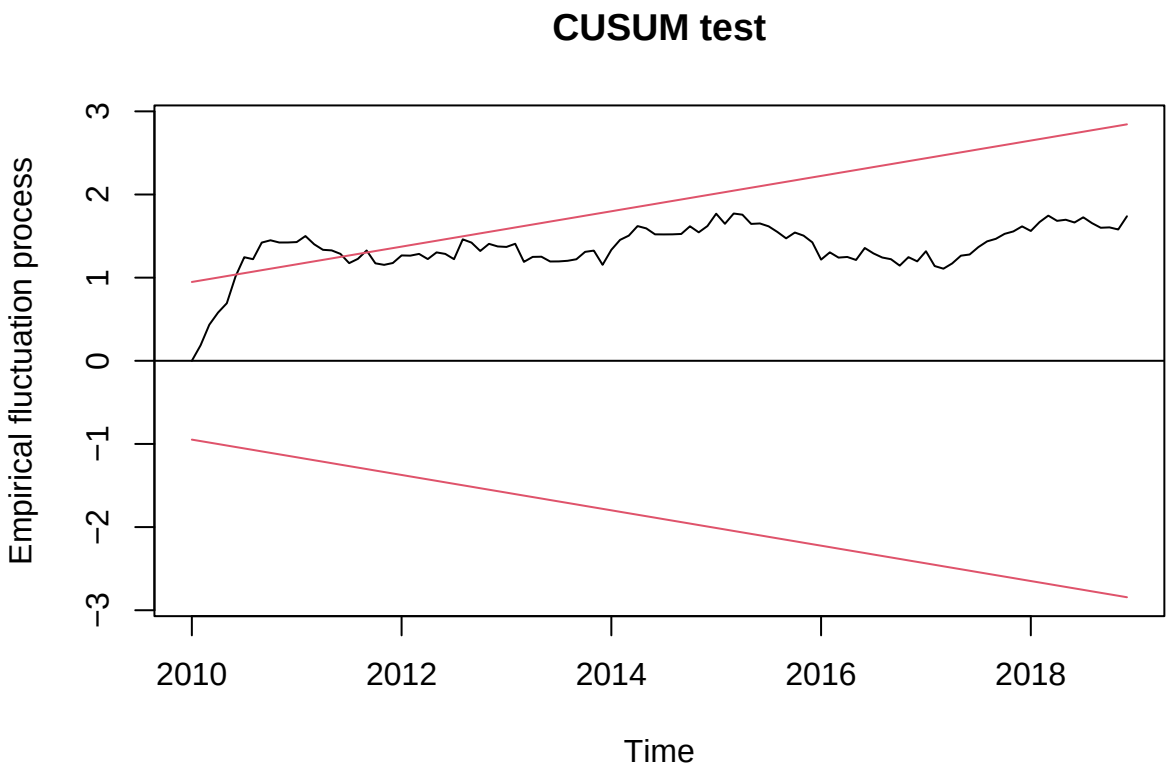



```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 39.632, df = 22, p-value = 0.01193
##
## Model df: 0.   Total lags used: 22
```

```
plot(efp(hw_add$residuals ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```




```
plot(efp(hw_mult$residuals ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```

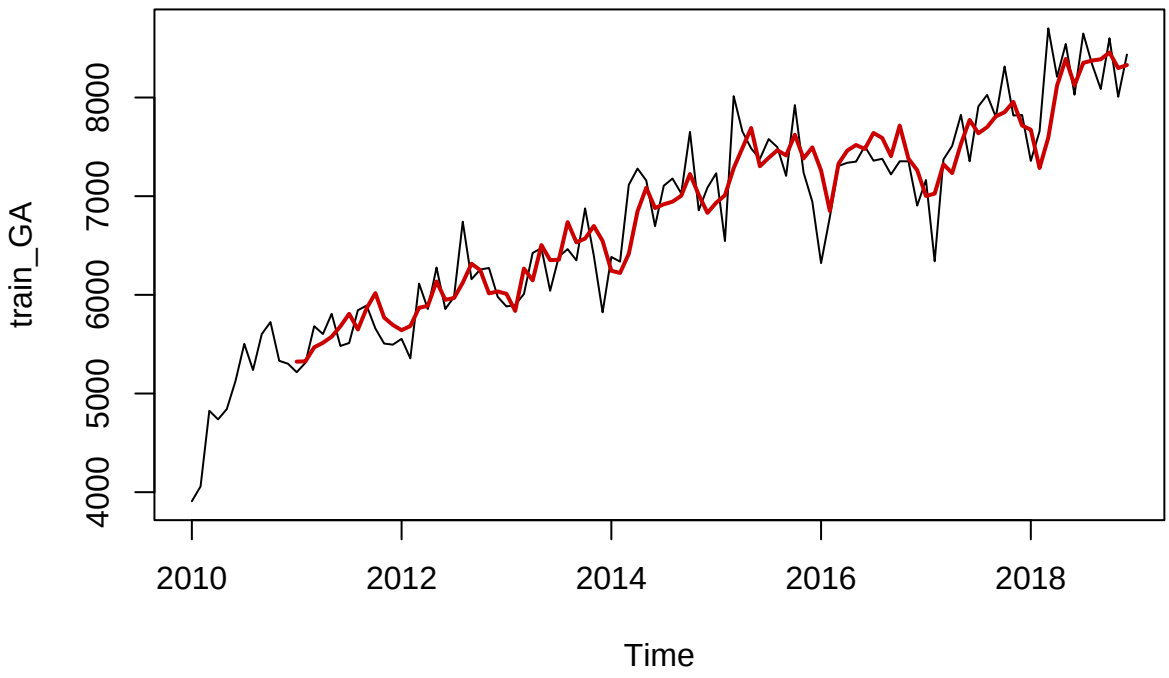


This is much better results compared to Holt linear model. In both ACF graphs, there are a few spikes that barely cross the threshold, but we can write this off as White noise. There is no pattern in the residuals, and they seem to centered at or close to 0. On top of all that, the Ljung-Box test fails to reject the null hypothesis on both models. Whether we do additive or multiplicative, this is a major upgrade from the original without seasonality.

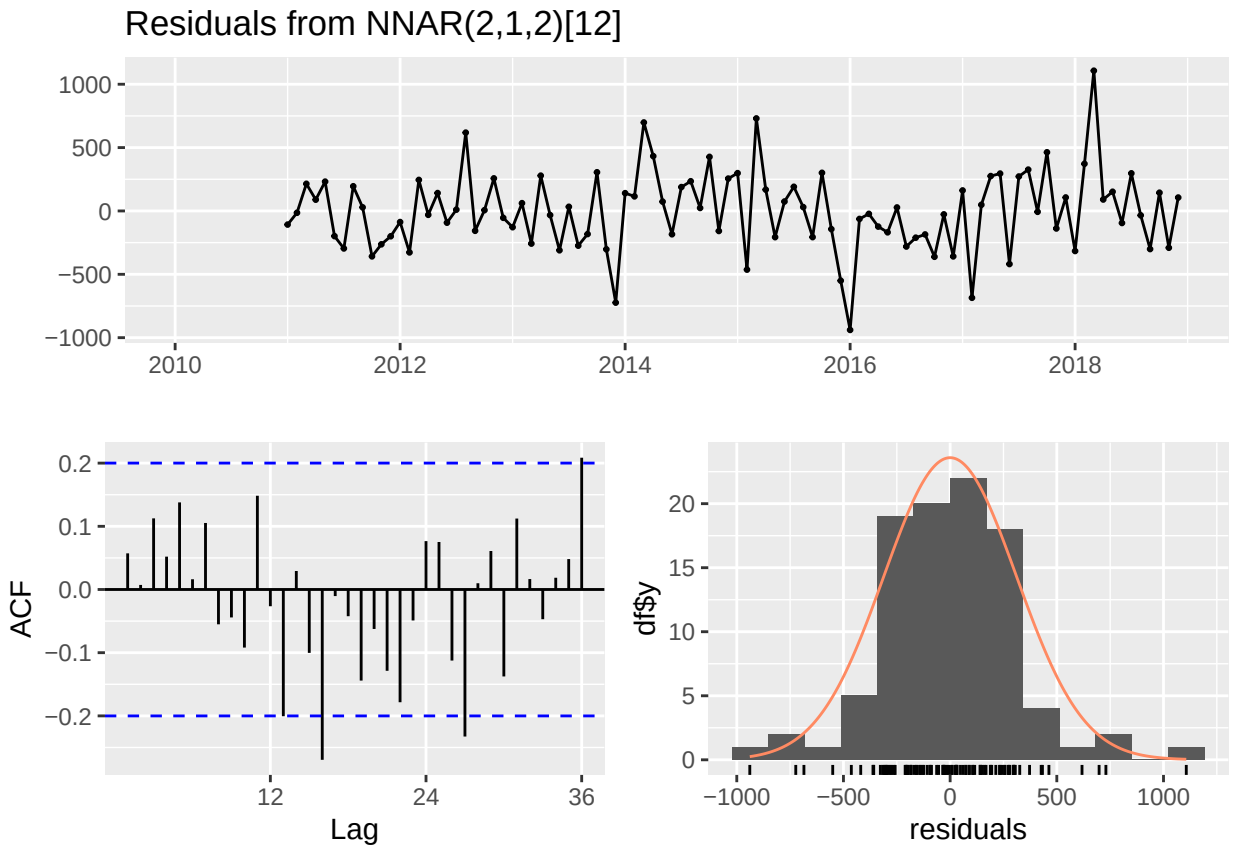
Looking at the cusum plots of both models, we can see the additive is stable, however the multiplicative model crosses the bounds slightly. This is alarming for model stability, however it is a very slight cross and it comes back in the end.

Nnetar Model

```
# Create the nnetar model by calling the nnetar function, and plot the training data to see the fit of
nnetar_ga <- nnetar(train_GA)
plot(train_GA)
lines(nnetar_ga$fitted,col="red3",lwd=2)
```

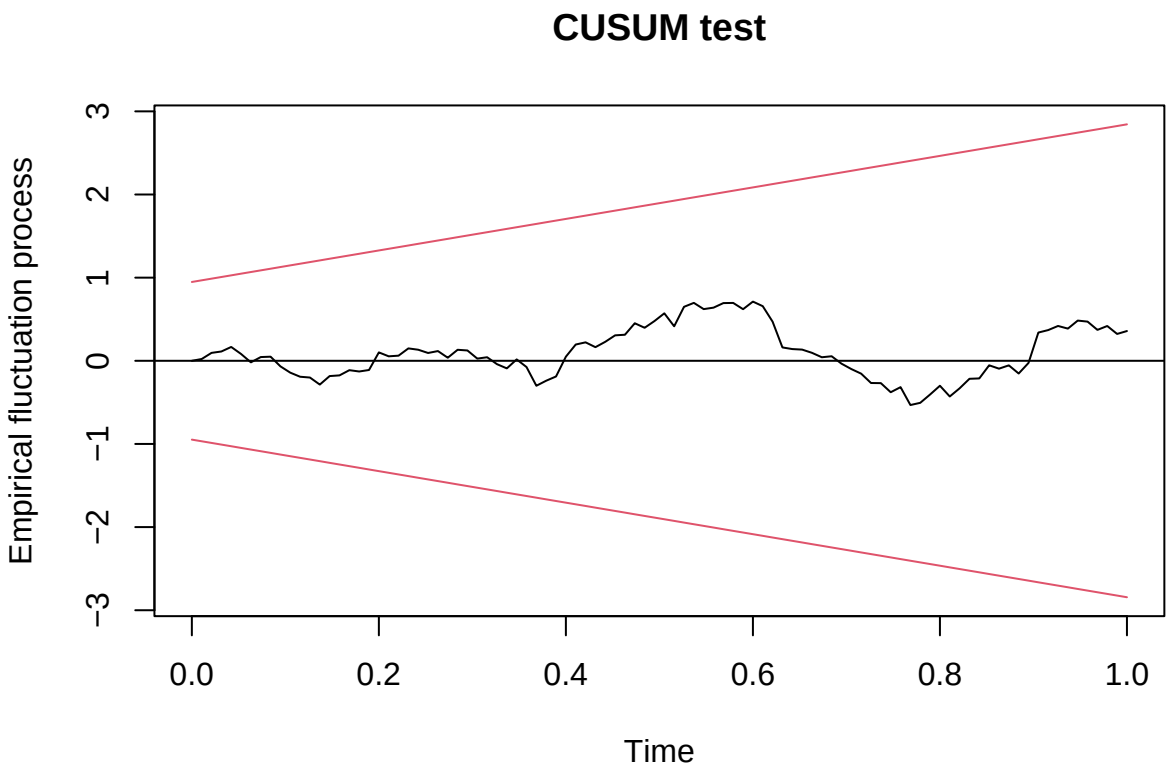



```
#Assess the nnetar model validity by using checkresiduals and the cusum plot
checkresiduals(nnetar_ga)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(2,1,2)[12]
## Q* = 32.767, df = 22, p-value = 0.06523
##
## Model df: 0.   Total lags used: 22
```

```
plot(efp(nnetar_ga$residuals ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```




```
cat("--- Training Dataset ---")
```

```
## --- Training Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = nnetar_ga$residuals, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = nnetar_ga$residuals, .actual = train_GA),
      "\nME:   ", ME(.resid = nnetar_ga$residuals, .actual = train_GA))
```

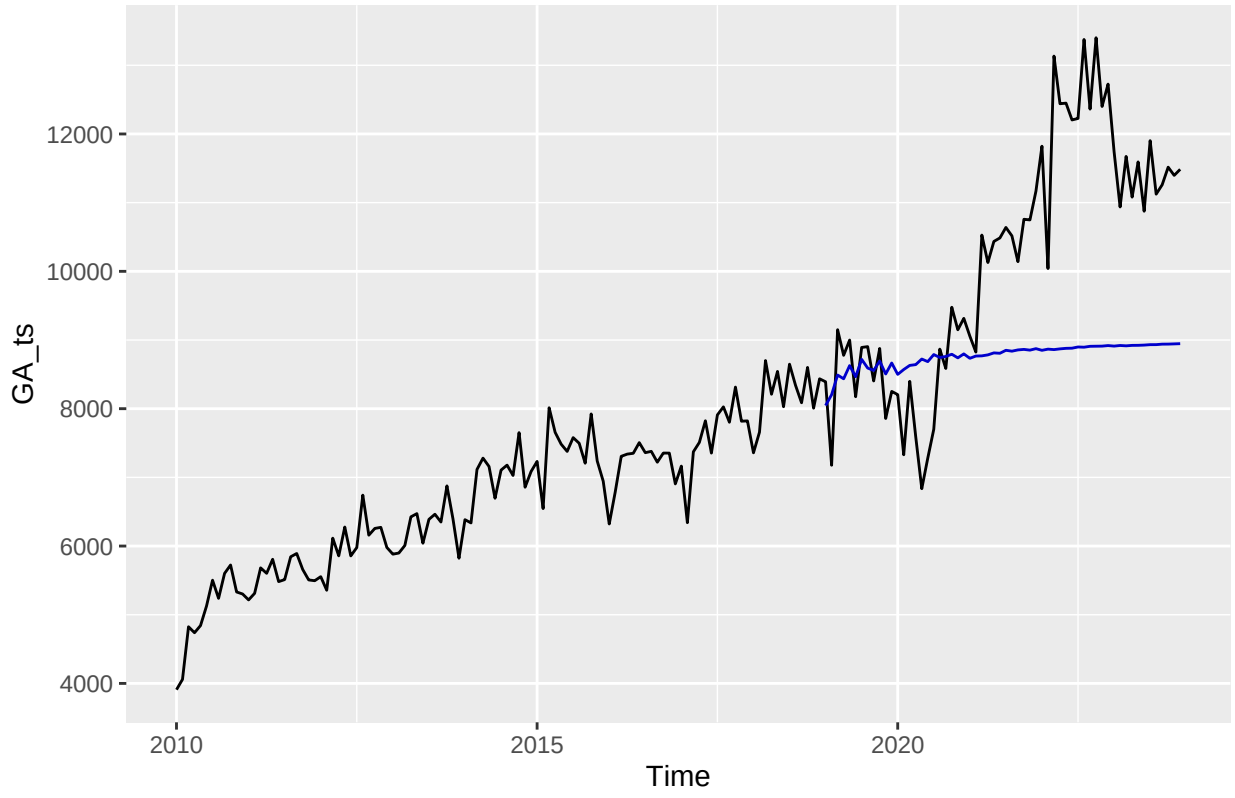
```
##
## MAE:  236.2234
## RMSE: 309.7247
## ME:   -0.007863011
```

Based off the residuals, we reject the null hypothesis in the Ljung-Box test and conclude serial correlation is still in our model, so our model is not completely effective. Looking at the ACF, it looks pretty clean with just a couple spikes, and the p value for the test was 0.03, so this shows that the model isn't that far off, it's pretty reasonable for the data. Looking at the diagnostic statistics, we have an MAE of 242 and an RMSE of 315, pretty good for the model although higher than some of our other models.

We can see from the Cusum plot that there are no structural breaks in the model

```
#Compute the forecast and plot the forecast with the original data to see forecast fit, and then look a
```

```
forecast_nn <- forecast(nnetar_ga, h = length(test_GA))

autoplot(GA_ts) +
  autolayer(forecast_nn)
```




```
nnetar_resid <- test_GA - forecast_nn$mean
nnetar_forecast_resid <- ts(nnetar_resid, start = 2019, frequency = 12)
cat("--- Testing Dataset ---")
```

```
## --- Testing Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = nnetar_forecast_resid, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = nnetar_forecast_resid, .actual = test_GA),
      "\nME:   ", ME(.resid = nnetar_forecast_resid, .actual = test_GA))
```

```
##
## MAE:  1716.947
## RMSE: 2129.426
## ME:   1386.715
```

```
summary(forecast_nn)
```

```
##
## Forecast method: NNAR(2,1,2)[12]
##
## Model Information:
##
## Average of 20 networks, each of which is
## a 3-2-1 network with 11 weights
## options were - linear output units
##
## Error measures:
##                         ME     RMSE      MAE        MPE    MAPE      MASE
## Training set -0.007863011 309.7247 236.2234 -0.2096272 3.43138 0.4810439
##                     ACF1
## Training set 0.05721419
##
## Forecasts:
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2019 8045.702 8203.941 8490.043 8435.099 8627.540 8465.651 8718.489 8591.694
## 2020 8499.500 8570.179 8630.012 8641.678 8722.869 8685.142 8788.187 8745.925
## 2021 8733.368 8766.409 8769.137 8783.418 8812.150 8806.969 8850.231 8837.397
## 2022 8850.025 8867.083 8860.450 8870.818 8878.551 8880.877 8898.344 8895.490
## 2023 8911.409 8920.044 8915.142 8921.816 8922.709 8926.056 8932.408 8932.811
##           Sep      Oct      Nov      Dec
## 2019 8556.336 8692.984 8505.350 8665.146
## 2020 8759.681 8792.785 8739.927 8797.858
## 2021 8855.708 8863.217 8852.491 8873.977
## 2022 8908.017 8909.818 8910.911 8918.916
## 2023 8939.554 8940.387 8943.393 8946.482
```

The (2, 1, 2) means 2 lagged values, 1 exogenous input, and 2 hidden nodes (12 means seasonality). When looking at the graph, we see that Neural Networks do a very poor job at predicting the testing set. In terms of the diagnostic statistics, we see the MAE and RMSE skyrocket to 1871 and 2317, pretty consistent with what we've seen with other models.

Prophet Model

```
# Create the prophet model
time_index <- time(train_GA) # Get the time index
data_values <- as.numeric(coredata(train_GA)) # Extract the core data
prophet_df <- data.frame(ds = as.Date(time_index, origin = "2010-01-01"), y = data_values)
prophet_df$ds <- as.Date(prophet_df$ds)
m <- prophet(prophet_df, yearly.seasonality = "auto", seasonality.mode = "additive")
```

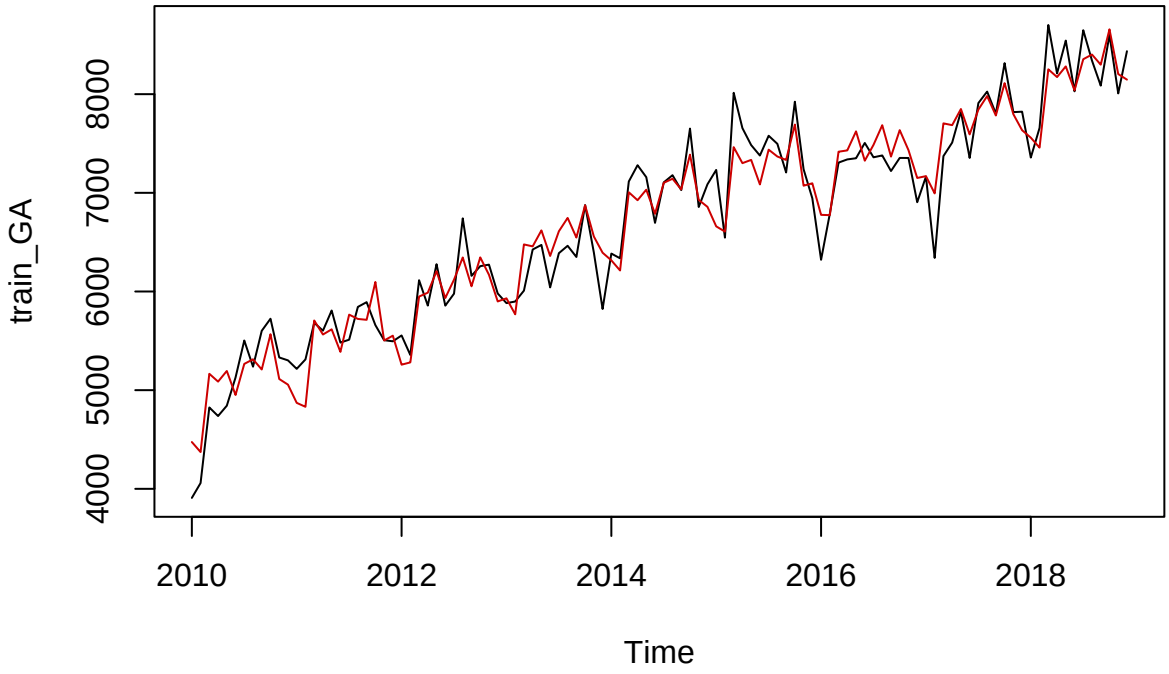
```
## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
future <- make_future_dataframe(m, periods = 60, freq = "month")
prophet_forecast <- predict(m, future)

#Prophet_forecast returned both the model on the training set, and a 5 year forecast so filter it to ge
```

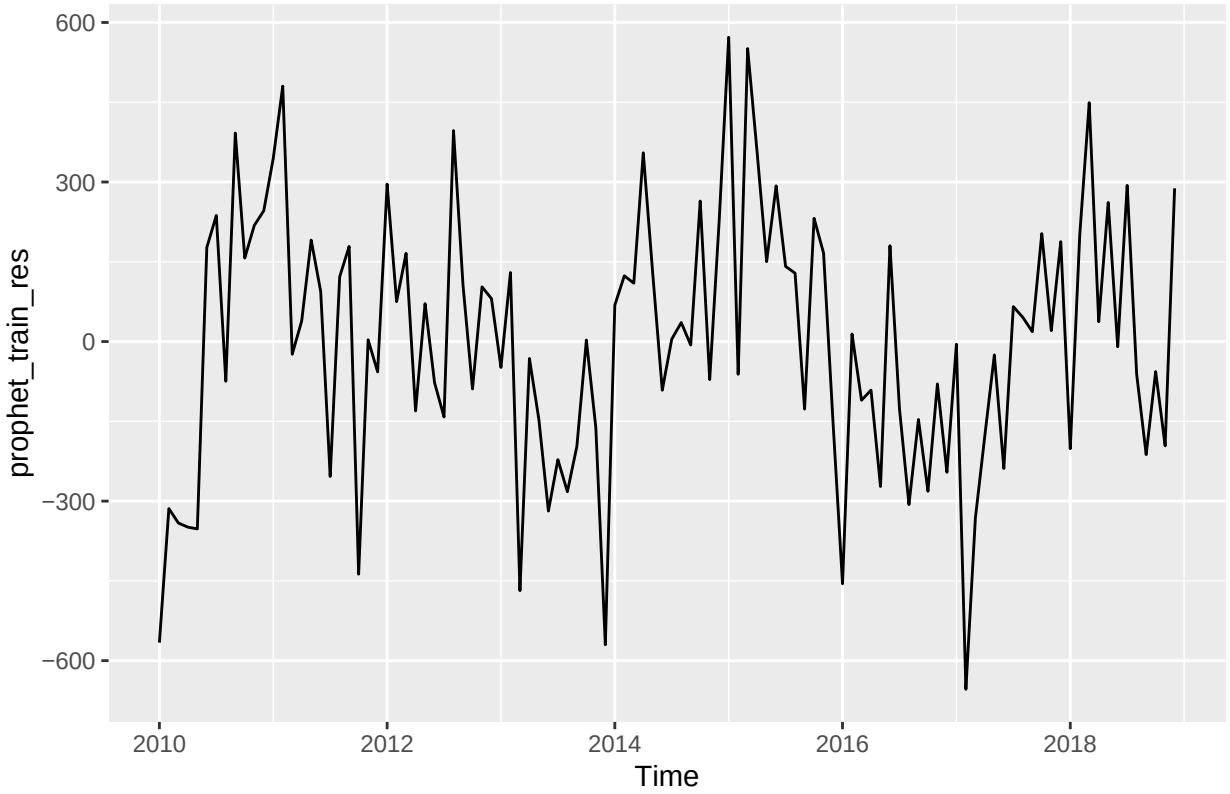
```r
train_results <- prophet_forecast %>%
  filter(ds <= "2018-12-01")

forecast_results <- prophet_forecast %>%
  filter(ds > "2018-12-01")
forecast_results$y <- test_GA
```

```r
# Plot the training data with the prophet model fit
train_prophet_ts <- ts(train_results$yhat, start = 2010, frequency = 12)
test_prophet_ts <- ts(forecast_results$yhat, start = 2019, frequency = 12)
plot(train_GA)
lines(train_prophet_ts, col="red3")
```

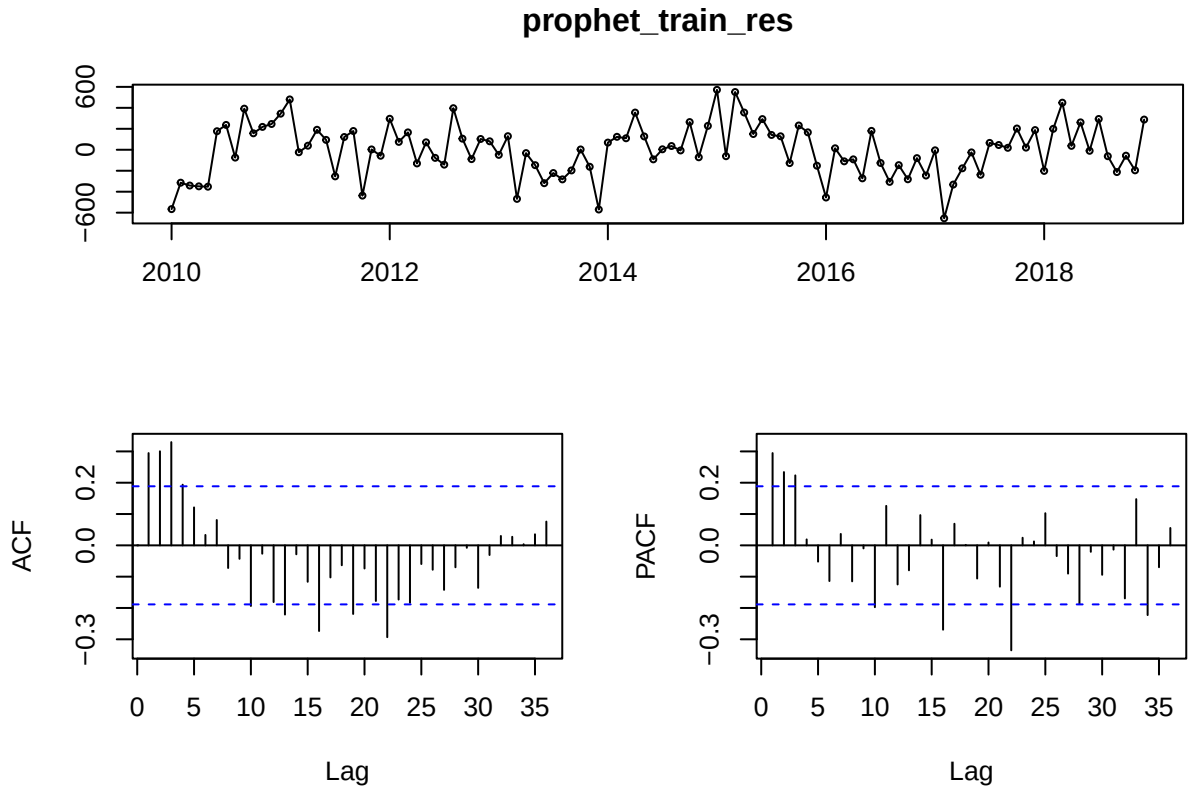


Looking at the model fit, the model fit seems pretty reasonable, however there seems to be a trend of the model not accounting for enough of the spikes.

```r
# Assess the model validity of the prophet model by checking the residuals and running the ljung box te

# Extract the residuals for both the training data
prophet_train_res <- train_GA - train_prophet_ts
prophet_test_res <- test_GA - test_prophet_ts
autoplot(prophet_train_res)
```



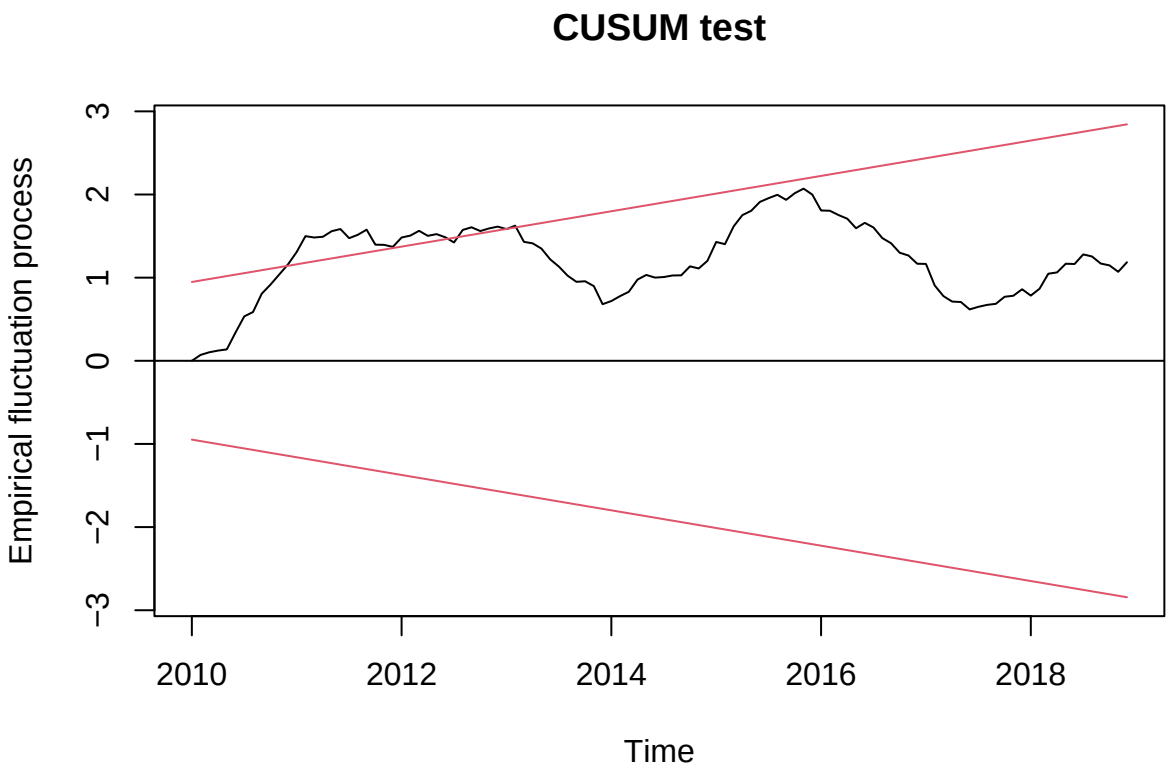

```
tsdisplay(prophet_train_res)
```




```
Box.test(prophet_train_res, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  prophet_train_res
## X-squared = 9.6335, df = 1, p-value = 0.001911
```

```
plot(efp(prophet_train_res ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```



From looking at the autoplot of the residuals, and the ACF, PACF and Ljung box test, we can see that the dynamics are not all accounted for in the model. We see some patterns in the time plot, and we see plenty of significant lags in the ACF and PACF plot of the residuals. In specific, around lags 15-25, there are numerous significant lags in both the ACF and PACF. Furthermore, the Ljung box test gives us a P value of 0.001 signifying that there is serial correlation in the residuals. So overall we see the prophet model does not fully cover the dynamics of the data

looking at the cusum plot, we see the residuals barely cross the bounds. This is alarming for model stability, despite the fact that it is ever so slightly crossing the border but it comes back towards the end.

```
# Get the MAE, RMSE and ME values for the prophet model on the training data
cat("--- Training Dataset ---")
```

```
## --- Training Dataset ---
```

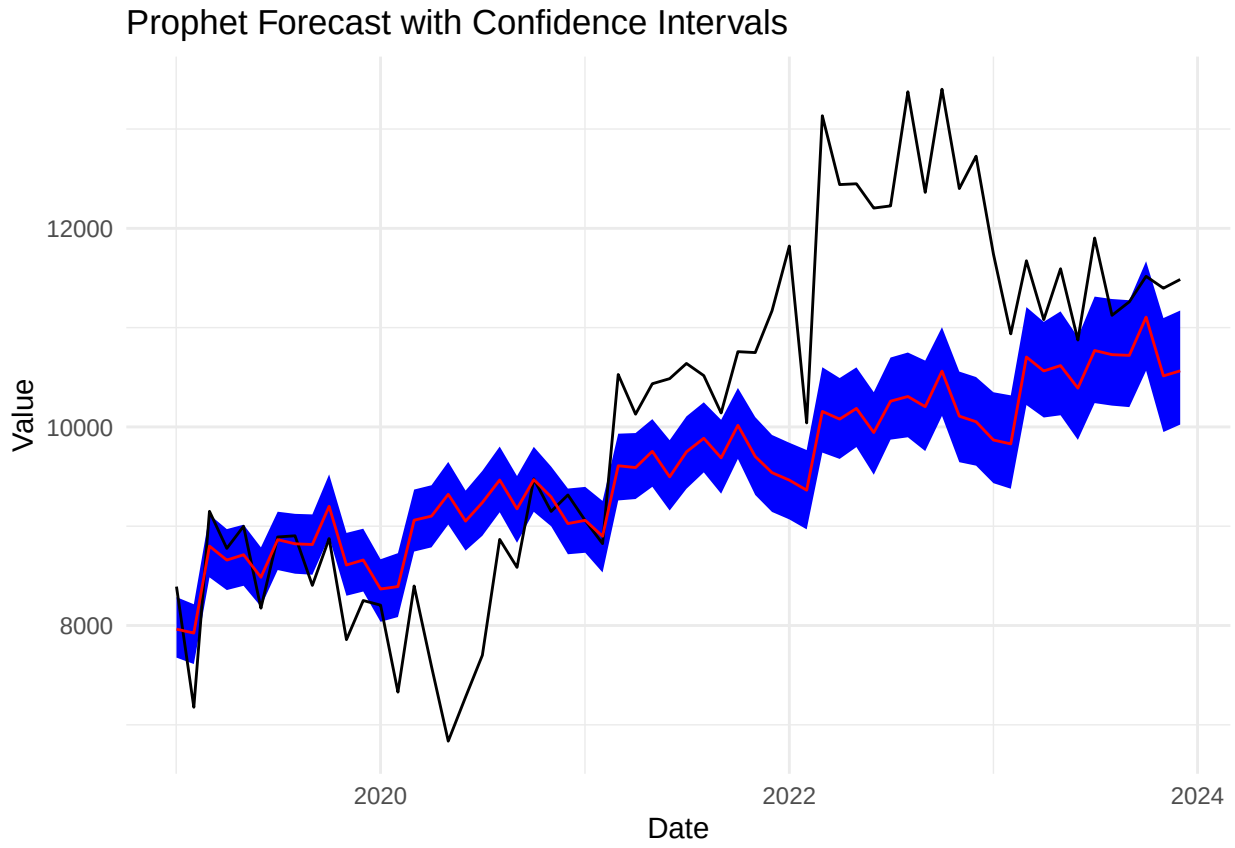
```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = prophet_train_res, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = prophet_train_res, .actual = train_GA),
      "\nME:   ", ME(.resid = prophet_train_res, .actual = train_GA))
```

```
##
## MAE:  193.7811
## RMSE: 243.0319
## ME:   0.2218004
```

Based on these diagnostic statistics, it seems the model does reasonably well for the training data, with

relatively low MAE and RMSE values of 193 and 243. We could see that by looking at the plot of the model with the original data as they overalapped pretty well.

```
# Plot the testing data with the forecast of the prophet model
ggplot() +
  geom_ribbon(data = forecast_results, aes(x = ds, ymin = yhat_lower, ymax = yhat_upper), fill = "blue")
  geom_line(data = forecast_results, aes(x = ds, y = y), color = "black") + # Actual values
  geom_line(data = forecast_results, aes(x = ds, y = yhat), color = "red") + # Forecasted values
  labs(x = "Date", y = "Value", title = "Prophet Forecast with Confidence Intervals") +
  theme_minimal()
```




```
prophet_test_res <- test_GA - test_prophet_ts
# Get the MAE, RMSE and ME values for the prophet model on the testing data
cat("--- Testing Dataset ---")
```

```
## --- Testing Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = prophet_test_res, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = prophet_test_res, .actual = test_GA),
      "\nME:   ", ME(.resid = prophet_test_res, .actual = test_GA))
```

```
##
## MAE:  1029.558
## RMSE: 1332.186
## ME:   577.5092
```

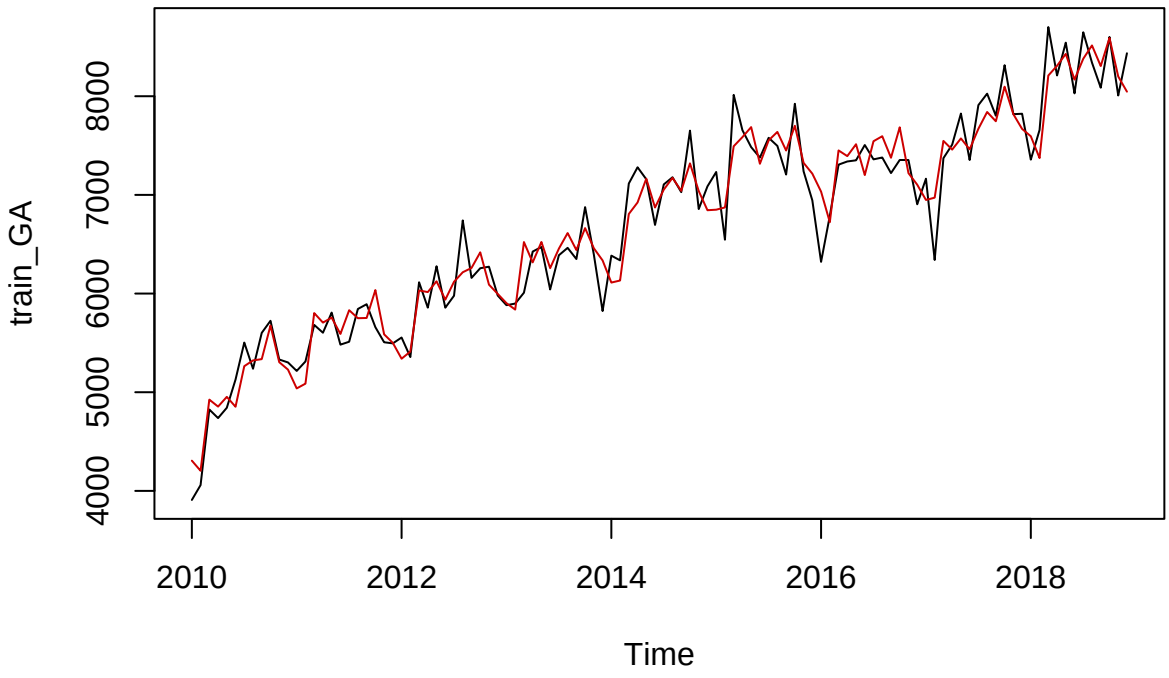
Similar to all of our previous models, we see the diagnostic statistic values skyrocket with the testing data. With our previous work, we know that the testing data is partly accountable for that as it is much more volatile than the training data. However by looking at the plot of the forecast, it looks like the prophet forecast gives a pretty linear forecast with not much volatility which correlates with the high error values we see.

Combination Forecast

To create the combination forecast, we took a regular average of the fitted values, and then the forecasted values, from each of our individual models.

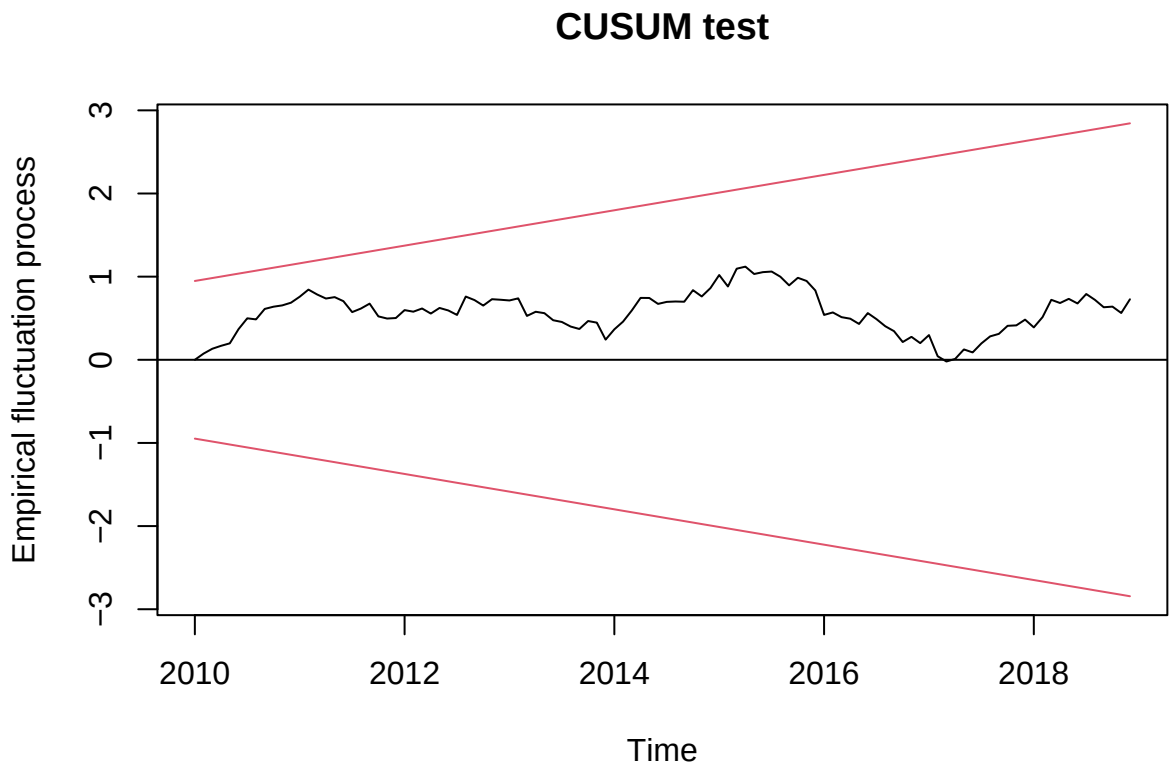
```
# First, get the fitted values of our combination forecast by combining the fitted values and taking an

# because of the way the nnetar function operates, there are no fitted values for the first yera of the

nnetar_ga$fitted[1:12] <- train_GA[1:12]
combined_fitted <- (GA_arima$fitted + ets_model$fitted + hw_mult$fitted + hw_add$fitted + nnetar_ga$fit
```

```
# Look at the fit of the combined model vs the original training data
plot(train_GA)
lines(combined_fitted, col="red3")
```

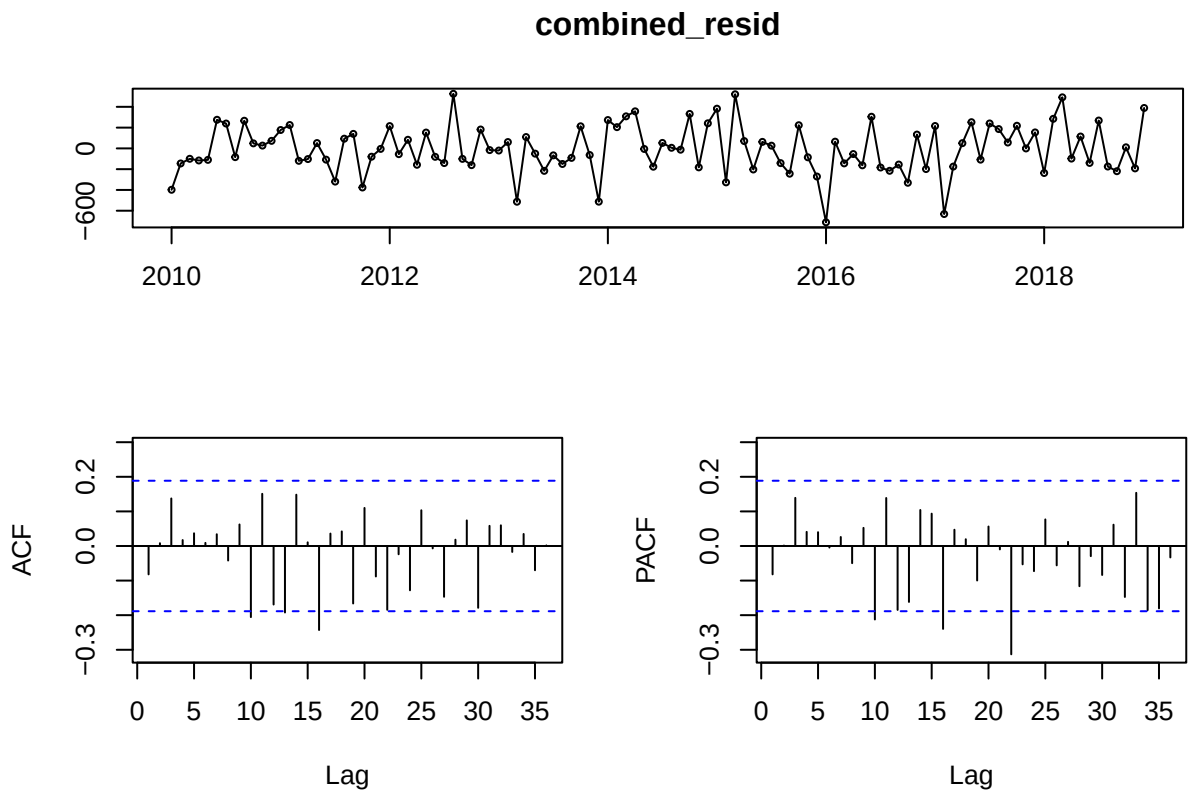


Looking at this model fit, it looks very reasonable and seems to do pretty good with the spikes as well.

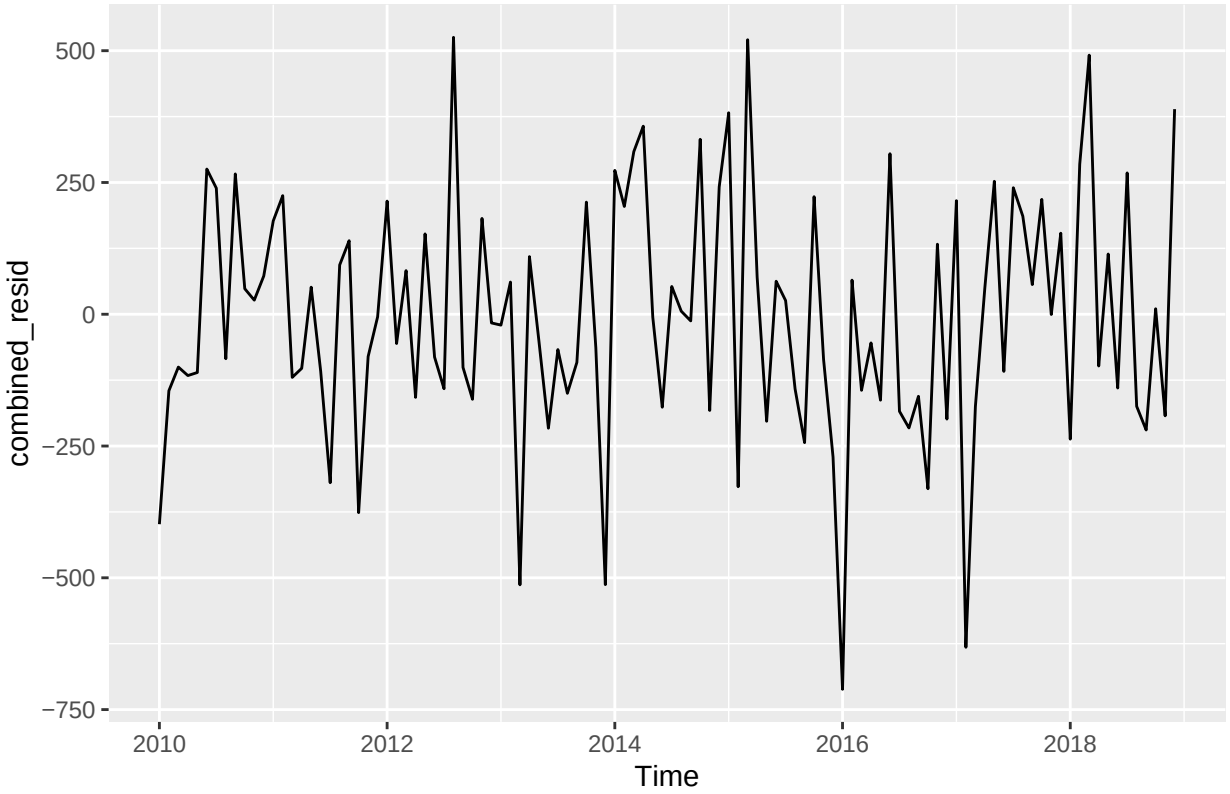
```r
# Assess the model validity of our combined model by looking and running analysis on the residuals.
combined_resid <- train_GA - combined_fitted
plot(efp(combined_resid ~ 1, type = "Rec-CUSUM"), main = "CUSUM test")
```




```r
tsdisplay(combined_resid)
```




```
autoplot(combined_resid)
```




```
Box.test(combined_resid, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  combined_resid
## X-squared = 0.74728, df = 1, p-value = 0.3873
```

Looking at the tsdisplay and the plot of the residuals, we actually see that our combined model does very well in capturing the dynamics of the data. We see that the ACF and Pacf are pretty clean, except for a couple lags where they slightly cross the threshold at around 16 and 22. Looking at the Ljung Box test, the test failed to reject the null therefore saying that there is no more serial correlation in the residuals. These results signify that our model does a good job with the training data. Although it must be said because we used the actual training data to fit the first year of fitted values for the nnetar model, that the results may be slightly misleading.

Looking at the cusum plot, there are no structural breaks in the model with no significant divergence

```
# Get the MAE, RMSE and ME values for the combined model on the training data
cat("--- Training Dataset ---")
```

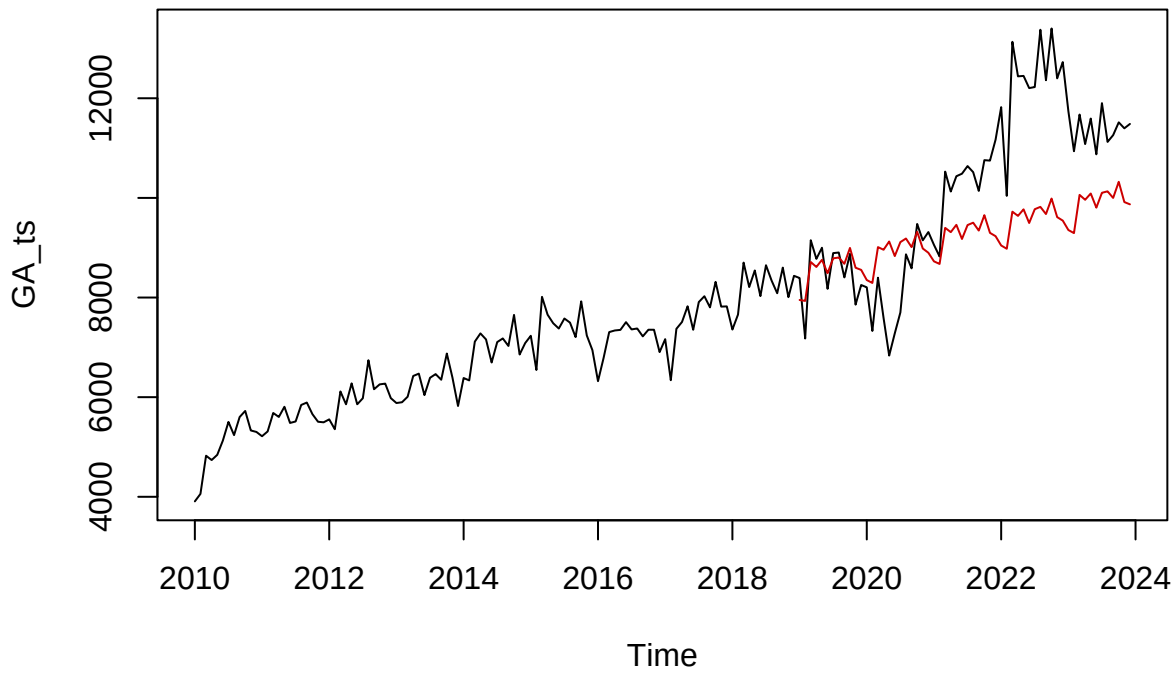
```
## --- Training Dataset ---
```

```
cat(sep = "",
      "\nMAE:  ", MAE(.resid = combined_resid, .actual = train_GA),
      "\nRMSE: ", RMSE(.resid = combined_resid, .actual = train_GA),
      "\nME:   ", ME(.resid = combined_resid, .actual = train_GA))
```

```
##
## MAE:  184.2418
## RMSE: 230.7908
## ME:   -5.5933
```

Looking at our diagnostic statistics for the training data, our MAE and RMSE are relatively low as we figured with the residual analysis we had done.

```
# Compute the forecasted values for our combined model
combined_forecast <- (GA_arima_forecast$mean + GA_ets_forecast$mean + hw_mult$mean + hw_add$mean + fore

# Plot the forecasted values with the original time series
plot(GA_ts)
lines(combined_forecast, col="red3")
```



After computing the forecast and plotting it, we can see the forecast does not do well with the testing data. Once again, we see that it isn't able to capture the stark increase we see in the testing data from 2019 onwards.

```r
combined_forecast_resid <- test_GA - combined_forecast
# Get the MAE, RMSE and ME values for the combined model on the testing data
cat("--- Testing Dataset ---")
```

```
## --- Testing Dataset ---
```

```r
cat(sep = "",
      "\nMAE:  ", MAE(.resid = combined_forecast_resid, .actual = test_GA),
      "\nRMSE: ", RMSE(.resid = combined_forecast_resid, .actual = test_GA),
      "\nME:   ", ME(.resid = combined_forecast_resid, .actual = test_GA))
```

```
##
## MAE:  1286.889
## RMSE: 1612.802
## ME:   900.3356
```

Again, we see the massive increase in the error statistics for the testing data. We will get into the comparisons between these error statistics shortly however the big trend is how all these models struggle with accouting for the dynamics of the testing data, which is not too surprising given how different the training and testing data are.

Model Comparision

Now, we will go into comparing all of these models between each other, specifically on the RMSE values on the testing data. We've seen up till now the similarities on how these models struggle with the testing data, but now let's compare them to see which ones do better relative to the others.

```r
# Store the error statistics for each model
arimaMAE <- MAE(.resid = arima_forecast_resid, .actual = test_GA)
arimaRMSE <- RMSE(.resid = arima_forecast_resid, .actual = test_GA)
arimaME <- ME(.resid = arima_forecast_resid, .actual = test_GA)

etsMAE <- MAE(.resid = ets_forecast_resid, .actual = test_GA)
etsRMSE <- RMSE(.resid = ets_forecast_resid, .actual = test_GA)
etsME <- ME(.resid = ets_forecast_resid, .actual = test_GA)

hwaddMAE <- MAE(.resid = hw_add_resid, .actual = test_GA)
hwaddRMSE <- RMSE(.resid = hw_add_resid, .actual = test_GA)
hwaddME <- ME(.resid = hw_add_resid, .actual = test_GA)

hwmultMAE <- MAE(.resid = hw_mult_resid, .actual = test_GA)
hwmultRMSE <- RMSE(.resid = hw_mult_resid, .actual = test_GA)
hwmultME <- ME(.resid = hw_mult_resid, .actual = test_GA)

nnetarMAE <- MAE(.resid = nnetar_forecast_resid, .actual = test_GA)
nnetarRMSE <- RMSE(.resid = nnetar_forecast_resid, .actual = test_GA)
nnetarME <- ME(.resid = nnetar_forecast_resid, .actual = test_GA)

prophMAE <- MAE(.resid = prophet_test_res, .actual = test_GA)
prophRMSE <-RMSE(.resid = prophet_test_res, .actual = test_GA)
prophME <-ME(.resid = prophet_test_res, .actual = test_GA)
```

```
combMAE <- MAE(.resid = combined_forecast_resid, .actual = test_GA)
combRMSE <- RMSE(.resid = combined_forecast_resid, .actual = test_GA)
combME <- ME(.resid = combined_forecast_resid, .actual = test_GA)
```

```
# Create a table storing the MAE, RMSE and ME values for each model and display them in ascending order

#NOTE: given the fact we did not choose between additive or multiplicative in the holt winters section,

model_names <- c("Arima", "ETS", "Holt Winters Additive", "Holt Winters Multiplicative",  "NNETAR", "Pr
mae_values <- c(arimaMAE, etsMAE, hwaddMAE, hwmultMAE, nnetarMAE, prophMAE, combMAE)
rmse_values <- c(arimaRMSE, etsRMSE, hwaddRMSE, hwmultRMSE, nnetarRMSE, prophRMSE, combRMSE)
me_values <- c(arimaME, etsME, hwaddME, hwmultME, nnetarME, prophME, combME)

# Create dataframe
results_df <- data.frame(Model = model_names, MAE = mae_values, RMSE = rmse_values, ME = me_values)

results_df <- results_df[order(results_df$RMSE), ]

# Output dataframe as a table
library(knitr) # for kable function
```

```
## Warning: package 'knitr' was built under R version 4.1.2
```

```
# Alternatively, you can use xtable package for more control over the table formatting
# library(xtable)

# Output table using kable
kable(results_df, caption = "Model Evaluation Metrics")
```

Table 1: Model Evaluation Metrics

	Model	MAE	RMSE	ME
3	Holt Winters Additive	970.5785	1271.149	455.9686
6	Prophet	1029.5575	1332.186	577.5092
4	Holt Winters Multiplicative	1046.8862	1353.048	565.4003
1	Arima	1156.5090	1473.696	784.1569
7	Combined	1286.8887	1612.802	900.3356
5	NNETAR	1716.9468	2129.426	1386.7151
2	ETS	1872.0649	2320.950	1632.2636

Here, we see that interestingly enough, the lowest RMSE, and also MAE and ME value was the Holt winters additive method. Given the fact that it had the lowest values across the board, it seems as though it is the clear preferred model over all the other models we tested in this project. We noticed with going through the testing for each model, that almost every model struggled with the volatility of the testing dataset, and we saw that the holt winters additive and multiplicative trend was better able to model that stark increase, which most likely is why it had the best error statistics.

Overall, we can see that we would choose the Holt Winters Additive method as our model of choice for forecasting Georgia monthly imports.

Conclusions and Future Work

In conclusion, we fitted 6 main models to Monthly Georgia imports from January of 2010 to December of 2023. These models were: an arima model, an ETS model, a Holt-Winters model(multiple types), a Prophet model, a neural network autoregression model, and a combination of these models averaged into one.

To assess our model performance, we used a training testing split, with our training period until 2018, and our testing period from 2019-2023. We evaluated our forecasts and model fit based on the MAE, RMSE, and ME values and overall, ended up coming to the conclusion as mentioned that our Holt-Winters Additive Method was our best and preferred model. One major source of issue was the fact that the data from 2019-2023 was quite a bit more volatile and pretty stark compared to our training data, thus that produced quite a bit of error with our models. It is important to take that into consideration when analyzing the model's performance on the testing data.

The discrepancy between the training data and testing data was our largest potential source of error when looking at the forecast performances. This discrepancy is likely due to the COVID-19 pandemic that had massive economic impact and shock.

As a result, going forward in terms of future work, it would be helpful to get future data that has a bit more structure over time and not such a big discrepancy like our data, although we want to make sure that we have some volatility in the data so we can see how our models account for that. Furthermore, future work in modeling and forecasting imports would be very beneficial on an economic note in terms of forecasting and modleing economic activity and gaining insight. ## References

Georgia monthly time series of imports data source: https://fred.stlouisfed.org/series/IMPTOTGA