Logistics - Predicting Demand of Key Ingredient

Morgan Kennedy

Overview of the Problem and Analysis

Data is provided on orders made between March and June at four stores of a fast-food chain restaurant. Two of these stores are located in New York and two in California. The data is split into multiple small databases containing information on each order made, the recipes and sub-recipes of menu items and the ingredients used in each of the recipes. The objective of this time series analysis is to develop accurate forecasts of lettuce demand for each of the four stores for the next two weeks.

The following steps will be used to analyse and forecast the time series data: 0. Pre-Processing and Data Wrangling - Manipulating the data sets provided to make it more appropriate for the analysis to be done. The end product of this step will be separate data sets with daily lettuce demand from each of the four stores.

1. Data Exploration and Visual Analysis - Exploring and visualizing the data to understand how it changes over time and to form hypothesis about trend and seasonal components

2. Model Building - The data will be split into training and test sets and a model will be built on the training data. These models will be built using best model detection algorithms, that will identify the model that best meets a particular criterion, along with models suspected of fitting the data well based on earlier visual analysis.

3. Forecasting - The models built on the training data will be used to forecast the test data.

4. Performance Evaluations - Model performance will need to be evaluated on in sample and out of sample data. The in-sample performance is measured on the training data, while the out of sample performance is measured against the test data. An appropriate error metric will be used to compare a model's performance

5. Model Selection - Once the models have been compared, the best performing model will be selected for that particular store. The selected model will be used to forecast future lettuce demand.

Introduction and Data Processing Overview

In simple terms, a time series is a sequence of data points observed over a period of time. When conducting time series analysis it is general understood that for some data, the observed demand = systematic component + random component. In time series analysis, the goal is to focus on trying to forecast the systematic component and that random component is any part of the forecast that deviates from the systematic part. The random part should not be forecasted.

The systematic component consists of three possible parts, the level (current demand), the trend (the rate of growth of decline) and seasonality (predictable seasonal fluctuations). The goal in time series analysis is to identify which of these components are present in the data and forecast estimates that can be used to approximate future observations.

The order data for each of the four restaurants was provided and broken down into 11 data sets. The data provided was aggregated to provide the daily demand for lettuce in each of the four stores. This process was done in python and is provided in a separate file. Below is a detailed overview of the steps involved in the processing: The required data was built from the smallest component, ingredients, up to the order level.

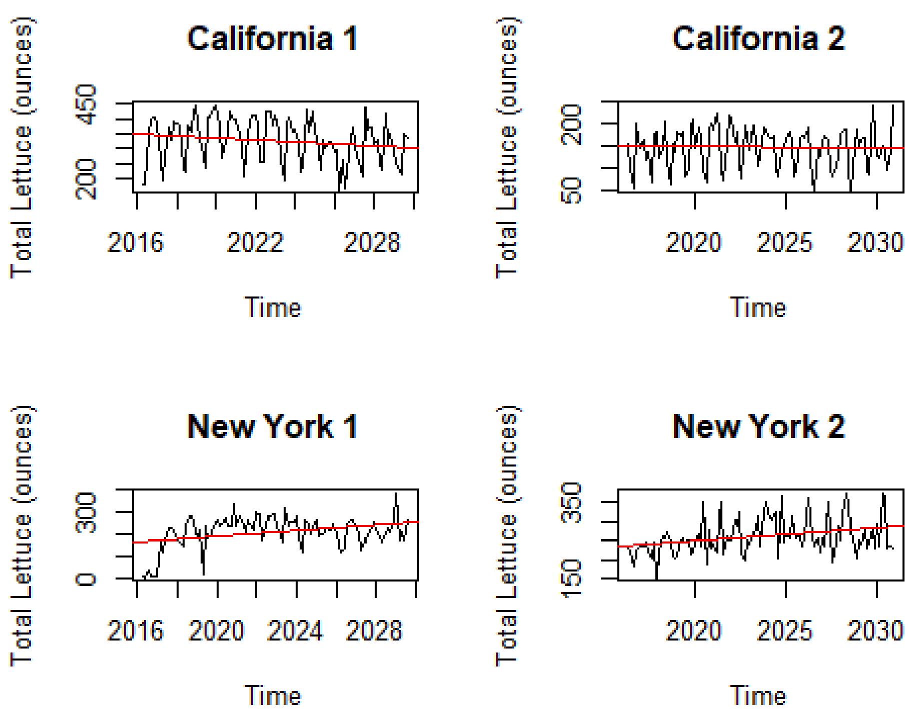
1. Identify the ID of lettuce in the ingredients dataframe
2. Identify all the recipes with sub-recipes that include lettuce
3. Find the total amount of lettuce used in these sub-recipes
4. Identify the recipes that involved lettuce and the total amount of lettuce used.
5. Join the two data frames together - amount of lettuce used in the sub-recipes and the amount of lettuce used in the recipes to get the total amount of lettuce used in an overall recipe.
6. Based on menu orders, identify the amount of lettuce required per order.
7. Group the data based on store number and data and aggregate, sum, the total lettuce required for orders on that day.
8. Separate the data based on store number and convert the data frame to a csv file.

Data Exploration and Visual Analysis

In order to have a better understanding of the data in terms of trend and seasonality we need to visualize the data for each store and observe how the data changes over time. In order to do this we must first convert each dataframe into a time series object. Since this is daily data for just over 3 months, 94 - 105 days, we can look for weekly seasonality.

```
#Converting the data sets into time series objects with a frequency of 52
(weekly)
calif_1_ts <- ts(calif_1[,c('Total_Lettuce')], frequency = 365, start =
c(2015,10))
calif_1_ts <- ts(as.vector(calif_1_ts), frequency = 7, start = c(2015, 10))

calif_2_ts <- ts(calif_2[,c('Total_Lettuce')], frequency = 365, start =
c(2015,10))
calif_2_ts <- ts(as.vector(calif_2_ts), frequency = 7, start = c(2015,10))

newY_1_ts <- ts(newY_1[,c('Total_Lettuce')], frequency = 365, start =
c(2015,10))
newY_1_ts <- ts(as.vector(newY_1_ts), frequency = 7, start = c(2015,10))

newY_2_ts <- ts(newY_2[,c('Total_Lettuce')], frequency = 365, start =
```

```
c(2015,10))
newY_2_ts <- ts(as.vector(newY_2_ts), frequency = 7, start = c(2015,10))

#Creating a 2x2 frame to plot featuring daily demand of lettuce in each of
the 4
#stores along with a fitted regression line for each store.

par(mfrow=c(2,2))
plot(calif_1_ts, main = 'California 1', ylab ='Total Lettuce (ounces)') +
  abline(reg= lm(calif_1_ts~time(calif_1_ts)), col = 'red')

## integer(0)

plot(calif_2_ts, main = 'California 2', ylab ='Total Lettuce (ounces)') +
  abline(reg= lm(calif_2_ts~time(calif_2_ts)), col = 'red')

## integer(0)

plot(newY_1_ts, main = 'New York 1', ylab ='Total Lettuce (ounces)') +
  abline(reg= lm(newY_1_ts~time(newY_1_ts)), col = 'red')

## integer(0)

plot(newY_2_ts, main = 'New York 2', ylab ='Total Lettuce (ounces)') +
  abline(reg= lm(newY_2_ts~time(newY_2_ts)), col = 'red')
```




```
## integer(0)
```

The above plots show the daily demand for lettuce in ounces in 4 stores of a popular fast food restaurant. The red line in each plot is a linear regression line fitted to the data which should show the general trend of the data. Across all four stores there seems to be semi regular rising and falling of demand which may indicate some seasonality in the data.

The store labelled as California 1 seems as it would have the highest total demand with daily demand ranging from a low of 200 to over 400 ounces on some days. However, it has very distinct seasonality type pattern which may indicate that seasonality would be an important part of the time series model that will be fit later on. In addition the downward slopping regression line indicates that there is a declining trend in the data.

The range of demand for California 2 seems to be lower ranging from 50 to 250 ounces. This store has a similar seasonality pattern to what was observed in the other california location. However, the regression line is flat, gradient close to 0. This may indicate that there is no trend in the data.

At the start of the New York 1 data, the values are very close to 0 which is inconsistent which the other data points in the series. Before continuing with the analysis it is likely best to remove these values so that the model fitted to the data is a better representation of what is present at the store. Ignoring the first few data points, that distinct and regular peaks and valleys seen in the California stores data, it not as present. While there are some general rises and falls they do not seem to occur at regular intervals. This may indicate that the seasonal component of the time series is not very strong. Based on the fitted regression line, there is an indication that the trend is increasing. However, this may have been influenced by the abnormally low levels at the beginning of the data and further exploration will need to be done to see if this trend is accurate.

Similar to New York 1, while there are some distinct rises and falls in the data, it is not as regular as in the California data sets. Thus while there may be seasonality present in the data, it may not be a significant component of the forecasted demand function. In addition the upward slope of the fitted regression line indicates that there may be some increasing trend present in the data.

Further analysis will need to be done to confirm the conjectures made from visually analysing these plots.

```
#Checking the data for New York 1 for abnormalities
head(newY_1_ts, 15)
```

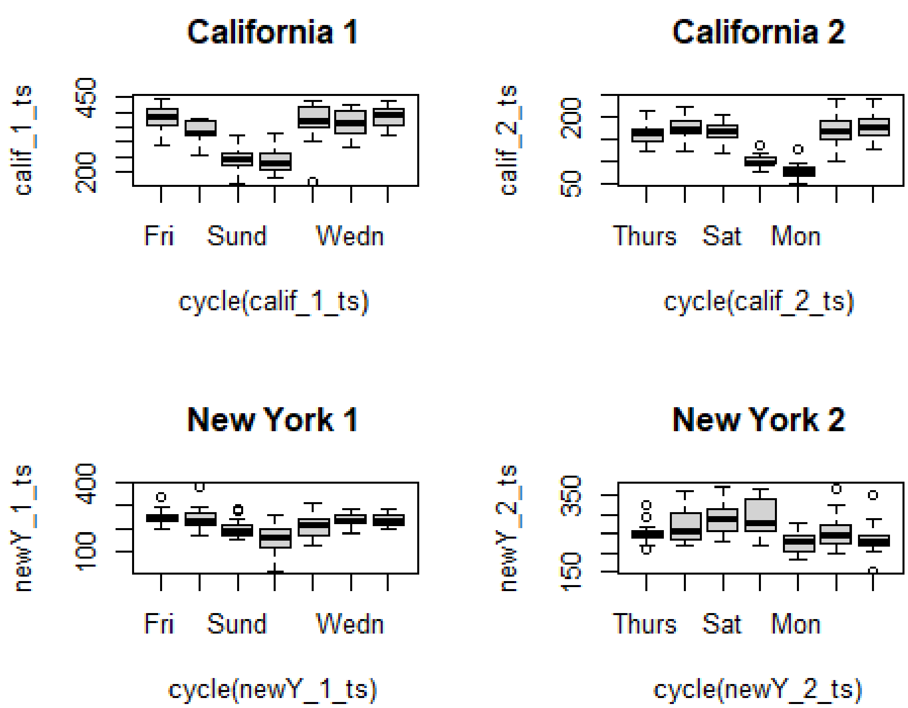
```
## Time Series:
## Start = c(2016, 3)
## End = c(2018, 3)
## Frequency = 7
##  [1]   4   0  29   6   8   4 164 112 168 222 222 194 172 158 142
```

Looking at the first 6 values of the New York 1 data set, we can see that these values are unusually low and do not coincide with other parts of the data. So that these values do not negatively influence modelling in the future we will remove them and start with the 8th daily demand entry.

```
newY_1_ts <- ts(newY_1[-c(1:6),c('Total_Lettuce')], frequency = 1, start =
c(2015,10))
newY_1_ts <- ts(as.vector(newY_1_ts), frequency = 7, start = c(2015,10))
head(newY_1_ts, 15)

## Time Series:
## Start = c(2016, 3)
## End = c(2018, 3)
## Frequency = 7
##  [1] 164 112 168 222 222 194 172 158 142 238 282 276 250 184 212
```

We can also look for seasonality factors by using a boxplot to observe the distribution of demand on each of the 7 days of the week.

```
par(mfrow=c(2,2), srt = 45)

boxplot(calif_1_ts~cycle(calif_1_ts), main = 'California 1', names = c('Fri',
'Sat', 'Sund', 'Mon', 'Tues', 'Wedn', 'Thurs'))
boxplot(calif_2_ts~cycle(calif_2_ts), main = 'California 2', names =
c('Thurs', 'Frid', 'Sat', 'Sun', 'Mon', 'Tues', 'Wedn'))
boxplot(newY_1_ts~cycle(newY_1_ts), main = 'New York 1', names = c('Fri',
'Sat', 'Sund', 'Mon', 'Tues', 'Wedn', 'Thurs'))
boxplot(newY_2_ts~cycle(newY_2_ts), main = 'New York 2',names = c('Thurs',
'Frid', 'Sat', 'Sun', 'Mon', 'Tues', 'Wedn') )
```



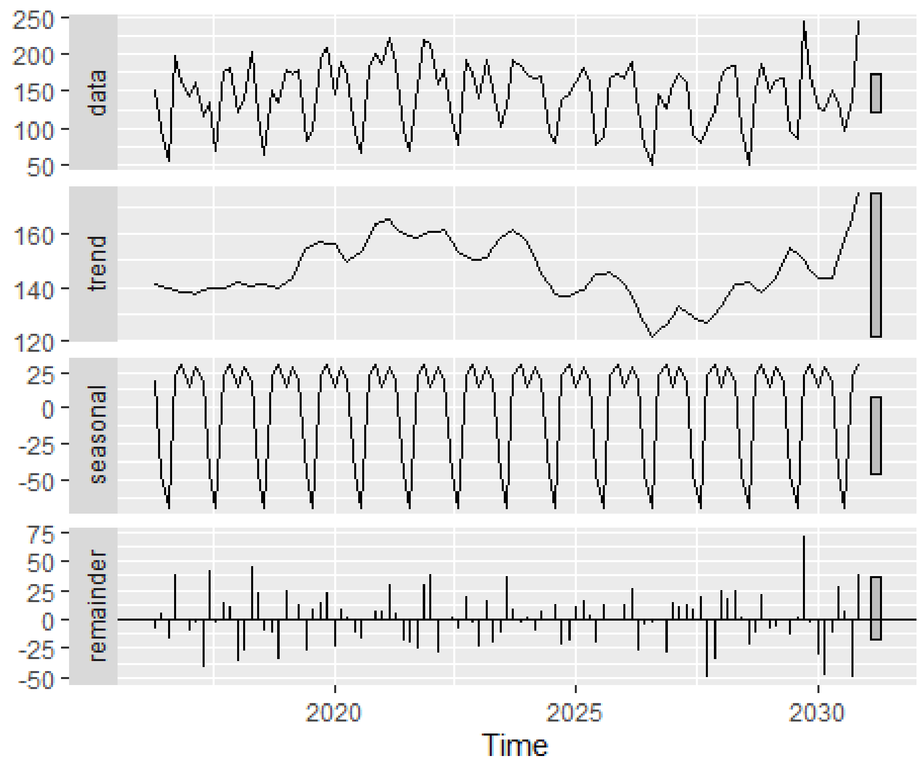
The above boxplots shows the distribution of demand at each store on different days of the week. As we can see based on the thick line of the box plot, which indicates median demand for that week day based on the data, the distribution of demand generally differs day to day and from store to store. Observing these graphs should provide further evidence of seasonality that may have been observed in the line plot above.

We can see that the seasonality differs from store to store. * California 1 - There is a distinct pattern shown in the data. Demand is its lowest on Sundays and Mondays than begins to increase and is relatively high for the rest of the week before beginning to fall on Friday and Saturday. * California 2 - There is a similar seasonal pattern as described in California 1. Demand is at the lowest on a Monday. On Tuesday, demand surges and is high for the following 4 days before falling significantly on Sunday before hitting the Monday lows. * New York 1 - There is minimal variability in the median demand per day. The lowest demand is on Sunday but there is no significant rise or fall in demand levels on the other days of the week. This may indicate that there is little seasonality effects in the time series. * New York 2 - There is a bit more variability in the demand than in the New York 1 store. On average, demand is its highest on a Saturday and its Lowest on a Monday. Nonetheless the variability in average demand is low and the distribution of demand on each day varies widely and is more than the other stores. This may indicate that there is little seasonality effects in the time series.

Summary of Observations in terms of the systematic component of the time series: * California 1- Slightly decreasing trend, high seasonality * California 2 - Very little to no trend, high seasonality * New York 1 - Slight upward trend, little to no seasonality * New York 2 - slightly upward trend, little to no seasonality

From here on, analysis will be done on a per store basis starting with California 2, store number 46673. ## California 2 - Store Number 46673 #### Step 1: Data Exploration A more in depth analysis cna be done on the trend and seasonal effects on the time series by decomposing it using the stl function. This function will also show the relative importance of each component.

```
calif_2_ts %>% stl(s.window = "periodic") %>% autoplot
```



The above graphs show a breakdown of the trend and seasonal components of the time series along with the remainder once these factors are accounted for. We can judge the relative importance, or the size of the effect each of these components has on the data, based on the bar on the right of each graph. The smaller the bar, the more critical this component is in the data, or the more variability in the data it is able to account for. The seasonal and remainder components have bars of roughly the same size. This means that the data has strong seasonal effects along with strong randomness. The size of the trend bar is vary large in comparison implying that trend is not as signfiicant to the time series. This is inline with the observations made earlier from the plot of the data.

Step 2: Model Building

Now that we have a better understanding of the data, we can try to fit time series models on the data. Before fitting the models, we need to separate the data into a train and test set. The train set will be used to fit the model. The test set will be used to compare the results of the two models that we fit to determine which model is the most appropriate for the data. Once the best model is chosen, we can then use this model to make forecasts of future demand.

```
#Splitting the data into a roughly 80-20 training-test split
calif_2_ts_train <- window(calif_2_ts, end = c(2027, 11))
calif_2_ts_test <- window(calif_2_ts, start = c(2027, 11))
```

Part a: Exponential Smoothing

We can first try to fit a model that uses simple exponential smoothing as the underlying algorithm for fitting its data. The Holt Winter’s Model is an extension of the simple exponential smoothing model to correct for trend and seasonality that may be present within the data. In the original model of simple exponential smoothing, there is assumed to be a constant level, as the data seems to have no observable trend or seasonality. This assumed level changes in each period after a demand level is observed by taking a weighted average of the observed demand and the forecasted level. The weights are determined by α, the smoothing constant for the level. The Holt winters model incorporates two more smoothing parameters, β and γ, the smoothing components of trend and seasonality respectively. Similar to the original model, the Holt Winters model updates its estimates for trend and seasonality by taking a weighted average of the observed trend and seasonality along with the forecasted trend and seasonality components for that period. Exponential smoothing methods can vary widely accounting for different characteristics of the trend, seasonality and error components. Each of the components can be of three different forms None, Additive, or Multiplicative allowing for 27 variations of the model that can be fitted to the data. The ets() function can build a model using estimates of the smoothing parameters α, β and γ, based on a specified model or it can try multiple combinations to determine the best model and most appropriate parameters to achieve some criterion. Or based on a particular form of the model, it can can estimate the best values of each parameter than will allow the model to achieve some criterion, either minimising the sum of squared error, or by maximizing the 'likelihood' (the probability that the data arises from the specified model) of the model. Further to that if the model specified is 'ZZZ' the aglorithm will also try to decide on which type of model should be fitted along with the best parameter values. The ets function will generally use the Akaike Information Criterion (AIC) score to choose the best performing model. The AIC is an estimator of prediction error and is calculated as $AIC = -2ln(L) + 2k$ where L is the likelihood and the model and k is the number of parameters. The higher the maximum likelihood the better fit a model provides to the data. Since in the AIC the log of the likelihood is used, the 'best model' will be the one with a low AIC

```r
calif_2_ts.ets1 <- ets(calif_2_ts_train, model = "ZZZ")
calif_2_ts.ets1
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = calif_2_ts_train, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.0976
##     gamma = 1e-04
##
##   Initial states:
##     l = 142.5273
##     s = 31.2824 18.4637 25.4649 20.4672 -67.7692 -46.6788
##            18.7697
##
##   sigma:  24.5387
##
```

```
##      AIC     AICc      BIC 
## 944.0108 946.9441 968.5542
```

The ets model has found an 'ANA' model to be the best for the data. This means an additive error firm (the first 'A'), no trend component (the 'N') and an additive seasonality (the second 'A'). An additive error means that the size of the remainder is not dependent on the period of the time series. There is no significant trend in the data to be accounted for and so the model performs best without accounting for a trend. An additive seasonality means that the size of the seasonal component is independent of the trend in the data. This is inline with the observations made earlier on the data and the decomposed elements. The smoothing parameters are $\alpha = 0.1223$ and $\gamma = 10^{-4}$. The model has also estimated initial values for the level and each of the seasonal states as shown above.

We can compare this other variations of the model to see how a different type of model may perform in sample and how this compares to the 'best' model out of sample.

Model 2-> To verify that trend is not there in the data, a model including trend as an addiitive component can be utilized. model = 'AAA'

```
calif_2_ts.ets2 <- ets(calif_2_ts_train, model = 'AAA')

print(calif_2_ts.ets2)
```

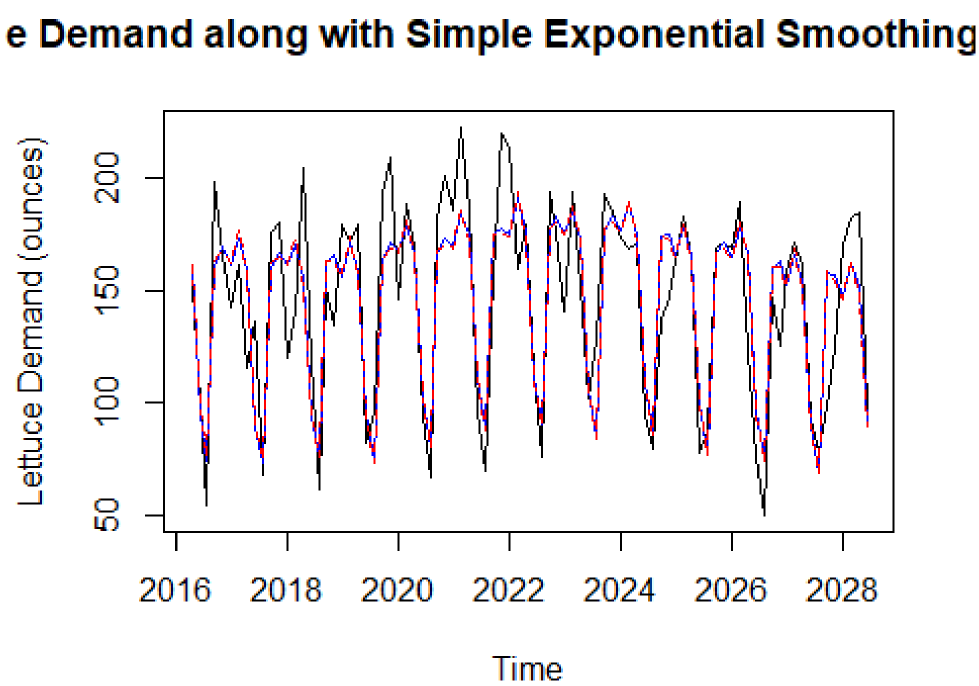
```
## ETS(A,A,A) 
## 
## Call:
##  ets(y = calif_2_ts_train, model = "AAA") 
## 
##   Smoothing parameters:
##     alpha = 0.0974 
##     beta  = 1e-04 
##     gamma = 1e-04 
## 
##   Initial states:
##     l = 144.6748 
##     b = -0.0609 
##     s = 34.0013 18.0709 23.1112 21.5881 -67.4612 -46.1644
##            16.8541
## 
##   sigma:  24.9205
## 
##      AIC     AICc      BIC 
## 948.4025 952.6765 977.8547
```

Model 2 -> $\alpha = 0.0452$ and $\gamma = 10^{-4}$ Model 3 -> $\alpha = 0.0974$, $beta = 10^{-4}$ and $\gamma = 10^{-4}$

The γ term in each of the models is the same and the initial states for trend and seasonality have very little variability. In fact, the performance of each model as measured by the AIC are very similar. However as expected, the model that was selected by the ets function has the lowest AIC and this would be the best model.

We can demonstrate the performance of these models on the train set by first plotting each model and seeing how well they approximate the original demand.

```
plot(calif_2_ts_train, xlab = 'Time', ylab='Lettuce Demand (ounces)', main =
'Daily Lettuce Demand along with Simple Exponential Smoothing Model
Variants')
lines(fitted(calif_2_ts.ets1), col = 'blue')
lines(fitted(calif_2_ts.ets2), col = 'red', lty = 2)
```



Explanation of the Graph:

The blue line represents the best fit model while the red dotted line represent model 2 #. As would be expected the two models track each other fairly well as the parameters estimated by the ets function were very similar, except for α, the smoothing parameter of the level. The models are able to explain a lot of the variation in the time series but generally underestimate the data, when it is at its peaks. That is the estimates for the 4th, 5th and 6th peak are below the observed levels. There are other points in the data where the forecasts are above observed values and this is found earlier in the data. There are a few areas that the blue line is slightly closer to the actual level rather than the red line. This implies that the best fit model is able to fit the data slightly better than the model with an additive trend component included.

Part b: ARIMA Model - Auto Regressive Integrated Moving Average Model

The second type of model that will be fitted to the data is an ARIMA model. ARIMA models are another approach for time series forecasting. While exponential smoothing focuses on trying to estimate the trend and seasonality of the model, ARIMA models try to account for autocorrelations of the data. Autocorrelation in a time series represents the degree of similarity between a time series and a lagged version of itself. The overall ARIMA model comes from the combination of two types of models the Autoregressive Models (AR) and the Moving Average Model (MA), along with differencing.

The AR model uses multiple linear regression to forecast the variable as a linear combination of past values of the series. It takes parameter p which refers to the number of lags that will be incorporated in the model. The MA model builds a regression using the moving average of past forecast errors. The number of past errors used in the model is determined by the parameter q. The ARIMA model combines both of these models along with a differencing of the data. Differencing is way of making a non-stationary time series, a time series whose properties are dependent on the time period, to a stationary one. Differencing can help to stabilize the mean of a time series by removing changes in the level and thus eliminating or reducing trend and seasonality. The differencing done to the data will inform the parameter d.

The combination of AR, MA and difference produces a non-seasonal ARIMA model. In order to determine the most appropriate ARIMA model, we will ensure that the time series is stationary and then identify the parameters p and q using the ACF and PACF plots.

Procedure for creating an ARIMA model ; In order to determine the parameter d, we will need to first identify if the time series is stationary. If not, we will need to difference the data till it is stationary. We will conduct the Augmented Dickey–Fuller (ADF) test to test for a unit root and thus trend stationarity and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for level stationarity.

For the ADF Test: $H_0: $ The unit root is present in the time series sample $H_1: $ The time series sample is trend stationary

For the KPSS Test: $H_0: $ The time series is level stationary $H_1: $ The time series is not level stationary

```
adf.test(calif_2_ts_train)

## Warning in adf.test(calif_2_ts_train): p-value smaller than printed p-
value

##
##  Augmented Dickey-Fuller Test
##
## data:  calif_2_ts_train
## Dickey-Fuller = -7.7668, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary

kpss.test(calif_2_ts_train, null="Level")

## Warning in kpss.test(calif_2_ts_train, null = "Level"): p-value greater
than
## printed p-value
```

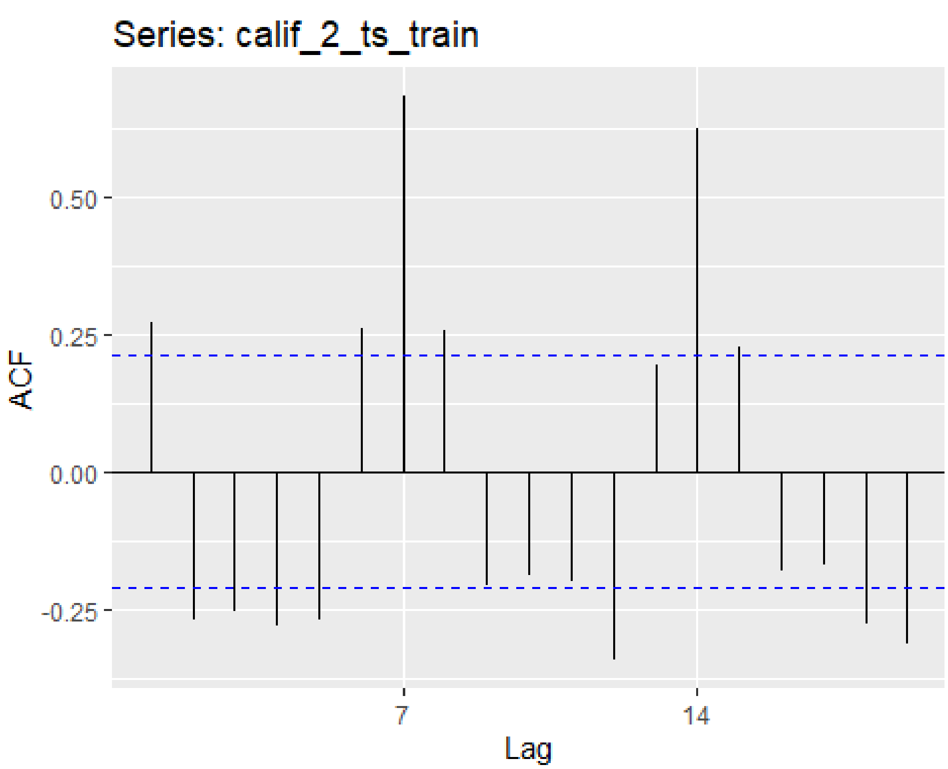
```
##
##  KPSS Test for Level Stationarity
##
## data:  calif_2_ts_train
## KPSS Level = 0.13795, Truncation lag parameter = 3, p-value = 0.1
```

The p-value of the ADF test is less than 0.01, which is less than the chosen significance level of 1%. Since the p-value is below the significance level, we reject the null hypothesis in favor of the alternative and therefore the time series seems to already be trend stationary. The p-value of the KPSS test is 0.1 which is higher than the significance level of 1%. Since the p-value is greater than the significance level, we fail to reject the null hypothesis and thus the time series seems to be level stationary.

Since the time series is already stationary, we do not need to difference the data and therefore the ARIMA parameter $d = 0$.

We can now use this time series to estimate that parameters p and q for the model. To estimate parameter q we will use the auto-correlated function (ACF).

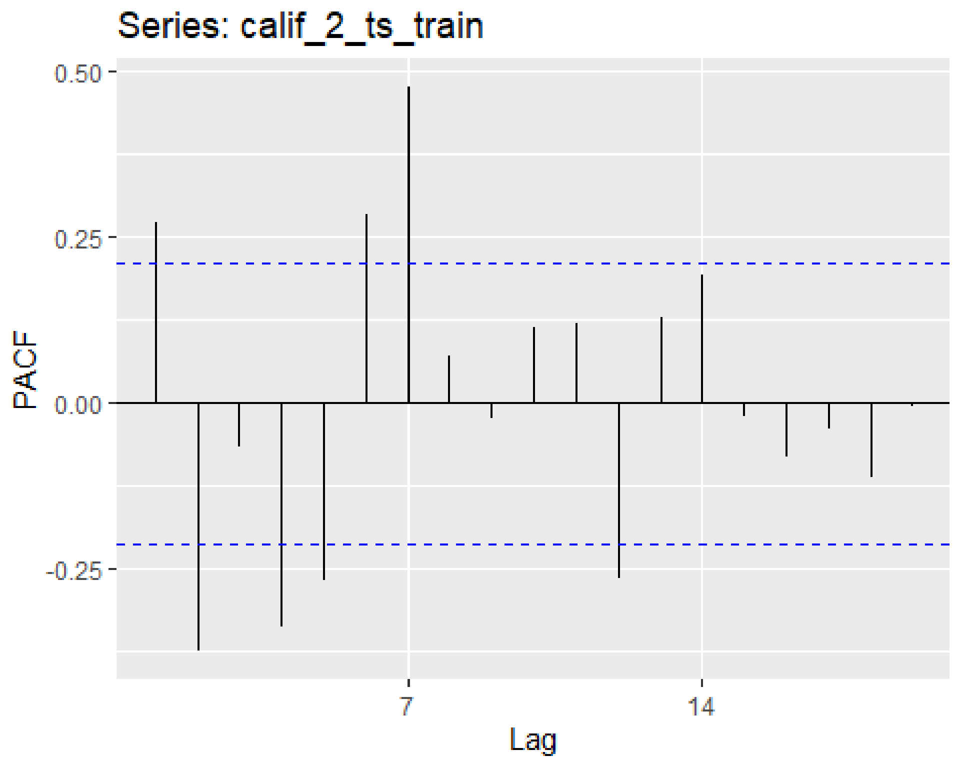
```
ggAcf(calif_2_ts_train)
```



The vertical lines, spikes, that exceed the horizontal blue lines are considered significant. Of the significant spikes, the ones at lags 7 and 14 are higher than the rest implying that there is some seasonality that needs to be accounted for in the model. The seasonality is at every 7 lags. In addition, the spikes at these lags (lags 7 and 14) are decreasing but still significant. There is also geometric decay in the ACF for Lag 1 and Lag 12 (at lag 8, 15 and 14). $q = 0$

To estimate the parameter p, the Partial Autocorrelated Function (PACF) will be used.

```
ggPacf(calif_2_ts_train)
```



The graph of the PACF also oscillated between positive and negative values. There are consistently significant spikes up to and including lag 7 and the majority of spikes after that point are insignificant. Spikes at lags of 7 decrease geometrically. Since there is an oscilation of significant spikes in the ACF graph and outside of the lags mentioned earlier as having geometric decay in the PACF graph has significant spikes up to lag _ therefore $q = 0$ and $p = $

Therefore this is a seasonal ARIMA model with a moving average component $(0, 0, 1)$. The non seasonal part of the ARIMA model has parameters $d = 0$, $q = $ and $p = $.

Based on the values of p, q, d detrmined above along with observations on the spikes in the ACF and the PACF, we can determine that the ideal model is a seasonally adjusted ARIMA model of the form ARIMA(1,0,0)(0,0,1)[7]

We can also use the auto.arima() function to estimate the best parameters of the ARIMA function using the information criterion. Therefore it seems that we need to fit a ARIMA model to the data.

```
auto.arima(calif_2_ts_train , d =0)
```

```
## Series: calif_2_ts_train
## ARIMA(1,0,0)(0,1,1)[7]
##
## Coefficients:
```

```
##          ar1     sma1
##       0.1783  -0.7183
## s.e.  0.1128   0.1209
##
## sigma^2 = 672.1:  log likelihood = -370.8
## AIC=747.6   AICc=747.92   BIC=754.71
```

The auto.arima() function has identified a model ARIMA(1,0,0)(0,1,1)[7] as the best fitting functions while the visual analysis done above has ARIMA(1,0,0)(0,0,1)[7] as the best function. We will fit the two ARIMA models and evaluate their performance.

```
#Fitting the three ARIMA models
calif_2_ts.arima1 <- Arima(calif_2_ts_train, order = c(1, 0, 0), seasonal =
c(0, 1, 1))
calif_2_ts.arima2 <- Arima(calif_2_ts_train, order = c(1, 0, 0), seasonal =
c(0, 0, 1))

#Calculating the MSE on these models
MSE_calif_2_AR1 <-
sqrt(sum(calif_2_ts.arima1$residuals^2)/(length(calif_2_ts_train)-2))
MSE_calif_2_AR2 <-
sqrt(sum(calif_2_ts.arima2$residuals^2)/(length(calif_2_ts_train)-2))
```

A critical component of how well a time series model is performing is how well it forecasts unseen data. That is the out of sampling evaluation. To do this, for each of the 5 models created above, we will forecast the next 23 days, which would be the equivalent to the test data. We can then compare how accurate the model forecasts the test data.

####Step 3:Forecasting

```
#Forecasting the next 18 days in each of the 5 models and then calculating
the accuracy of each forecast.
n_calif_2_test <- length(calif_2_ts_test)
calif_2_ets1.forecast <- forecast(calif_2_ts.ets1, h=n_calif_2_test)
calif_2_ets2.forecast <- forecast(calif_2_ts.ets2, h=n_calif_2_test)

calif_2_AR1.forecast <- forecast(calif_2_ts.arima1, h=n_calif_2_test)
calif_2_AR2.forecast <- forecast(calif_2_ts.arima2, h=n_calif_2_test)

Acc_calif_2_ets1 <- t(accuracy(calif_2_ets1.forecast,
calif_2_ts_test))[2,c('Test set')]
Acc_calif_2_ets2 <- t(accuracy(calif_2_ets2.forecast,
calif_2_ts_test))[2,c('Test set')]
Acc_calif_2_AR1 <- t(accuracy(calif_2_AR1.forecast,
calif_2_ts_test))[2,c('Test set')]
Acc_calif_2_AR2 <- t(accuracy(calif_2_AR2.forecast,
calif_2_ts_test))[2,c('Test set')]
```

####Step 4: Performance Evaluation

We can begin evaluating which model would be the best/ most appropriate for the data by measuring the forecast error. A good forecasting method should aim to only capture the systematic component of the time series and not the random component. The In-Sample error is based on the fitted values the model has on the train data. The most popular approach for measuring forecast errors is using the mean squared error. The mean squared error is most suited for data where the cost of a large error is larger than gains from very accurate data as there is a penalty built into the formula for errors that are 'very large'. This an appropriate measure for this item, lettuce as there are significant costs involved in either significantly overestimating or underestimating demand: These are described below:

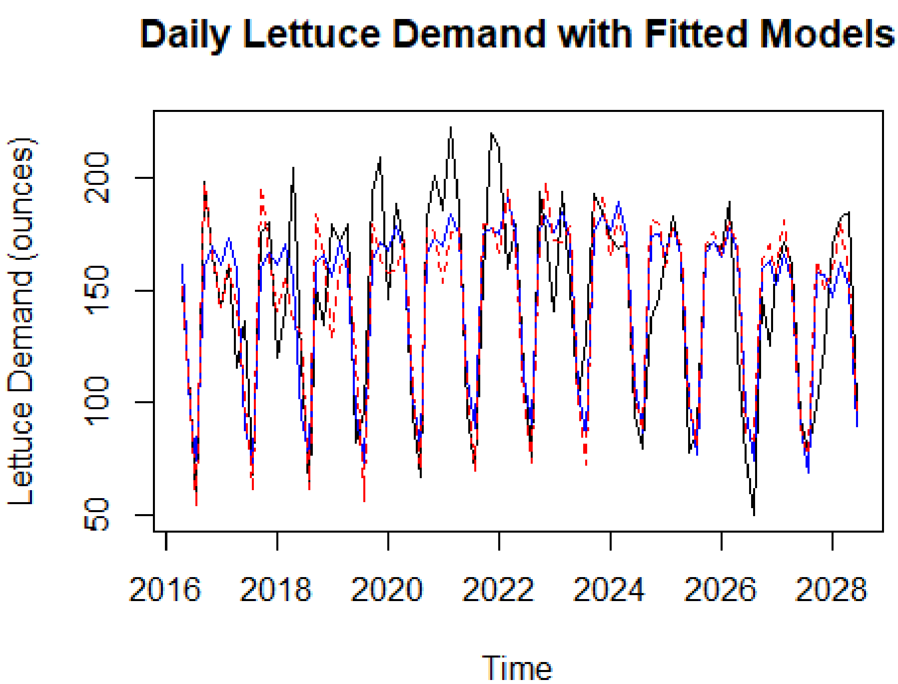
1. Overestimating demand - Lettuce is a fresh vegetable and is highly perishable. On average it is estimated that a head of lettuce will stay fresh and crisp, and therefore in ideal condition for a food restaurant, for 7- 10 days. Therefore if the company were to overestimate demand and therefore order too much such that a head of lettuce may stay beyond its lifespan without being used, then the restaurant would be unable to serve it and therefore it would not be able to generate revenue from the lettuce and only costs.

2. Underestimating demand - Lettuce is generally used as a garnish a dish or in fast food in burgers, sandwiches, salads and similar dishes. While the food can be prepared without lettuce, not having it may turn away customers and therefore a lack of lettuce may cause the company to actually lose out on revenue.

```
#We can begin evaluating the mean squared errors for each of the models
fitted above
MSE_calif_2_ets1 <-
sqrt(sum(calif_2_ts.ets1$residuals^2)/(length(calif_2_ts_train)-2))
MSE_calif_2_ets2 <-
sqrt(sum(calif_2_ts.ets2$residuals^2)/(length(calif_2_ts_train)-2))
#We will save these estimates later for a comparison of all models

model_names_c2 <- c('ETS Best - ANA', 'ETS - AAA',
                     'ARIMA Best - (1,0,0)(0,1,1)[7]', 'ARIMA -
(1,0,0)(0,0,1)[7] ')


RMSE_train_c2 <- c(MSE_calif_2_ets1, MSE_calif_2_ets2 ,MSE_calif_2_AR1,
MSE_calif_2_AR2)

RMSE_test_c2 <- c(Acc_calif_2_ets1, Acc_calif_2_ets2, Acc_calif_2_AR1,
Acc_calif_2_AR2)


data.frame(model_names_c2, RMSE_train_c2, RMSE_test_c2)

##                    model_names_c2 RMSE_train_c2 RMSE_test_c2
## 1                   ETS Best - ANA      23.49405     35.66287
## 2                        ETS - AAA      23.54763     36.62440
```

```
## 3 ARIMA Best - (1,0,0)(0,1,1)[7]      24.82203     41.18578
## 4     ARIMA - (1,0,0)(0,0,1)[7]       36.13760     46.55152
```

```
plot(calif_2_ts_train, xlab = 'Time', ylab='Lettuce Demand (ounces)', main =
'Daily Lettuce Demand with Fitted Models')
lines(calif_2_ets1.forecast$fitted, col = 'blue')
lines(calif_2_AR1.forecast$fitted, col = 'red', lty = 2)
```



The black line represents the original model data. The blue line is for the Exponential smoothing model while the red dotted line represents the best fit ARIMA model. The grpah shows that both models fit the data differently. The ARIMA has a tendency to have more extreme values for demand on both the high and low side, particularly at the beginning of the time series. In the middle of the series, both models fit similarly in places. This explains the very similar in sampling training RMSE. However, on the out of sample training, the RMSE of the ARIMA model of 41.18 is larger than the RMSE of the ets model, 35.66. Due to the increased disparity between the in sample and out of sample error, the ARIMA model is more than likely fitting more of the randomness, or noise, in the series over the exponential smoothing model that does not seem to capture as much noise. Therefor the best model for this store is the exponential smoothing model with an additive error component, no trend and an additive seasonality. Starting off by comparing the best Exponential Smoothing and ARIMA models (ETS Best and ARIMA Best. On the train data the ETS Model offers lowest forecast error, comparing the RMSE, in both the train and test data and would therefore be the best model for future predictions.

####Step 5: Model Selection

```r
#Fitting the model to all of the data available
calif_2_model <- ets(calif_2_ts, model = 'ANA')
calif_2_model
```

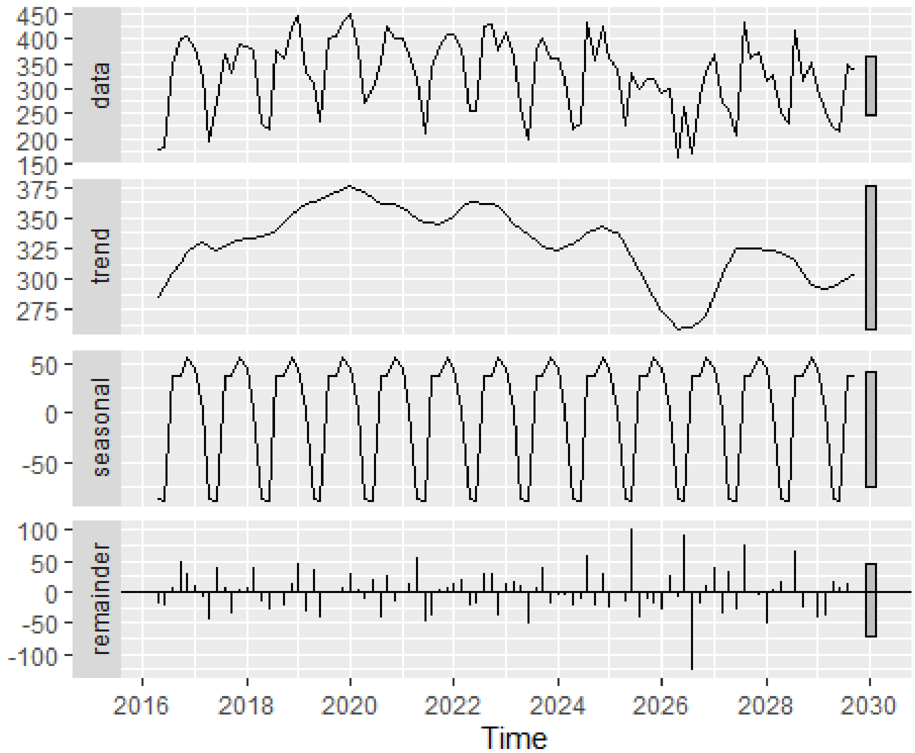
```
## ETS(A,N,A)
##
## Call:
##  ets(y = calif_2_ts, model = "ANA")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 145.9345
##     s = 27.1516 13.8629 29.8353 24.1902 -69.0184 -44.6583
##            18.6367
##
##   sigma:  26.6343
##
##      AIC     AICc      BIC
## 1164.093 1166.484 1190.440
```

Using the similar procedure, models can be built for each of the other stores.

California 1 - Store Number 4904

Step 1: Data Exploration

```r
calif_1_ts %>% stl(s.window = "periodic") %>% autoplot
```



Based on the decomposition above each component of the data can be ranked in terms of its significance, how much of the variation in the data it is able to explain. In terms of significance, the remainder component is the most important in understanding the variation, the seasonality is of second highest importance while the trend is the least important component.

Step 2: Model Building

```
#Splitting the data into a roughly 80-20 training-test split
calif_1_ts_train <- window(calif_1_ts, end = c(2027, 1))
calif_1_ts_test <- window(calif_1_ts, start = c(2027, 1))
```

Part a: Exponential smoothing models.

Model 1 -> Model determined by the ets function as having the best information criterion

```
calif_1_ts.ets1 <- ets(calif_1_ts_train, model = "ZZZ")
calif_1_ts.ets1

## ETS(A,A,A)
##
## Call:
##  ets(y = calif_1_ts_train, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.0127
##     beta  = 0.0125
##     gamma = 5e-04
```

```
##
##   Initial states:
##     l = 315.9648
##     b = 3.6247
##     s = 8.1102 50.6401 57.8214 40.4994 18.3055 -80.5061
##            -94.8704
##
##   sigma:  42.8103
##
##      AIC     AICc      BIC
## 912.2838 917.2362 940.2526
```

The best fit model is one with additive error, trend and seasonality. The values of the model parameters are $\alpha = 0.0127, \beta = 0.0125$ and $\gamma = 5x10^{-4}$. This model therefore asserts that it is critical to include trend and seasonality in the model unlike the California 2 model that implied that there was no trend in the data.

Model 2 -> 'ANA' - Since the trend did not account for much variation it may be useful to look at a model without this component and see its performance.

Model 3 -> 'ANN' - The trend and seasonality components were small in comparison to the total variation in the model so it may be of interest to see a model with only level included.

```
calif_1_ts.ets2 <- ets(calif_1_ts_train, model = "ANA")
calif_1_ts.ets2

## ETS(A,N,A)
##
## Call:
##  ets(y = calif_1_ts_train, model = "ANA")
##
##   Smoothing parameters:
##     alpha = 0.1935
##     gamma = 1e-04
##
##   Initial states:
##     l = 335.4537
##     s = 10.9948 54.9311 56.7306 38.9975 20.9684 -86.7199
##            -95.9024
##
##   sigma:  43.9461
##
##      AIC     AICc      BIC
## 914.5671 917.9517 937.8744

calif_1_ts.ets3 <- ets(calif_1_ts_train, model = "ANN")
calif_1_ts.ets3

## ETS(A,N,N)
##
## Call:
```

```
##  ets(y = calif_1_ts_train, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.0792
##
##   Initial states:
##     l = 321.4838
##
##   sigma:  77.2118
##
##       AIC      AICc       BIC
##  993.7849  994.1182 1000.7771
```

Part b: ARIMA Models

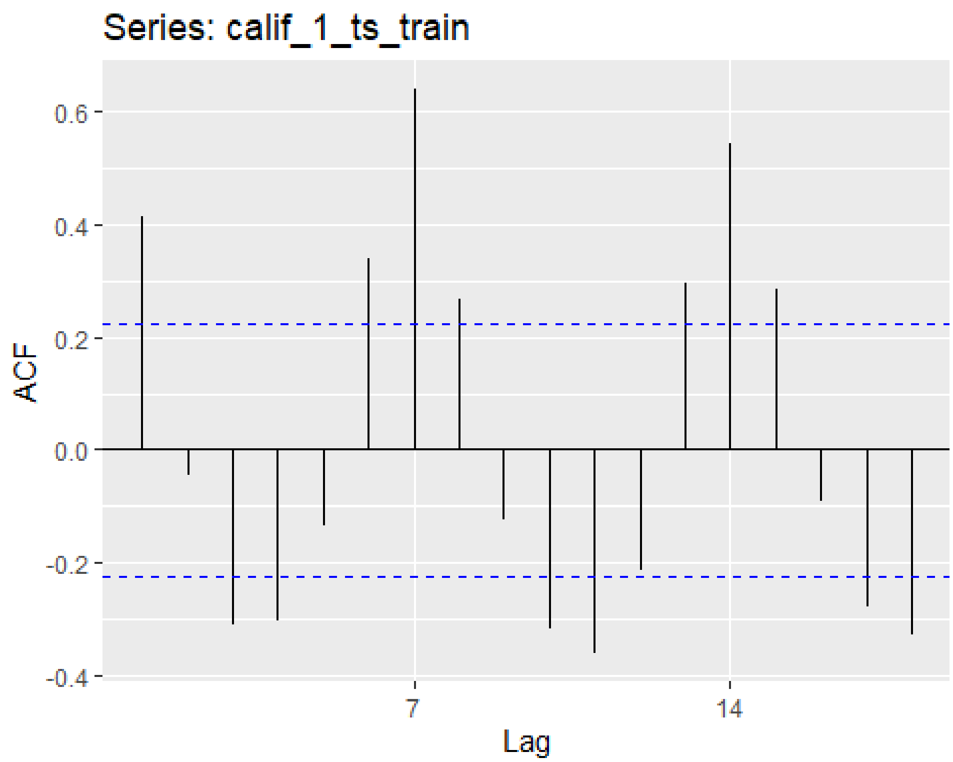
1. Test for stationarity. If not stationary, difference the data until it is stationary.

```
#Testing for stationarity
adf.test(calif_1_ts_train)

## Warning in adf.test(calif_1_ts_train): p-value smaller than printed p-
value

##
##  Augmented Dickey-Fuller Test
##
## data:  calif_1_ts_train
## Dickey-Fuller = -5.1616, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary

kpss.test(calif_1_ts_train, null="Level")

## Warning in kpss.test(calif_1_ts_train, null = "Level"): p-value greater
than
## printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  calif_1_ts_train
## KPSS Level = 0.23935, Truncation lag parameter = 3, p-value = 0.1
```

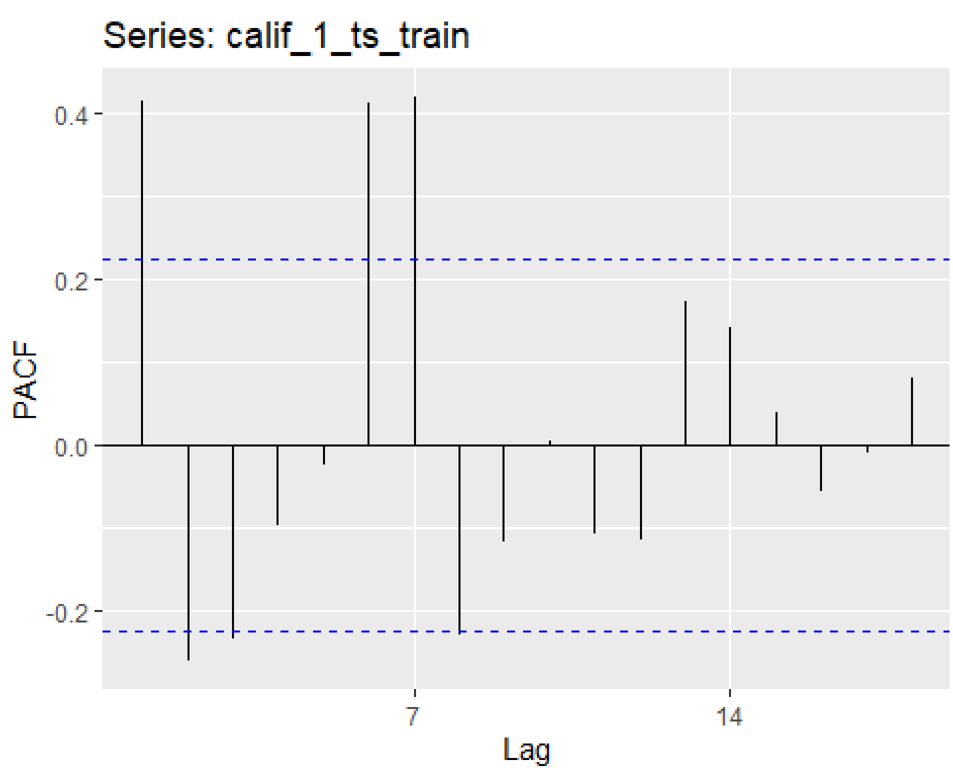
Based on the results of the ADF and KPSS tests, the model is both level and trend stationary so $d = 0$.

2. Determine the values of the parameters p and q and possible seasonal parameters

```
ggAcf(calif_1_ts_train)
```




```
ggPacf(calif_1_ts_train)
```



Observations from the ACF and PACF: * The most significant spikes in the ACF are at lags 7 and 14 indicating seasonality needs to be accounted for. * In the PACF most of the spikes till lag 7 are significant. * Most of the spikes in the ACF are significant and the values oscillate regularly between positive and negative values * The first insignificant spike in the PACF occurs at spike 3.

So for the non seasonal component of the time series, $p = 2$, $q = 0$ and $d = 0$. For the seasonal component: $P = 0$, $D = 0$ and $Q = 1$. Visual ARIMA model - ARIMA (2,0,0)(0,0,1)[7]

3. Fit ARIMA models Model 1- Automatically fitted best ARIMA model Model 2- Visually identified ARIMA model

```
auto.arima(calif_1_ts_train , d =0)

## Series: calif_1_ts_train
## ARIMA(1,0,1)(0,1,1)[7]
##
## Coefficients:
##          ar1      ma1     sma1
##        0.967  -0.8165  -0.6168
## s.e.   0.052   0.0976   0.2187
##
## sigma^2 = 2236:  log likelihood = -363.97
## AIC=735.94   AICc=736.57   BIC=744.88

calif_1_ts.arima1 <- Arima(calif_1_ts_train, order = c(1, 0, 1), seasonal =
c(0, 1, 1))
calif_1_ts.arima2 <- Arima(calif_1_ts_train, order = c(2, 0, 1), seasonal =
c(0, 0, 1))

#Calculating the RMSE of the arima models
MSE_calif_1_AR1 <-
sqrt(sum(calif_1_ts.arima1$residuals^2)/(length(calif_1_ts_train)-2))
MSE_calif_1_AR2 <-
sqrt(sum(calif_1_ts.arima2$residuals^2)/(length(calif_1_ts_train)-2))
```

####Step 3: Forecasting

```
n_calif_2_test <- length(calif_1_ts_test)
calif_1_ets1.forecast <- forecast(calif_1_ts.ets1, h=n_calif_2_test)
calif_1_ets2.forecast <- forecast(calif_1_ts.ets2, h=n_calif_2_test)
calif_1_ets3.forecast <- forecast(calif_1_ts.ets3, h=n_calif_2_test)
calif_1_AR1.forecast <- forecast(calif_1_ts.arima1, h=n_calif_2_test)
calif_1_AR2.forecast <- forecast(calif_1_ts.arima2, h=n_calif_2_test)
```

####Step 4: Performance Evaluation

```
Acc_calif_1_ets1 <- t(accuracy(calif_1_ets1.forecast, calif_1_ts_test))[2,]
Acc_calif_1_ets2 <- t(accuracy(calif_1_ets2.forecast, calif_1_ts_test))[2,]
Acc_calif_1_ets3 <- t(accuracy(calif_1_ets2.forecast, calif_1_ts_test))[2,]
Acc_calif_1_AR1 <- t(accuracy(calif_1_AR1.forecast, calif_1_ts_test))[2]
Acc_calif_1_AR2 <- t(accuracy(calif_1_AR2.forecast, calif_1_ts_test))[2]
```

```
model_names_c2 <- c('ETS Best - AAA', 'ETS - ANA', 'ETS -ANN',
                    'ARIMA BEST - (1,0,1)(0,1,1)[7]', 'ARIMA-
(2,0,1)(0,0,1)[7]')

RMSE_train_c1 <- c(Acc_calif_1_ets1['Training set'],
Acc_calif_1_ets2['Training set'], Acc_calif_1_ets2['Training set'],
MSE_calif_1_AR1 ,MSE_calif_1_AR2)

RMSE_test_c1 <- c(Acc_calif_1_ets1['Test set'], Acc_calif_1_ets2['Test set'],
Acc_calif_1_ets2['Test set'], Acc_calif_1_AR1 ,Acc_calif_1_AR2)

data.frame(model_names_c2,RMSE_train_c1,RMSE_test_c1 )
```

```
##                    model_names_c2 RMSE_train_c1 RMSE_test_c1
## 1                    ETS Best - AAA      39.59116    101.04021
## 2                         ETS - ANA      41.26204     55.95950
## 3                          ETS -ANN      41.26204     55.95950
## 4 ARIMA BEST - (1,0,1)(0,1,1)[7]      44.65272     44.06127
## 5       ARIMA- (2,0,1)(0,0,1)[7]      56.91637     56.16248
```

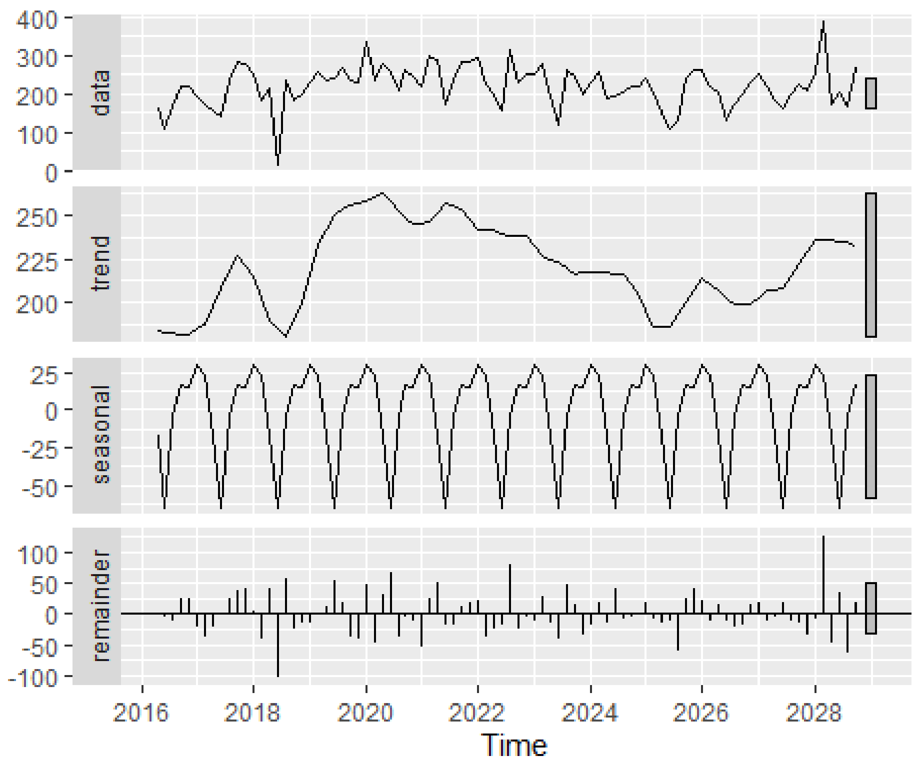
The table above shows the RMSE of each of the models fitted above on the train and test data. In comparing the performance on the train test, overall the exponential smoothing models fit the data better than the ARIMA models. However, their performance breaks down on the test data. The best fit Exponential smoothing model has its RMSE increase from 39.59 to 101.04. This implies that there was a significant amount of randomness that was being accounted for in the ETS model which should not have been accounted for allowing the train error to be very low while the test error is high. Therefore this model is not appropriate for forecasting as it breaks down on unseen data. The best fit ARIMA model, on the other hand has consistent results with a RMSE of 44 on both the train and test data. This model is able to find a balance in being able to fit the historic data and perform well on unseen data, replicating how it would perform in the future. Therefore the ARIMA(1,0,1)(0,1,1)[7] model will be used to forecast future results for this store.

####Step 5: Model Selection

```
#Fitting the model on all the data available
calif_1_model <- Arima(calif_1_ts, order = c(1, 0, 1), seasonal = c(0, 1, 1))
```

##New York 1: Store Number 20974 ####Step 1: Data Exploration

```
newY_1_ts %>% stl(s.window = "periodic") %>% autoplot
```



Based on the decomposition of the time series shown above, from most significant to least significant the components are: * Trend - there seems to be a strong trend component that accounts for a significant portion of the variability in the data. * Remainder - There is a lot of randomness in the data that cannot be accounted for using trend and seasonality jointly * Seasonality - the seasonality that the function can identify seems to account for very little of the variation in the data.

Therefore the trend is expected to be necessary in the model while the seasonality may not be as necessary.

####Step 2: Model Building

```
#Splitting the data into a roughly 80-20 training-test split
newY_1_ts_train <- window(newY_1_ts, end = c(2026, 8))
newY_1_ts_test <- window(newY_1_ts, start = c(2026, 8))
```

#####Part a: Exponential Smoothing

```
newY_1_ts.ets1 <- ets(newY_1_ts_train, model = "ZZZ")
newY_1_ts.ets1

## ETS(A,N,A)
##
## Call:
##  ets(y = newY_1_ts_train, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.189
```

```
##     gamma = 1e-04
##
##   Initial states:
##     l = 213.706
##     s = 12.1713 30.4977 17.559 18.2047 3.2766 -68.0867
##            -13.6225
##
##   sigma:  42.9002
##
##      AIC     AICc      BIC
## 910.9059 914.2905 934.2132
```

In contrast to what was expected, the best fit model has an additive error, additive seasonality but no trend component.

Model 2<- A model with an additive dampened trend component since based on the decomposition it seemed as if trend would be critical to the model. A dampened additive trend allows for the value of the trend to decrease over time so that it becomes flat.

```
newY_1_ts.ets2<- ets(newY_1_ts_train, model = "AAA", damped = TRUE)
print(newY_1_ts.ets2)
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = newY_1_ts_train, model = "AAA", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.1455
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.9351
##
##   Initial states:
##     l = 182.426
##     b = 4.58
##     s = 11.4168 29.0107 17.1479 15.8 4.0672 -67.7095
##            -9.7332
##
##   sigma:  43.0206
##
##      AIC     AICc      BIC
## 913.8503 919.7212 944.1498
```

Part b: ARIMA Models

```
#Firstly we need to check that the data is stationary. We will use three
test, the adf test, pp test and ndiffs.
adf.test(newY_1_ts_train)
```

```
##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  newY_1_ts_train
## Dickey-Fuller = -3.7675, Lag order = 4, p-value = 0.02489
## alternative hypothesis: stationary
```

```
kpss.test(newY_1_ts_train, null="Level")
```

```
## Warning in kpss.test(newY_1_ts_train, null = "Level"): p-value greater
than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  newY_1_ts_train
## KPSS Level = 0.23687, Truncation lag parameter = 3, p-value = 0.1
```

```
ndiffs(newY_1_ts_train)
```

```
## [1] 0
```

Since the p-value of the ADF test is greater than the 1% significance level, we fail to reject the null in favor of the alternative hypothesis and therefore the data is not stationary. Similarly for the KPSS test, the p-value of this test is less than the 5% significant level, we reject the null hypothesis, however, for this test this means that the data does not seen to be level stationary.

Therefore we can difference the data once to see if it becomes stationary.

```
newY_1_ts_traind <- diff(newY_1_ts_train)
adf.test(newY_1_ts_traind)
```

```
## Warning in adf.test(newY_1_ts_traind): p-value smaller than printed p-
value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  newY_1_ts_traind
## Dickey-Fuller = -7.5369, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(newY_1_ts_traind, null="Level")
```

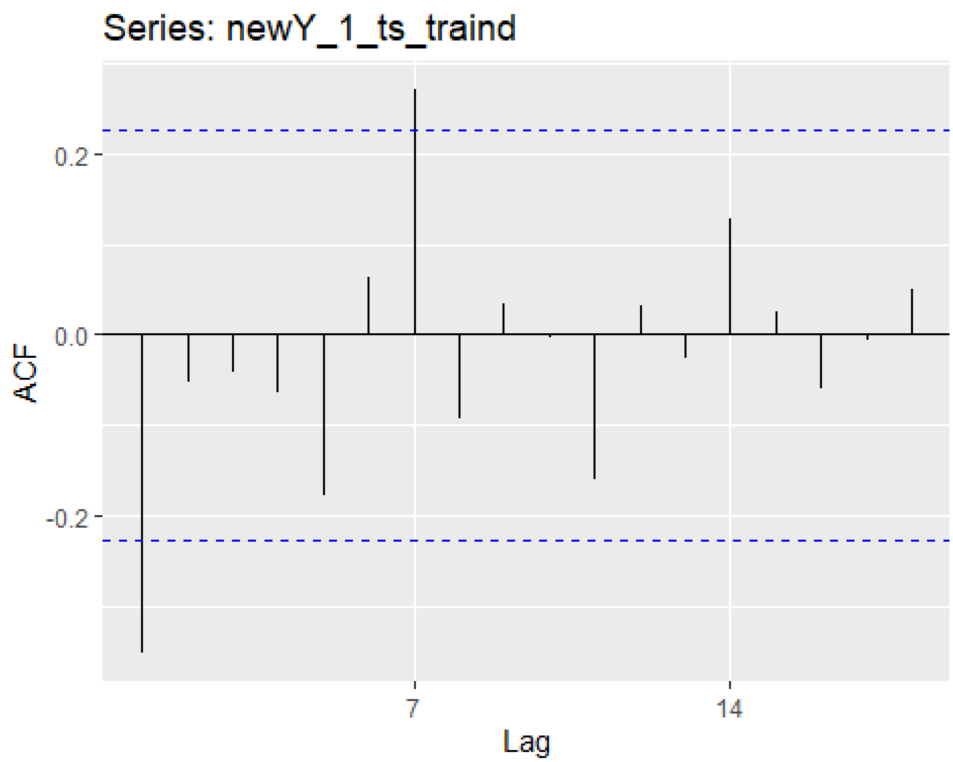
```
## Warning in kpss.test(newY_1_ts_traind, null = "Level"): p-value greater
than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
```

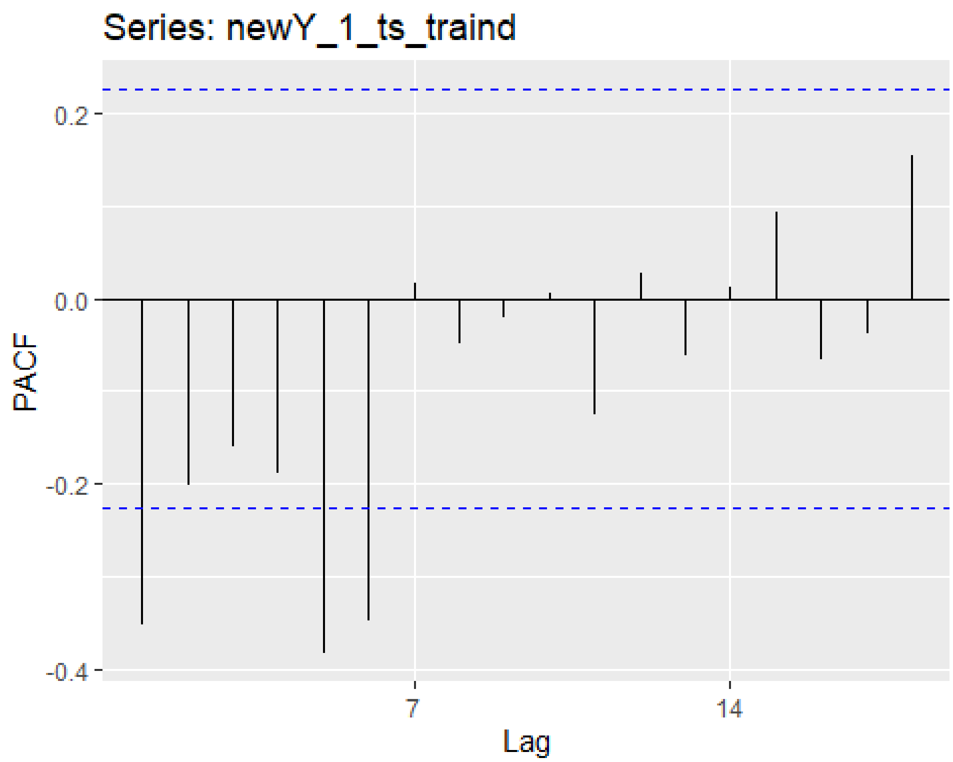
```
## data:  newY_1_ts_traind
## KPSS Level = 0.031142, Truncation lag parameter = 3, p-value = 0.1
```

Now that the data has been differenced, the p-value of the ADF test is still less than the 5% significance level, we reject the null in favor of the alternative hypothesis and therefore the data seems to be trend stationary. Similarly for the KPSS test, the p-value of this test is greater than the 5% significant level (at 10%), we fail to reject the null hypothesis, and thus the series is not level stationary.

```
ggAcf(newY_1_ts_traind)
```




```
ggPacf(newY_1_ts_traind)
```



Observations from the ACF and PACF: * In the ACF there is a single significant spike at lag 1. $q = 1$. * There are only two significant spikes in the PACF at lags 1 and 5 with the length of the spikes decaying over time. $p = 0$ * There does not seem to be any evidence of seasonality in the data Visually the ACF and PACF are indicating an ARIMA(0,1,1) model.

Finding the best arima function to the data.

```
auto.arima(newY_1_ts_train, d = 1)

## Series: newY_1_ts_train
## ARIMA(1,1,1)(0,0,2)[7]
##
## Coefficients:
##          ar1      ma1    sma1    sma2
##       0.2055  -0.9431  0.3434  0.2287
## s.e.  0.1430   0.0876  0.1248  0.1371
##
## sigma^2 = 2370:  log likelihood = -397.05
## AIC=804.1   AICc=804.97   BIC=815.69
```

Model 1 -> best fit function -> ARIMA(0, 1, 1)(1, 0, 0)[7] Model 2 -> model found visually -> ARIMA (0, 1, 1)(0,0,1) - No seasonal component

```
newY_1_ts.arima1 <- Arima(newY_1_ts_train, order = c(1, 1, 1), seasonal =
c(0, 0,2 ))
newY_1_ts.arima2 <- Arima(newY_1_ts_train, order = c(1, 1, 1), seasonal =
c(1, 0, 0))
```

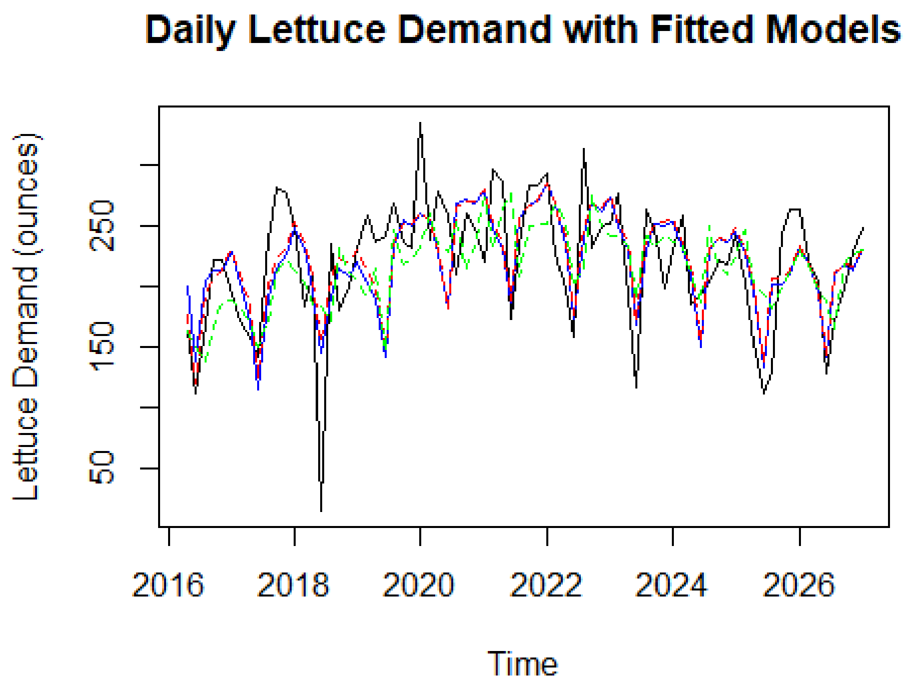
```r
#Calculating the MSE on these models
MSE_newY_1_AR1 <-
sqrt(sum(newY_1_ts.arima1$residuals^2)/(length(newY_1_ts_train)-2))
MSE_newY_1_AR2 <-
sqrt(sum(newY_1_ts.arima2$residuals^2)/(length(newY_1_ts_train)-2))
```

####Step 3: Forecasting

```r
n_newY_1_test <- length(newY_1_ts_test)
newY_1_ets1.forecast <- forecast(newY_1_ts.ets1, h=n_newY_1_test)
newY_1_ets2.forecast <- forecast(newY_1_ts.ets2, h=n_newY_1_test)
newY_1_AR1.forecast <- forecast(newY_1_ts.arima1, h=n_newY_1_test)
newY_1_AR2.forecast <- forecast(newY_1_ts.arima2, h=n_newY_1_test)
```

####Step 4: Performance Evaluation

```r
Acc_newY_1_ets1 <- t(accuracy(newY_1_ets1.forecast, newY_1_ts_test))[2,]
Acc_newY_1_ets2 <- t(accuracy(newY_1_ets2.forecast, newY_1_ts_test))[2,]
Acc_newY_1_AR1 <- t(accuracy(newY_1_AR1.forecast, newY_1_ts_test))[2]
Acc_newY_1_AR2 <- t(accuracy(newY_1_AR2.forecast, newY_1_ts_test))[2]

model_names_ny1 <- c('ETS Best - ANA', 'ETS - AAdA',
                     'ARIMA Best (1,1,1)(0,0,2)[7]', 'ARIMA(1,1,1)(1,0,0)[7]')

RMSE_train_ny1 <- c(Acc_newY_1_ets1['Training set'],
Acc_newY_1_ets2['Training set'], MSE_newY_1_AR1 ,MSE_newY_1_AR2)

RMSE_test_ny1 <- c(Acc_newY_1_ets1['Test set'], Acc_newY_1_ets2['Test set'],
Acc_newY_1_AR1 ,Acc_newY_1_AR2)

data.frame(model_names_ny1,RMSE_train_ny1,RMSE_test_ny1 )

##                 model_names_ny1 RMSE_train_ny1 RMSE_test_ny1
## 1                ETS Best - ANA       40.28006      57.48008
## 2                    ETS - AAdA       39.47840      57.66227
## 3 ARIMA Best (1,1,1)(0,0,2)[7]       47.68797      47.05632
## 4        ARIMA(1,1,1)(1,0,0)[7]       48.11095      47.47369

plot(newY_1_ts_train, xlab = 'Time', ylab='Lettuce Demand (ounces)', main =
'Daily Lettuce Demand with Fitted Models')
lines(newY_1_ets1.forecast$fitted, col = 'blue')
lines(newY_1_ets2.forecast$fitted, col = 'red', lty = 2)
lines(newY_1_AR1.forecast$fitted, col = 'green', lty = 2)
```



The blue line represents the exponential smoothing model with only an additive by dampened trend component, the red line represents the exponential smoothing model with additive trend and seasonality and the green line represents the ARIMA(0,1,1)(1,0,0)[7] model. The lines all follow a similar pattern, some going closer to the observed values than others. The ARIMA function, should by the green line does not track the as well as the exponentially smoothed models but is still able to capture the overall variation of the data. This shows as the RMSE of the exponentially smoothed models are both lower than that of the bets fit ARIMA models. However, in the out of sample evaluation, not only does the ARIMA model have a more consistent result, the RMSE does not vary widely, but for this store the ARIMA model performs the best on the test set and has the lowest RMSE of the models being evaluated.

The model chosen is the ARIMA(1,1,1)(0,0,2)[7].

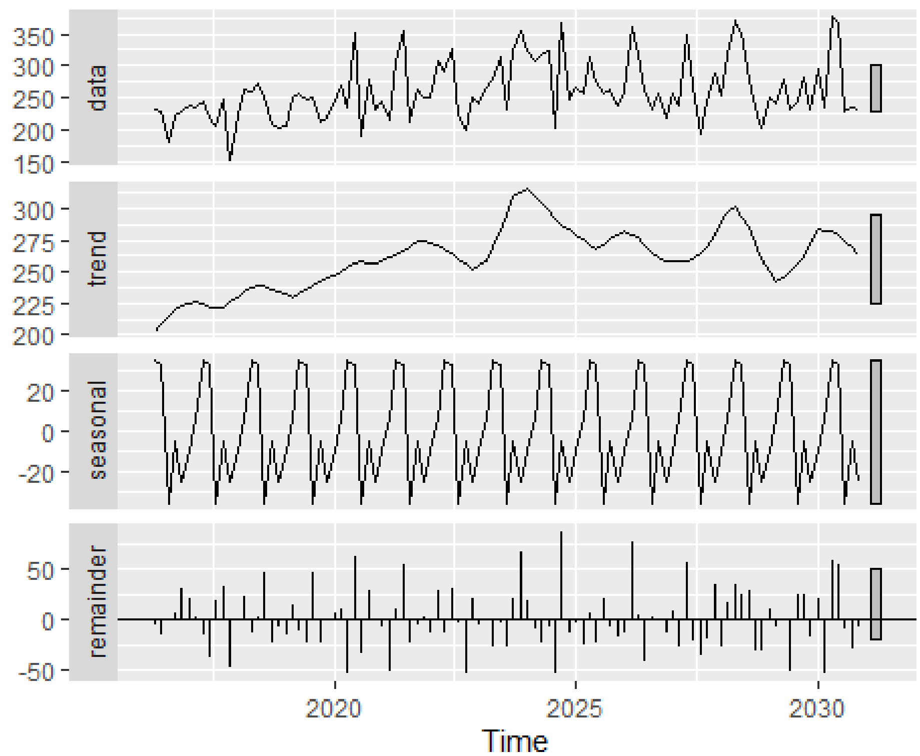
####Step 5: Model Selection

```
ny_1_model <- Arima(newY_1_ts_train, order = c(1, 1, 1), seasonal = c(0, 0,2
))
```

New York 2 - Store Number : 12631

Step 1: Data Exploration

```
newY_2_ts %>% stl(s.window = "periodic") %>% autoplot
```



Based on the decomposition above, the following observations are made: * The remainder seems to be the most critical component of the data, implying that there is significant amount of randomness in the data. * The trend component is of 2nd most importance * The seasonal component is of the leasT significant, through we can see the peaks of valleys of the seasonal component in the data.

This implies that the model may require a trend component but may not require a seasonal one.

Step 2: Model Building

```
#Splitting the data into a roughly 80-20 training-test split
newY_2_ts_train <- window(newY_2_ts, end = c(2027, 7))
newY_2_ts_test <- window(newY_2_ts, start = c(2027, 7))
```

#####Part a: Exponential Smoothing using ets

```
newY_2_ts.ets1 <- ets(newY_2_ts_train, model = "ZZZ")
newY_2_ts.ets1

## ETS(M,N,M)
##
## Call:
##  ets(y = newY_2_ts_train, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.1549
##     gamma = 1e-04
```

```
##
##   Initial states:
##     l = 240.4624
##     s = 1.0391 0.9567 0.9337 1.0025 0.8566 1.1237
##            1.0877
##
##   sigma:  0.1438
##
##      AIC     AICc      BIC
## 962.0482 965.1467 986.1153
```

The ets function best fit on the data has: * Multiplicative errors, i.e. errors whose size is expected to change over time * No trend component * Multiplicative seasonality - the size of the peaks and valleys in the seasonality changes over time

Since trend was more important that seasonality based on the decomposition we can fit a model with trend included.

Model 2 - multiplicative error, multiplicative trend and no seasonality Model 3 - additive error, additive trend and no seasonality

```
newY_2_ts.ets2<- ets(newY_2_ts_train, model = "MMN")
newY_2_ts.ets3 <- ets(newY_2_ts_train, model = "AAA")

print(newY_2_ts.ets2)

## ETS(M,M,N)
##
## Call:
##  ets(y = newY_2_ts_train, model = "MMN")
##
##   Smoothing parameters:
##     alpha = 0.0718
##     beta  = 1e-04
##
##   Initial states:
##     l = 221.301
##     b = 1.003
##
##   sigma:  0.1636
##
##      AIC     AICc      BIC
## 981.1046 981.8941 993.1382

print('**************************************************')

## [1] "**************************************************"

print(newY_2_ts.ets3)
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = newY_2_ts_train, model = "AAA")
##
##   Smoothing parameters:
##     alpha = 2e-04
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.9719
##
##   Initial states:
##     l = 210.2123
##     b = 2.1878
##     s = 11.1716 -10.2557 -26.4981 -1.2744 -31.0897 31.4424
##            26.504
##
##   sigma:  36.5091
##
##      AIC     AICc      BIC
## 964.3767 969.7297 995.6641
```

Part b: ARIMA model

Testing for stationarity

```
print(adf.test(newY_2_ts_train))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  newY_2_ts_train
## Dickey-Fuller = -3.9522, Lag order = 4, p-value = 0.01598
## alternative hypothesis: stationary
```

```
print(kpss.test(newY_2_ts_train, null='Level'))
```

```
## Warning in kpss.test(newY_2_ts_train, null = "Level"): p-value smaller
than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  newY_2_ts_train
## KPSS Level = 0.97892, Truncation lag parameter = 3, p-value = 0.01
```

```
print(ndiffs(newY_2_ts_train))
```

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

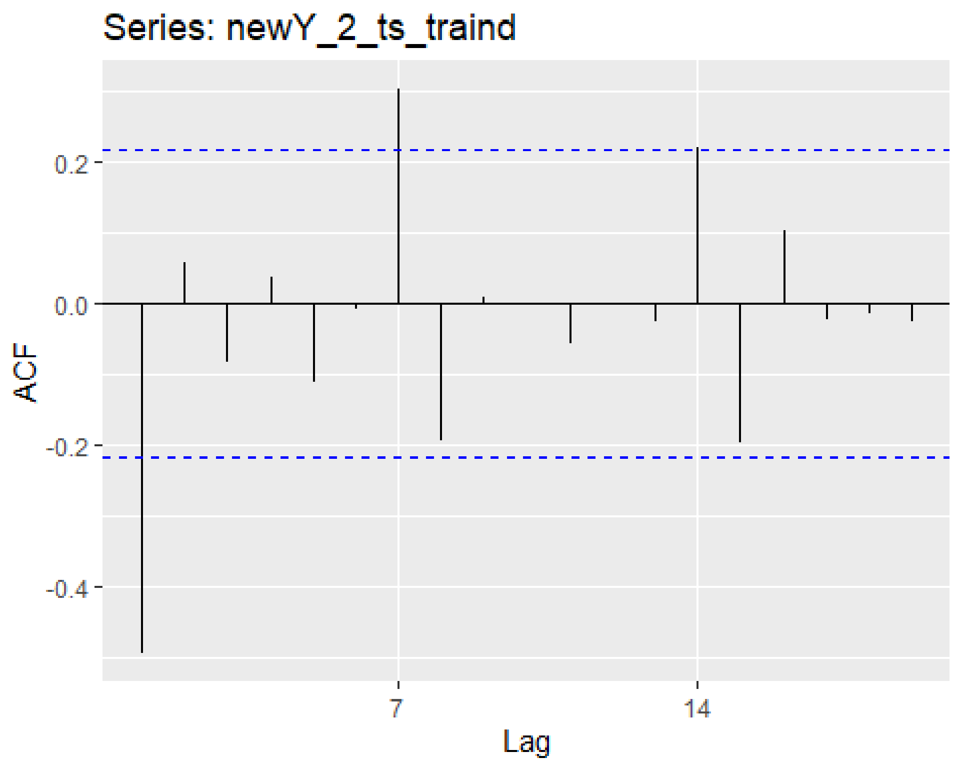
Since the p-value of the ADF test is greater than the 1% significance level, we fail to reject the null in favor of the alternative hypothesis and therefore the data is not stationary. Similarly for the KPSS test, the p-value of this test is less than the 5% significant level, we reject the null hypothesis, however, for this test this means that the data does not seen to be level stationary.

Therefore we can difference the data once to see if it becomes stationary.

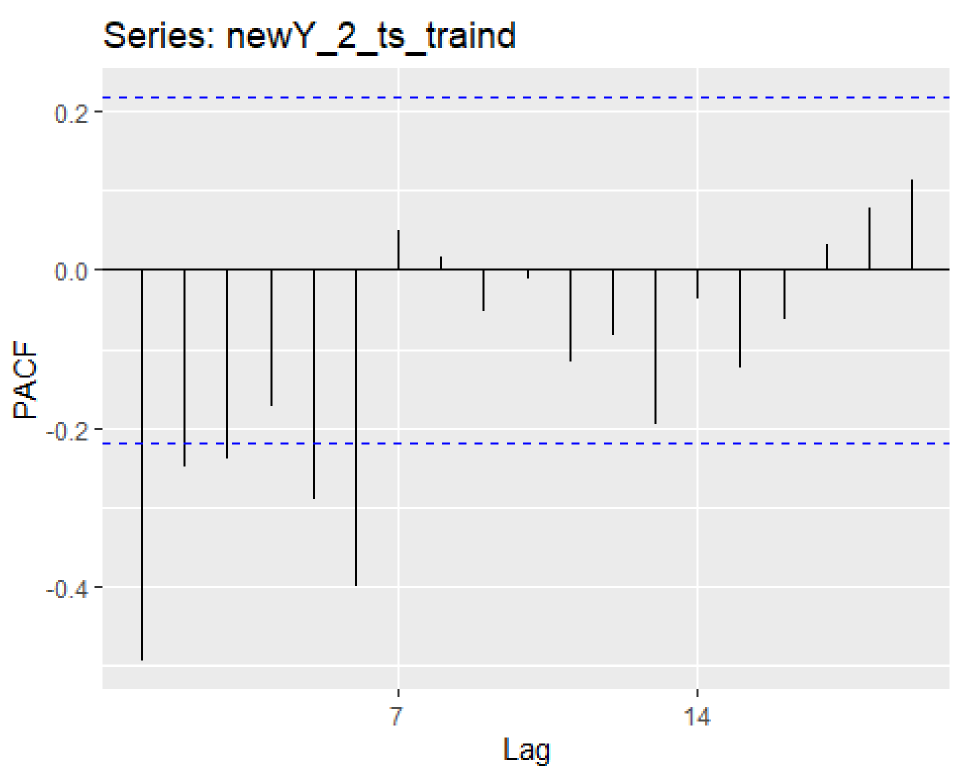
```
newY_2_ts_traind <- diff(newY_2_ts_train)
print(adf.test(newY_2_ts_traind))

## Warning in adf.test(newY_2_ts_traind): p-value smaller than printed p-
value

##
##  Augmented Dickey-Fuller Test
##
## data:  newY_2_ts_traind
## Dickey-Fuller = -7.1694, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary

print(kpss.test(newY_2_ts_traind, null='Level'))

## Warning in kpss.test(newY_2_ts_traind, null = "Level"): p-value greater
than
## printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  newY_2_ts_traind
## KPSS Level = 0.024078, Truncation lag parameter = 3, p-value = 0.1
```

After differencing the series once, the series is now both trend and level stationary. $d = 1$

```
ggAcf(newY_2_ts_traind)
```




```
ggPacf(newY_2_ts_traind)
```



Observations: * ACF - Significant spikes at Lags 1 and 7 * ACF - Exponential decay of spikes at 7 and 14 implying some weekly seasonality * ACF - first insignificant spike is at lag 2 $q = 1$ * PACF - All lags up to lag 7 are significant. * PACF - Geometric decay of lags

Expected ARIMA Model from observations ARIMA(0,1,1)(0,0,1)[7]

```
auto.arima(newY_2_ts_train, d= 1)
```

```
## Series: newY_2_ts_train
## ARIMA(0,1,1)(0,0,2)[7]
##
## Coefficients:
##           ma1    sma1    sma2
##       -0.9085  0.2822  0.1444
## s.e.   0.0470  0.1188  0.0906
##
## sigma^2 = 1685:  log likelihood = -415.37
## AIC=838.74   AICc=839.27   BIC=848.32
```

The non seasonal part of the ARIMA model from the auto.arima function is as expected.

Model 1- best fit model - ARIMA(0,1,1)(0,0,2)[7] Model 2- other model - ARIMA(0,1,1)(0,0,1)[7]

```
newY_2_ts.arima1 <- Arima(newY_1_ts_train, order = c(0, 1, 1), seasonal =
c(0, 0, 2))
newY_2_ts.arima2 <- Arima(newY_1_ts_train, order = c(0, 1, 1), seasonal =
c(0, 0, 1))

#Calculating the MSE on these models
MSE_newY_2_AR1 <-
sqrt(sum(newY_2_ts.arima1$residuals^2)/(length(newY_2_ts_train)-2))
MSE_newY_2_AR2 <-
sqrt(sum(newY_2_ts.arima2$residuals^2)/(length(newY_2_ts_train)-2))
```

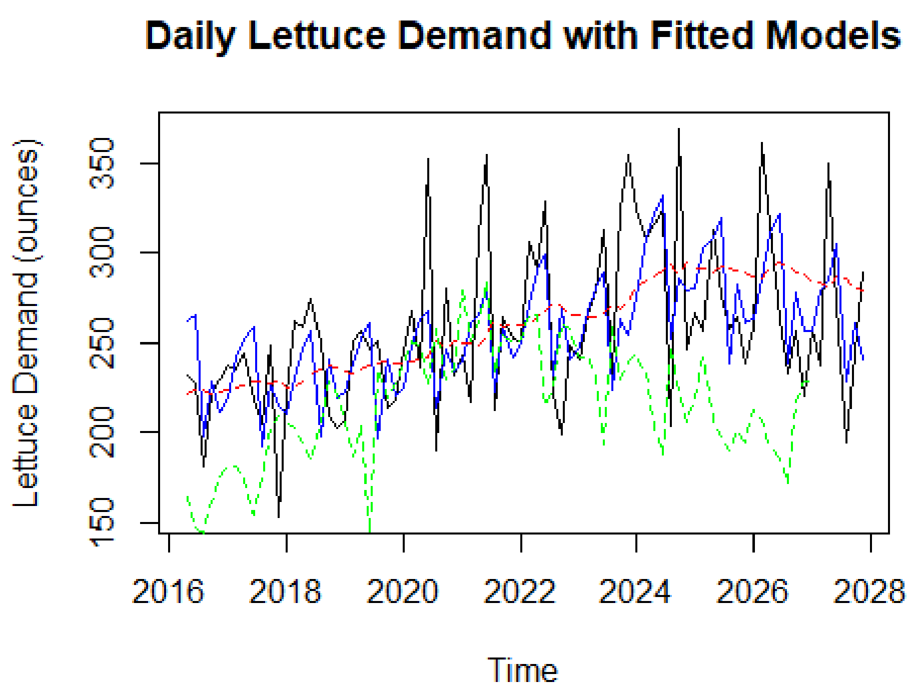
####Step 3: Forecasting

```
n_newY_2_test <- length(newY_2_ts_test)
newY_2_ets1.forecast <- forecast(newY_2_ts.ets1, h=n_newY_2_test)
newY_2_ets2.forecast <- forecast(newY_2_ts.ets2, h=n_newY_2_test)
newY_2_ets3.forecast <- forecast(newY_2_ts.ets3, h=n_newY_2_test)
newY_2_AR1.forecast <- forecast(newY_2_ts.arima1, h=n_newY_2_test)
newY_2_AR2.forecast <- forecast(newY_2_ts.arima2, h=n_newY_2_test)
```

Step 4: Performance Evaluation

```
Acc_newY_2_ets1 <- t(accuracy(newY_2_ets1.forecast, newY_2_ts_test))[2,]
Acc_newY_2_ets2 <- t(accuracy(newY_2_ets2.forecast, newY_2_ts_test))[2,]
Acc_newY_2_ets3 <- t(accuracy(newY_2_ets2.forecast, newY_2_ts_test))[2,]
Acc_newY_2_AR1 <- t(accuracy(newY_2_AR1.forecast, newY_2_ts_test))[2]
Acc_newY_2_AR2 <- t(accuracy(newY_2_AR2.forecast, newY_2_ts_test))[2]

model_names_ny2 <- c('ETS Best - MNM', 'ETS - MMN', 'ETS - AAN',
```

```
                     'ARIMA Best - (0,1,1)(0,0,2)[7]',
'ARIMA(0,1,1)(0,0,1)[7]')

RMSE_train_ny2 <- c(Acc_newY_2_ets1['Training set'],
Acc_newY_2_ets2['Training set'], Acc_newY_2_ets2['Training set'],
MSE_newY_2_AR1, MSE_newY_2_AR2)

RMSE_test_ny2 <- c(Acc_newY_2_ets1['Test set'], Acc_newY_2_ets2['Test set'],
Acc_newY_2_ets2['Test set'], Acc_newY_2_AR1, Acc_newY_2_AR2)

data.frame(model_names_ny2,RMSE_train_ny2,RMSE_test_ny2)

##                   model_names_ny2 RMSE_train_ny2 RMSE_test_ny2
## 1                  ETS Best - MNM       35.05182      43.66539
## 2                       ETS - MMN       42.23461      55.09357
## 3                       ETS - AAN       42.23461      55.09357
## 4 ARIMA Best - (0,1,1)(0,0,2)[7]       46.53838      47.74737
## 5          ARIMA(0,1,1)(0,0,1)[7]       48.08490      49.33406

plot(newY_2_ts_train, xlab = 'Time', ylab='Lettuce Demand (ounces)', main =
'Daily Lettuce Demand with Fitted Models')
lines(newY_2_ets1.forecast$fitted, col = 'blue')
lines(newY_2_ets2.forecast$fitted, col = 'red', lty = 2)
lines(newY_2_AR1.forecast$fitted, col = 'green', lty = 2)
```



In the graph above the comparison is between the best fit ets model with ‘MNM’ parameters (blue solid line), the ets model with ‘MMN’ parameters (red dotted line) and the best fit

ARIMA(0,1,1)(0,0,2)[7] model (green dotted line). Since the ETS 'MMN' model only has a trend component, it is only following the direction of the demand but does not account for much variability in the model. The ARIMA model seems to match some of the variability in the series, however, it seems to be generally underestimating the series particularly in the later part of the series. The blue line, the ETS 'MNM' model, in comparison to the other two seems to fit the data best and is able to account for more variability than the aforementioned models. This is also shown in the values of the RMSE. On both the train and test data, the ets 'MNM' model has the lowest RMSE and therefore it should be selected as the best model. It should be noted that the ARIMA models have more consistency in errors between the train and test sets.

Step 5: Model Selection

```
#Fitting the best performing model on all the data available
model_ny2 <- ets(newY_2_ts, model = 'MNM')
```

Creating the forecasts of demand for the next 2 weeks

From the fitted models we can forecast another 2 weeks, 14 days of demand

```
calif_1.forecast <- forecast(calif_1_model, h=14)
calif_2.forecast <- forecast(calif_2_model, h=14)
newY_1.forecast <- forecast (ny_1_model, h=14)
newY_2.forecast <- forecast(model_ny2, h=14)

#Store 4094
calif_1.forecast

##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2029.857       357.9536 298.9453 416.9619 267.7082 448.1990
## 2030.000       344.3136 284.3659 404.2614 252.6314 435.9959
## 2030.143       306.3195 245.5488 367.0902 213.3787 399.2603
## 2030.286       220.5880 159.1483 282.0277 126.6241 314.5519
## 2030.429       222.0952 160.0187 284.1716 127.1574 317.0330
## 2030.571       346.8948 284.2572 409.5325 251.0988 442.6909
## 2030.714       342.5645 279.4320 405.6970 246.0116 439.1173
## 2030.857       361.9443 297.1447 426.7439 262.8418 461.0467
## 2031.000       348.0818 282.7538 413.4098 248.1712 447.9924
## 2031.143       309.8776 244.0843 375.6709 209.2554 410.4998
## 2031.286       223.9477 157.8005 290.0949 122.7843 325.1111
## 2031.429       225.2676 158.7559 291.7793 123.5467 326.9885
## 2031.571       349.8904 283.0564 416.7244 247.6766 452.1042
## 2031.714       345.3930 278.2742 412.5119 242.7436 448.0424

#Store 46673
calif_2.forecast

##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 2031.000      159.79583 125.66254 193.9291 107.59347 211.9982
## 2031.143      173.08458 138.95130 207.2179 120.88223 225.2869
## 2031.286      164.56924 130.43595 198.7025 112.36688 216.7716
```

```
## 2031.429      101.27350  67.14021 135.4068  49.07114 153.4758
## 2031.571       76.91392  42.78064 111.0472  24.71157 129.1163
## 2031.714      170.12068 135.98740 204.2540 117.91833 222.3230
## 2031.857      175.76910 141.63581 209.9024 123.56674 227.9715
## 2032.000      159.79583 125.66254 193.9291 107.59347 211.9982
## 2032.143      173.08458 138.95130 207.2179 120.88223 225.2869
## 2032.286      164.56924 130.43595 198.7025 112.36688 216.7716
## 2032.429      101.27350  67.14021 135.4068  49.07114 153.4759
## 2032.571       76.91392  42.78064 111.0472  24.71156 129.1163
## 2032.714      170.12068 135.98740 204.2540 117.91833 222.3230
## 2032.857      175.76910 141.63581 209.9024 123.56674 227.9715
```

```
#Store Number 20974
print(newY_1.forecast)
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 2027.143       211.8566 149.4641 274.2492 116.43541 307.2779
## 2027.286       204.5843 140.0800 269.0885 105.93354 303.2350
## 2027.429       171.0648 106.1913 235.9382  71.84934 270.2802
## 2027.571       191.6759 126.6125 256.7393  92.17005 291.1818
## 2027.714       208.3522 143.1287 273.5758 108.60139 308.1031
## 2027.857       221.1012 155.7235 286.4788 121.11465 321.0877
## 2028.000       226.1212 160.5909 291.6514 125.90129 326.3410
## 2028.143       214.6622 144.2034 285.1211 106.90474 322.4197
## 2028.286       213.6497 142.4727 284.8268 104.79384 322.5056
## 2028.429       197.5044 125.9997 269.0092  88.14737 306.8615
## 2028.571       208.6922 136.9217 280.4627  98.92871 318.4557
## 2028.714       205.0356 133.0119 277.0592  94.88491 315.1863
## 2028.857       208.9354 136.6618 281.2089  98.40248 319.4683
## 2029.000       213.9382 141.4161 286.4604 103.02515 324.8513
```

```
#Store Number 12631
newY_2.forecast
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2031.000       258.1965 209.7455 306.6475 184.0971 332.2959
## 2031.143       276.1955 224.0706 328.3205 196.4773 355.9138
## 2031.286       302.6475 245.2074 360.0877 214.8004 390.4947
## 2031.429       307.1361 248.5179 365.7542 217.4873 396.7848
## 2031.571       233.2254 188.4671 277.9838 164.7735 301.6774
## 2031.714       266.3445 214.9505 317.7386 187.7441 344.9449
## 2031.857       246.8641 198.9710 294.7571 173.6179 320.1102
## 2032.000       258.1966 207.8360 308.5572 181.1766 335.2165
## 2032.143       276.1956 222.0386 330.3526 193.3696 359.0216
## 2032.286       302.6476 242.9923 362.3029 211.4127 393.8826
## 2032.429       307.1361 246.2815 367.9908 214.0670 400.2053
## 2032.571       233.2255 186.7775 279.6735 162.1894 304.2616
## 2032.714       266.3446 213.0306 319.6586 184.8079 347.8813
## 2032.857       246.8641 197.2003 296.5279 170.9099 322.8183
```