Demand Forecasting for A Fast-Food Restaurant Chain

Contents

Introduction

This report aims to forecast the lettuce demand for four US-based branches of a fast-food restaurant chain to support inventory replenishment decisions. We are dealing with data provided by the American fast-food chain on ingredients, recipes, transaction information, and restaurant information, with a time frame ranging from early March 2015 to the 15th of June 2015.

This report uses the previously conducted data wrangling for usable data. For each of the four restaurants, we build ARIMA and Holt-Winters models and evaluate them against each other to obtain the best possible forecast of lettuce demand over the following fourteen days (from 06/16/2015 to 06/29/2015).

California 1 (ID:46673)

To forecast demand for restaurant 46673, we train both Holt-Winters and ARIMA models and assess their performance on out-of-sample data. The Holt-Winters model is an exponential smoothing model using three components to forecast time series data, the error (or noise), trend and seasonality. This report uses the Exponential Smoothing State Space (ETS) model as our exponential smoothing model due to its greater flexibility. The Auto-Regressive Integrated Moving Average (ARIMA) model also uses three components to forecast time series data, an auto-regressive component, an integrated component, and a moving average component. Throughout this report, we consider a variation of this model, the seasonal ARIMA (SARIMA), to better handle patterns in the data.

Note that the process for restaurant 46673 will be identical and carried over to the remaining three.

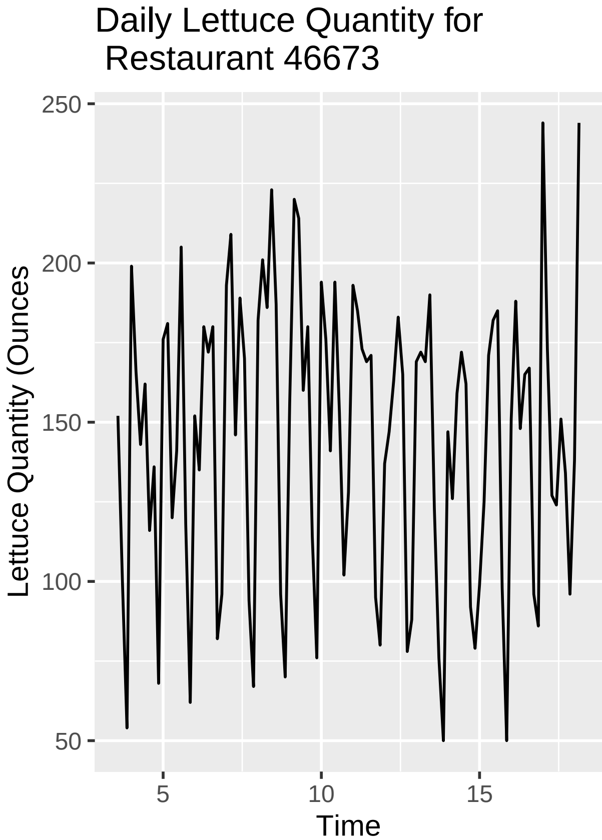
```
# Import the data
restaurant_46673 <- read_csv('restaurant_46673.csv')
restaurant_46673[1, 2] # Find the start date
```

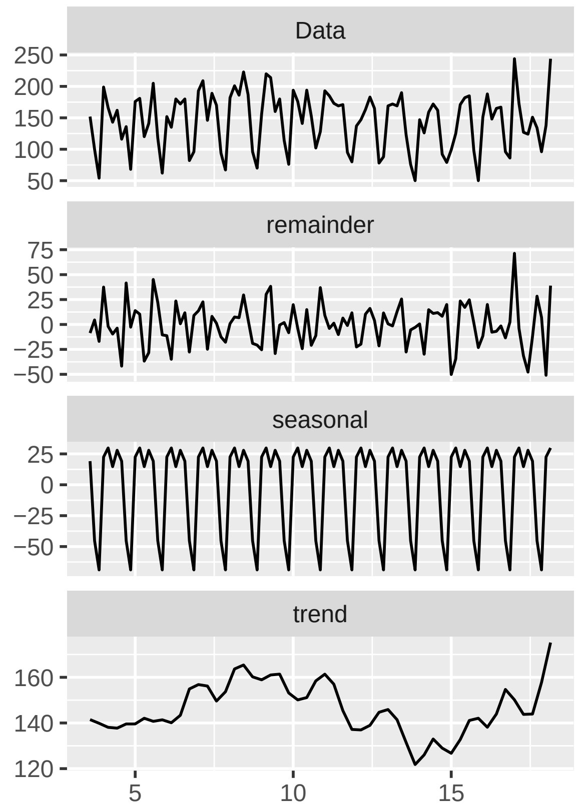
```
## # A tibble: 1 x 1
##   date
##   <date>
## 1 2015-03-05
```

Holt-Winters Model

The time series object for restaurant 46673 is created with a frequency of 7, corresponding to the cycle's length, and starting on the 5th of March 2015. The plot of the time series object and the plot of the seasonal decomposition of the time series object allow us to better understand the data fluctuations.

```
restaurant_46673_ts <- ts(restaurant_46673$`Quantity (ounces)`,
                          frequency = 7, # 7-day cycle
                          start = c(03, 05)) # Starts on the third of March

# Plot of time-series
restaurant_46673_ts.plot1 <- autoplot(restaurant_46673_ts) +
  ggtitle('Daily Lettuce Quantity for \n Restaurant 46673') +
  xlab('Time') +
  ylab('Lettuce Quantity (Ounces')

# Plot of seasonal decomposition
restaurant_46673_ts.plot2 <- restaurant_46673_ts %>% stl(s.window = "period") %>% autoplot

grid.arrange(restaurant_46673_ts.plot1, restaurant_46673_ts.plot2, ncol = 2)
```





From the above two plots, no significant trend stands out; thus, the trend component can be disregarded later on when building our models. Nonetheless, there appears to be an additive seasonality component that will need to be accounted for in the exponential smoothing model.

Before doing any of the models, however, we split the data between a training and a testing set. The commonly used splits are based on a 70/30, 80/20 or 90/10 ratio. Because the data set is so small, we want to maximise the data points in our training set and keep the minimum number of variables necessary in our test set. A trade-off we need to consider is that the more data points in our training set, the more a model is likely to learn the noise in the data. We therefore want to find a balance and would prefer the 80/20 ratio as our train/test split if we followed conventional splitting patterns. Nonetheless, there is another way we can think about the split for our data. Bringing this back to the context of our report, we want to forecast for a 14-day period. Taking a 20% test split however will include more than 14 data points. Therefore, when asking the model to forecast for the upcoming 14 days, the difference between that and the number of data points in the test set will be unused data. It could therefore be interesting here to split our data such that the last 14 days are the test set and the remainder are the training set. Although this is an interesting idea, going back to the previous mentioned overfitting issue, making our training set such a large proportion of the data will likely introduce noise into our model. To avoid this, we choose to keep a conventional train/test split ratio of 80/20.

```
# Define the length of our training set
n_train_46673 <- round(length(restaurant_46673_ts) * 0.8)

# Splitting into train and test
# Train set: 80%
restaurant_46673_ts.train <- subset(restaurant_46673_ts, end = n_train_46673)

# Test set: 20%
restaurant_46673_ts.test <- subset(restaurant_46673_ts, start = n_train_46673+1)
```

Then, we fit the Holt-Winters exponential smoothing model with the *ets()* function, pre-defining an additive error and an additive seasonality. We can also ensure that our model specification is correct by building another model where we set the model parameter to be 'ZZZ' such that the error, trend, and seasonality are not pre-defined.

```
restaurant_46673.ets1 <- ets(restaurant_46673_ts.train, model = 'ANA')
restaurant_46673.ets2 <- ets(restaurant_46673_ts.train, model = 'ZZZ')
restaurant_46673.ets2
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = restaurant_46673_ts.train, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.1223
##     gamma = 1e-04
##
##   Initial states:
##     l = 144.2834
##     s = 31.9515 15.5058 26.5879 24.4611 -70.2816 -46.5225
##            18.2978
##
##   sigma:  24.6121
##
##      AIC     AICc      BIC
## 897.1487 900.2473 921.2159
```

Because the 'ZZZ' specification output is 'ANA' for both models, we know we correctly defined our exponential smoothing model. Thus, the error is additive, there is no trend, and the seasonality is additive. The resulting smoothing constants are $\alpha = 0.1123$, $\beta = 0$ and $\gamma = 0.0001$. They control the degree of smoothing applied to the error and seasonal components of the model.

The output also gives the initial states, representing the starting values for the error and seasonal components. The model assumes that the level starts at 144.2834 and has seven seasonal components with starting values of 31.9515, 15.5058, 26.5879, 24.4611, -70.2816, -46.5225, and 18.2978. In addition, the estimated standard deviation of the error term, sigma, is equal to 24.6121.

Then, we can conduct an out-of-sample evaluation for the model.

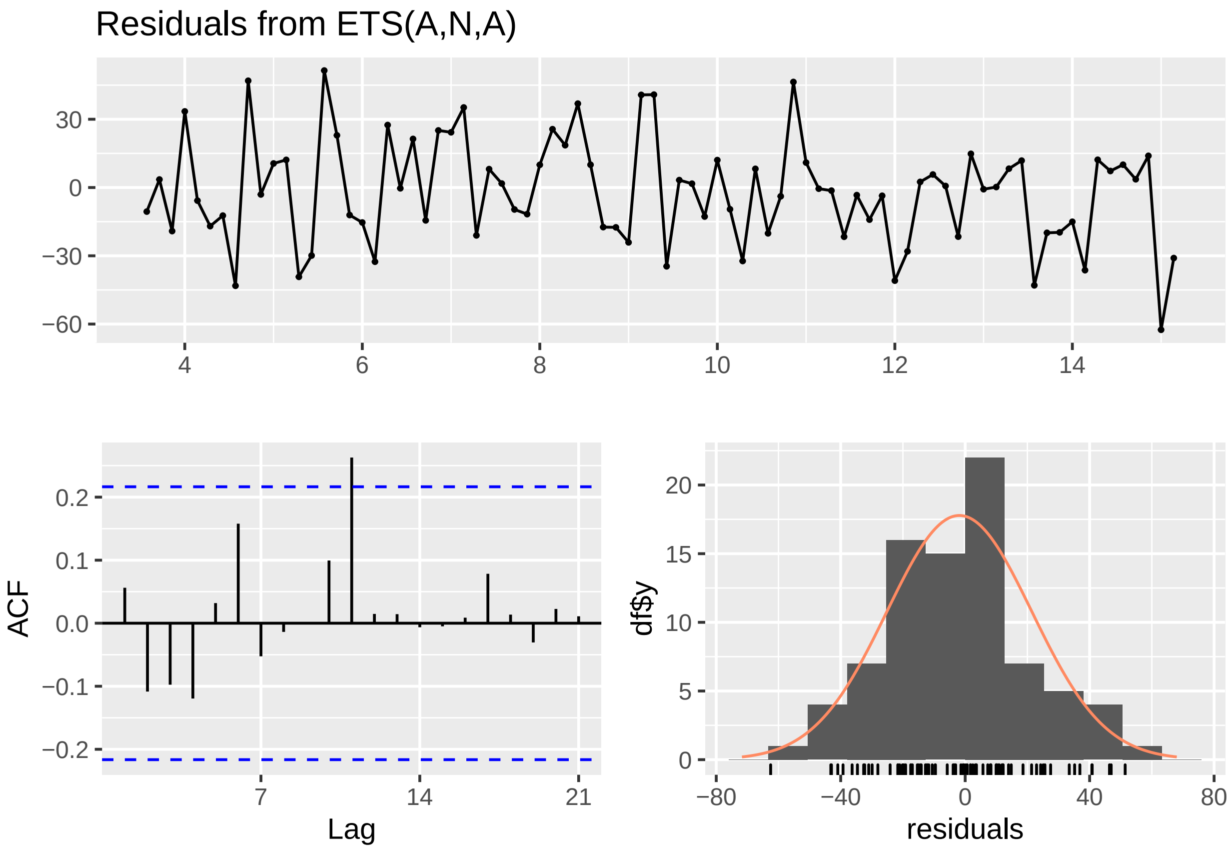
```
# Out-of-sample evaluation
restaurant_46673.ets.f <- forecast(restaurant_46673.ets1, h = 14)
accuracy(restaurant_46673.ets.f, restaurant_46673_ts.test)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -1.861661 23.22218 18.45667 -4.154372 14.30555 0.6970043
## Test set     24.672842 33.76837 25.43478 14.869195 16.39307 0.9605279
##                    ACF1 Theil's U
## Training set 0.05622874        NA
## Test set     0.10348431 0.4214732
```

Essential outputs in the above out-of-sample evaluation are those on the second line, **test set**. Indeed, even if our model performed exceptionally well on the training set (in-sample), we would need more to understand its performance with new unseen data. Furthermore, it may indicate overfitting, where the model learns too much of the noise in the data and cannot deal with out-of-sample data points.

We can conduct a residuals analysis for this model to see if it is satisfactory before re-calibrating with the entire sample.

```
# Analyse the residuals
checkresiduals(restaurant_46673.ets.f)
```



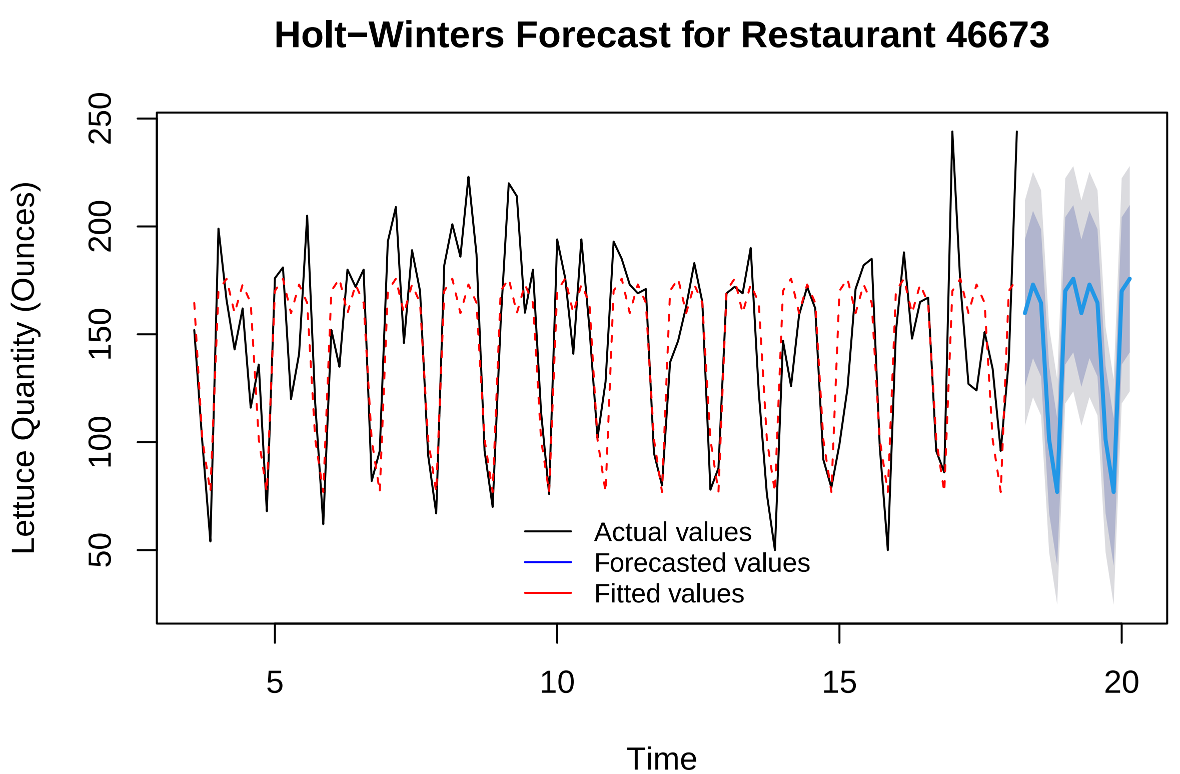

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,A)
## Q* = 13.685, df = 14, p-value = 0.4735
##
## Model df: 0.   Total lags used: 14
```

The Ljung-Box test suggests that the exponential smoothing model's residuals will likely be independently and identically distributed with no significant autocorrelation. Indeed, a p-value greater than 0.05 fails to reject the null hypothesis that the residuals are independently distributed. Furthermore, the errors appear to have a constant variance and are normally distributed with a mean of zero.

Therefore, we conclude that this model is satisfactory for forecasting lettuce demand for restaurant 46673.

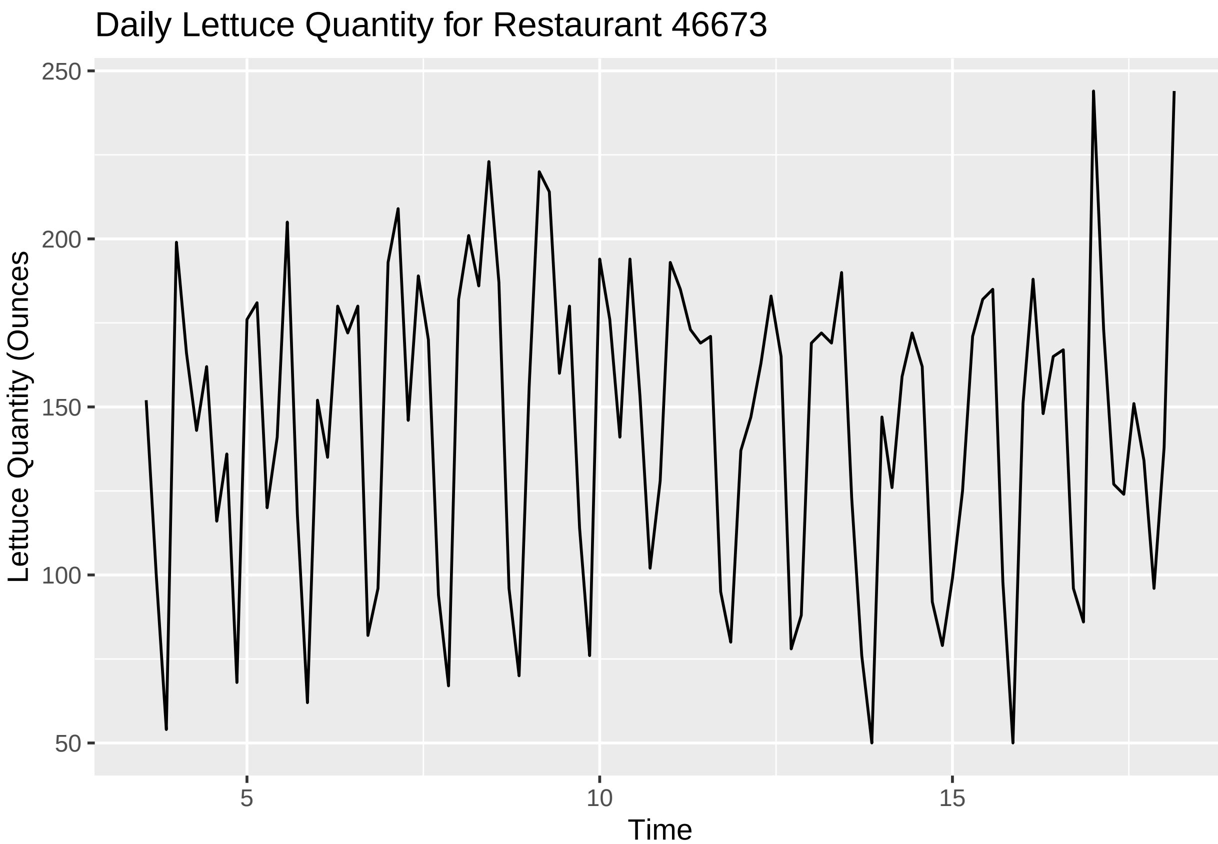
Then, we can re-calibrate it on the entire sample and forecast the quantity of lettuce demanded by restaurant 46673 for the upcoming 14 days. Plotting the output further allows visualising the model's performance. This is done by plotting the actual data as a black line, the fitted data as a red dotted line, and the forecasted data as a blue line, surrounded by the 95% confidence interval.

```r
# Forecast
# Re-calibrate model with the entire sample
restaurant_46673.selected.model <- ets(restaurant_46673_ts, model = "ANA")
restaurant_46673.selected.model.f <- forecast(restaurant_46673.selected.model, h = 14)

# Summary plot of actual, fitted, and forecasted values
plot(restaurant_46673.selected.model.f, main = "Holt-Winters Forecast for Restaurant 46673",
     xlab="Time", ylab="Lettuce Quantity (Ounces)", lty = 1, col = "black")
lines(fitted(restaurant_46673.selected.model.f), lty = 2, col = "red")
legend("bottom", legend=c("Actual values","Forecasted values", "Fitted values"),
       col=c("black", "blue", "red"), box.lty=0, lty=1, cex=0.8)
```



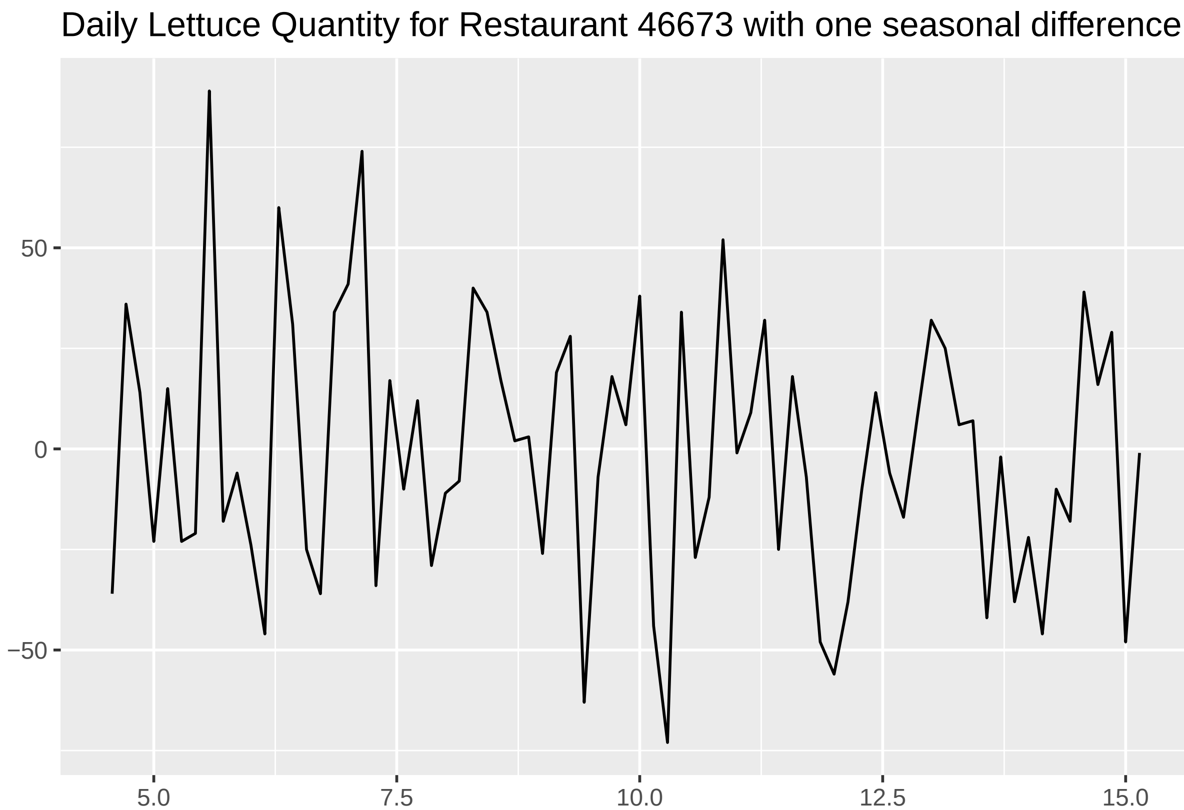
ARIMA Model

```r
# Plot of time-series
autoplot(restaurant_46673_ts) +
  ggtitle('Daily Lettuce Quantity for Restaurant 46673') +
  xlab('Time') +
  ylab('Lettuce Quantity (Ounces')
```



From the above plot and our previous seasonal decomposition of the time series object, we can conclude that the object does not display any trend but does display an additive seasonal component (we thus need to apply seasonal differencing). Also, the time series seems to be stationary in terms of mean and variance, which can be verified with several tests. Therefore, we start by taking the difference once at the seasonal lag (lag of 7). Then, we test the stationarity of the data with the Augmented Dickey-Fuller (ADF) test, Phillips-Perron Unit Root (PP) test, and the KPSS test. Both the ADF and PP tests test for the presence of unit roots in the data. The PP test, however, is non-parametric and accounts for autocorrelation and heteroscedasticity. The KPSS test looks at whether or not the data is trend-stationary. We also use the *ndiffs()* function on our training set to determine the number of differences necessary to stationaries the data. Finally, the *nsdiffs()* function tells us the number of differences necessary to stationaries the data in terms of seasonality.

```r
# Apply one seasonal difference
restaurant_46673_ts.diff <- diff(restaurant_46673_ts.train, differences = 1, lag = 7)
autoplot(restaurant_46673_ts.diff) + # No seasonal trend
  ggtitle('Daily Lettuce Quantity for Restaurant 46673 with one seasonal difference')
```




```r
# Stationarity tests
adf.test(restaurant_46673_ts.diff) # Augmented Dickey-Fuller Test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  restaurant_46673_ts.diff
## Dickey-Fuller = -4.1898, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pp.test(restaurant_46673_ts.diff) # Phillips-Perron Unit Root Test
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  restaurant_46673_ts.diff
## Dickey-Fuller Z(alpha) = -70.036, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

```r
kpss.test(restaurant_46673_ts.diff) # KPSS Test for Level Stationarity
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  restaurant_46673_ts.diff
## KPSS Level = 0.21189, Truncation lag parameter = 3, p-value = 0.1
```

```r
ndiffs(restaurant_46673_ts.train)
```

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

```r
# Seasonal stationarity
nsdiffs(restaurant_46673_ts.train)
```

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

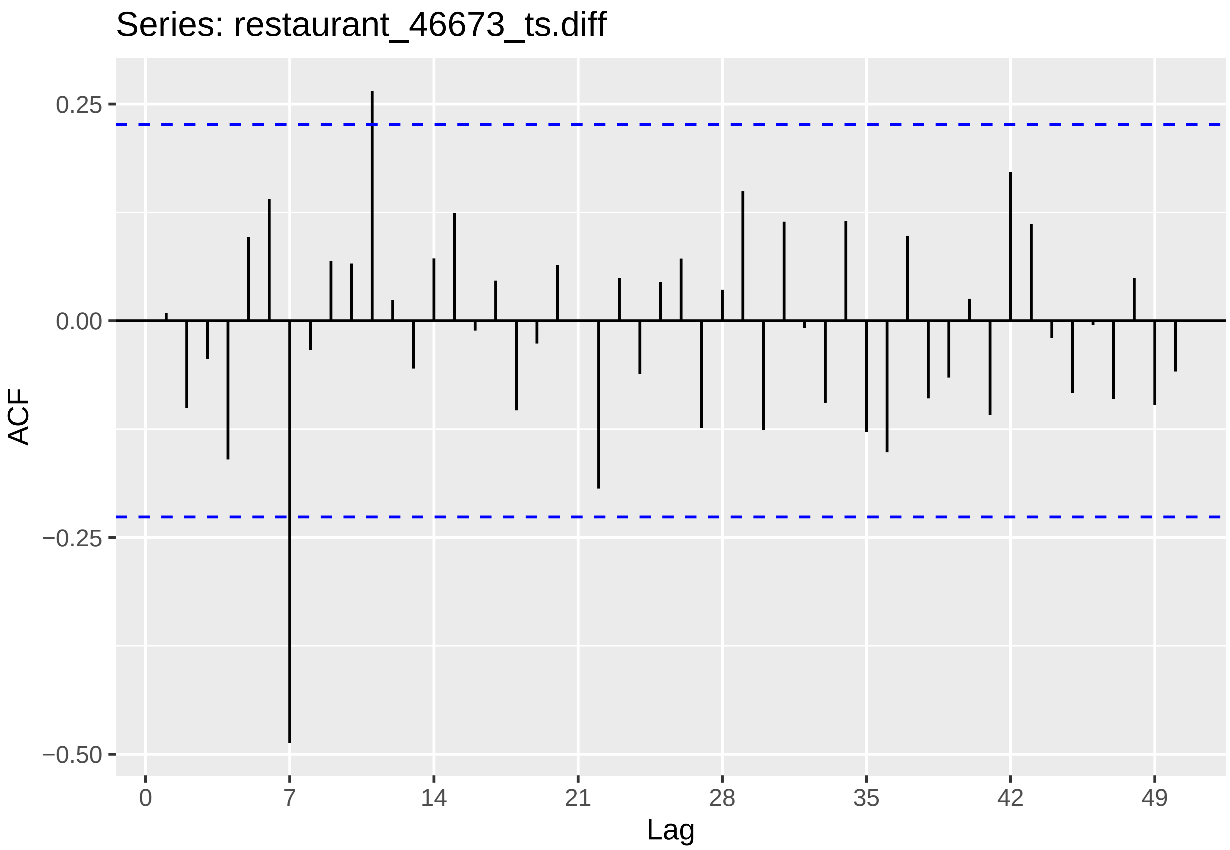
From the above tests, we have the following conclusions:

- ADF: sufficient evidence to reject the null that we have a unit root; thus, the time series is stationary.
- PP: sufficient evidence to reject the null; thus, the time series is stationary.
- KPSS: insufficient evidence to reject the null; thus, the time series is stationary.

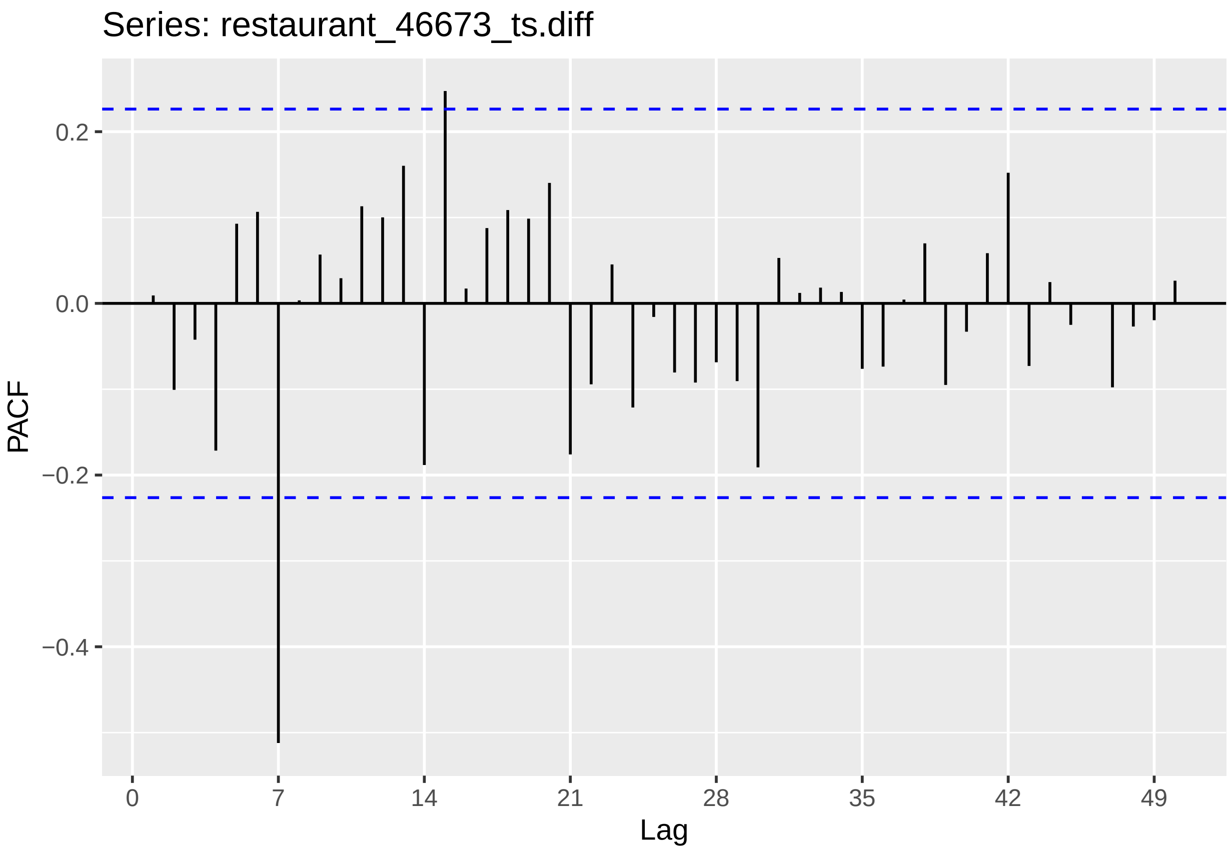
The above tests show that the time series proved stationary in terms of trend. Nevertheless, it is not stationary regarding seasonality, which is consistent with our preliminary decomposition analysis. The output of 1 from the *nsdiffs()* function confirms that we needed to apply one seasonal difference. We therefore expect an ARIMA model of the form ARIMA(p, d, q)(P, D, Q)[7] where d = 0 and D = 1.

Now that the d and D parameters are determined, we need to determine the ARIMA model's p, q, P, and Q parameters by looking at the ACF and PACF plots. We also automatically select the best ARIMA model using the *auto.ARIMA()* function for comparison based on the AIC_C. We choose to compare based on the AIC_C and not AIC or BIC as the AIC_C adjusts for small sample sizes by adding a correcting factor to the AIC and thus accounts for the AIC's tendency to overfit with small sample sizes.

```r
# Choice of p, q, P and Q
# Define lag.max=50 to have a better idea of how the plots decay
ggAcf(restaurant_46673_ts.diff, lag.max = 50)
```




```
ggPacf(restaurant_46673_ts.diff, lag.max = 50)
```




```
# Find the best ARIMA model
auto.arima(restaurant_46673_ts.train, trace = TRUE, ic = "aicc")
```

```
##
##  ARIMA(2,0,2)(1,1,1)[7] with drift         : 723.8686
##  ARIMA(0,0,0)(0,1,0)[7] with drift         : 738.1812
##  ARIMA(1,0,0)(1,1,0)[7] with drift         : 720.4236
##  ARIMA(0,0,1)(0,1,1)[7] with drift         : 714.7828
##  ARIMA(0,0,0)(0,1,0)[7]                    : 736.1684
##  ARIMA(0,0,1)(0,1,0)[7] with drift         : 740.3446
##  ARIMA(0,0,1)(1,1,1)[7] with drift         : 716.818
##  ARIMA(0,0,1)(0,1,2)[7] with drift         : 716.8367
##  ARIMA(0,0,1)(1,1,0)[7] with drift         : 720.3518
##  ARIMA(0,0,1)(1,1,2)[7] with drift         : Inf
##  ARIMA(0,0,0)(0,1,1)[7] with drift         : 714.6913
##  ARIMA(0,0,0)(1,1,1)[7] with drift         : 716.8559
##  ARIMA(0,0,0)(0,1,2)[7] with drift         : 716.8779
##  ARIMA(0,0,0)(1,1,0)[7] with drift         : 718.949
##  ARIMA(0,0,0)(1,1,2)[7] with drift         : Inf
##  ARIMA(1,0,0)(0,1,1)[7] with drift         : 714.7543
##  ARIMA(1,0,1)(0,1,1)[7] with drift         : 717.0476
##  ARIMA(0,0,0)(0,1,1)[7]                    : 713.2449
##  ARIMA(0,0,0)(1,1,1)[7]                    : 715.1981
##  ARIMA(0,0,0)(0,1,2)[7]                    : 715.2319
##  ARIMA(0,0,0)(1,1,0)[7]                    : 717.0312
##  ARIMA(0,0,0)(1,1,2)[7]                    : 717.2849
```

```
##  ARIMA(1,0,0)(0,1,1)[7]                     : 713.1908
##  ARIMA(1,0,0)(0,1,0)[7]                     : 738.2727
##  ARIMA(1,0,0)(1,1,1)[7]                     : 714.9377
##  ARIMA(1,0,0)(0,1,2)[7]                     : 714.9216
##  ARIMA(1,0,0)(1,1,0)[7]                     : 718.4058
##  ARIMA(1,0,0)(1,1,2)[7]                     : 717.215
##  ARIMA(2,0,0)(0,1,1)[7]                     : 715.4216
##  ARIMA(1,0,1)(0,1,1)[7]                     : 715.4218
##  ARIMA(0,0,1)(0,1,1)[7]                     : 713.2317
##  ARIMA(2,0,1)(0,1,1)[7]                     : 717.719
##
##  Best model: ARIMA(1,0,0)(0,1,1)[7]
```

```
## Series: restaurant_46673_ts.train
## ARIMA(1,0,0)(0,1,1)[7]
##
## Coefficients:
##          ar1     sma1
##       0.1793  -0.6995
## s.e.  0.1187   0.1229
##
## sigma^2 = 699.8:  log likelihood = -353.43
## AIC=712.85   AICc=713.19   BIC=719.81
```

The orders of p and P are determined by looking at the PACF plot, and the orders of q and Q are determined by looking at the ACF plot. The above ACF plot displays a sinusoidal and not an exponential decay. Furthermore, by analysing the spikes, the ACF plot shows spikes at lags 7 and 11. The spike at lag 11 is most likely due to white noise, so we choose not to define a corresponding model. Therefore, we can define a seasonal MA component of 1 (Q = 1) and a non-seasonal MA component of 0 (q = 0). The PACF plot also decays in a sinusoidal manner, and we see a spike at a lag of 7 and a spike at a lag of 15. Thus, it may be necessary to integrate a seasonal AR component of 1 (P = 1). Again, the spike at lag of 15 is most likely due to white noise, so we choose not to define a corresponding model and define models with a non-seasonal AR component of 0 (p = 0).

We can define the following model from these two plots to be tested against the two best models defined from the auto.ARIMA function: ARIMA(0,0,0)(1,1,1)[7].

The best ARIMA model according to the *auto.ARIMA()* function is ARIMA(1,0,0)(0,1,1)[7], and the second best is ARIMA(0,0,1)(0,1,1)[7]. We train and forecast lettuce demand for restaurant 46673 to compare each model's performance.

```
# Candidate models
restaurant_46673.arima1 <- Arima(restaurant_46673_ts.train, order = c(1, 0, 0),
                                 seasonal = list(order = c(0, 1, 1), period = 7),
                                 include.drift = FALSE)
restaurant_46673.arima2 <- Arima(restaurant_46673_ts.train, order = c(0, 0, 1),
                                 seasonal = list(order = c(0, 1, 1), period = 7),
                                 include.drift = FALSE)
restaurant_46673.arima3 <- Arima(restaurant_46673_ts.train, order = c(0, 0, 0),
                                 seasonal = list(order = c(1, 1, 1), period = 7),
                                 include.drift = FALSE)

# Model evaluation
# Forecast
restaurant_46673.arima1.f <- forecast(restaurant_46673.arima1, h = 14)
```

```r
restaurant_46673.arima2.f <- forecast(restaurant_46673.arima2, h = 14)
restaurant_46673.arima3.f <- forecast(restaurant_46673.arima3, h = 14)

# Out-of-sample performance
accuracy(restaurant_46673.arima1.f, restaurant_46673_ts.test)
```

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.7949027 24.95910 18.28022 -2.980306 13.47389 0.6903407
## Test set     13.7331324 32.31547 21.75614  4.592271 14.93708 0.8216063
##                     ACF1 Theil's U
## Training set 0.005226471        NA
## Test set     0.067914914  0.437933
```

```r
accuracy(restaurant_46673.arima2.f, restaurant_46673_ts.test)
```

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.8097159 24.98450 18.30272 -2.989476 13.52350 0.6911902
## Test set     13.6005814 32.47706 21.75134  4.523346 14.94033 0.8214250
##                     ACF1 Theil's U
## Training set 0.007210826        NA
## Test set     0.070032512 0.4406956
```

```r
accuracy(restaurant_46673.arima3.f, restaurant_46673_ts.test)
```

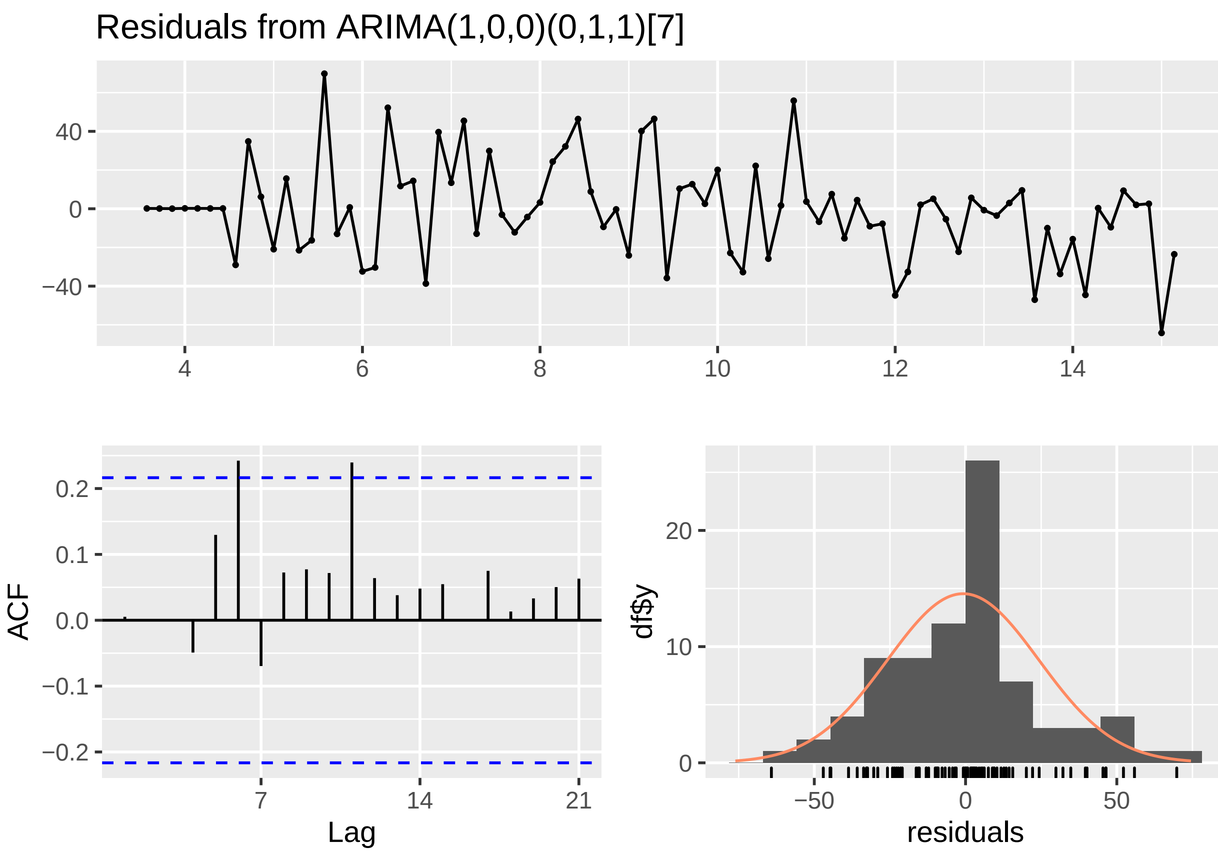
```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -1.287945 25.54684 18.88651 -3.500201 14.10580 0.7132369
## Test set     17.342891 36.78959 25.02225  7.190461 16.67359 0.9449491
##                    ACF1 Theil's U
## Training set 0.17012673        NA
## Test set     0.09562759 0.4953024
```

To choose the best of our ARIMA models, we focus on the test set values for the metrics, as these will better indicate the model's ability to perform with new unseen data. Because the ranking of the best model varies depending on the metric, we first need to understand what each metric tells us:

- Mean error (ME): average of the forecast errors;
- Root Mean Squared Error (RMSE): square root of the average of the forecast errors;
- Mean Absolute Error (MAE): average of the absolute values of the forecast errors;
- Mean Percentage Error (MPE): average of the forecast errors as a percentage of the actual values;
- Mean Absolute Percentage Error (MAPE): average of the absolute values of the forecast errors as a percentage of the actual values;
- Mean Absolute Scaled Error (MASE): a measure of the accuracy of the forecast against a naive forecast;
- Autocorrelation of Residuals (ACF1): autocorrelation of the forecast errors at lag 1;
- Theil's U: RMSE of the forecast over RMSE of a naive forecast.

The most commonly used and generally regarded as the best metrics to evaluate the accuracy of a forecast are the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). Therefore, we focus on the values for these metrics when evaluating the best model. Here, the first two models perform very similarly. Model 1, however, slightly outperforms model 2 in RMSE and MAPE. Therefore, we choose ARIMA(1, 0, 0)(0, 1, 1)[7] as the best of the three ARIMA models put against each other. We will likely use this to re-calibrate, although we first conduct a residuals analysis to see if it is satisfactory.

```r
# Analyse the residuals
checkresiduals(restaurant_46673.arima1)
```



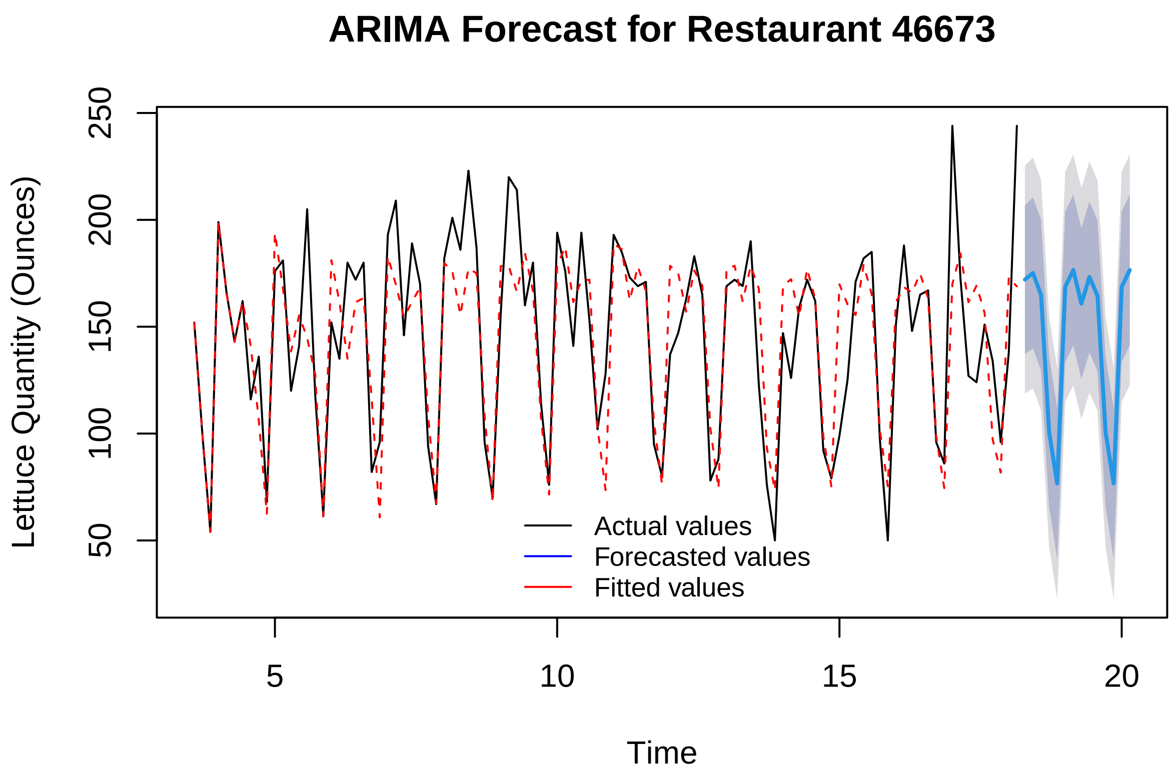

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,1)[7]
## Q* = 15.376, df = 12, p-value = 0.2215
##
## Model df: 2.   Total lags used: 14
```

The Ljung-Box test conducted for the ARIMA model suggests that the residuals are likely to be independently and identically distributed with no significant autocorrelation. Furthermore, the errors appear to have a constant variance and are normally distributed with a mean of zero. Finally, according to the ACF plot, there are spikes at lags 6 and 11. However, because there is no autocorrelation issue, we can assume these are due to white noise.

Therefore, the first model is satisfactory for forecasting lettuce demand for restaurant 46673. We can thus re-calibrate model 1 on the entire sample and forecast lettuce demand for the next 14 days.

```r
# Forecast
# Re-calibrate model with the entire sample
# Best model forecast (train on the whole dataset)
restaurant_46673.arima <- Arima(restaurant_46673_ts, order = c(1, 0, 0),
                                seasonal = list(order = c(0, 1, 1), period = 7),
                                include.drift = FALSE)
```

```r
restaurant_46673.arima.f <- forecast(restaurant_46673.arima, h = 14)

# Summary plot of actual, fitted, and forecasted values
plot(restaurant_46673.arima.f, main = "ARIMA Forecast for Restaurant 46673",
     xlab="Time", ylab="Lettuce Quantity (Ounces)", lty = 1, col = "black")
lines(fitted(restaurant_46673.arima.f), lty = 2, col = "red")
legend("bottom", legend=c("Actual values","Forecasted values", "Fitted values"),
       col=c("black", "blue", "red"), box.lty=0, lty=1, cex=0.8)
```



Final Forecast

Finally, because both the Holt-Winters and ARIMA models appear to be satisfactory, we compare the forecasting error of each sample on new unseen data (the test set) to choose the best one.

```r
# Forecasting error
accuracy(restaurant_46673.arima1.f, restaurant_46673_ts.test)
```

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.7949027 24.95910 18.28022 -2.980306 13.47389 0.6903407
## Test set     13.7331324 32.31547 21.75614  4.592271 14.93708 0.8216063
##                     ACF1 Theil's U
## Training set 0.005226471        NA
## Test set     0.067914914  0.437933
```

```r
accuracy(restaurant_46673.ets.f, restaurant_46673_ts.test)
```

```
##                     ME     RMSE       MAE        MPE      MAPE       MASE
```

```
## Training set -1.861661 23.22218 18.45667 -4.154372 14.30555 0.6970043
## Test set     24.672842 33.76837 25.43478 14.869195 16.39307 0.9605279
##                   ACF1 Theil's U
## Training set 0.05622874        NA
## Test set     0.10348431 0.4214732
```

We compare the performance of each model based on the previously defined preferred metrics of RMSE, MAE, and MAPE. Because the ARIMA model performs better in all these metrics, we choose this one and re-calibrate with the entire sample to forecast lettuce demand for restaurant 46673 for the upcoming 14 days.

```r
# Re-calibrate model with the entire sample
restaurant_46673.model <- Arima(restaurant_46673_ts, order = c(1, 0, 0),
                                seasonal = list(order = c(0, 1, 1), period = 7),
                                include.drift = FALSE)
restaurant_46673.model.f <- forecast(restaurant_46673.model, h = 14)

# Convert to a data frame
restaurant_46673.model.f.df <- as.data.frame(restaurant_46673.model.f)
rownames(restaurant_46673.model.f.df) <- c('2015-06-16', '2015-06-17', '2015-06-18',
                                           '2015-06-19', '2015-06-20', '2015-06-21',
                                           '2015-06-22', '2015-06-23', '2015-06-24',
                                           '2015-06-25', '2015-06-26', '2015-06-27',
                                           '2015-06-28', '2015-06-29')
restaurant_46673.model.f.df
```

```
##            Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 2015-06-16      172.06567 137.22069 206.9107 118.77488 225.3565
## 2015-06-17      175.15870 139.83185 210.4855 121.13095 229.1864
## 2015-06-18      164.80336 129.53950 200.0672 110.87194 218.7348
## 2015-06-19      100.78962  65.52539 136.0539  46.85763 154.7216
## 2015-06-20       76.67597  41.41172 111.9402  22.74396 130.6080
## 2015-06-21      168.66758 133.40333 203.9318 114.73557 222.5996
## 2015-06-22      176.52969 141.26544 211.7939 122.59768 230.4617
## 2015-06-23      160.78112 125.44050 196.1217 106.73230 214.8299
## 2015-06-24      173.27133 137.93071 208.6120 119.22251 227.3201
## 2015-06-25      164.48770 129.22345 199.7519 110.55568 218.4197
## 2015-06-26      100.73682  65.47257 136.0011  46.80480 154.6688
## 2015-06-27       76.66714  41.40288 111.9314  22.73512 130.5992
## 2015-06-28      168.66610 133.40185 203.9304 114.73408 222.5981
## 2015-06-29      176.52944 141.26519 211.7937 122.59742 230.4615
```

We can use the above table to find the forecasted lettuce demand for an expected date and to understand the possible range of this demand. Indeed, the *Lo 80* and *Hi 80* columns define the limits of the 80% forecast interval, and the *Lo 95* and *Hi 95* columns represent the limits of the 95% forecast interval. For example, on the 16th of June 2015, we expect a demand for lettuce of around 172 ounces and are 95% confident that the demand would be between 119 and 225 ounces. If we were to generate more forecasts based on the same model and data, 95% would contain the actual value.

California 2 (ID:4904)

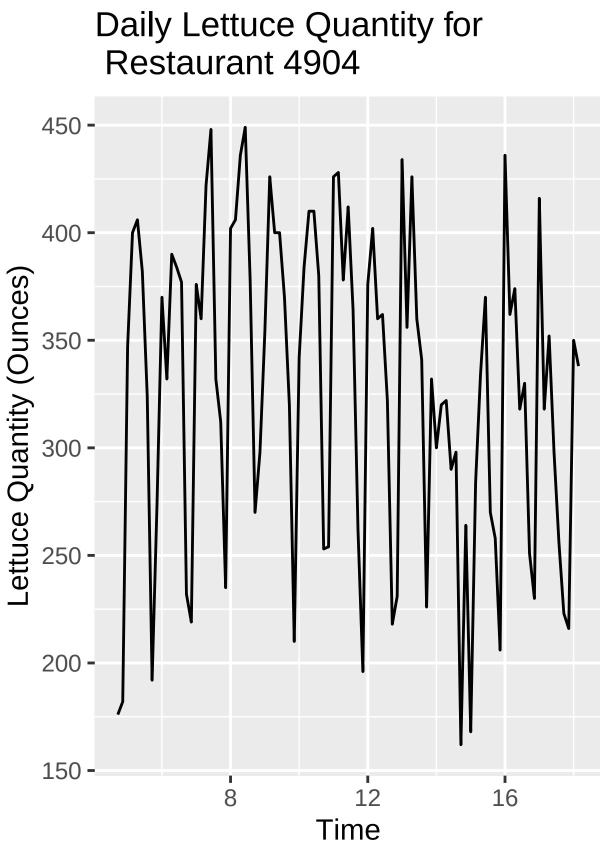
```
restaurant_4904 <- read_csv('restaurant_4904.csv')
restaurant_4904[1, 2] # Find the start date
```

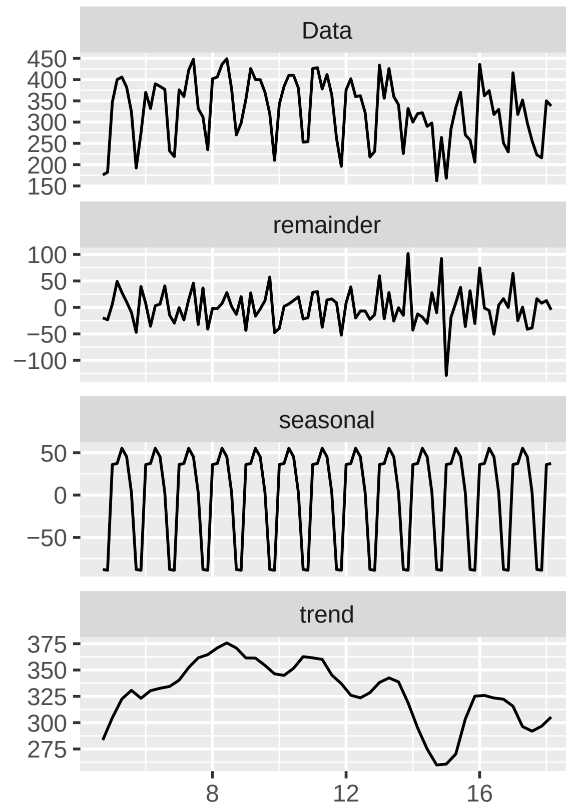
```
## # A tibble: 1 x 1
##   date
##   <date>
## 1 2015-03-13
```

Holt-Winters Model

We start by creating a time series object for restaurant 4904 and studying any trend or seasonality in the data. The time series object for restaurant 4904 is created with a frequency of 7, corresponding to the cycle's length and starting on the 13th of March 2015.

```
restaurant_4904_ts <- ts(restaurant_4904$`Quantity (ounces)`,
                         frequency = 7, # 7-day cycle
                         start = c(03, 13)) # Starts on the 13th of March

# Plot of time-series
restaurant_4904_ts.plot1 <- autoplot(restaurant_4904_ts) +
  ggtitle('Daily Lettuce Quantity for \n Restaurant 4904') +
  xlab('Time') +
  ylab('Lettuce Quantity (Ounces)')

# Plot of seasonal decomposition
restaurant_4904_ts.plot2 <-restaurant_4904_ts %>% stl(s.window = "period") %>% autoplot

grid.arrange(restaurant_4904_ts.plot1, restaurant_4904_ts.plot2, ncol = 2)
```





From the time series decomposition, there doesn't appear to be a trend, so the trend component can be ignored. However, an additive seasonality component will need to be accounted for in the exponential smoothing model. First, however, we split the data between a training and a testing set according to an 80/20 ratio.

```
# Calculate the number of observations in the training set
n_train_4904 <- round(length(restaurant_4904_ts) * 0.8)

# Splitting into train and test
# Train set: 80%
restaurant_4904_ts.train <- subset(restaurant_4904_ts, end = n_train_4904)

# Test set: 20%
restaurant_4904_ts.test <- subset(restaurant_4904_ts, start = n_train_4904+1)
```

Then, we fit the exponential smoothing model with the ets() function. In the ets() function, we set the model to be 'ANA' such that we pre-define an additive error, no trend, and additive seasonality. We can also check that our model specification is correct by building another model where we set the model to be 'ZZZ' such that the error, trend, and seasonality are not pre-defined.

```
restaurant_4904.ets1 <- ets(restaurant_4904_ts.train, model = 'ANA')
restaurant_4904.ets2 <- ets(restaurant_4904_ts.train, model = 'ZZZ')

restaurant_4904.ets2
```

```
## ETS(A,A,A)
##
```

```
## Call:
##  ets(y = restaurant_4904_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
```

Because the 'ZZZ' specification output is 'AAA', the model identifies an additive trend which may have been missed in the decomposition analysis. Thus, the error, trend, and seasonality are additive. The resulting smoothing constants are $\alpha = 0.0127$, $\beta = 0.0125$ and $\gamma = 0.0005$. The initial states, and thus starting values given by the 'ZZZ' specification for the error, trend, and seasonal components are 315.9648 for the level; 3.6247 for the trend; 8.1102, 50.6401, 57.8214, 40.4994, 18.3055, -80.5061, -94.8704 for the seven seasonal components. The estimated standard deviation of the error term is equal to 42.8103.

We can also find the values given by the 'ANA' model specification.

```
restaurant_4904.ets1
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = restaurant_4904_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
```

The smoothing constants for the 'ANA' specification are $\alpha = 0.1935$, $\beta = 0$ and $\gamma = 0.0001$. With this specification, the initial states are 335.4537 for the level and 10.9948, 54.9311, 56.7306, 38.9975, 20.9684, -86.7199, and -95.9024 for the seven seasonal components. The estimated standard deviation of the error term is equal to 43.9461.

Because the two models' definitions differ, we can conduct an out-of-sample evaluation for both the 'ANA' and 'AAA' models and see which performs better when presented with new data.

```
# Out-of-sample evaluation
restaurant_4904.ets1.f <- forecast(restaurant_4904.ets1, h = 14)
restaurant_4904.ets2.f <- forecast(restaurant_4904.ets2, h = 14)

# Forecasting error
accuracy(restaurant_4904.ets1.f, restaurant_4904_ts.test)
```

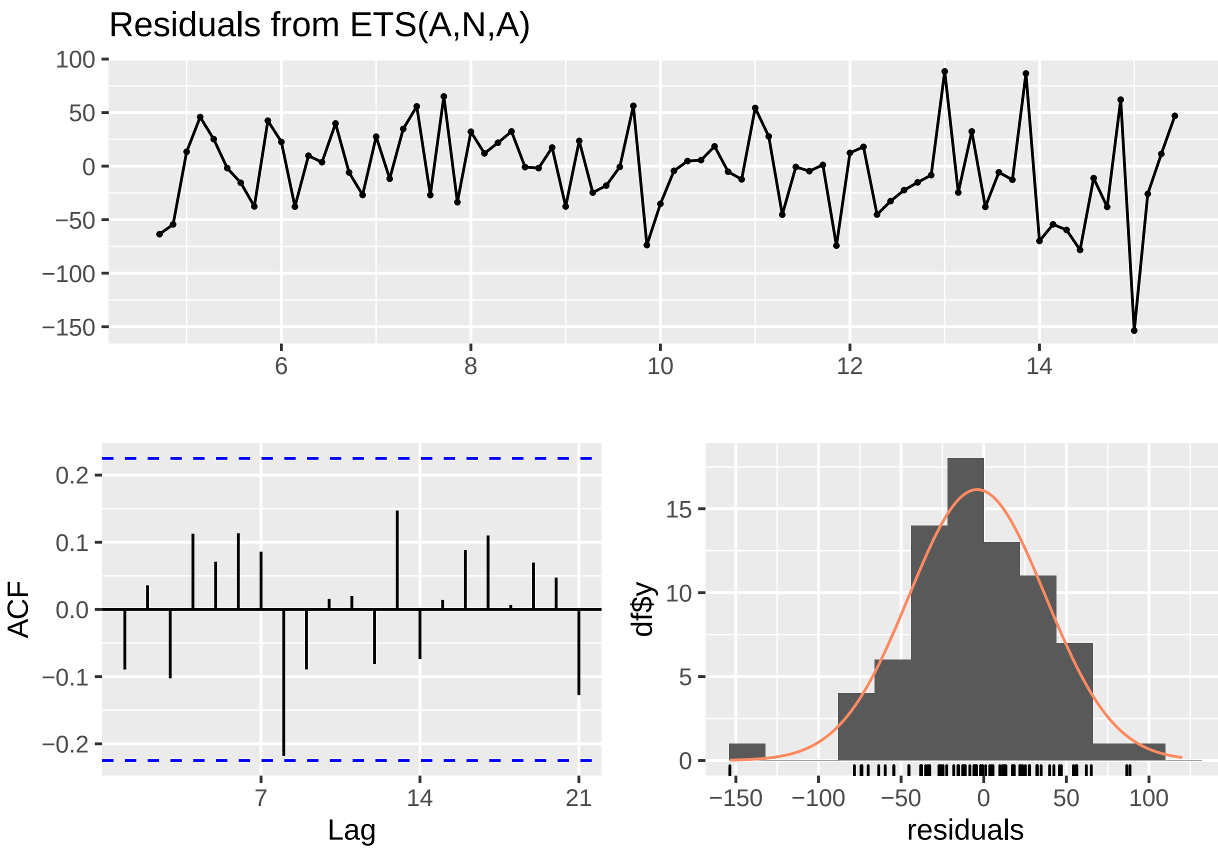
```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -3.960539 41.26204 31.63586 -3.438702 10.98139 0.7382057
## Test set     38.446807 61.47467 47.93970 11.519435 14.75281 1.1186471
##                     ACF1 Theil's U
## Training set -0.08936168        NA
## Test set     -0.16452289 0.6533893
```

```
accuracy(restaurant_4904.ets2.f, restaurant_4904_ts.test)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -8.866754 39.59116 30.57049 -4.868275 10.79776 0.7133458
## Test set     82.284751 95.46099 82.66336 25.587658 25.72788 1.9289048
##                    ACF1 Theil's U
## Training set -0.0497177        NA
## Test set     -0.2138363   0.95271
```

The first of the two models, 'ANA', performs significantly better than the other one regarding out-of-sample data. The 'AAA' specification however does perform better in-sample, explaining the choice of parameters when setting model to 'ZZZ'. Therefore, the 'ANA' model is the one we are most likely to use to re-calibrate the entire sample and forecast the quantity of lettuce demanded by restaurant 4904 for the upcoming 14 days. However, before that, we analyse the residuals to determine if the model is satisfactory.

```
# Analyse the residuals
checkresiduals(restaurant_4904.ets1.f)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,A)
## Q* = 12.858, df = 14, p-value = 0.5378
##
## Model df: 0.   Total lags used: 14
```

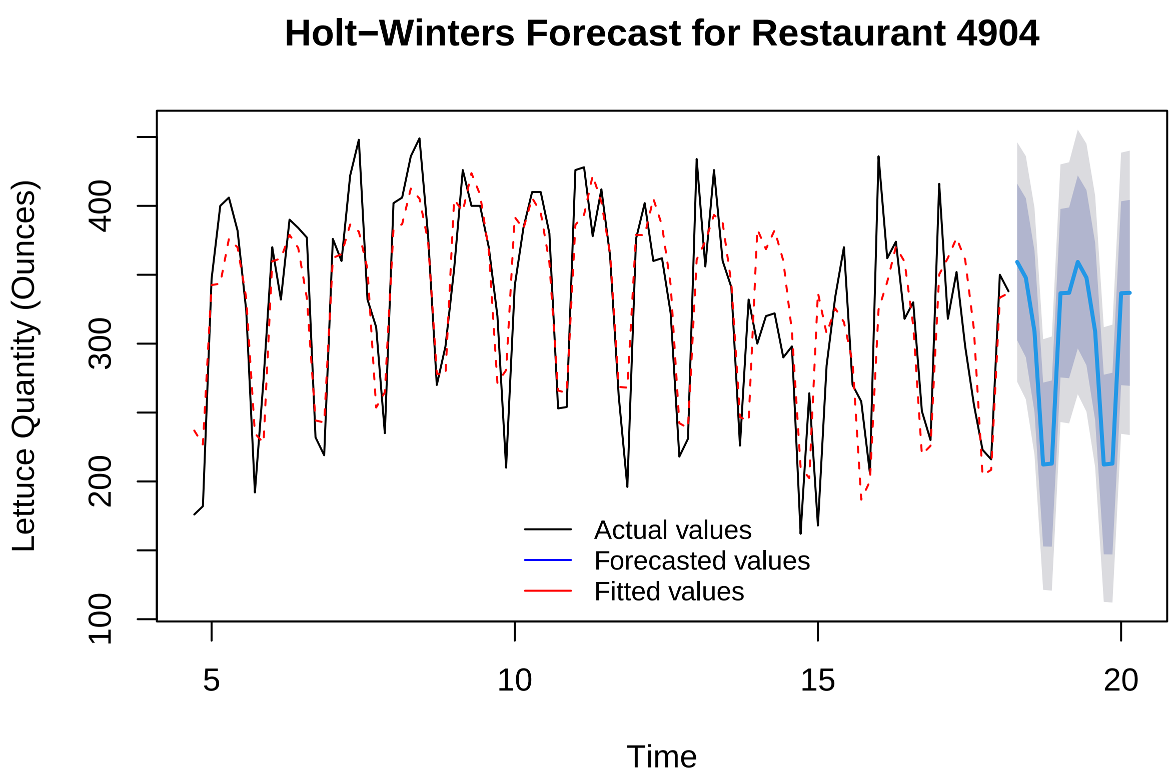
The Ljung-Box test suggests that the Holt Winter model's residuals are likely to be independently and identically distributed with no significant autocorrelation as we fail to reject the null of independent residuals. Furthermore, the errors appear to have a constant variance and are normally distributed with a mean of zero. In addition, looking at the ACF plot of the first model, the absence of spike further confirms our choice of the first model as the best to use for the forecast.

Therefore, we conclude that the exponential smoothing model with the 'ANA' specification is satisfactory for forecasting lettuce demand for restaurant 4904.

We can then re-calibrate the entire sample and forecast lettuce demand for the next 14 days. Plotting the output further allows us to visualise the model's performance. Again, the real data is plotted as a black line, the fitted data as a red dotted line, and the forecasted data as a blue line.

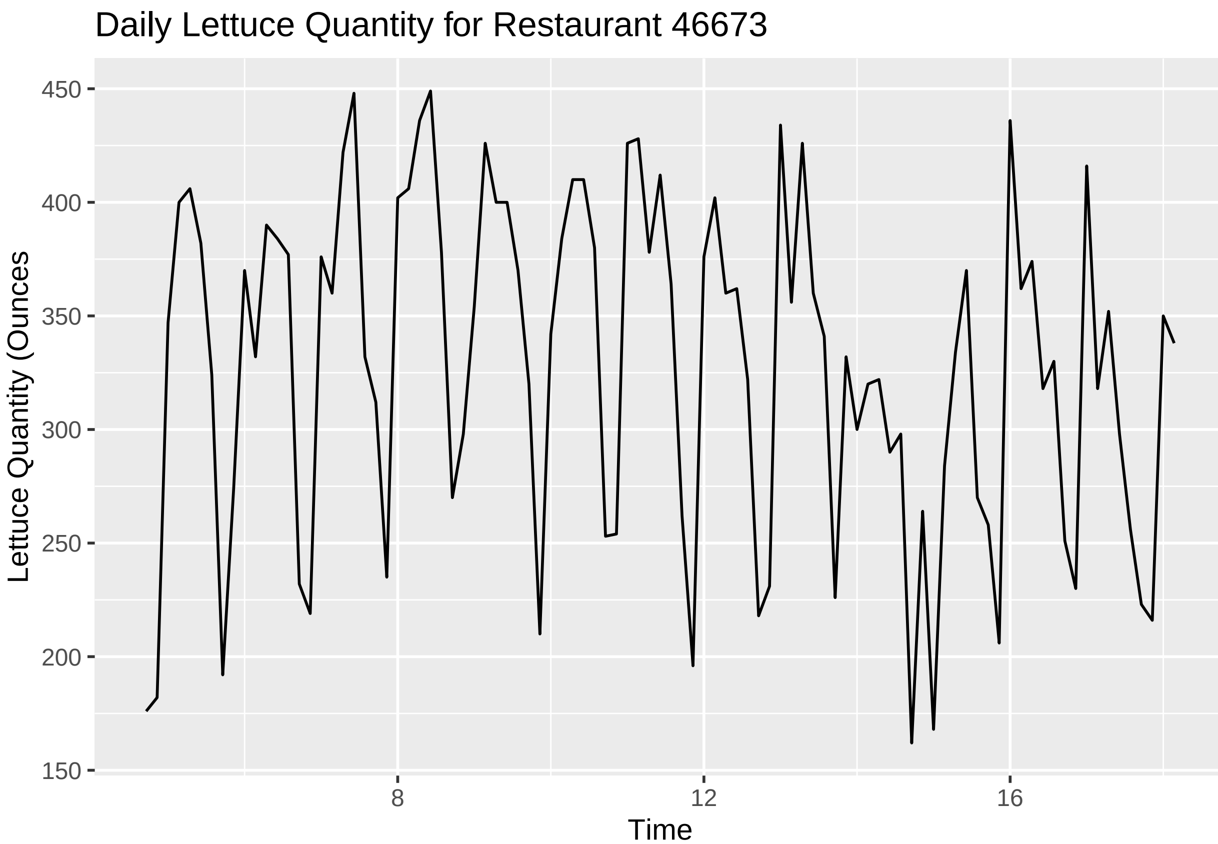
```
# Forecast
# Re-calibrate model with the entire sample
restaurant_4904.selected.model <- ets(restaurant_4904_ts, model = "ANA")
restaurant_4904.selected.model.f <- forecast(restaurant_4904.selected.model, h = 14)

# Summary plot of actual, fitted, and forecasted values
```

```r
plot(restaurant_4904.selected.model.f, main = "Holt-Winters Forecast for Restaurant 4904",
     xlab="Time", ylab="Lettuce Quantity (Ounces)", lty = 1, col = "black")
lines(fitted(restaurant_4904.selected.model.f), lty = 2, col = "red")
legend("bottom", legend=c("Actual values","Forecasted values", "Fitted values"),
       col=c("black", "blue", "red"), box.lty=0, lty=1, cex=0.8)
```



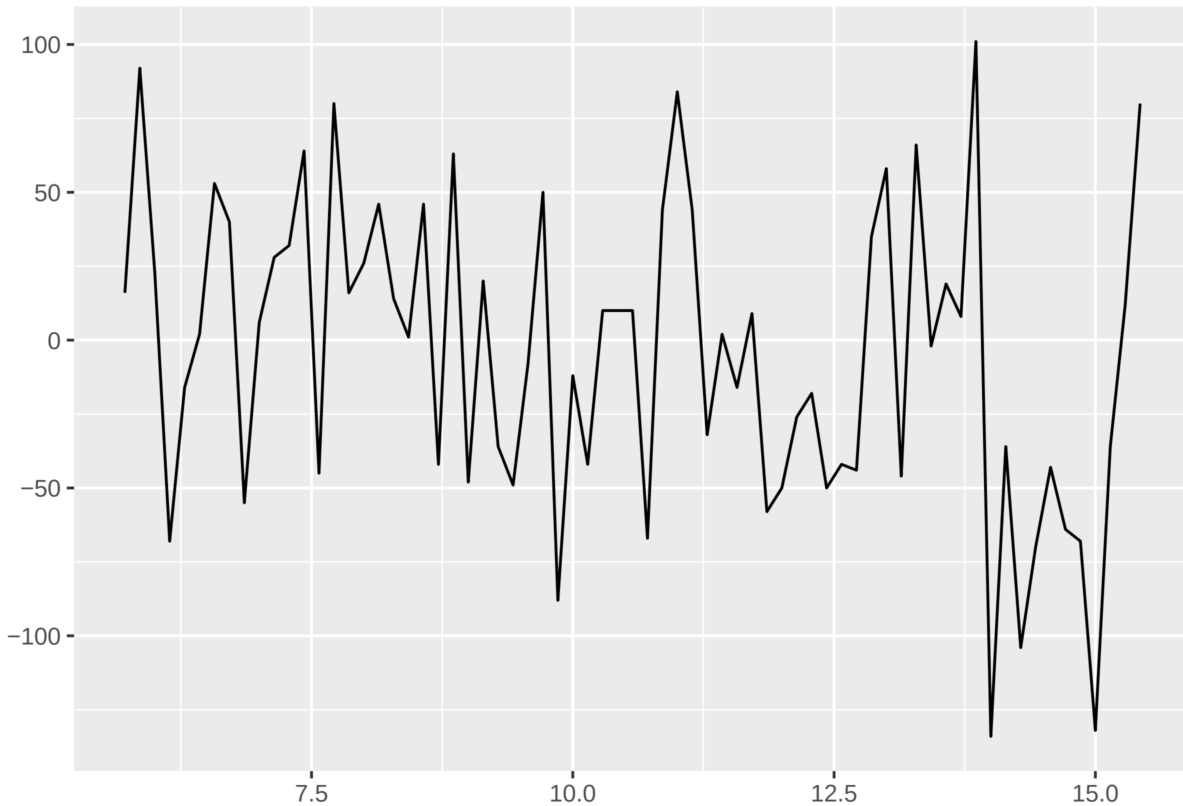
ARIMA Model

```r
# Plot of time-series
autoplot(restaurant_4904_ts) +
  ggtitle('Daily Lettuce Quantity for Restaurant 46673') +
  xlab('Time') +
  ylab('Lettuce Quantity (Ounces')
```



The above plot shows that the time series seems stationary in terms of mean and variance, which can be verified with several tests. Also, we can remember that no trend was observed when decomposing the time series, but that we observed seasonality. Therefore, we need to apply one seasonal difference.

```
# Apply one seasonal difference
restaurant_4904_ts.diff <- diff(restaurant_4904_ts.train, differences = 1, lag = 7)
autoplot(restaurant_4904_ts.diff) # no seasonal trend
```




```r
# Stationarity test
adf.test(restaurant_4904_ts.diff) # Augmented Dickey-Fuller Test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  restaurant_4904_ts.diff
## Dickey-Fuller = -2.8934, Lag order = 4, p-value = 0.2121
## alternative hypothesis: stationary
```

```r
pp.test(restaurant_4904_ts.diff) # Phillips-Perron Unit Root Test
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  restaurant_4904_ts.diff
## Dickey-Fuller Z(alpha) = -68.18, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

```r
kpss.test(restaurant_4904_ts.diff) # KPSS Test for Level Stationarity
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  restaurant_4904_ts.diff
## KPSS Level = 0.50699, Truncation lag parameter = 3, p-value = 0.04009
```

```r
ndiffs(restaurant_4904_ts.train)
```

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

```r
# Seasonal stationarity
nsdiffs(restaurant_4904_ts.train)
```

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

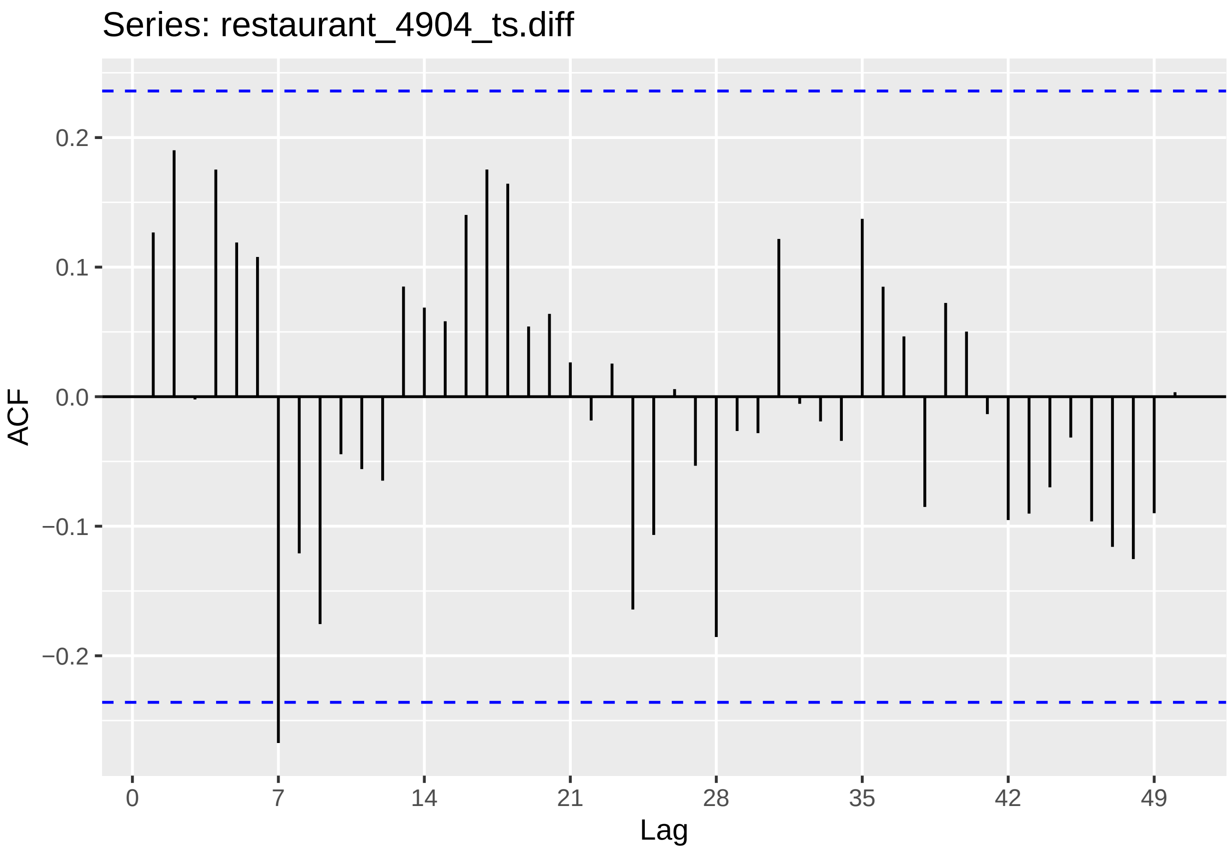
From the above tests, we have the following conclusions:

- ADF: insufficient evidence to reject the null that we have a unit root; thus, the time series may not be stationary.
- PP: sufficient evidence to reject the null; thus, the time series is stationary.
- KPSS: sufficient evidence to reject the null; thus, we accept the alternative that the data is non-stationary.

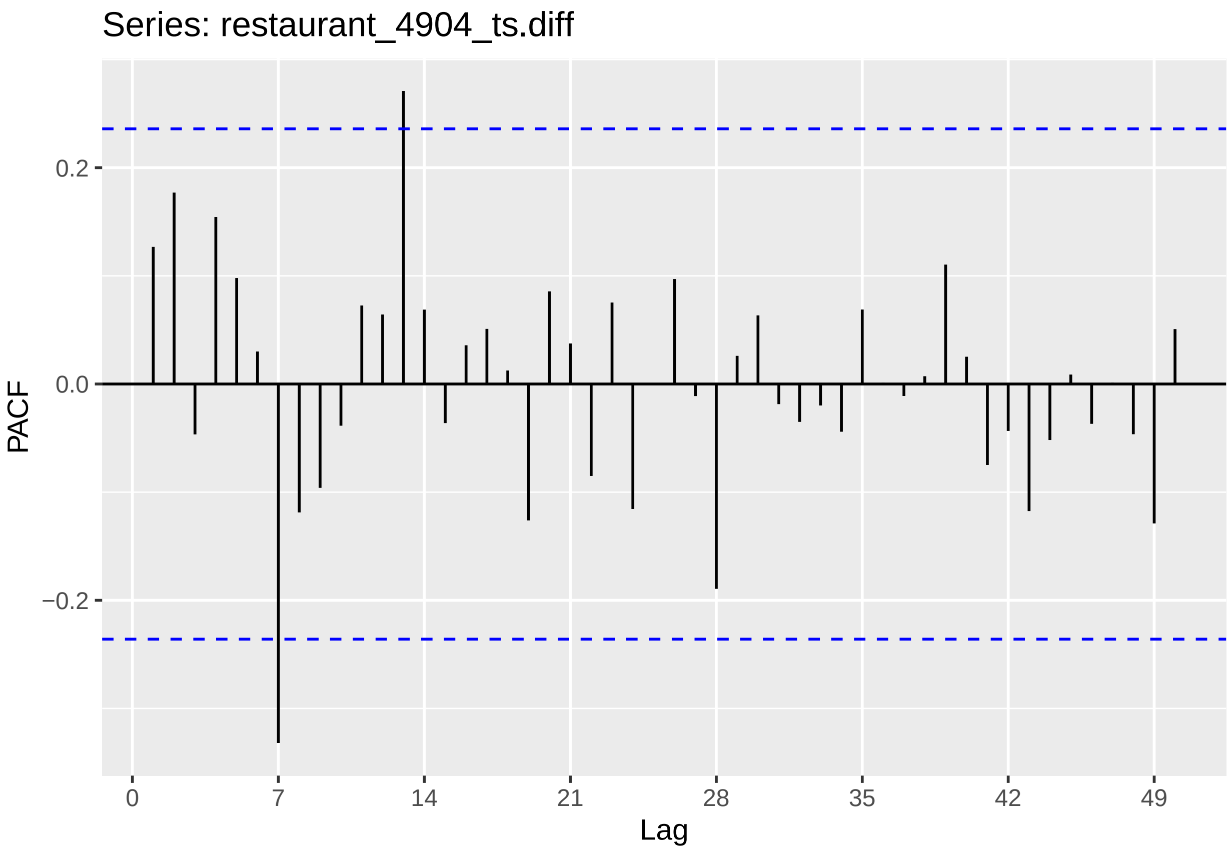
From the above-described tests, the time series is likely not stationary regarding trend. However, the *ndiffs()* function, an automated way to determine the appropriate number of times to differentiate a time series data, returns a value of 0, thus suggesting that the data is stationary. Looking at an ARIMA model of the form ARIMA(p, d, q)(P, D, Q)[7], we are not certain whether to expect a value of d = 0 or d = 1. Because the two outcomes seem justified, we will test models with each value. In addition, it is not stationary in terms of seasonality, which is consistent with our preliminary decomposition analysis. The output of 1 from the *nsdiffs()* function confirms that we needed to apply one seasonal difference. We, therefore, expect an ARIMA model where D = 1.

We now need to determine the ARIMA model's p, q, P, and Q parameters. We also automatically select the best ARIMA model for comparison based on the AIC_c.

```r
# Choice of p, q, P and Q
ggAcf(restaurant_4904_ts.diff, lag.max = 50)
```




```
ggPacf(restaurant_4904_ts.diff, lag.max = 50)
```




```r
# Find the best ARIMA model
auto.arima(restaurant_4904_ts.train, trace = TRUE, ic = "aicc")
```

```
##
##  ARIMA(2,1,2)(1,1,1)[7]                    : 734.3954
##  ARIMA(0,1,0)(0,1,0)[7]                    : 769.7228
##  ARIMA(1,1,0)(1,1,0)[7]                    : 743.4943
##  ARIMA(0,1,1)(0,1,1)[7]                    : 726.6895
##  ARIMA(0,1,1)(0,1,0)[7]                    : 736.6688
##  ARIMA(0,1,1)(1,1,1)[7]                    : 728.243
##  ARIMA(0,1,1)(0,1,2)[7]                    : 728.3977
##  ARIMA(0,1,1)(1,1,0)[7]                    : 728.8278
##  ARIMA(0,1,1)(1,1,2)[7]                    : 730.6263
##  ARIMA(0,1,0)(0,1,1)[7]                    : 762.5365
##  ARIMA(1,1,1)(0,1,1)[7]                    : 728.492
##  ARIMA(0,1,2)(0,1,1)[7]                    : 728.4945
##  ARIMA(1,1,0)(0,1,1)[7]                    : 740.8211
##  ARIMA(1,1,2)(0,1,1)[7]                    : 730.36
##
##  Best model: ARIMA(0,1,1)(0,1,1)[7]
```

```
## Series: restaurant_4904_ts.train
## ARIMA(0,1,1)(0,1,1)[7]
##
## Coefficients:
##           ma1     sma1
```

```
##       -0.8413  -0.5869
## s.e.   0.0720   0.1838
##
## sigma^2 = 2248:  log likelihood = -360.16
## AIC=726.31   AICc=726.69   BIC=732.97
```

The above ACF plot displays a sinusoidal and not an exponential decay. Furthermore, by analysing the spikes, the ACF plot shows a spike at lag 7. Therefore, we can define a seasonal MA component of 1 (Q = 1) and a non-seasonal MA component of 0 (q = 0). The PACF plot also decays in a sinusoidal manner, and we see a spike at a lag of 7 and a spike at a lag of 13. Thus, it may be necessary to integrate a seasonal AR component of 1 (P = 1). However, the spike at lag 13 is most likely due to white noise, so we choose not to define a corresponding model and only take a non-seasonal AR component of 0 (p = 0).

From these two plots, we can define the models ARIMA(0,0,0)(1,1,1)[7] and ARIMA(0,1,0)(1,1,1)[7] to be tested against the two best models defined from the auto.ARIMA function.

The best ARIMA model as determined by the *auto.ARIMA()* function is ARIMA(0,1,1)(0,1,1)[7], and the second best is ARIMA(0,1,1)(1,1,1)[7]. We train and forecast for both models to find the best one.

```r
# Candidate models
restaurant_4904.arima1 <- Arima(restaurant_4904_ts.train, order = c(0, 1, 1),
                               seasonal = list(order = c(0, 1, 1), period = 7),
                               include.drift = FALSE)
restaurant_4904.arima2 <- Arima(restaurant_4904_ts.train, order = c(0, 1, 1),
                               seasonal = list(order = c(1, 1, 1), period = 7),
                               include.drift = FALSE)
restaurant_4904.arima3 <- Arima(restaurant_4904_ts.train, order = c(0, 0, 0),
                               seasonal = list(order = c(1, 1, 1), period = 7),
                               include.drift = FALSE)
restaurant_4904.arima4 <- Arima(restaurant_4904_ts.train, order = c(0, 1, 0),
                               seasonal = list(order = c(1, 1, 1), period = 7),
                               include.drift = FALSE)

# Model evaluation
# Forecast
restaurant_4904.arima1.f <- forecast(restaurant_4904.arima1, h = 14)
restaurant_4904.arima2.f <- forecast(restaurant_4904.arima2, h = 14)
restaurant_4904.arima3.f <- forecast(restaurant_4904.arima3, h = 14)
restaurant_4904.arima4.f <- forecast(restaurant_4904.arima4, h = 14)

# Out-of-sample performance
accuracy(restaurant_4904.arima1.f, restaurant_4904_ts.test)
```

```
##                    ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
## Training set -7.63880 44.18003 31.90097 -3.637262 10.72865 0.744392 -0.1074127
## Test set     68.93341 94.85250 72.55725 19.942468 21.54263 1.693084 -0.3804041
##              Theil's U
## Training set        NA
## Test set     0.9968718
```

```r
accuracy(restaurant_4904.arima2.f, restaurant_4904_ts.test)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -7.644354 43.34083 31.34353 -3.553584 10.53472 0.7313845
```

```
## Test set     59.314728 85.02060 63.64462 17.355464 18.91362 1.4851128
##                     ACF1 Theil's U
## Training set -0.09763621        NA
## Test set     -0.30061699 0.9087127
```

```
accuracy(restaurant_4904.arima3.f, restaurant_4904_ts.test)
```

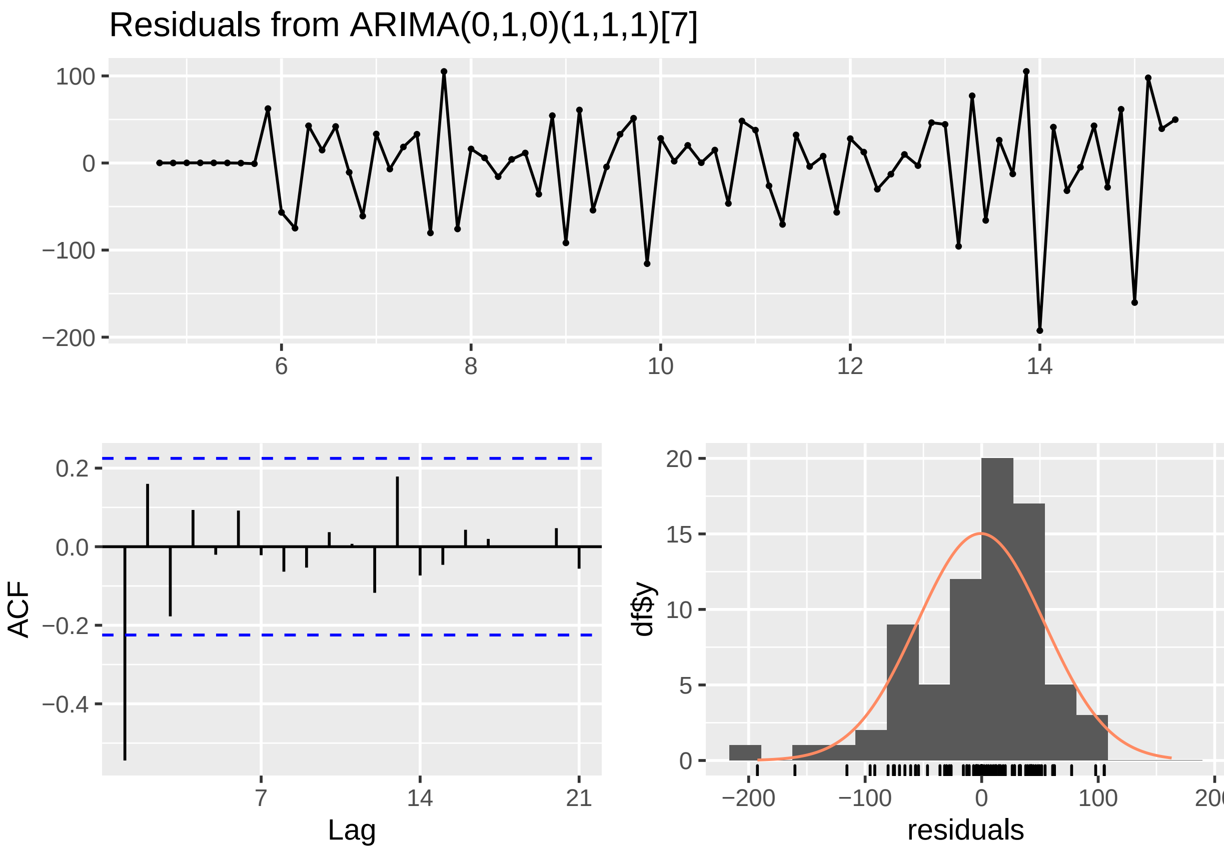
```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -4.816394 47.42281 35.56333 -3.121887 11.98290 0.8298512
## Test set     36.707332 96.06832 72.42244  7.877955 22.35198 1.6899387
##                    ACF1 Theil's U
## Training set  0.1469180        NA
## Test set     -0.3819224  1.054555
```

```
accuracy(restaurant_4904.arima4.f, restaurant_4904_ts.test)
```

```
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  -0.8067078 54.26392 39.35234  -1.692753 13.23252 0.9182657
## Test set     -27.5761788 69.10602 58.17567 -11.147944 18.88997 1.3574978
##                    ACF1 Theil's U
## Training set -0.5442289        NA
## Test set     -0.1373644 0.6211783
```

The best of the four ARIMA models put against each other in terms of RMSE, MAE, and MAPE is the fourth model, ARIMA(0, 1, 0),(1, 1, 1)[7]. We are likely to use this to re-calibrate with the entire sample. Before doing so, however, we check whether it is satisfactory by analysing the residuals.

```
# Analyse the residuals
checkresiduals(restaurant_4904.arima4.f)
```

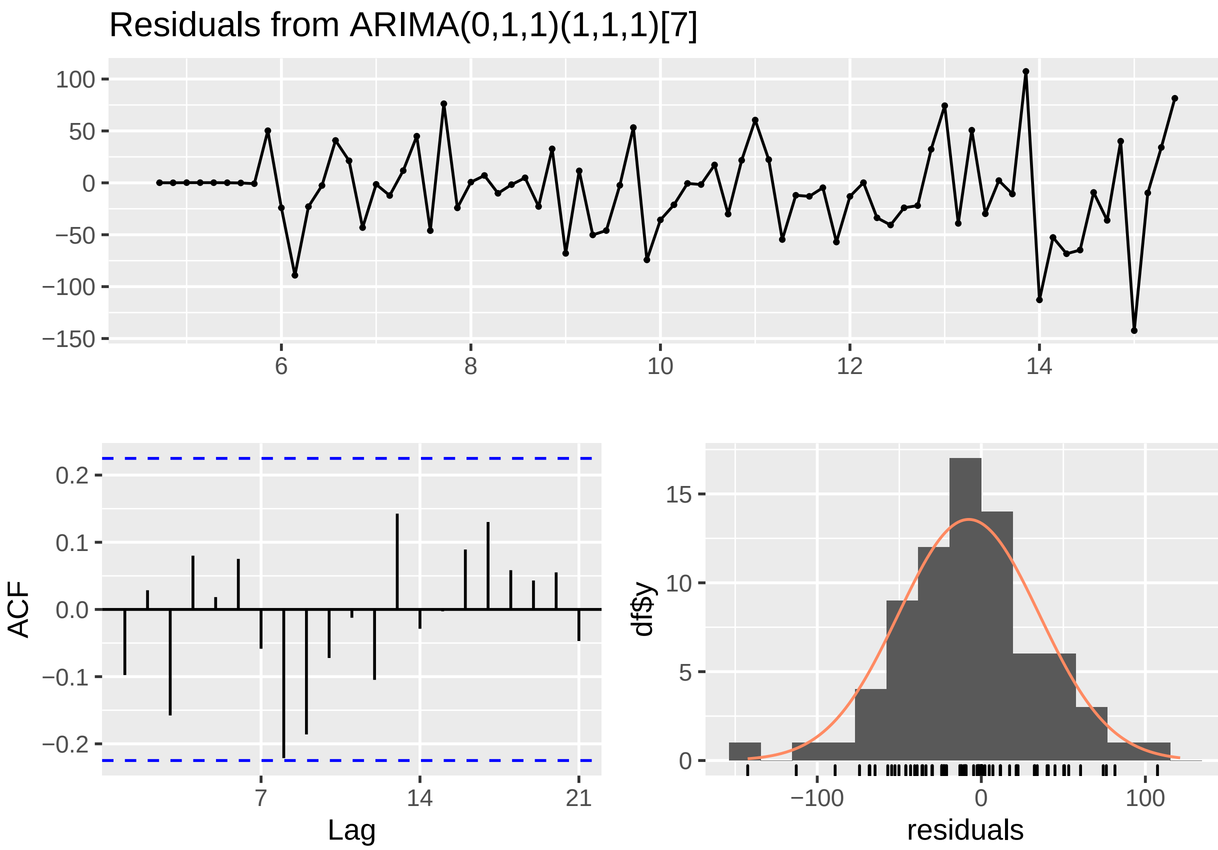



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)(1,1,1)[7]
## Q* = 35.046, df = 12, p-value = 0.0004605
##
## Model df: 2.   Total lags used: 14
```

The Ljung-Box tests suggests that the ARIMA model's residuals are likely not to be independently and identically distributed. Indeed, the Ljung-Box test shows a p-value of 0.0004605, significantly lower than the 0.05 threshold, thus rejecting the null hypothesis of independent residuals in favour of the alternative that the residuals are not independent.

Therefore, we conclude that the fourth model is not satisfactory for forecasting lettuce demand for restaurant 4904, and thus that we should choose the second-best in terms of out-of-sample performance if the residuals are satisfactory, which in this case is model 2, ARIMA(0,1,1)(1,1,1)[7].

```
# Analyse the residuals
checkresiduals(restaurant_4904.arima2.f)
```



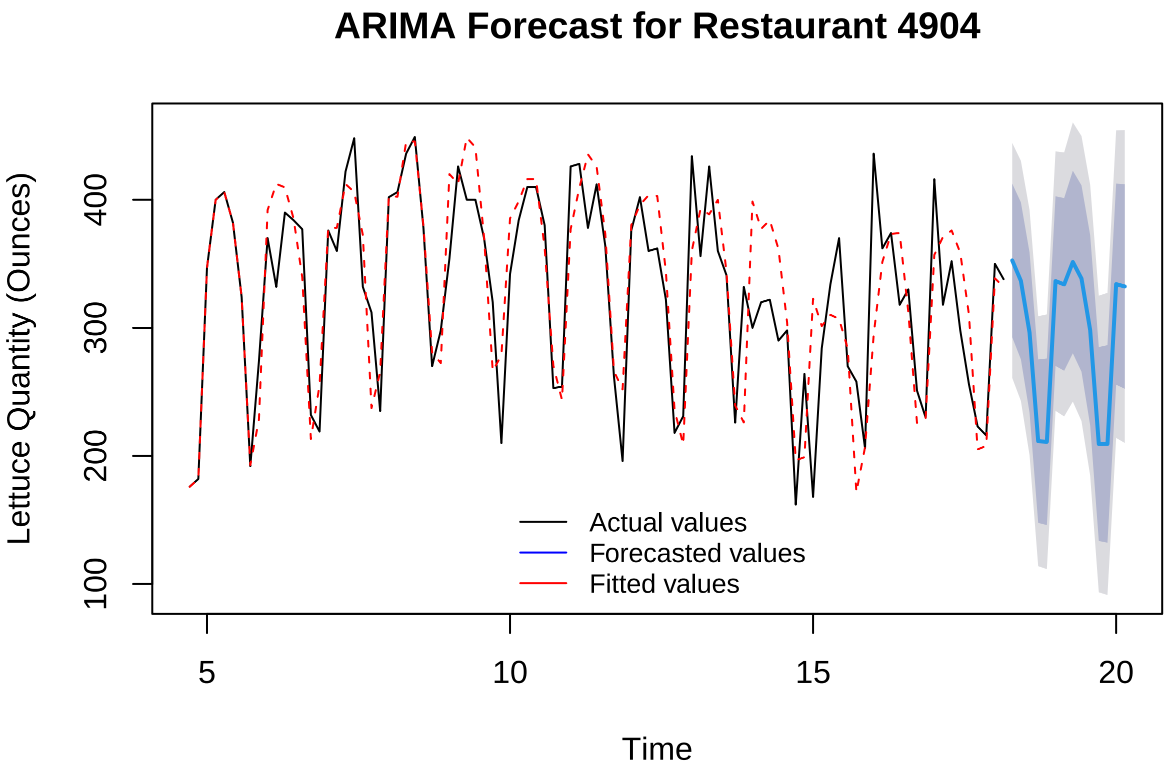

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(1,1,1)[7]
## Q* = 14.989, df = 11, p-value = 0.183
##
## Model df: 3.   Total lags used: 14
```

The Ljung-Box test suggests that the ARIMA model's residuals are likely independently and identically distributed, and the errors appear to have a constant variance and are normally distributed with a mean of zero for all models.

Therefore, we re-calibrate the model on our entire dataset.

```
# Forecast
# Re-calibrate model with the entire sample
# Best model forecast (train on the whole dataset)
restaurant_4904.arima <- Arima(restaurant_4904_ts, order = c(0, 1, 1),
                                seasonal = list(order = c(1, 1, 1), period = 7),
                                include.drift = FALSE)
restaurant_4904.arima.f <- forecast(restaurant_4904.arima, h = 14)

# Summary plot of actual, fitted, and forecasted values
plot(restaurant_4904.arima.f, main = "ARIMA Forecast for Restaurant 4904",
     xlab="Time", ylab="Lettuce Quantity (Ounces)", lty = 1, col = "black")
lines(fitted(restaurant_4904.arima.f), lty = 2, col = "red")
```

```r
legend("bottom", legend=c("Actual values","Forecasted values", "Fitted values"),
       col=c("black", "blue", "red"), box.lty=0, lty=1, cex=0.8)
```



Final Forecast

Finally, we compare the forecasting error of each sample on new unseen data (the test set) to choose the best one.

```r
# Forecasting error
accuracy(restaurant_4904.arima2.f, restaurant_4904_ts.test)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -7.644354 43.34083 31.34353 -3.553584 10.53472 0.7313845
## Test set     59.314728 85.02060 63.64462 17.355464 18.91362 1.4851128
##                     ACF1 Theil's U
## Training set -0.09763621        NA
## Test set     -0.30061699 0.9087127
```

```r
accuracy(restaurant_4904.ets1.f, restaurant_4904_ts.test)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -3.960539 41.26204 31.63586 -3.438702 10.98139 0.7382057
## Test set     38.446807 61.47467 47.93970 11.519435 14.75281 1.1186471
##                     ACF1 Theil's U
## Training set -0.08936168        NA
## Test set     -0.16452289 0.6533893
```

As previously mentioned, the most commonly used and generally regarded as the best metrics to evaluate the accuracy of a forecast are the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). Because we want to minimise these values, we choose the Holt-Winters model and find the forecasted lettuce demand for restaurant 4904 for the next 14 days.

```r
# Re-calibrate model with the entire sample
restaurant_4904.model <- ets(restaurant_4904_ts, model = 'ANA')
restaurant_4904.model.f <- forecast(restaurant_4904.model, h = 14)

# Convert to a data frame
restaurant_4904.model.f.df <- as.data.frame(restaurant_4904.model.f)
rownames(restaurant_4904.model.f.df) <- c('2015-06-16', '2015-06-17', '2015-06-18',
                                          '2015-06-19', '2015-06-20', '2015-06-21',
                                          '2015-06-22', '2015-06-23', '2015-06-24',
                                          '2015-06-25', '2015-06-26', '2015-06-27',
                                          '2015-06-28', '2015-06-29')
restaurant_4904.model.f.df
```

```
##            Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015-06-16       359.2945 302.4655 416.1235 272.3820 446.2069
## 2015-06-17       347.8354 290.1187 405.5521 259.5654 436.1055
## 2015-06-18       308.8660 250.2751 367.4569 219.2589 398.4731
## 2015-06-19       212.2648 152.8125 271.7171 121.3403 303.1892
## 2015-06-20       212.9511 152.6498 273.2525 120.7282 305.1741
## 2015-06-21       336.5967 275.4580 397.7353 243.0932 430.1002
## 2015-06-22       336.8785 274.9129 398.8440 242.1103 431.6466
## 2015-06-23       359.2945 296.5138 422.0751 263.2798 455.3092
## 2015-06-24       347.8354 284.2501 411.4207 250.5901 445.0807
## 2015-06-25       308.8660 244.4861 373.2459 210.4055 407.3265
## 2015-06-26       212.2648 147.1000 277.4295 112.6038 311.9257
## 2015-06-27       212.9511 147.0108 278.8915 112.1041 313.7982
## 2015-06-28       336.5967 269.8898 403.3035 234.5773 438.6160
## 2015-06-29       336.8785 269.4129 404.3440 233.6988 440.0581
```

New York 1 (ID:12631)

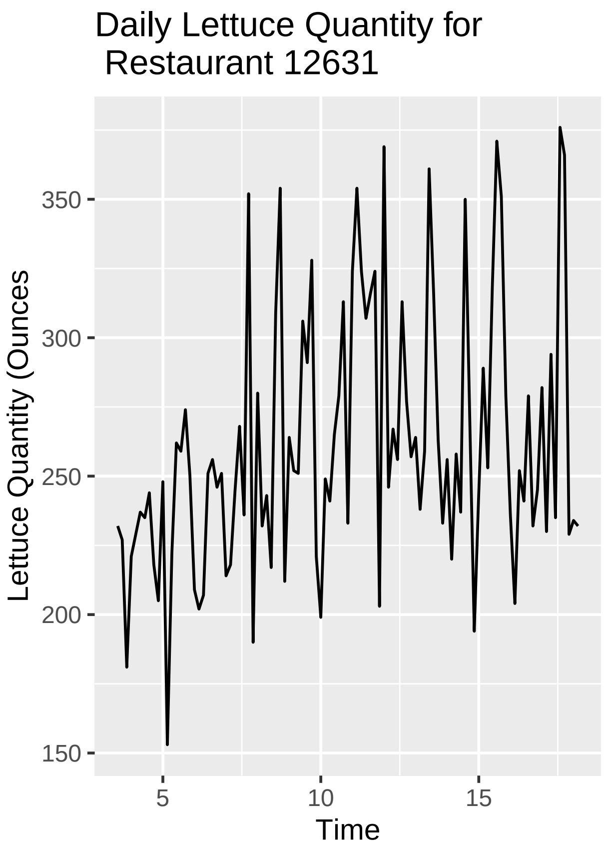
```
restaurant_12631 <- read_csv('restaurant_12631.csv')
restaurant_12631[1, 2] # Find the start date
```

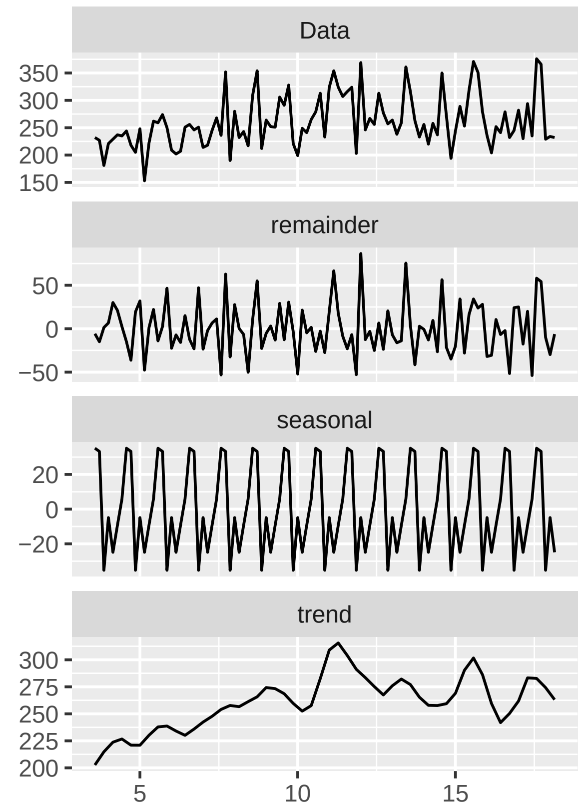
```
## # A tibble: 1 x 1
##   date
##   <date>
## 1 2015-03-05
```

Holt-Winters Model

The time series object for restaurant 12631 is created with a frequency of 7, corresponding to the cycle's length and starting on the 5th of March 2015. The plot of the time series object alongside that of the seasonal decomposition of the time series object allows us to better understand the fluctuations in the data.

```
restaurant_12631_ts <- ts(restaurant_12631$`Quantity (ounces)`,
                          frequency = 7, # 7-day cycle
                          start = c(03, 05)) # Starts on the 5th of March

# Plot of time-series
restaurant_12631_ts.plot1 <- autoplot(restaurant_12631_ts) +
  ggtitle('Daily Lettuce Quantity for \n Restaurant 12631') +
  xlab('Time') +
  ylab('Lettuce Quantity (Ounces')

# Plot of seasonal decomposition
restaurant_12631_ts.plot2 <- restaurant_12631_ts %>% stl(s.window = "period") %>% autoplot

grid.arrange(restaurant_12631_ts.plot1, restaurant_12631_ts.plot2, ncol = 2)
```





From the time series decomposition, there doesn't appear to be a trend (although one could also argue for a slight positive trend), and there seems to be a seasonality component that will need to be accounted for in the exponential smoothing model. First, however, we split the data between a training and a testing set according to an 80/20 ratio.

```
# Calculate the number of observations in the training set
n_train_12631 <- round(length(restaurant_12631_ts) * 0.8)

# Splitting into train and test
# Train set: 80%
restaurant_12631_ts.train <- subset(restaurant_12631_ts, end = n_train_12631)

# Test set: 20%
restaurant_12631_ts.test <- subset(restaurant_12631_ts, start = n_train_12631+1)
```

Then, we fit the Holt-Winters Model with ets() function. Looking back at the plot of the time series, we identify that there may be a non-linear relationship and thus that the error is multiplicative. Also, the amplitude of the seasonal cycles seem to be increasing, thus suggesting a multiplicative seasonality. Thus, in the ets() function, we pre-define the model to be 'MNM'. We can also create a 'MAM' model where we define an additive trend and evaluate its out-of sample performance. Finally, to ensure our analysis, we construct another model where we set 'ZZZ' such that the model defines the error, trend, and seasonality.

```
restaurant_12631.ets1 <- ets(restaurant_12631_ts.train, model = 'MNM')
restaurant_12631.ets2 <- ets(restaurant_12631_ts.train, model = 'MAM')
restaurant_12631.ets3 <- ets(restaurant_12631_ts.train, model = 'ZZZ')

restaurant_12631.ets3
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = restaurant_12631_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
```

Because the 'ZZZ' specification output is 'MNM', elements might have been missed. The resulting smoothing constants for the 'MNM' model are $\alpha = 0.1549$, $\beta = 0.0001$ and $\gamma = 0.0001$. The initial states given by the 'ZZZ' specification for the error and seasonal components are 240.4624 for the error and 1.0391, 0.9567, 0.9337, 1.0025, 0.8566, 1.1237, and 1.0877 for the seven seasonal components. The estimated standard deviation of the error term is equal to 0.1438.

We can also find the values given by the 'MAM' model.

```
restaurant_12631.ets2
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = restaurant_12631_ts.train, model = "MAM")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.9762
##
##   Initial states:
##     l = 215.1624
##     b = 1.9084
##     s = 1.0242 0.9686 0.9256 1.0231 0.8399 1.134
##            1.0847
##
##   sigma:  0.1381
##
##      AIC     AICc      BIC
## 960.2097 965.5627 991.4971
```

First, this model is defined as 'MAdM', thus implying multiplicative error and seasonality components, and a dampened trend. The resulting smoothing constants for the 'MAdM' model are $\alpha = 0.0001$, $\beta = 0.0001$, $\gamma = 0.0001$, and $\phi = 0.972$. The initial states given by this model's specification are 215.1624 for the error, 1.9084 for the trend, and 1.0242, 0.9686, 0.9256, 1.0231, 0.8399, 1.134, and 1.0847 for the seven seasonal components. The estimated standard deviation of the error term is equal to 0.1381.

We can conduct an out-of-sample evaluation for the models.

```
# Out-of-sample evaluation
restaurant_12631.ets1.f <- forecast(restaurant_12631.ets1, h = 14)
restaurant_12631.ets2.f <- forecast(restaurant_12631.ets2, h = 14)

accuracy(restaurant_12631.ets1.f, restaurant_12631_ts.test)
```

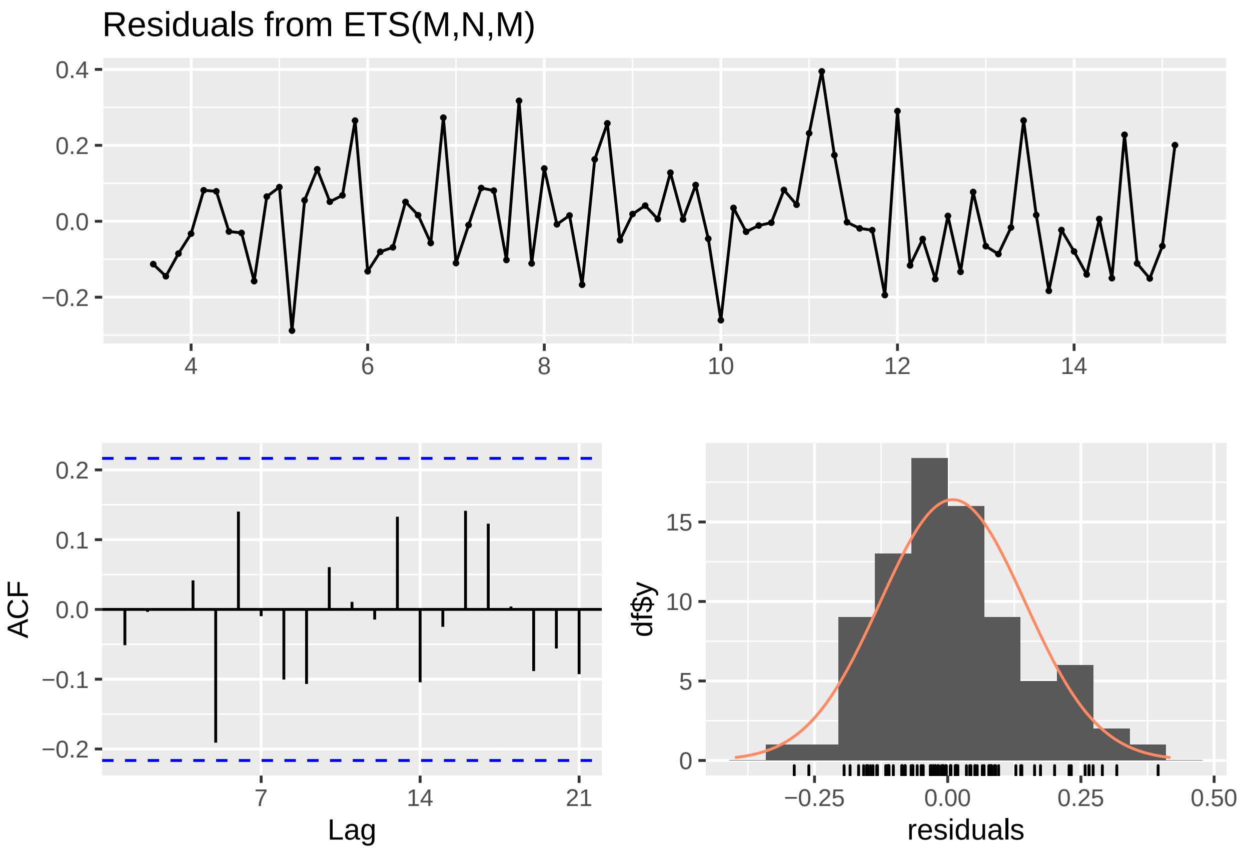
```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 2.025111 35.05182 26.63933 -0.8240822 10.28761 0.7491376
## Test set     3.693812 40.85532 33.47162 -0.6996635 12.28321 0.9412718
##                    ACF1 Theil's U
## Training set -0.0429736        NA
## Test set      0.4267731 0.8832205
```

```
accuracy(restaurant_12631.ets2.f, restaurant_12631_ts.test)
```

```
##                      ME     RMSE      MAE       MPE      MAPE      MASE
## Training set  -1.356915 33.63185 25.40108 -2.149226  9.881108 0.7143162
## Test set     -14.744351 44.64154 38.67174 -7.704227 15.023037 1.0875067
##                    ACF1 Theil's U
## Training set 0.02318681        NA
## Test set     0.38014412  1.020357
```

Based on their out-of-sample performance, the first model ('MNM') displays a better performance based on the RMSE, MAE, and MAPE metrics. We make sure the model is satisfactory by conducting a residuals analysis before re-calibrating with the entire sample.

```
# Analyse the residuals
checkresiduals(restaurant_12631.ets1.f)
```



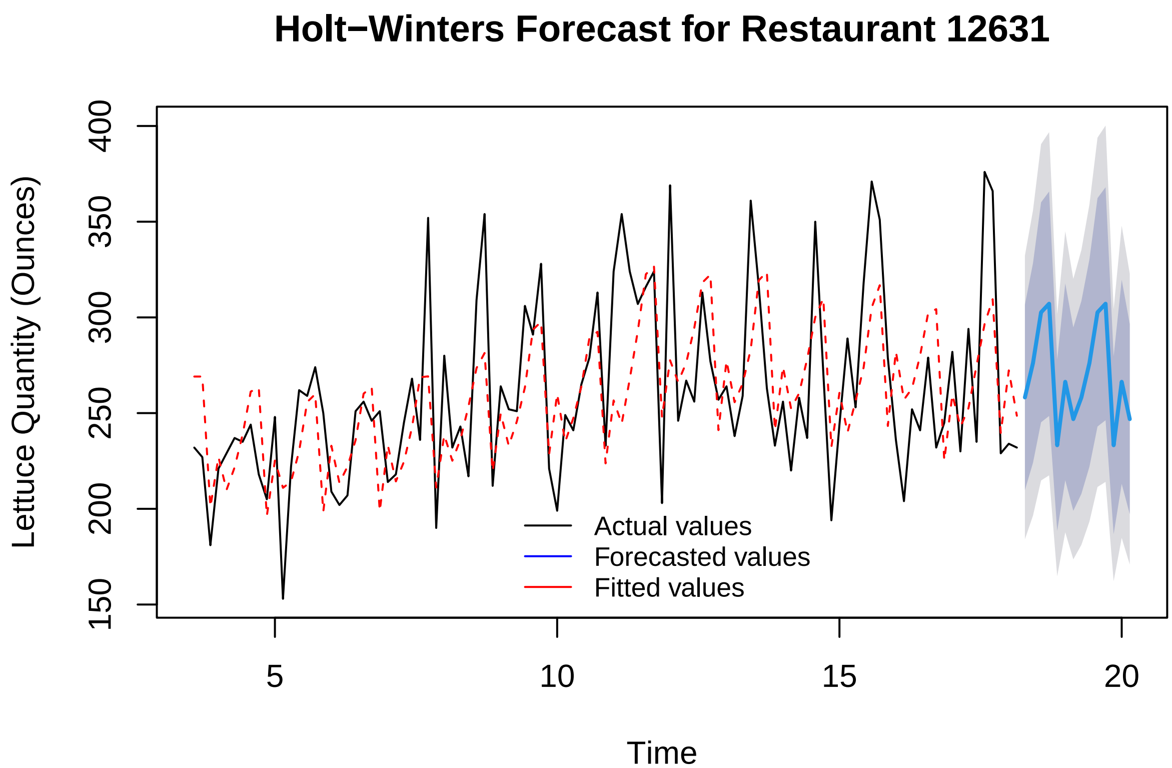

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,M)
## Q* = 10.696, df = 14, p-value = 0.7097
##
## Model df: 0.   Total lags used: 14
```

The Ljung-Box test has a p-value of 0.7097, superior to 0.05, suggesting that the model's residuals are likely to be independently and identically distributed with no significant autocorrelation. Furthermore, the errors appear to have a constant variance and are normally distributed with a mean of zero. Finally, there are no significant spikes in the ACF plot.

Therefore, the 'MNM' model is satisfactory for forecasting lettuce demand for restaurant 12631. Next, we re-calibrate it on the entire sample and forecast the quantity of lettuce demanded for restaurant 12631 for the upcoming 14 days. Plotting the output with a black line as the real data, a dashed red line as the fitted values, and a blue line as the forecasted values further allows us to visualise the model's performance.

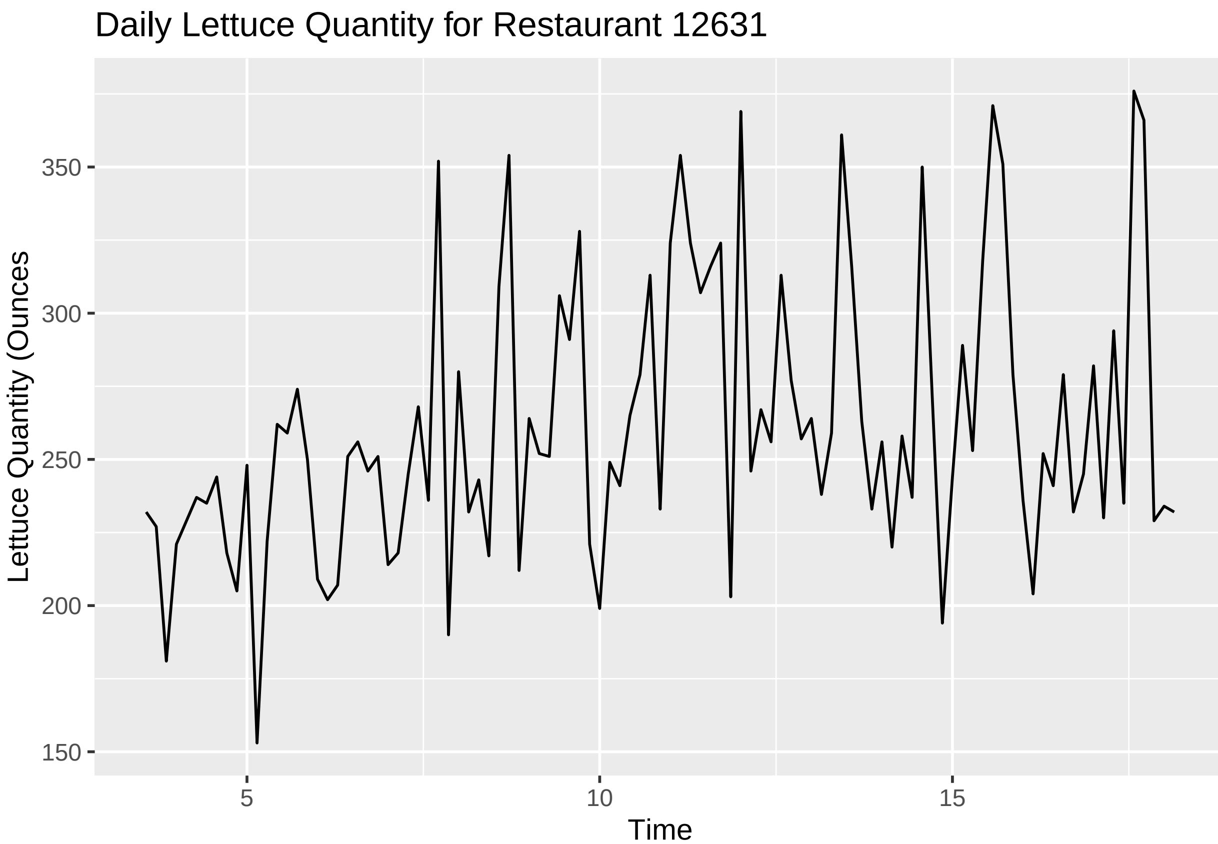
```
# Forecast
# Re-calibrate model with the entire sample
restaurant_12631.selected.model <- ets(restaurant_12631_ts, model = "MNM")
restaurant_12631.selected.model.f <- forecast(restaurant_12631.selected.model, h = 14)

# Summary plot of actual, fitted, and forecasted values
plot(restaurant_12631.selected.model.f, main = "Holt-Winters Forecast for Restaurant 12631",
     xlab="Time", ylab="Lettuce Quantity (Ounces)", lty = 1, col = "black")
lines(fitted(restaurant_12631.selected.model.f), lty = 2, col = "red")
```

```r
legend("bottom", legend=c("Actual values","Forecasted values", "Fitted values"),
       col=c("black", "blue", "red"), box.lty=0, lty=1, cex=0.8)
```



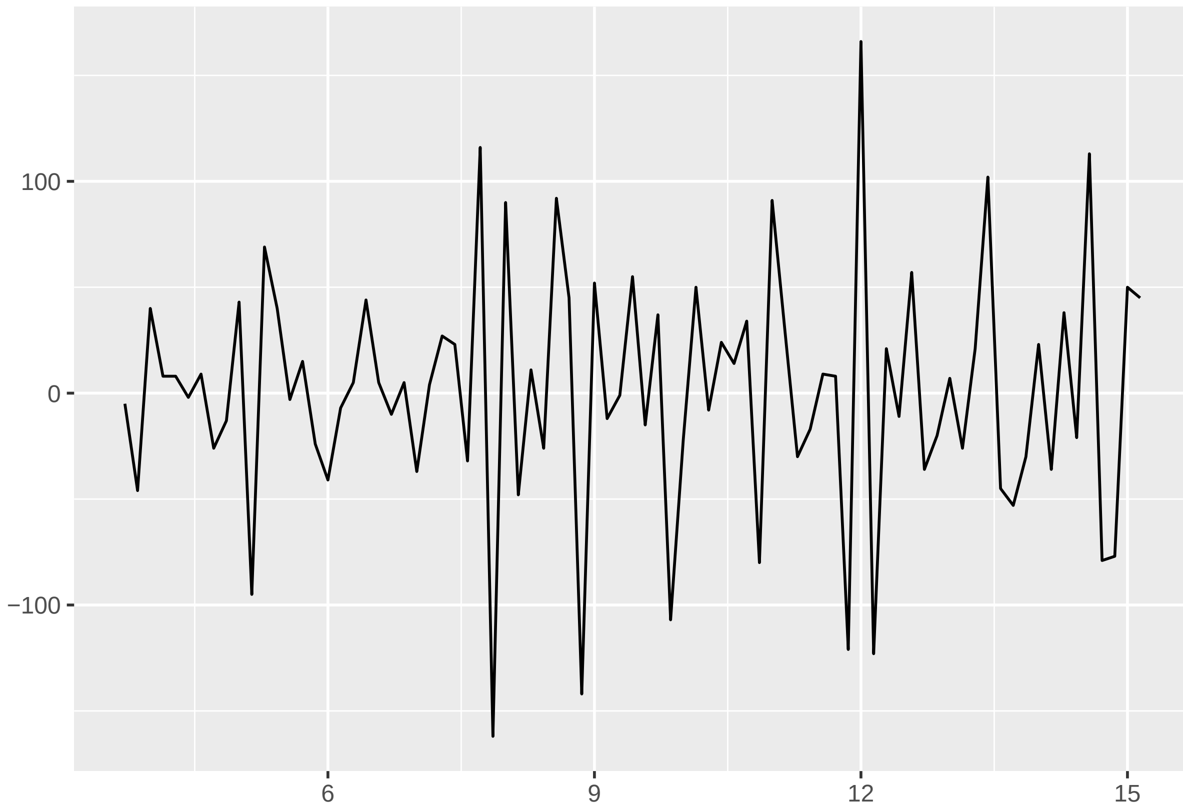
ARIMA Model

```r
# Plot of time-series
autoplot(restaurant_12631_ts) +
  ggtitle('Daily Lettuce Quantity for Restaurant 12631') +
  xlab('Time') +
  ylab('Lettuce Quantity (Ounces)')
```



From the above plot, the time series seems stationary regarding seasonality but not in terms of mean and variance. Therefore, we need to apply first-order differencing.

```
# Apply a first-order difference
restaurant_12631_ts.diff <- diff(restaurant_12631_ts.train, differences = 1)
autoplot(restaurant_12631_ts.diff)
```




```r
# Stationary test
adf.test(restaurant_12631_ts.diff) # Augmented Dickey-Fuller Test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  restaurant_12631_ts.diff
## Dickey-Fuller = -7.1694, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pp.test(restaurant_12631_ts.diff) # Phillips-Perron Unit Root Test
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  restaurant_12631_ts.diff
## Dickey-Fuller Z(alpha) = -104.26, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

```r
kpss.test(restaurant_12631_ts.diff) # KPSS Test for Level Stationarity
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  restaurant_12631_ts.diff
## KPSS Level = 0.024078, Truncation lag parameter = 3, p-value = 0.1
```

```r
ndiffs(restaurant_12631_ts.train)
```

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

```r
# Seasonal stationarity
nsdiffs(restaurant_12631_ts.train)
```

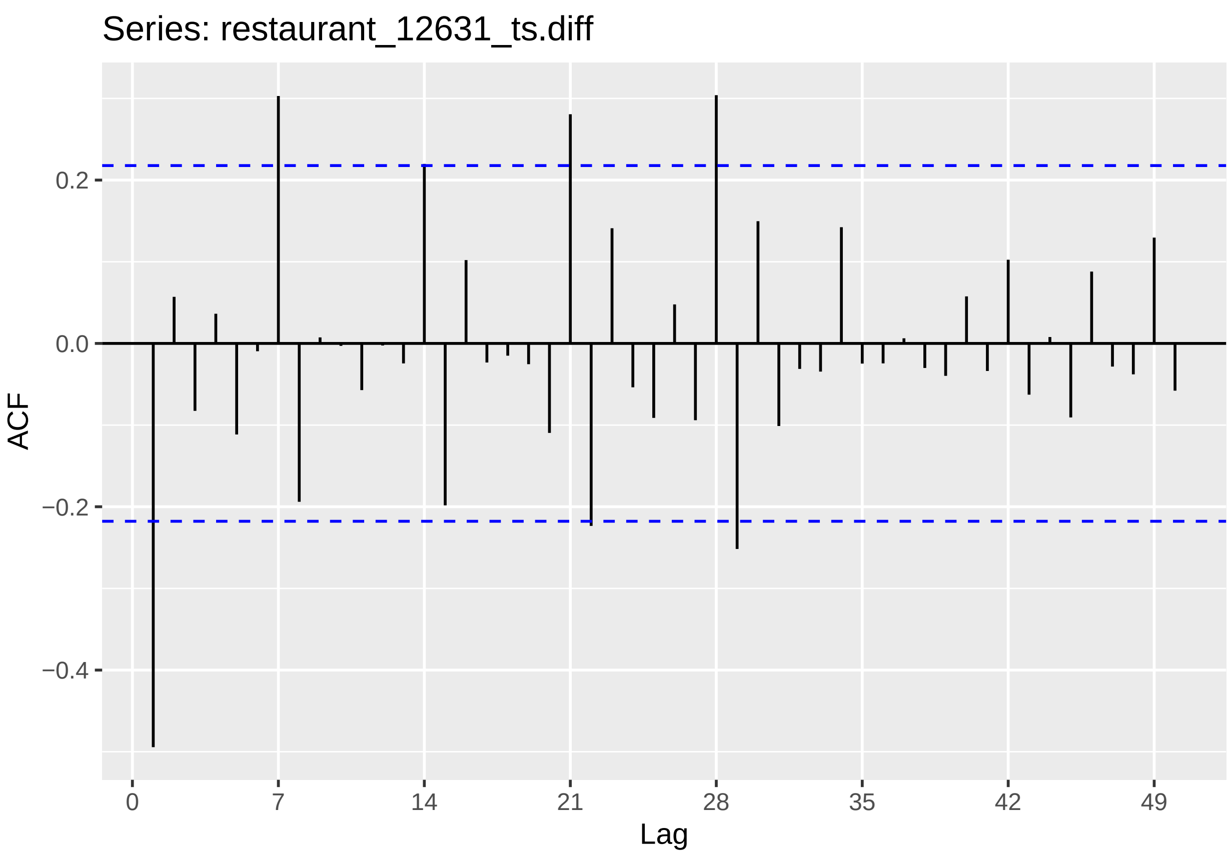
```
## [1] 0
```

From the above tests, we have the following conclusions:

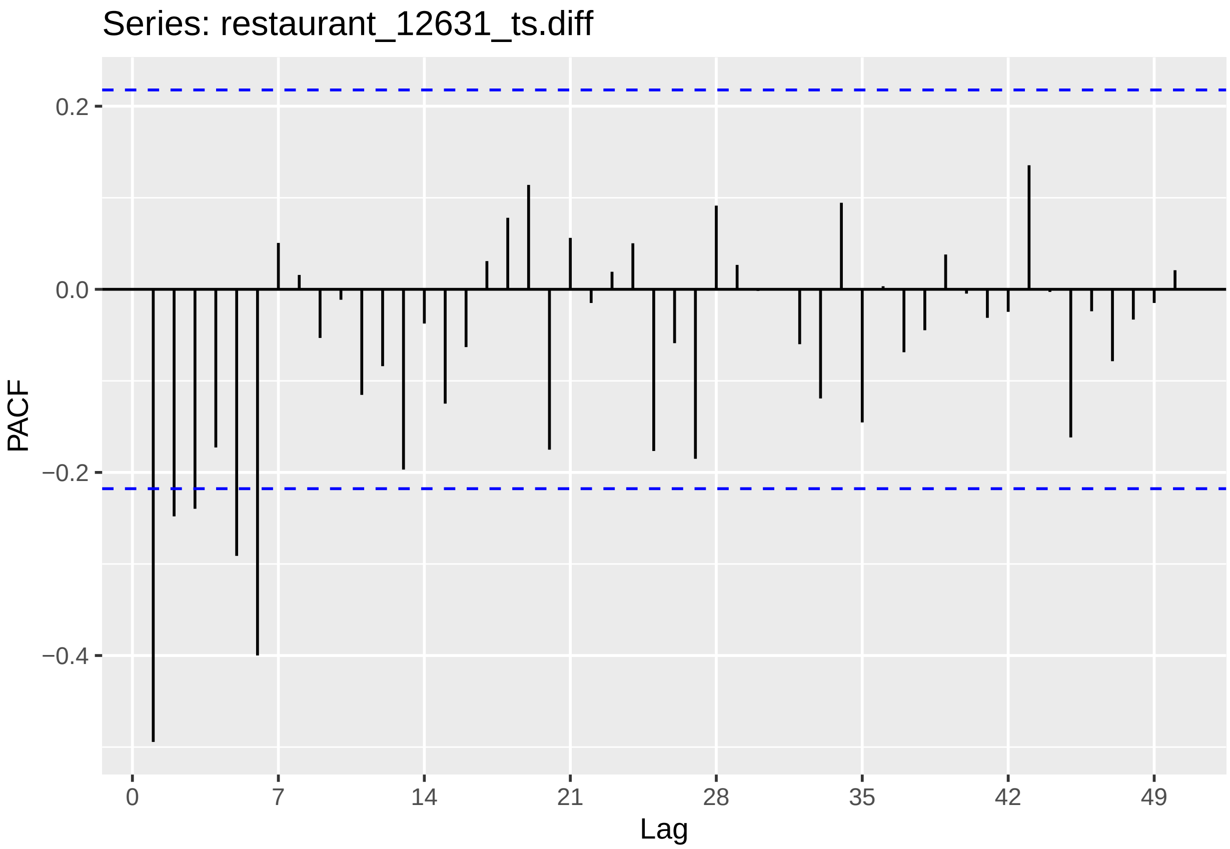
- ADF: sufficient evidence to reject the null that we have a unit root; thus, the time series is stationary.
- PP: sufficient evidence to reject the null; thus, the time series is stationary.
- KPSS: insufficient evidence to reject the null; thus, the time series is stationary.

From the above tests, we see that the time series with a first-order difference were proved to be stationary in terms of trend. Moreover, the output of 1 from the *ndiffs* function confirms that we needed to apply first-order differencing. We, therefore, expect an ARIMA model of the form ARIMA(p, d, q)(P, D, Q)[7] where d = 1 and D = 0.

```r
# Choice of p, q, P and Q
ggAcf(restaurant_12631_ts.diff, lag.max = 50)
```



```r
ggPacf(restaurant_12631_ts.diff, lag.max = 50)
```




```r
# Find the best ARIMA model
auto.arima(restaurant_12631_ts.train, trace = TRUE, ic = "aicc")
```

```
##
##  ARIMA(2,1,2)(1,0,1)[7] with drift         : Inf
##  ARIMA(0,1,0)           with drift         : 889.8583
##  ARIMA(1,1,0)(1,0,0)[7] with drift         : 858.615
##  ARIMA(0,1,1)(0,0,1)[7] with drift         : 840.4414
##  ARIMA(0,1,0)                              : 887.7673
##  ARIMA(0,1,1)           with drift         : 844.7933
##  ARIMA(0,1,1)(1,0,1)[7] with drift         : Inf
##  ARIMA(0,1,1)(0,0,2)[7] with drift         : Inf
##  ARIMA(0,1,1)(1,0,0)[7] with drift         : Inf
##  ARIMA(0,1,1)(1,0,2)[7] with drift         : Inf
##  ARIMA(0,1,0)(0,0,1)[7] with drift         : 885.5329
##  ARIMA(1,1,1)(0,0,1)[7] with drift         : Inf
##  ARIMA(0,1,2)(0,0,1)[7] with drift         : Inf
##  ARIMA(1,1,0)(0,0,1)[7] with drift         : 861.1074
##  ARIMA(1,1,2)(0,0,1)[7] with drift         : Inf
##  ARIMA(0,1,1)(0,0,1)[7]                    : 839.5062
##  ARIMA(0,1,1)                              : 843.9588
##  ARIMA(0,1,1)(1,0,1)[7]                    : Inf
##  ARIMA(0,1,1)(0,0,2)[7]                    : 839.2685
##  ARIMA(0,1,1)(1,0,2)[7]                    : Inf
```

```
##  ARIMA(0,1,0)(0,0,2)[7]                    : 884.5932
##  ARIMA(1,1,1)(0,0,2)[7]                    : 841.5416
##  ARIMA(0,1,2)(0,0,2)[7]                    : 841.5417
##  ARIMA(1,1,0)(0,0,2)[7]                    : 859.7941
##  ARIMA(1,1,2)(0,0,2)[7]                    : Inf
##
##  Best model: ARIMA(0,1,1)(0,0,2)[7]
```

```
## Series: restaurant_12631_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 ACF seems to decay in a sinusoidal manner, and by analysing the spikes of the above ACF plot, we see a spike at lag 1 and spikes at lags 7, 14, 21, 22, 28, 29 (although there is barely a spike at lag 14). Therefore, due to the spikes at the seasonal cycles (7, 14, 21, 28) we can define a seasonal MA component of 4 or 3 (Q = 4 or Q = 3) and a non-seasonal MA component of 1 (q = 1). Indeed, the spikes at lags 22 and 29 are likely due to whit noise. The PACF also decays in a sinusoidal manner, with 5 spikes at lags 1, 2, 3, 5, and 6. Because the spikes at lags 2 and 3 are not extremely significant, they may be due to white noise. Thus, we can define a non-seasonal AR component of either 5 or 3 and a seasonal AR component of 0 (p = 5 or p = 3, P = 0).

From these two plots, we can define the models: ARIMA(5,1,1)(0,0,4)[7], ARIMA(5,1,1)(0,0,3)[7], ARIMA(3,1,1)(0,0,4)[7] and ARIMA(3,1,1)(0,0,3)[7] to be tested against the two best models defined from the *auto.ARIMA()* function.

The best ARIMA model according to the *auto.ARIMA()* function is ARIMA(0,1,1)(2,0,1)[7], and the second best is ARIMA(0,1,1)(1,0,2)[7]. We train and forecast for both models to find the best one.

```
# Candidate models
restaurant_12631.arima1 <- Arima(restaurant_12631_ts.train, order = c(0, 1, 1),
                                 seasonal = list(order = c(0, 0, 2), period = 7),
                                 include.drift = TRUE)
restaurant_12631.arima2 <- Arima(restaurant_12631_ts.train, order = c(1, 1, 1),
                                 seasonal = list(order = c(0, 0, 2), period = 7),
                                 include.drift = TRUE)
restaurant_12631.arima3 <- Arima(restaurant_12631_ts.train, order = c(5, 1, 1),
                                 seasonal = list(order = c(0, 0, 4), period = 7),
                                 include.drift = FALSE)
restaurant_12631.arima4 <- Arima(restaurant_12631_ts.train, order = c(5, 1, 1),
                                 seasonal = list(order = c(0, 0, 3), period = 7),
                                 include.drift = FALSE)
restaurant_12631.arima5 <- Arima(restaurant_12631_ts.train, order = c(3, 1, 1),
                                 seasonal = list(order = c(0, 0, 4), period = 7),
                                 include.drift = FALSE)
restaurant_12631.arima6 <- Arima(restaurant_12631_ts.train, order = c(3, 1, 1),
                                 seasonal = list(order = c(0, 0, 3), period = 7),
                                 include.drift = FALSE)
```

```r
# Model evaluation
# Forecast
restaurant_12631.arima1.f <- forecast(restaurant_12631.arima1, h = 14)
restaurant_12631.arima2.f <- forecast(restaurant_12631.arima2, h = 14)
restaurant_12631.arima3.f <- forecast(restaurant_12631.arima3, h = 14)
restaurant_12631.arima4.f <- forecast(restaurant_12631.arima4, h = 14)
restaurant_12631.arima5.f <- forecast(restaurant_12631.arima5, h = 14)
restaurant_12631.arima6.f <- forecast(restaurant_12631.arima6, h = 14)

# Out-of-sample performance
accuracy(restaurant_12631.arima1.f, restaurant_12631_ts.test)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set   1.85433 39.06280 30.16930 -1.409787 11.68378 0.8484054
## Test set     -17.46160 48.99997 44.11929 -9.293899 17.11575 1.2407000
##                    ACF1 Theil's U
## Training set 0.02996393        NA
## Test set     0.57957151  1.139853
```

```r
accuracy(restaurant_12631.arima2.f, restaurant_12631_ts.test)
```

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set   1.760367 39.05215 30.03743 -1.450999 11.65125 0.8446971
## Test set     -17.685890 49.14888 44.30935 -9.385516 17.19330 1.2460448
##                     ACF1 Theil's U
## Training set -0.01667599        NA
## Test set      0.57852051  1.143437
```

```r
accuracy(restaurant_12631.arima3.f, restaurant_12631_ts.test)
```

```
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.004243317 36.03610 26.67804 -1.782955 10.49959 0.7502261
## Test set     24.437384560 44.45003 31.32104  7.546477 10.54244 0.8807942
##                      ACF1 Theil's U
## Training set -0.007716865        NA
## Test set      0.390175071 0.9024513
```

```r
accuracy(restaurant_12631.arima4.f, restaurant_12631_ts.test)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  2.17278 39.09825 30.03798 -1.196069 11.68278 0.8447127
## Test set     13.58500 42.73521 32.21687  2.872195 11.21903 0.9059863
##                     ACF1 Theil's U
## Training set 0.003618436        NA
## Test set     0.545357400 0.8873399
```

```r
accuracy(restaurant_12631.arima5.f, restaurant_12631_ts.test)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  2.589832 35.54697 26.47255 -0.7212281 10.28662 0.7444474
```

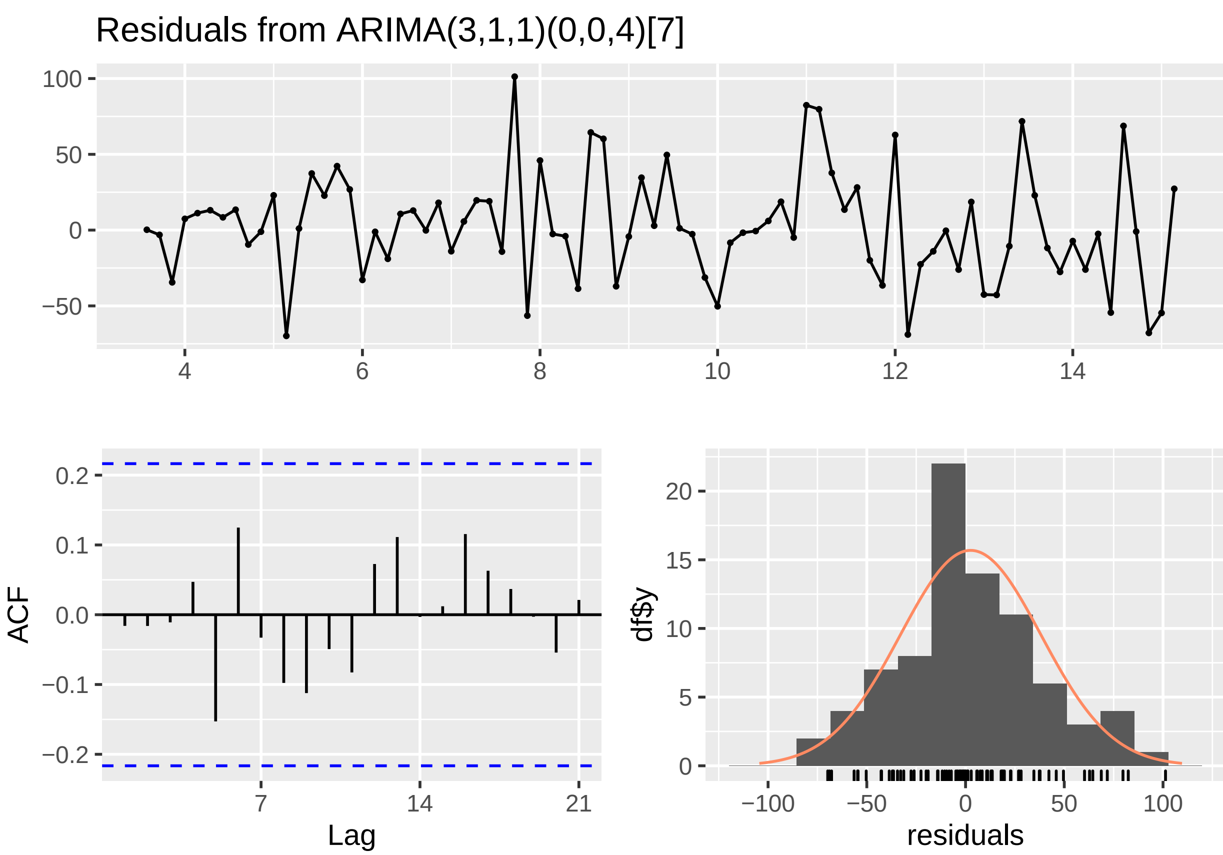
```
## Test set     13.197883 39.74545 30.72207  3.1478522 10.64084 0.8639503
##                     ACF1 Theil's U
## Training set -0.01596466        NA
## Test set      0.34994389 0.8049121
```

```
accuracy(restaurant_12631.arima6.f, restaurant_12631_ts.test)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 3.270944 39.49360 30.72868 -0.8215885 11.92992 0.8641362
## Test set     7.719948 42.64536 32.86343  0.5360647 11.66347 0.9241686
##                     ACF1 Theil's U
## Training set -0.01618415        NA
## Test set      0.52310467 0.8965777
```

From the above output, if we focus on the performance of the models on the test set and the RMSE, MAE, and MAPE metrics, model 5 appears to achieve the best values. We therefore choose ARIMA(3, 1, 1),(0, 0, 4)[7] as our best model. We are most likely to use this to re-calibrate with the entire sample if it is satisfactory when analysing the residuals.

```
# Analyse the residuals
checkresiduals(restaurant_12631.arima5.f)
```



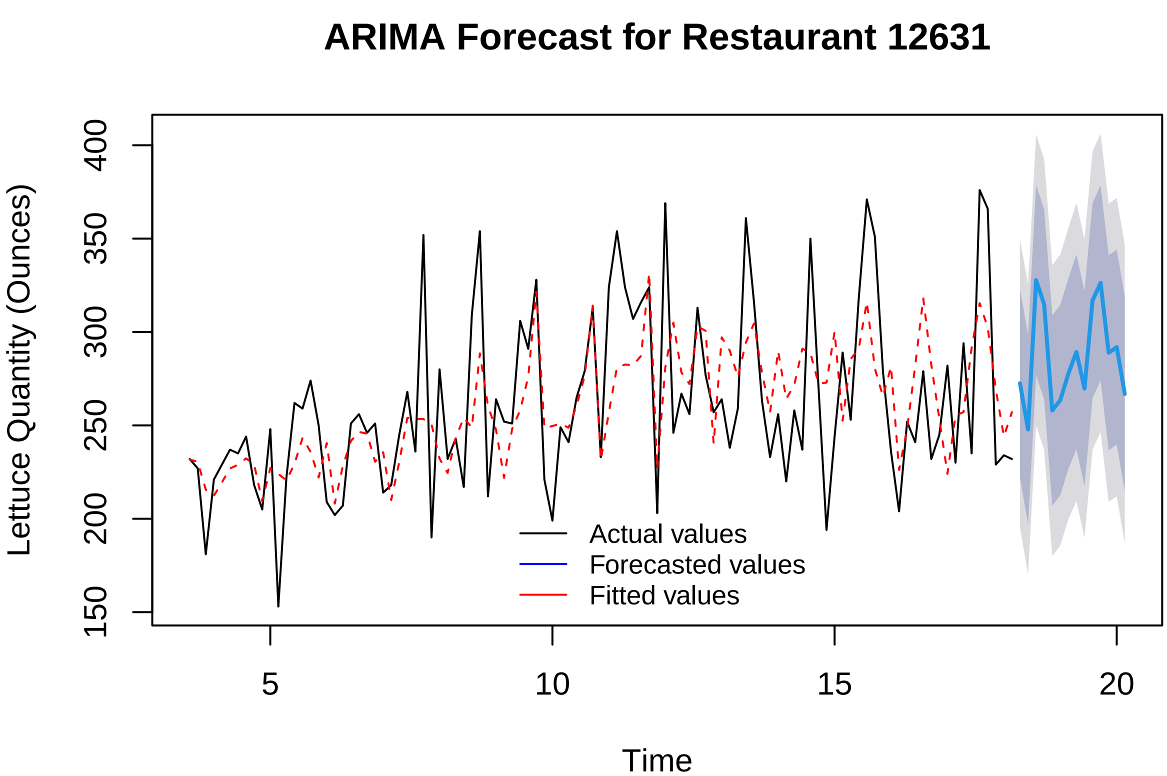

```
##
##  Ljung-Box test
##
```

```
## data:  Residuals from ARIMA(3,1,1)(0,0,4)[7]
## Q* = 8.582, df = 6, p-value = 0.1985
##
## Model df: 8.   Total lags used: 14
```

The Box-Ljung tests suggest that the fifth ARIMA model's residuals are likely to be independently and identically distributed with constant variance and normally distributed with a mean of zero. It is also satisfactory when looking at the ACF plot.

Therefore, we conclude that ARIMA(3,1,1)(0,0,4)[7] is satisfactory for forecasting lettuce demand for restaurant 12631 and is thus used to re-calibrate with the entire sample.

```r
# Forecast
# Re-calibrate model with the entire sample
# Best model forecast (train on the whole dataset)
restaurant_12631.arima <- Arima(restaurant_12631_ts, order = c(3, 1, 1),
                                seasonal = list(order = c(0, 0, 4), period = 7),
                                include.drift = TRUE)
restaurant_12631.arima.f <- forecast(restaurant_12631.arima, h = 14)

# Summary plot of actual, fitted, and forecasted values
plot(restaurant_12631.arima.f, main = "ARIMA Forecast for Restaurant 12631",
     xlab="Time", ylab="Lettuce Quantity (Ounces)", lty = 1, col = "black")
lines(fitted(restaurant_12631.arima.f), lty = 2, col = "red")
legend("bottom", legend=c("Actual values","Forecasted values", "Fitted values"),
       col=c("black", "blue", "red"), box.lty=0, lty=1, cex=0.8)
```



Final Forecast

Finally, because both the Holt-Winters and ARIMA models appear to be satisfactory, we compare the forecasting error of each sample on new unseen data (the test set) to choose the best one.

```
# Forecasting error
accuracy(restaurant_12631.arima5.f, restaurant_12631_ts.test)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  2.589832 35.54697 26.47255 -0.7212281 10.28662 0.7444474
## Test set     13.197883 39.74545 30.72207  3.1478522 10.64084 0.8639503
##                     ACF1 Theil's U
## Training set -0.01596466        NA
## Test set      0.34994389 0.8049121
```

```
accuracy(restaurant_12631.ets1.f, restaurant_12631_ts.test)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 2.025111 35.05182 26.63933 -0.8240822 10.28761 0.7491376
## Test set     3.693812 40.85532 33.47162 -0.6996635 12.28321 0.9412718
##                    ACF1 Theil's U
## Training set -0.0429736        NA
## Test set      0.4267731 0.8832205
```

The ARIMA model performs better in terms of RMSE, MAE, and MAPE, although there isn't a major difference. The ME of the ARIMA model is a lot higher, indicating that it is less precise than the ETS model. We therefore choose the ETS model over the ARIMA model and re-calibrate with the entire sample.

```
# Re-calibrate model with the entire sample
restaurant_12631.model <- ets(restaurant_12631_ts, model = "MNM")
restaurant_12631.model.f <- forecast(restaurant_12631.model, h = 14)

# Convert to a data frame
restaurant_12631.model.f.df <- as.data.frame(restaurant_12631.model.f)
rownames(restaurant_12631.model.f.df) <- c('2015-06-16', '2015-06-17', '2015-06-18',
                                           '2015-06-19', '2015-06-20', '2015-06-21',
                                           '2015-06-22', '2015-06-23', '2015-06-24',
                                           '2015-06-25', '2015-06-26', '2015-06-27',
                                           '2015-06-28', '2015-06-29')
restaurant_12631.model.f.df
```

```
##            Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015-06-16       258.1965 209.7455 306.6475 184.0971 332.2959
## 2015-06-17       276.1955 224.0706 328.3205 196.4773 355.9138
## 2015-06-18       302.6475 245.2074 360.0877 214.8004 390.4947
## 2015-06-19       307.1361 248.5179 365.7542 217.4873 396.7848
## 2015-06-20       233.2254 188.4671 277.9838 164.7735 301.6774
## 2015-06-21       266.3445 214.9505 317.7386 187.7441 344.9449
## 2015-06-22       246.8641 198.9710 294.7571 173.6179 320.1102
## 2015-06-23       258.1966 207.8360 308.5572 181.1766 335.2165
## 2015-06-24       276.1956 222.0386 330.3526 193.3696 359.0216
## 2015-06-25       302.6476 242.9923 362.3029 211.4127 393.8826
```

```
## 2015-06-26       307.1361 246.2815 367.9908 214.0670 400.2053
## 2015-06-27       233.2255 186.7775 279.6735 162.1894 304.2616
## 2015-06-28       266.3446 213.0306 319.6586 184.8079 347.8813
## 2015-06-29       246.8641 197.2003 296.5279 170.9099 322.8183
```

New York 2 (ID:20974)

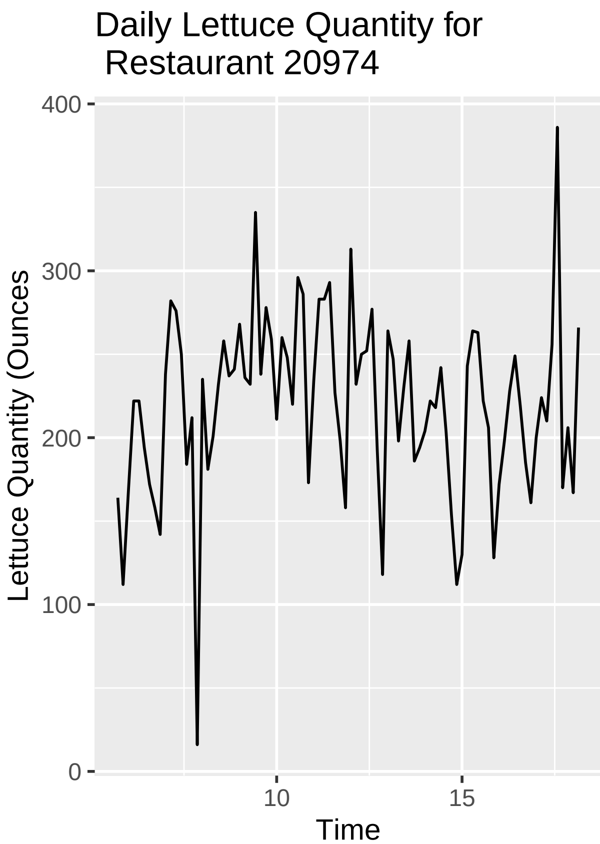
```
restaurant_20974 <- read_csv('restaurant_20974.csv')
restaurant_20974[1, 2] # Find the start date
```

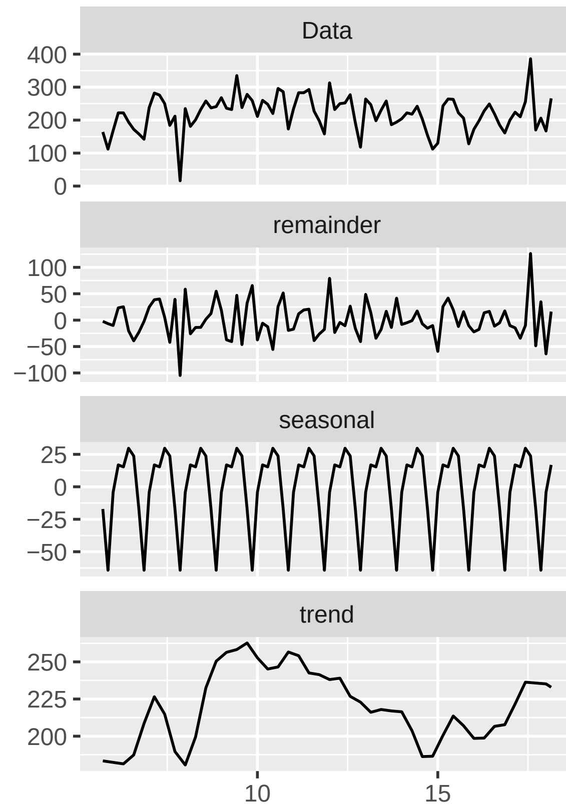
```
## # A tibble: 1 x 1
##   date
##   <date>
## 1 2015-03-20
```

Holt-Winters Model

The time series object for restaurant 20974 is again created with a frequency of 7, corresponding to the cycle's length and starting on the 20th of March 2015. The plot of the time series object alongside that of the seasonal decomposition of the time series object allows us to better understand the fluctuations in the data.

```
restaurant_20974_ts <- ts(restaurant_20974$`Quantity (ounces)`,
                          frequency = 7, # 7-day cycle
                          start = c(03, 20)) # Starts on the 20th of March

# Plot of time-series
restaurant_20974_ts.plot1 <- autoplot(restaurant_20974_ts) +
  ggtitle('Daily Lettuce Quantity for \n Restaurant 20974') +
  xlab('Time') +
  ylab('Lettuce Quantity (Ounces')

# Plot of seasonal decomposition
restaurant_20974_ts.plot2 <- restaurant_20974_ts %>% stl(s.window = "period") %>% autoplot

grid.arrange(restaurant_20974_ts.plot1, restaurant_20974_ts.plot2, ncol = 2)
```





There doesn't appear to be a trend from the time series decomposition, and as such the trend component can be ignored. Still, there appears to be an additive seasonality component that will need to be accounted for in the exponential smoothing model.

First, we split the data between a training and a testing set according to an 80/20 ratio.

```
# Calculate the number of observations in the training set
n_train_20974 <- round(length(restaurant_20974_ts) * 0.8)

# Splitting into train and test
# Train set: 80%
restaurant_20974_ts.train <- subset(restaurant_20974_ts, end = n_train_20974)

# Test set: 20%
restaurant_20974_ts.test <- subset(restaurant_20974_ts, start = n_train_20974+1)
```

Then, we fit the Holt-Winters Model with the *ets()* function. As we mentioned from the seasonal decomposition, the seasonal component appears to be additive, and there doesn't appear to be a trend. Thus, in the *ets()* function, we pre-define the model to be 'ANA'. Then, as we did with all other restaurants, to ensure our analysis, we construct another model where we set 'ZZZ' such that the function defines the error, trend, and seasonality.

```
restaurant_20974.ets1 <- ets(restaurant_20974_ts.train, model = 'ANA')
restaurant_20974.ets2 <- ets(restaurant_20974_ts.train, model = 'ZZZ')

restaurant_20974.ets2
```

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

```
##
## Call:
##  ets(y = restaurant_20974_ts.train, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.1895
##     gamma = 1e-04
##
##   Initial states:
##     l = 215.2573
##     s = 13.4722 30.6874 15.4489 18.7483 6.1767 -69.3617
##            -15.1719
##
##   sigma:  44.4909
##
##      AIC     AICc      BIC
## 839.1011 842.8299 861.5861
```

Because the 'ZZZ' specification output is 'ANA', we know that the error and seasonal components are additive. The resulting smoothing constants are $\alpha = 0.1895$, $\beta = 0$ and $\gamma = 0.0001$; and the initial states are 215.2573 for the level and 13.4722, 30.6874, 15.4489, 18.7483, 6.1767, -69.3617, and -15.1719 for the seven seasonal components. The estimated standard deviation of the error term is equal to 44.4909.

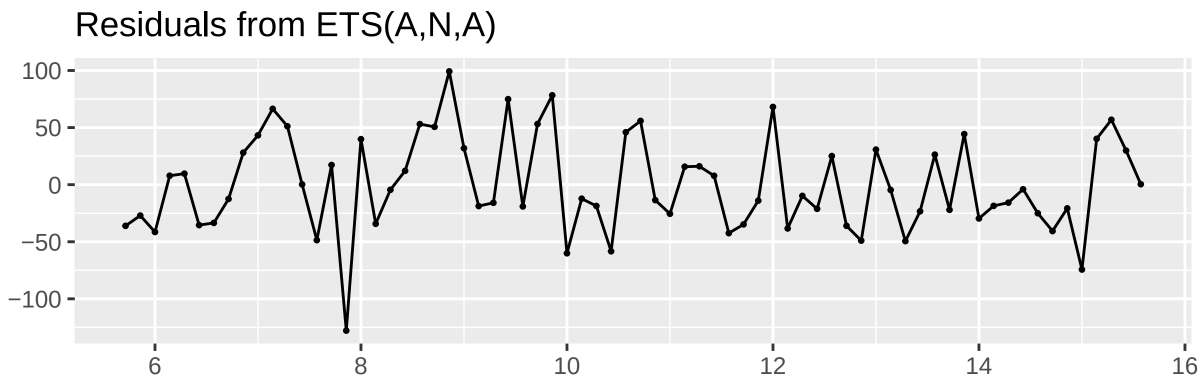
Then, we can conduct an out-of-sample evaluation for the model.

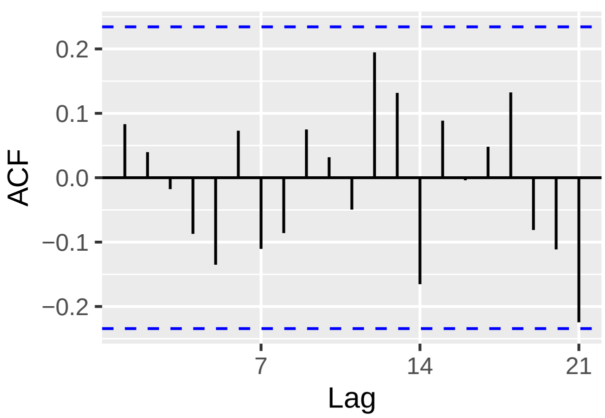
```
# Out-of-sample evaluation
restaurant_20974.ets1.f <- forecast(restaurant_20974.ets1, h = 14)

# Forecasting error
accuracy(restaurant_20974.ets1.f, restaurant_20974_ts.test)
```

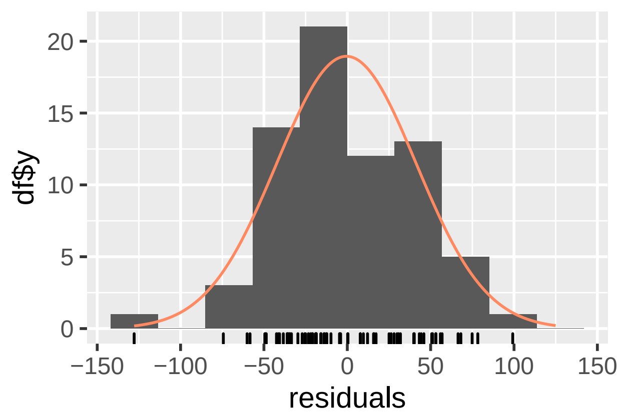
```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.5314202 41.53243 34.24962 -13.4603626 26.57567 0.7226142
## Test set      7.6512299 47.28392 25.34920   0.4970002 10.25627 0.5348290
##                    ACF1 Theil's U
## Training set 0.08314736        NA
## Test set     0.13453932 0.8841927
```

If the residuals analysis is satisfactory, no further improvements need to be made, and we can re-calibrate on the entire sample.

```
# Analyse the residuals
checkresiduals(restaurant_20974.ets1.f)
```





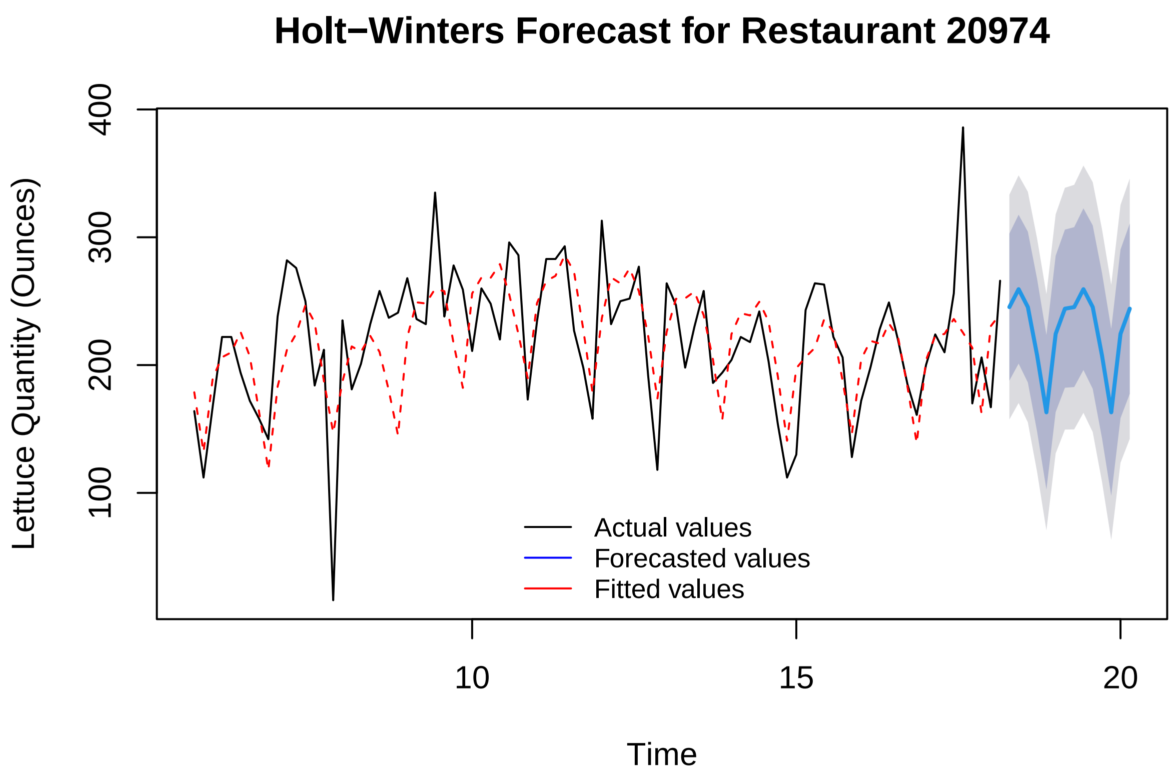



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,A)
## Q* = 12.682, df = 14, p-value = 0.5517
##
## Model df: 0.   Total lags used: 14
```

The Ljung-Box test suggests that the ARIMA model's residuals will likely be independently and identically distributed. Furthermore, the errors have constant variance and are normally distributed with a mean of zero, and there are no spikes in the ACF plot.

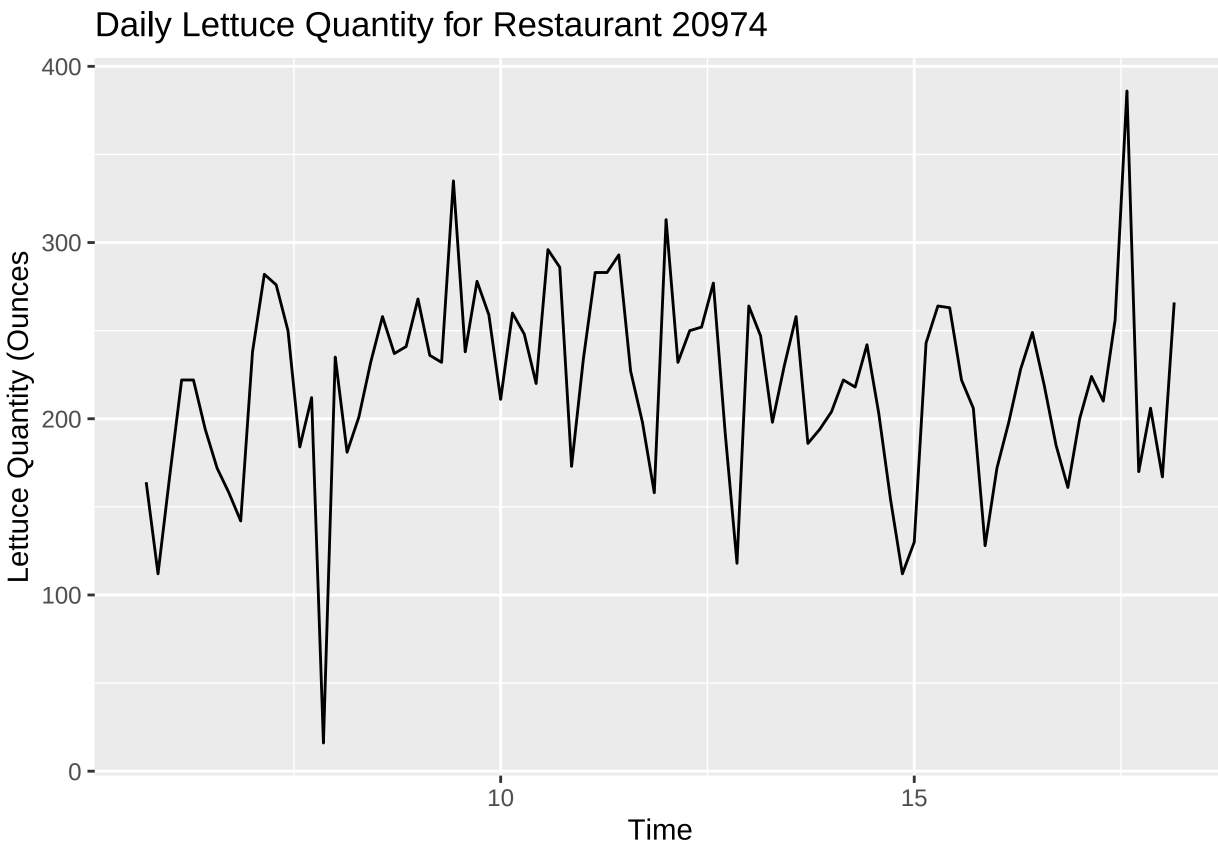
Therefore, this model is satisfactory for forecasting lettuce demand for restaurant 20974 and can be re-calibrated with the entire sample.

```
# Forecast
# Re-calibrate model with the entire sample
restaurant_20974.selected.model <- ets(restaurant_20974_ts, model = "ANA")
restaurant_20974.selected.model.f <- forecast(restaurant_20974.selected.model, h = 14)

# Summary plot of actual, fitted, and forecasted values
plot(restaurant_20974.selected.model.f, main = "Holt-Winters Forecast for Restaurant 20974",
     xlab="Time", ylab="Lettuce Quantity (Ounces)", lty = 1, col = "black")
lines(fitted(restaurant_20974.selected.model.f), lty = 2, col = "red")
legend("bottom", legend=c("Actual values","Forecasted values", "Fitted values"),
       col=c("black", "blue", "red"), box.lty=0, lty=1, cex=0.8)
```



ARIMA Model

```
# Plot of time-series
autoplot(restaurant_20974_ts) +
  ggtitle('Daily Lettuce Quantity for Restaurant 20974') +
  xlab('Time') +
  ylab('Lettuce Quantity (Ounces')
```



The above plot shows that the time series seems stationary in terms of mean and variance, which can be verified with several tests.

```
# Stationary test
adf.test(restaurant_20974_ts) # Augmented Dickey-Fuller Test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  restaurant_20974_ts
## Dickey-Fuller = -4.2042, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
pp.test(restaurant_20974_ts) # Phillips-Perron Unit Root Test
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  restaurant_20974_ts
## Dickey-Fuller Z(alpha) = -61.65, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(restaurant_20974_ts) # KPSS Test for Level Stationarity
```

```
##
```

```
##  KPSS Test for Level Stationarity
##
## data:  restaurant_20974_ts
## KPSS Level = 0.18726, Truncation lag parameter = 3, p-value = 0.1
```

```
ndiffs(restaurant_20974_ts.train)
```

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

```
# Seasonal stationarity
nsdiffs(restaurant_20974_ts.train)
```

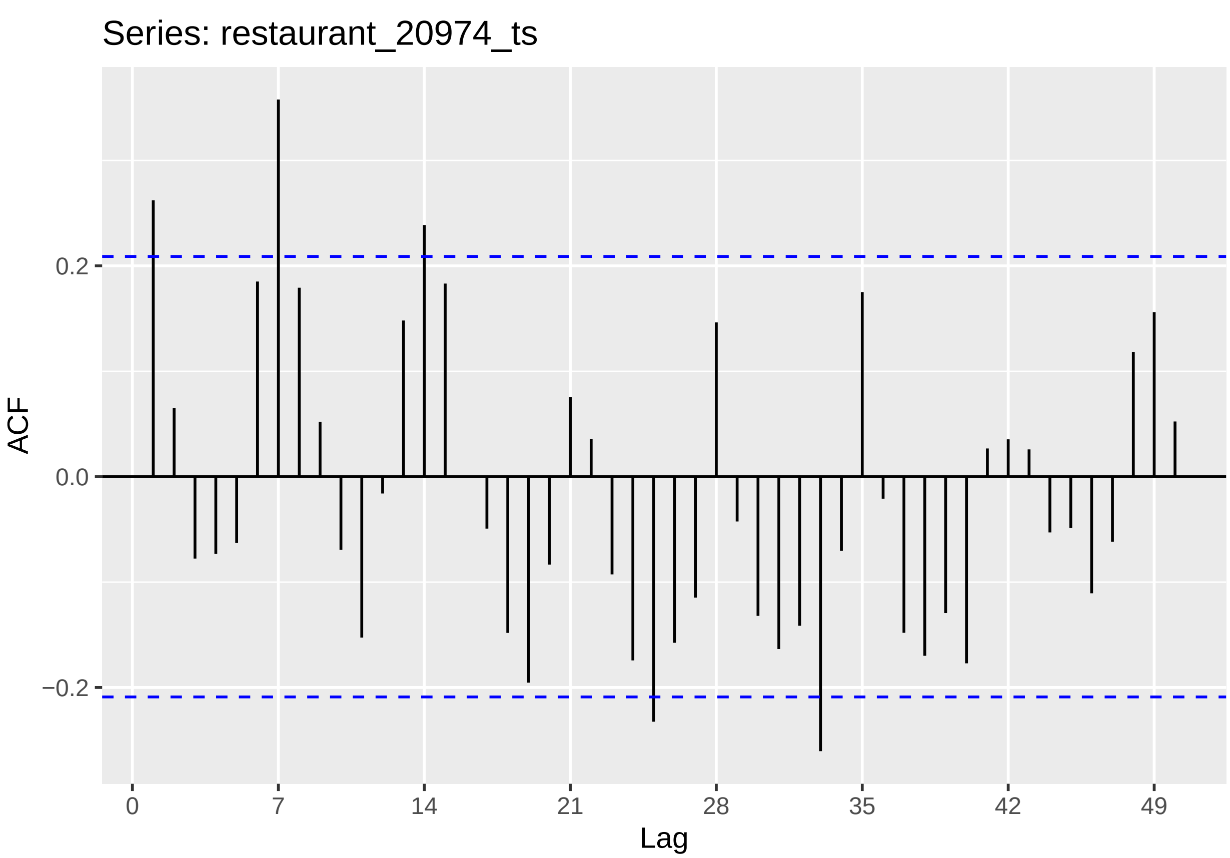
```
## [1] 0
```

From the above tests, we have the following conclusions:

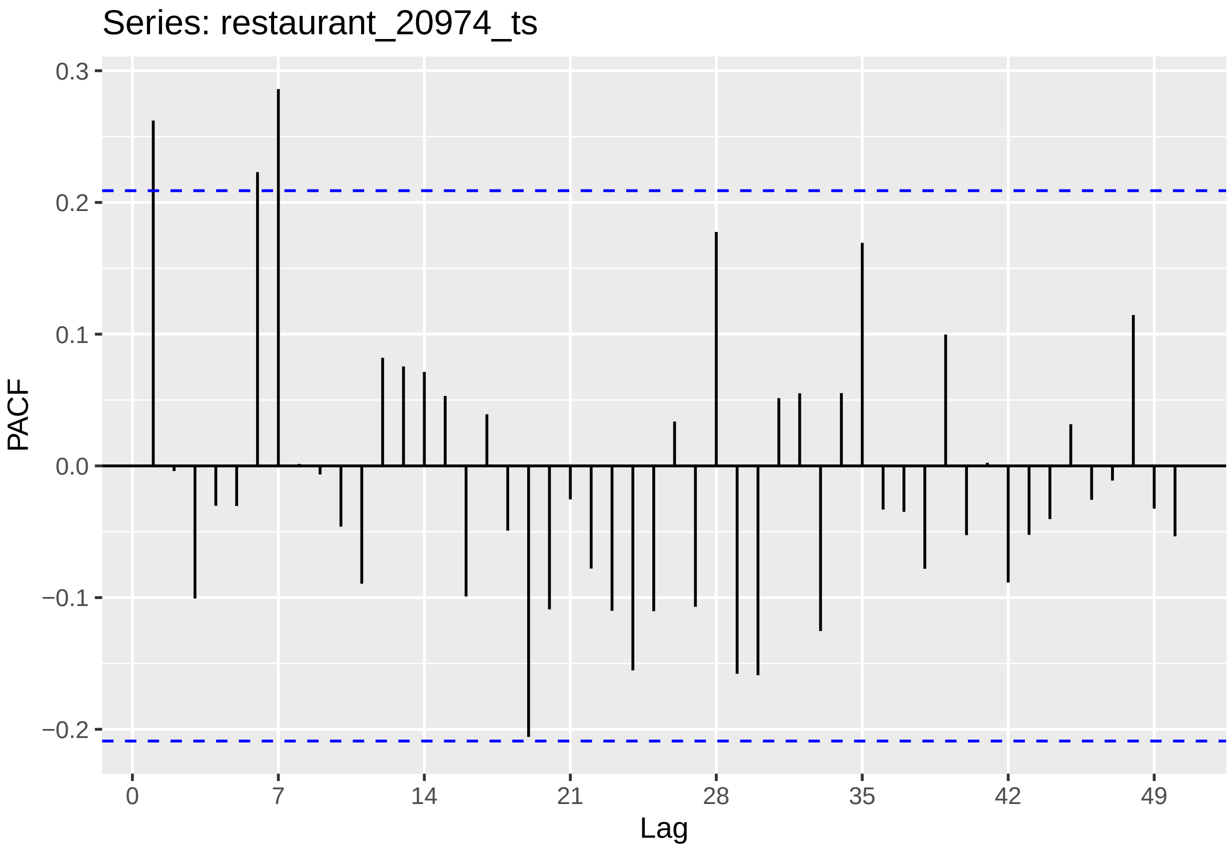
- ADF: sufficient evidence to reject the null that we have a unit root; thus, the time series is stationary.
- PP: sufficient evidence to reject the null; thus, the time series is stationary.
- KPSS: insufficient evidence to reject the null; thus, the time series is stationary.

From the above tests, we see that the time series was proved to be stationary in terms of trend. Furthermore, the time series is stationary in terms of seasonality. We, therefore, expect an ARIMA model of the form ARIMA(p, d, q)(P, D, Q)[7] where d = 0 and D = 0.

```
# Choice of p, q, P and Q
ggAcf(restaurant_20974_ts, lag.max = 50)
```



```r
ggPacf(restaurant_20974_ts, lag.max = 50)
```




```r
# Find the best ARIMA model
auto.arima(restaurant_20974_ts.train, trace = TRUE, ic = "aicc")
```

```
##
##  ARIMA(2,0,2)(1,0,1)[7] with non-zero mean : Inf
##  ARIMA(0,0,0)           with non-zero mean : 760.1464
##  ARIMA(1,0,0)(1,0,0)[7] with non-zero mean : 751.0966
##  ARIMA(0,0,1)(0,0,1)[7] with non-zero mean : 753.6909
##  ARIMA(0,0,0)           with zero mean     : 960.7132
##  ARIMA(1,0,0)           with non-zero mean : 755.6719
##  ARIMA(1,0,0)(2,0,0)[7] with non-zero mean : 752.4866
##  ARIMA(1,0,0)(1,0,1)[7] with non-zero mean : Inf
##  ARIMA(1,0,0)(0,0,1)[7] with non-zero mean : 752.8448
##  ARIMA(1,0,0)(2,0,1)[7] with non-zero mean : Inf
##  ARIMA(0,0,0)(1,0,0)[7] with non-zero mean : 753.5912
##  ARIMA(2,0,0)(1,0,0)[7] with non-zero mean : 753.158
##  ARIMA(1,0,1)(1,0,0)[7] with non-zero mean : 753.2154
##  ARIMA(0,0,1)(1,0,0)[7] with non-zero mean : 751.8502
##  ARIMA(2,0,1)(1,0,0)[7] with non-zero mean : 754.3766
##  ARIMA(1,0,0)(1,0,0)[7] with zero mean     : 788.0655
##
##  Best model: ARIMA(1,0,0)(1,0,0)[7] with non-zero mean
```

```
## Series: restaurant_20974_ts.train
```

```
## ARIMA(1,0,0)(1,0,0)[7] with non-zero mean
##
## Coefficients:
##          ar1    sar1      mean
##       0.2595  0.3230  218.4136
## s.e.  0.1167  0.1195   11.0479
##
## sigma^2 = 2443:  log likelihood = -371.24
## AIC=750.48   AICc=751.1   BIC=759.48
```

The above ACF plot displays a sinusoidal and not an exponential decay. Furthermore, by analysing the spikes, the ACF plot shows spikes at lags 1, 7, and 14. Therefore, we can define a seasonal MA component of 2 (Q = 2) and a non-seasonal MA component of 1 (q = 1). Because the spike at lag 14 is not extremely significant, we can also try a model with a seasonal MA component of 1 (Q = 1). The PACF plot also decays sinusodially, and we see spikes at lags of 1, 6, and 7. Thus, it may be necessary to integrate a seasonal AR component of 1 (P = 1). The spike at lag 6 is likely due to white noise, so we choose not to define a corresponding model and only take a non-seasonal AR component of 1 (p = 1).

From these two plots, we can define the models ARIMA(1,0,1)(1,0,2)[7] and ARIMA(1,0,1)(1,0,1)[7] to be tested against the two best models defined from the *auto.ARIMA()* function.

The best ARIMA model according to the *auto.ARIMA()* function is ARIMA(1,0,0)(1,0,0)[7], and the second best is ARIMA(0,0,1)(1,0,0)[7]. We train and forecast for both models to find the best one.

```r
# Candidate models
restaurant_20974.arima1 <- Arima(restaurant_20974_ts.train, order = c(1, 0, 0),
                                 seasonal = list(order = c(1, 0, 0), period = 7),
                                 include.drift = FALSE)
restaurant_20974.arima2 <- Arima(restaurant_20974_ts.train, order = c(0, 0, 1),
                                 seasonal = list(order = c(1, 0, 0), period = 7),
                                 include.drift = TRUE)
restaurant_20974.arima3 <- Arima(restaurant_20974_ts.train, order = c(1, 0, 1),
                                 seasonal = list(order = c(1, 0, 2), period = 7),
                                 include.drift = FALSE)
restaurant_20974.arima4 <- Arima(restaurant_20974_ts.train, order = c(1, 0, 1),
                                 seasonal = list(order = c(1, 0, 1), period = 7),
                                 include.drift = FALSE)

# model evaluation
# forecast
restaurant_20974.arima1.f <- forecast(restaurant_20974.arima1, h = 14)
restaurant_20974.arima2.f <- forecast(restaurant_20974.arima2, h = 14)
restaurant_20974.arima3.f <- forecast(restaurant_20974.arima3, h = 14)
restaurant_20974.arima4.f <- forecast(restaurant_20974.arima4, h = 14)

# out-of-sample performance
accuracy(restaurant_20974.arima1.f, restaurant_20974_ts.test)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 1.304185 48.35115 38.73376 -18.79179 33.82831 0.8172227 -0.0171058
## Test set     1.461356 51.33048 30.75369  -4.22295 13.79735 0.6488556  0.2219546
##              Theil's U
## Training set        NA
## Test set     0.9392753
```

```r
accuracy(restaurant_20974.arima2.f, restaurant_20974_ts.test)
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  0.9991089 48.47608 38.49692 -18.856084 33.65216 0.8122258
## Test set     -8.3863864 51.11250 34.60997  -8.947454 16.32705 0.7302170
##                    ACF1 Theil's U
## Training set 0.02555759        NA
## Test set     0.20352214 0.9781993
```

```r
accuracy(restaurant_20974.arima3.f, restaurant_20974_ts.test)
```

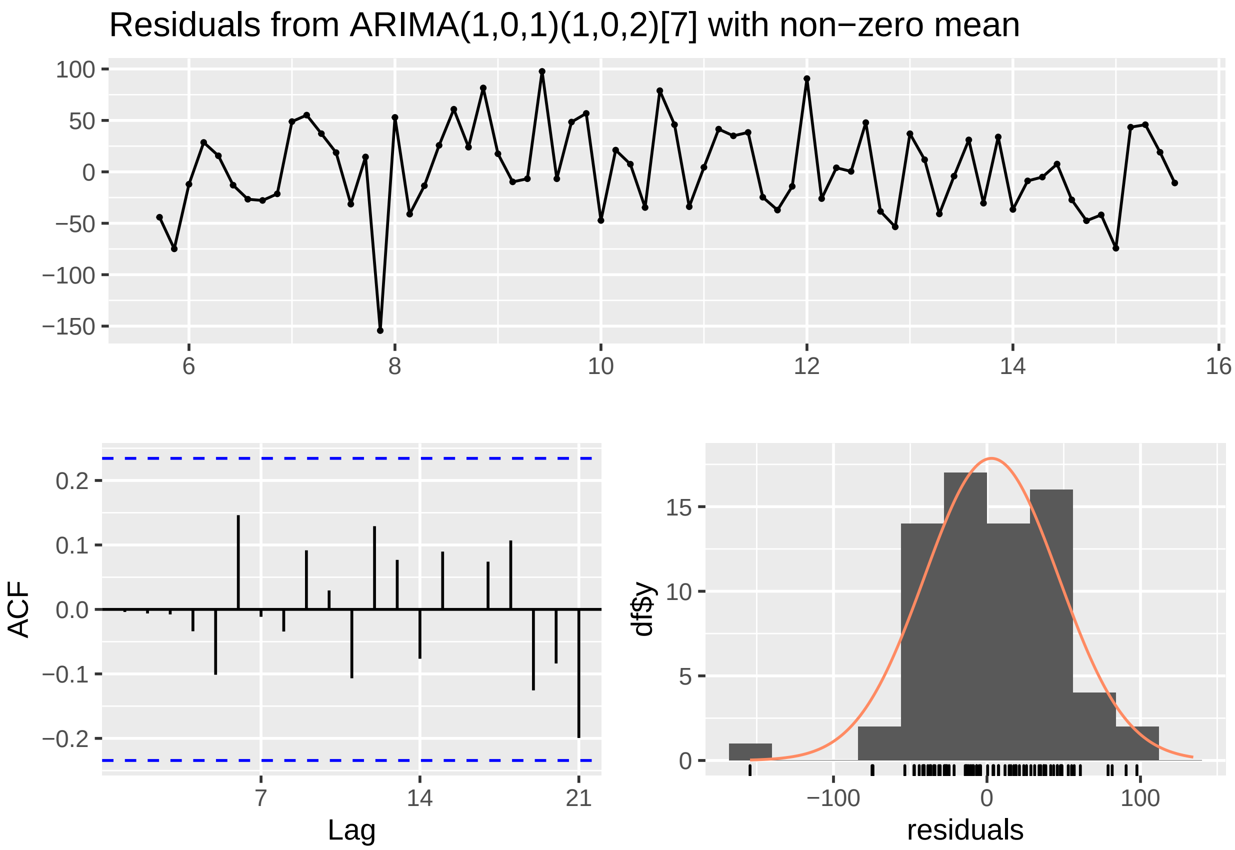
```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  2.969251 43.59166 35.00677 -15.072155 29.40658 0.7385889
## Test set     -3.768077 47.68789 28.68710  -5.661049 12.44291 0.6052537
##                      ACF1 Theil's U
## Training set -0.003941697        NA
## Test set      0.176800946 0.9225553
```

```r
accuracy(restaurant_20974.arima4.f, restaurant_20974_ts.test)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  3.324818 43.26962 34.75944 -14.658123 29.03737 0.7333707
## Test set     -4.368336 48.15276 29.21976  -6.009884 12.77501 0.6164919
##                    ACF1 Theil's U
## Training set 0.01035557        NA
## Test set     0.18145969 0.9524823
```

Looking at the RMSE, MAE and MAPE metrics, the third model has the lowest values, suggesting it has the best fit. We therefore choose ARIMA(1, 0, 1),(1, 0, 2)[7] as our best model. We will likely use this to re-calibrate with the entire sample if the residuals analysis is satisfactory.

```r
# Analyse the residuals
checkresiduals(restaurant_20974.arima3.f)
```



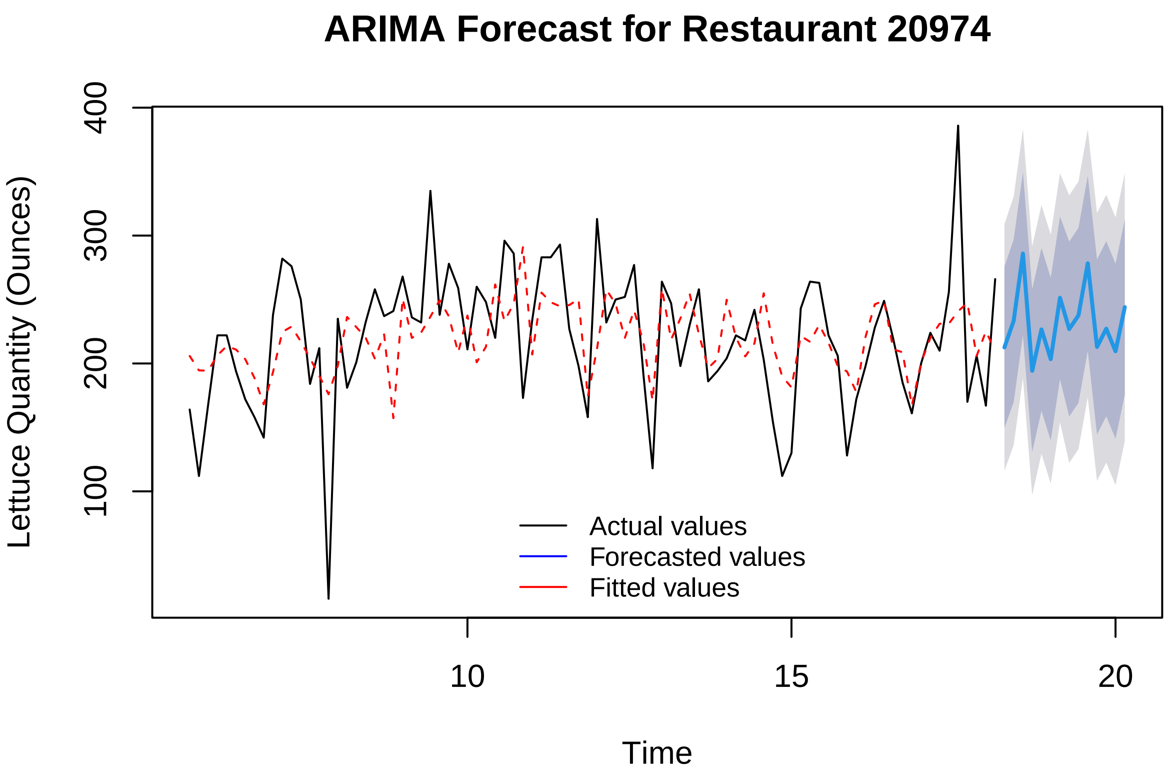

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(1,0,2)[7] with non-zero mean
## Q* = 6.9184, df = 9, p-value = 0.6456
##
## Model df: 5.   Total lags used: 14
```

The ARIMA model tested passed the Ljung-Box test, its residuals are likely to be independently and identically distributed with no significant autocorrelation. Furthermore, it appears that the errors have constant variance and are normally distributed with a mean of zero and that there are no spikes in the ACF plots.

Therefore, we conclude that ARIMA(1, 0, 1),(1, 0, 2)[7] is satisfactory for forecasting lettuce demand for restaurant 20974.

```r
# Forecast
# Re-calibrate model with the entire sample
# Best model forecast (train on the whole dataset)
restaurant_20974.arima <- Arima(restaurant_20974_ts, order = c(1, 0, 1),
                                seasonal = list(order = c(1, 0, 2), period = 7),
                                include.drift = TRUE)
restaurant_20974.arima.f <- forecast(restaurant_20974.arima, h = 14)

# Summary plot of actual, fitted, and forecasted values
plot(restaurant_20974.arima.f, main = "ARIMA Forecast for Restaurant 20974",
     xlab="Time", ylab="Lettuce Quantity (Ounces)")
lines(fitted(restaurant_20974.arima.f), lty = 2, col = "red")
```

```
legend("bottom", legend=c("Actual values","Forecasted values", "Fitted values"),
       col=c("black", "blue", "red"), box.lty=0, lty=1, cex=0.8)
```



Final Forecast

Finally, because both the Holt-Winters and ARIMA models are satisfactory, we compare the forecasting error of each sample on new unseen data to choose the best one.

```
# Forecasting error
accuracy(restaurant_20974.arima3.f, restaurant_20974_ts.test)
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  2.969251 43.59166 35.00677 -15.072155 29.40658 0.7385889
## Test set     -3.768077 47.68789 28.68710  -5.661049 12.44291 0.6052537
##                      ACF1 Theil's U
## Training set -0.003941697        NA
## Test set      0.176800946 0.9225553
```

```
accuracy(restaurant_20974.ets1.f, restaurant_20974_ts.test)
```

```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.5314202 41.53243 34.24962 -13.4603626 26.57567 0.7226142
## Test set      7.6512299 47.28392 25.34920   0.4970002 10.25627 0.5348290
##                    ACF1 Theil's U
## Training set 0.08314736        NA
## Test set     0.13453932 0.8841927
```

Because we want to minimise the RMSE, MAE, and MAPE, we choose the Holt-Winters model and re-calibrate with the entire sample.

```
# Re-calibrate model with the entire sample
restaurant_20974.model <- ets(restaurant_20974_ts, model = 'ANA')
restaurant_20974.model.f <- forecast(restaurant_20974.model, h = 14)

# Convert to a data frame
restaurant_20974.model.f.df <- as.data.frame(restaurant_20974.model.f)
rownames(restaurant_20974.model.f.df) <- c('2015-06-16', '2015-06-17', '2015-06-18',
                                           '2015-06-19', '2015-06-20', '2015-06-21',
                                           '2015-06-22', '2015-06-23', '2015-06-24',
                                           '2015-06-25', '2015-06-26', '2015-06-27',
                                           '2015-06-28', '2015-06-29')
restaurant_20974.model.f.df
```

```
##            Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 2015-06-16       245.3543 187.86551 302.8431 157.43279 333.2758
## 2015-06-17       259.3771 201.13770 317.6166 170.30761 348.4466
## 2015-06-18       245.3595 186.37898 304.3400 155.15658 335.5624
## 2015-06-19       207.5218 147.80939 267.2342 116.19955 298.8440
## 2015-06-20       163.0179 102.58243 223.4533  70.58984 255.4459
## 2015-06-21       224.4164 163.26647 285.5663 130.89565 317.9371
## 2015-06-22       244.1267 182.26971 305.9837 149.52459 338.7289
## 2015-06-23       245.3543 182.79903 307.9096 149.68427 341.0243
## 2015-06-24       259.3771 196.13132 322.6229 162.65102 356.1032
## 2015-06-25       245.3595 181.43061 309.2884 147.58870 343.1303
## 2015-06-26       207.5218 142.91705 272.1265 108.71737 306.3262
## 2015-06-27       163.0179  97.74427 228.2915  63.19051 262.8452
## 2015-06-28       224.4164 158.48071 290.3521 123.57647 325.2563
## 2015-06-29       244.1267 177.53475 310.7187 142.28309 345.9704
```

Conclusion and Summary of Forecasts

To conclude, the goal of this report was to use ARIMA and Holt-Winters models to forecast lettuce demand over two weeks for four US-based branches of a fast-food chain. Once we obtained the historical data for each restaurant, we transformed the values to a time-series object. Next, we split the data between a training and a test set to train the models and evaluate their out-of-sample performance, emphasising out-of-sample performance over in-sample performance. Then, using the time series decomposition to support our model specifications, we trained both Holt-Winters and ARIMA models until they could not be improved and evaluated them against each other on unseen data. Finally, the best-performing models were used to forecast lettuce demand for the Berkeley and New York restaurants.

Nevertheless, despite the attention to detail brought to the construction of the forecasting models, some limitations associated with each of the models used can be addressed. First, the ARIMA model assumes a linear time series and does not work well with long-term forecasting. Then, the exponential smoothing model is sensitive to the initial values of the smoothing parameters. Also, a limitation of our application of these models is that we did not consider external factors that may have affected the historical data or influenced our forecasts. Although the ETS model does not consider external factors, these could have been integrated with the ARIMA model using ARIMAX (ARIMA with exogenous variables). Finally, the restrictive time frame of the historical data likely acted both in our favour and against us as it mitigated the limitations due to excluding external factors but may have overlooked some additional seasonal elements (for example, in a yearly cycle).

A summary of the forecasts for each restaurant is provided below.

```r
dates <- as.Date(c('2015-06-16', '2015-06-17', '2015-06-18', '2015-06-19',
                   '2015-06-20', '2015-06-21', '2015-06-22', '2015-06-23',
                   '2015-06-24', '2015-06-25', '2015-06-26', '2015-06-27',
                   '2015-06-28', '2015-06-29'))

# Create the data frame with the rounded values to the nearest whole number
final_forecasts <- data.frame(
  Date = dates,
  `46673` = c(round(restaurant_46673.model.f.df$`Point Forecast`)),
  `4904` = c(round(restaurant_4904.model.f.df$`Point Forecast`)),
  `12631` = c(round(restaurant_12631.model.f.df$`Point Forecast`)),
  `20974` = c(round(restaurant_20974.model.f.df$`Point Forecast`))
)

colnames(final_forecasts) <- c('Store', 'California 1 (ID:46673)', 'California 2 (ID:4904)',
                               'New York 1 (ID:12631)', 'New York 1 (ID:20974)')

final_forecasts
```

```
##         Store California 1 (ID:46673) California 2 (ID:4904)
## 1  2015-06-16                     172                    359
## 2  2015-06-17                     175                    348
## 3  2015-06-18                     165                    309
## 4  2015-06-19                     101                    212
## 5  2015-06-20                      77                    213
## 6  2015-06-21                     169                    337
## 7  2015-06-22                     177                    337
## 8  2015-06-23                     161                    359
## 9  2015-06-24                     173                    348
## 10 2015-06-25                     164                    309
```

```
## 11 2015-06-26                    101                    212
## 12 2015-06-27                     77                    213
## 13 2015-06-28                    169                    337
## 14 2015-06-29                    177                    337
##    New York 1 (ID:12631) New York 1 (ID:20974)
## 1                    258                   245
## 2                    276                   259
## 3                    303                   245
## 4                    307                   208
## 5                    233                   163
## 6                    266                   224
## 7                    247                   244
## 8                    258                   245
## 9                    276                   259
## 10                   303                   245
## 11                   307                   208
## 12                   233                   163
## 13                   266                   224
## 14                   247                   244
```