Demand Forecasting for a Fast-Food Restaurant Chain

Logistics and Supply Chain Analytics - Individual Project

Konstantinos Paganopoulos

Solutions

We first load the necessary libraries.

We have a dataset, which includes daily sales for lettuce at a store in New York from a fast-food restaurant chain from 5 March 2015 to 15 June 2015. Each observation includes two values: day pair, and sales in that particular day.

Then, we load and split the data set into train and test set.

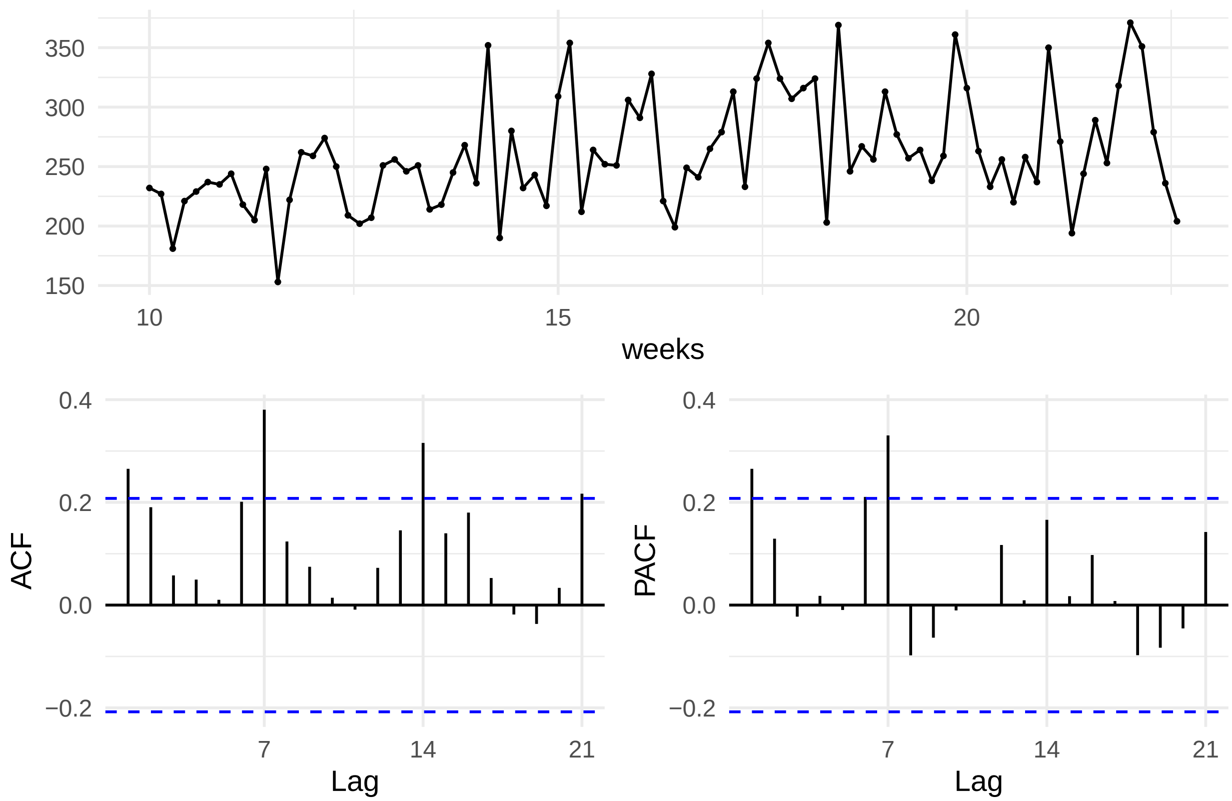
```
# read csv file
data <- read.csv(file = "NewYork1_final.csv", header = TRUE, stringsAsFactors = FALSE)

# convert column date of data set to type date
data$date <- as.Date(data$date)

# convert sales into a time series object
lettuce <- ts(data[, 2], frequency = 7, start = c(10, 1)) # 10th week 1st day

# split data set into train and test set
lettuce_train <- subset(lettuce, end = 89)
lettuce_test <- subset(lettuce, start = 90) # last 14 lines-days
```

ARIMA

We visually inspect the time series.

```
autoplot(lettuce_train, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 1 Store (12631) - Time series plot")
```


```r
ggtsdisplay(lettuce_train, xlab = "weeks", theme = theme_minimal())
```



Due to the trend in time series, it is non-stationary. We can get rid of trend by taking first-order difference. We plot the time series after the difference, and observe that there is no trend and appears to be stationary. We run ADF, PP and KPSS tests to formally test the stationarity of time series after the first-order difference, and all suggest that the time series is stationary.

```r
# stationary test
adf.test(lettuce_train)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  lettuce_train
## Dickey-Fuller = -4.8181, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pp.test(lettuce_train)
```

```
## Warning in pp.test(lettuce_train): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
```

```
## data:  lettuce_train
## Dickey-Fuller Z(alpha) = -75.753, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

```r
kpss.test(lettuce_train)
```

```
## Warning in kpss.test(lettuce_train): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  lettuce_train
## KPSS Level = 0.96112, Truncation lag parameter = 3, p-value = 0.01
```

The two automatic functions, ndiffs() and nsdiffs() tell us how many first-order differences, and how many seasonal differences, respectively, we need to take to make the time series stationary. We use those functions below:

```r
ndiffs(lettuce_train)
```

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

```r
# seasonal stationarity
nsdiffs(lettuce_train)
```

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

We need to differentiate one time.

```r
### stationarize time series
# take first order difference
lettuce_train.diff1 <- diff(lettuce_train, differences = 1)
```

Check again the tests for stationarity:

```r
# stationary test
adf.test(lettuce_train.diff1)
```

```
## Warning in adf.test(lettuce_train.diff1): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  lettuce_train.diff1
## Dickey-Fuller = -7.178, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pp.test(lettuce_train.diff1)
```

```
## Warning in pp.test(lettuce_train.diff1): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  lettuce_train.diff1
## Dickey-Fuller Z(alpha) = -112.24, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

```r
kpss.test(lettuce_train.diff1)
```

```
## Warning in kpss.test(lettuce_train.diff1): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  lettuce_train.diff1
## KPSS Level = 0.046459, Truncation lag parameter = 3, p-value = 0.1
```

Check again the two automatic functions for stationarity:

```r
ndiffs(lettuce_train.diff1)
```

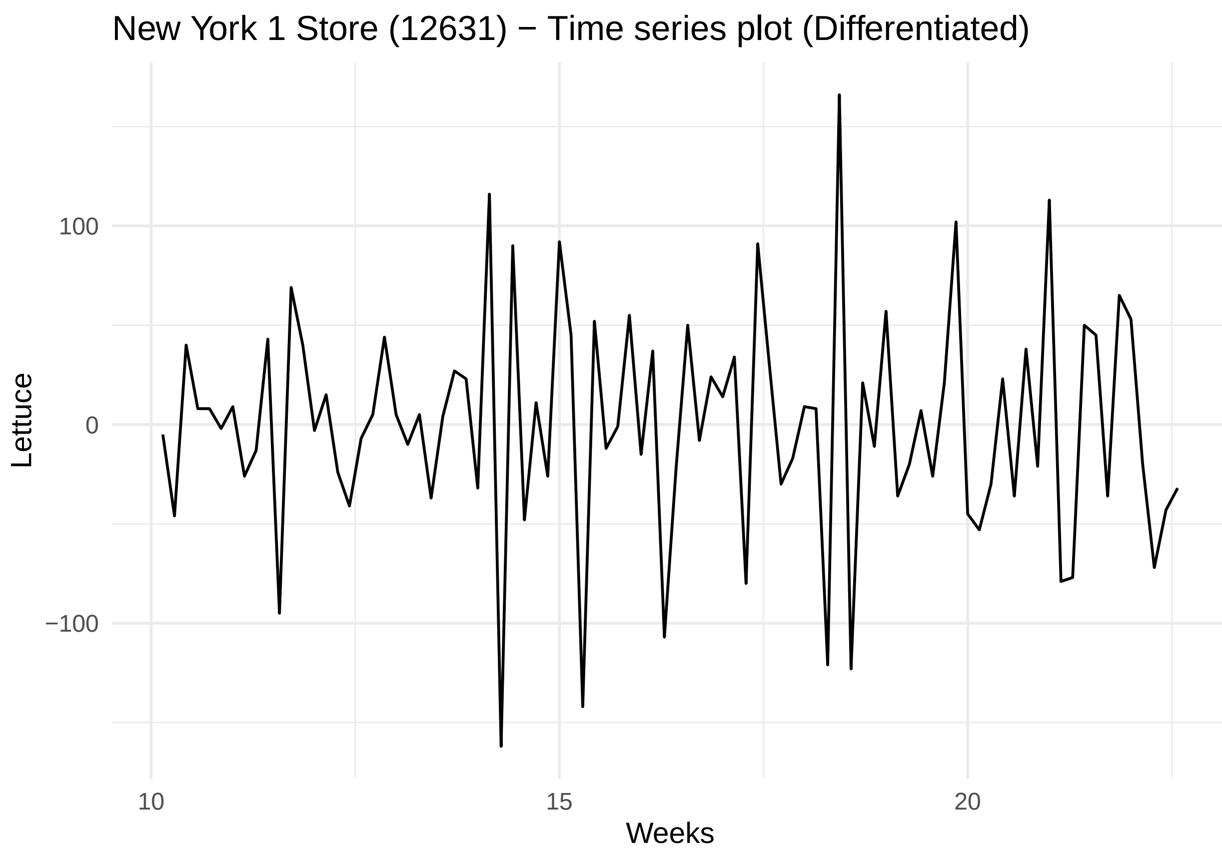
```
## [1] 0
```

```r
nsdiffs(lettuce_train.diff1)
```

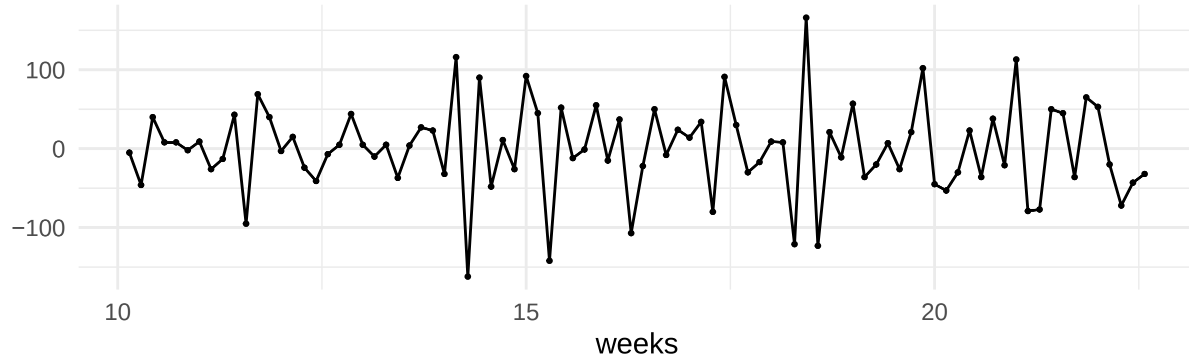
```
## [1] 0
```

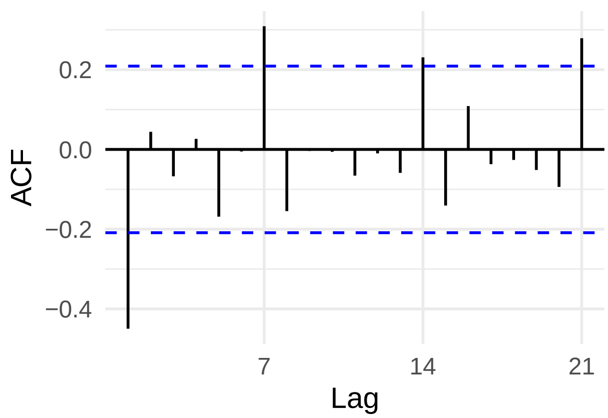
We now visually inspect the differentiated time series.

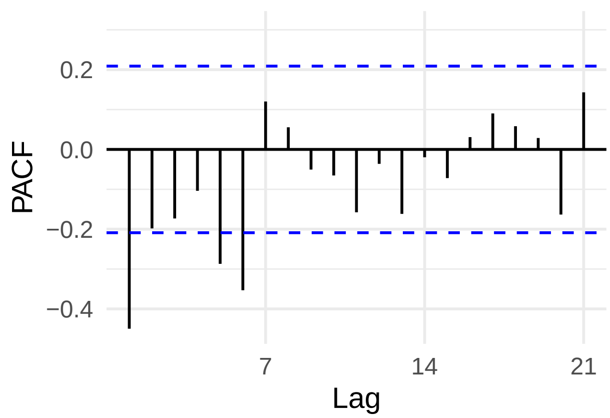
```r
autoplot(lettuce_train.diff1, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 1 Store (12631) - Time series plot (Differentiated)")
```




```r
ggtsdisplay(lettuce_train.diff1, xlab = "weeks", theme = theme_minimal())
```



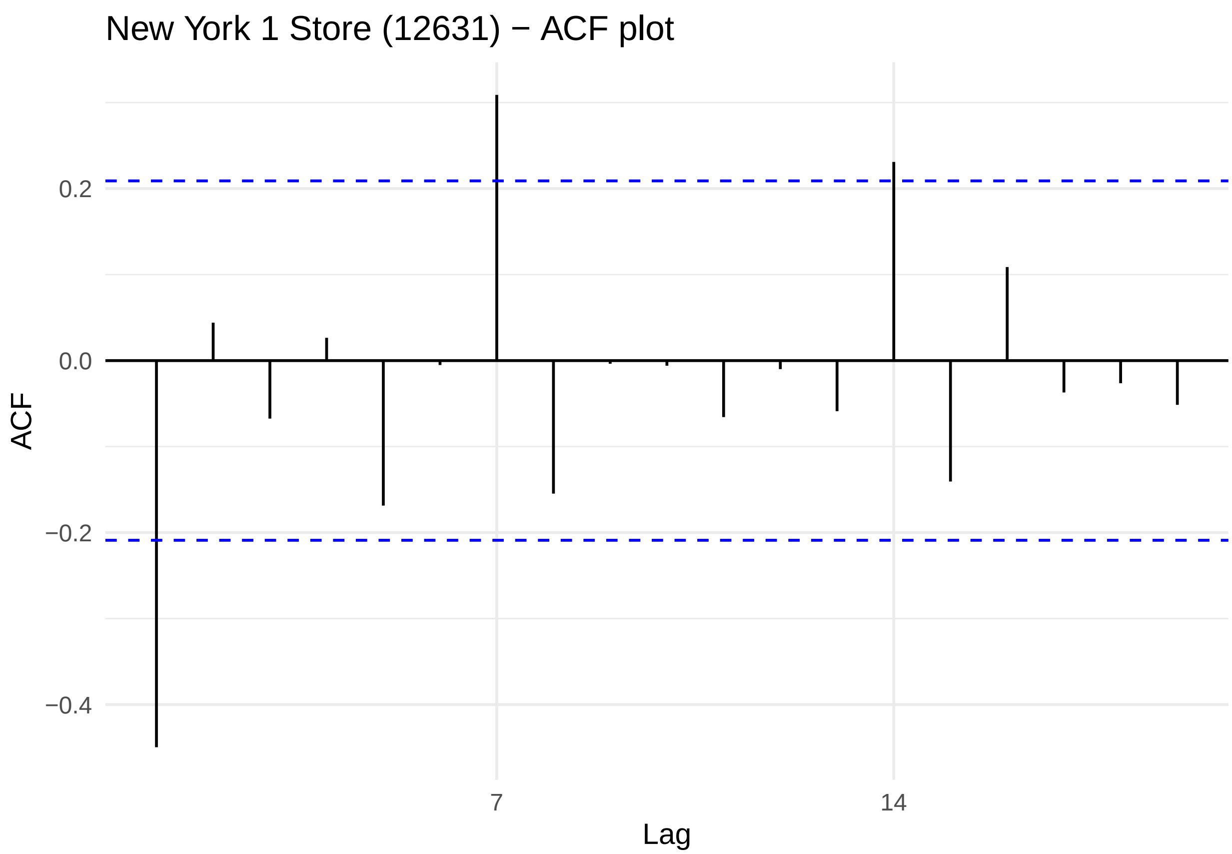




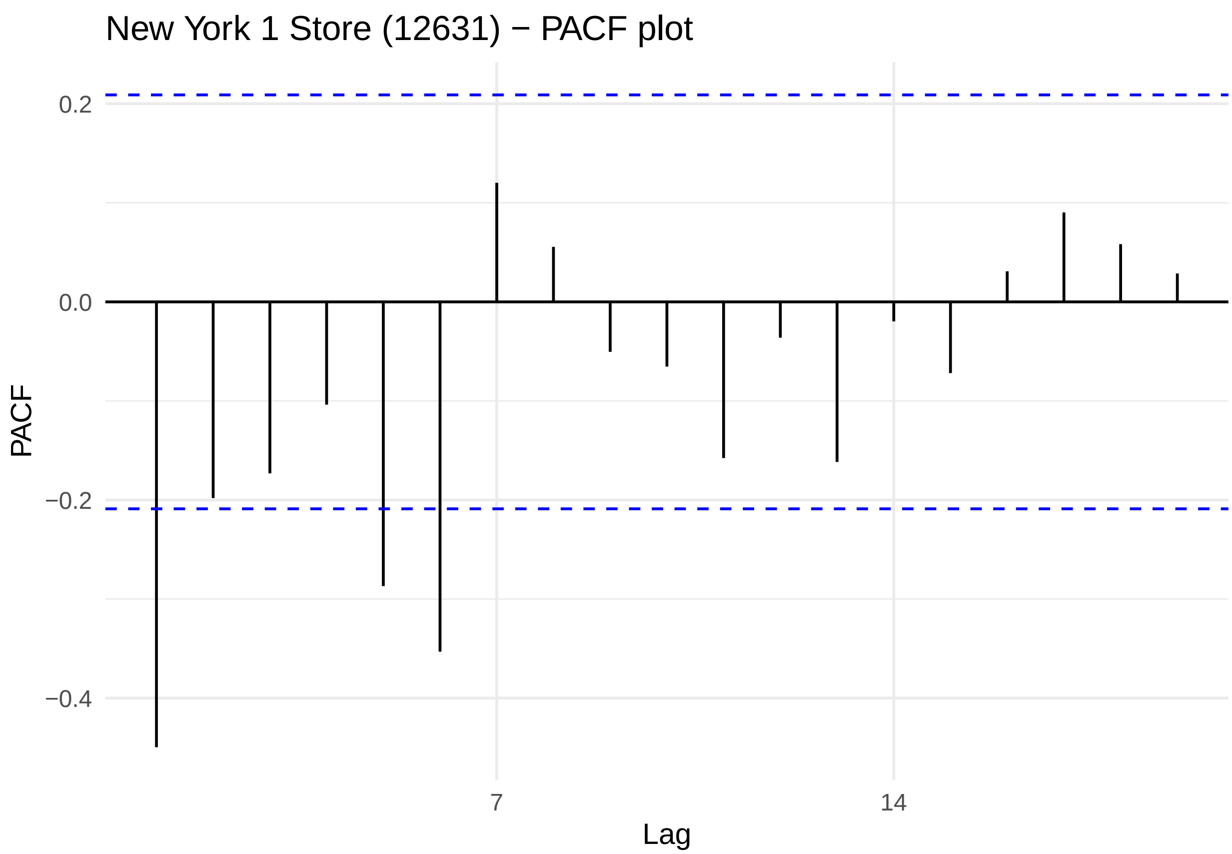
Looks stationary.

Once we have a stationary time series, the next step is to determine the optimal orders of MA and AR components. We first plot the ACF and PACF of the time series.

```
# acf plot
ggAcf(lettuce_train.diff1) + theme_minimal() + ggtitle("New York 1 Store (12631) - ACF plot")
```




```r
# pacf plot
ggPacf(lettuce_train.diff1) + theme_minimal() + ggtitle("New York 1 Store (12631) - PACF plot")
```



Next we use *auto.arima*() to search for the best ARIMA models.

The default procedure uses some approximations to speed up the search. These approximations can be avoided with the argument approximation = FALSE. It is possible that the minimum AIC model will not be found due to these approximations, or because of the stepwise procedure. A much larger set of models will be searched if the argument stepwise = FALSE is used. We also use d = 1 and D = 0 since we had first-differencing but no seasonal-differencing.

```
auto.arima(lettuce_train, trace = TRUE, ic = 'aic', approximation = FALSE, stepwise = FALSE, d=1, D=0)
```

```
##
##  ARIMA(0,1,0)                               : 962.3509
##  ARIMA(0,1,0)            with drift         : 964.3481
##  ARIMA(0,1,0)(0,0,1)[7]                     : 957.5856
##  ARIMA(0,1,0)(0,0,1)[7] with drift          : 959.5792
##  ARIMA(0,1,0)(0,0,2)[7]                     : 958.2074
##  ARIMA(0,1,0)(0,0,2)[7] with drift          : 960.2018
##  ARIMA(0,1,0)(1,0,0)[7]                     : 955.5445
##  ARIMA(0,1,0)(1,0,0)[7] with drift          : 957.5381
##  ARIMA(0,1,0)(1,0,1)[7]                     : Inf
##  ARIMA(0,1,0)(1,0,1)[7] with drift          : Inf
##  ARIMA(0,1,0)(1,0,2)[7]                     : Inf
##  ARIMA(0,1,0)(1,0,2)[7] with drift          : Inf
##  ARIMA(0,1,0)(2,0,0)[7]                     : 955.1941
##  ARIMA(0,1,0)(2,0,0)[7] with drift          : 957.1903
##  ARIMA(0,1,0)(2,0,1)[7]                     : Inf
```

```
## ARIMA(0,1,0)(2,0,1)[7] with drift         : Inf
## ARIMA(0,1,0)(2,0,2)[7]                    : Inf
## ARIMA(0,1,0)(2,0,2)[7] with drift         : Inf
## ARIMA(0,1,1)                              : 923.6733
## ARIMA(0,1,1)           with drift         : 923.4671
## ARIMA(0,1,1)(0,0,1)[7]                    : 919.049
## ARIMA(0,1,1)(0,0,1)[7] with drift         : Inf
## ARIMA(0,1,1)(0,0,2)[7]                    : 917.2619
## ARIMA(0,1,1)(0,0,2)[7] with drift         : Inf
## ARIMA(0,1,1)(1,0,0)[7]                    : 915.5441
## ARIMA(0,1,1)(1,0,0)[7] with drift         : Inf
## ARIMA(0,1,1)(1,0,1)[7]                    : Inf
## ARIMA(0,1,1)(1,0,1)[7] with drift         : Inf
## ARIMA(0,1,1)(1,0,2)[7]                    : Inf
## ARIMA(0,1,1)(1,0,2)[7] with drift         : Inf
## ARIMA(0,1,1)(2,0,0)[7]                    : 912.3795
## ARIMA(0,1,1)(2,0,0)[7] with drift         : Inf
## ARIMA(0,1,1)(2,0,1)[7]                    : Inf
## ARIMA(0,1,1)(2,0,1)[7] with drift         : Inf
## ARIMA(0,1,1)(2,0,2)[7]                    : Inf
## ARIMA(0,1,1)(2,0,2)[7] with drift         : Inf
## ARIMA(0,1,2)                              : 924.7507
## ARIMA(0,1,2)           with drift         : Inf
## ARIMA(0,1,2)(0,0,1)[7]                    : 920.0795
## ARIMA(0,1,2)(0,0,1)[7] with drift         : Inf
## ARIMA(0,1,2)(0,0,2)[7]                    : 918.644
## ARIMA(0,1,2)(0,0,2)[7] with drift         : Inf
## ARIMA(0,1,2)(1,0,0)[7]                    : 916.7102
## ARIMA(0,1,2)(1,0,0)[7] with drift         : Inf
## ARIMA(0,1,2)(1,0,1)[7]                    : Inf
## ARIMA(0,1,2)(1,0,1)[7] with drift         : Inf
## ARIMA(0,1,2)(1,0,2)[7]                    : Inf
## ARIMA(0,1,2)(1,0,2)[7] with drift         : Inf
## ARIMA(0,1,2)(2,0,0)[7]                    : 913.8304
## ARIMA(0,1,2)(2,0,0)[7] with drift         : Inf
## ARIMA(0,1,2)(2,0,1)[7]                    : Inf
## ARIMA(0,1,2)(2,0,1)[7] with drift         : Inf
## ARIMA(0,1,3)                              : 926.4351
## ARIMA(0,1,3)           with drift         : Inf
## ARIMA(0,1,3)(0,0,1)[7]                    : 920.5319
## ARIMA(0,1,3)(0,0,1)[7] with drift         : Inf
## ARIMA(0,1,3)(0,0,2)[7]                    : 919.5866
## ARIMA(0,1,3)(0,0,2)[7] with drift         : Inf
## ARIMA(0,1,3)(1,0,0)[7]                    : 916.7909
## ARIMA(0,1,3)(1,0,0)[7] with drift         : Inf
## ARIMA(0,1,3)(1,0,1)[7]                    : Inf
## ARIMA(0,1,3)(1,0,1)[7] with drift         : Inf
## ARIMA(0,1,3)(2,0,0)[7]                    : 915.0381
## ARIMA(0,1,3)(2,0,0)[7] with drift         : Inf
## ARIMA(0,1,4)                              : 927.5227
## ARIMA(0,1,4)           with drift         : Inf
## ARIMA(0,1,4)(0,0,1)[7]                    : 922.5064
## ARIMA(0,1,4)(0,0,1)[7] with drift         : Inf
## ARIMA(0,1,4)(1,0,0)[7]                    : 918.7093
```

```
##  ARIMA(0,1,4)(1,0,0)[7] with drift         : Inf
##  ARIMA(0,1,5)                              : 929.2936
##  ARIMA(0,1,5)           with drift         : Inf
##  ARIMA(1,1,0)                              : 944.6207
##  ARIMA(1,1,0)           with drift         : 946.6182
##  ARIMA(1,1,0)(0,0,1)[7]                    : 935.3688
##  ARIMA(1,1,0)(0,0,1)[7] with drift         : 937.3575
##  ARIMA(1,1,0)(0,0,2)[7]                    : 935.7208
##  ARIMA(1,1,0)(0,0,2)[7] with drift         : 937.7102
##  ARIMA(1,1,0)(1,0,0)[7]                    : 932.152
##  ARIMA(1,1,0)(1,0,0)[7] with drift         : 934.1414
##  ARIMA(1,1,0)(1,0,1)[7]                    : Inf
##  ARIMA(1,1,0)(1,0,1)[7] with drift         : Inf
##  ARIMA(1,1,0)(1,0,2)[7]                    : Inf
##  ARIMA(1,1,0)(1,0,2)[7] with drift         : Inf
##  ARIMA(1,1,0)(2,0,0)[7]                    : 932.7419
##  ARIMA(1,1,0)(2,0,0)[7] with drift         : 934.7357
##  ARIMA(1,1,0)(2,0,1)[7]                    : Inf
##  ARIMA(1,1,0)(2,0,1)[7] with drift         : Inf
##  ARIMA(1,1,0)(2,0,2)[7]                    : Inf
##  ARIMA(1,1,0)(2,0,2)[7] with drift         : Inf
##  ARIMA(1,1,1)                              : 924.6769
##  ARIMA(1,1,1)           with drift         : Inf
##  ARIMA(1,1,1)(0,0,1)[7]                    : 919.8443
##  ARIMA(1,1,1)(0,0,1)[7] with drift         : Inf
##  ARIMA(1,1,1)(0,0,2)[7]                    : 918.5174
##  ARIMA(1,1,1)(0,0,2)[7] with drift         : Inf
##  ARIMA(1,1,1)(1,0,0)[7]                    : 916.4674
##  ARIMA(1,1,1)(1,0,0)[7] with drift         : Inf
##  ARIMA(1,1,1)(1,0,1)[7]                    : Inf
##  ARIMA(1,1,1)(1,0,1)[7] with drift         : Inf
##  ARIMA(1,1,1)(1,0,2)[7]                    : Inf
##  ARIMA(1,1,1)(1,0,2)[7] with drift         : Inf
##  ARIMA(1,1,1)(2,0,0)[7]                    : 913.728
##  ARIMA(1,1,1)(2,0,0)[7] with drift         : Inf
##  ARIMA(1,1,1)(2,0,1)[7]                    : 915.7907
##  ARIMA(1,1,1)(2,0,1)[7] with drift         : Inf
##  ARIMA(1,1,2)                              : 926.6681
##  ARIMA(1,1,2)           with drift         : Inf
##  ARIMA(1,1,2)(0,0,1)[7]                    : 921.5654
##  ARIMA(1,1,2)(0,0,1)[7] with drift         : Inf
##  ARIMA(1,1,2)(0,0,2)[7]                    : 920.3171
##  ARIMA(1,1,2)(0,0,2)[7] with drift         : Inf
##  ARIMA(1,1,2)(1,0,0)[7]                    : 917.9811
##  ARIMA(1,1,2)(1,0,0)[7] with drift         : Inf
##  ARIMA(1,1,2)(1,0,1)[7]                    : Inf
##  ARIMA(1,1,2)(1,0,1)[7] with drift         : Inf
##  ARIMA(1,1,2)(2,0,0)[7]                    : Inf
##  ARIMA(1,1,2)(2,0,0)[7] with drift         : Inf
##  ARIMA(1,1,3)                              : 928.1161
##  ARIMA(1,1,3)           with drift         : Inf
##  ARIMA(1,1,3)(0,0,1)[7]                    : Inf
##  ARIMA(1,1,3)(0,0,1)[7] with drift         : Inf
##  ARIMA(1,1,3)(1,0,0)[7]                    : 917.4552
```

```
##  ARIMA(1,1,3)(1,0,0)[7] with drift         : Inf
##  ARIMA(1,1,4)                              : 929.5955
##  ARIMA(1,1,4)           with drift         : Inf
##  ARIMA(2,1,0)                              : 942.9122
##  ARIMA(2,1,0)           with drift         : 944.9121
##  ARIMA(2,1,0)(0,0,1)[7]                    : 934.9384
##  ARIMA(2,1,0)(0,0,1)[7] with drift         : 936.9341
##  ARIMA(2,1,0)(0,0,2)[7]                    : 934.606
##  ARIMA(2,1,0)(0,0,2)[7] with drift         : 936.6031
##  ARIMA(2,1,0)(1,0,0)[7]                    : 931.1616
##  ARIMA(2,1,0)(1,0,0)[7] with drift         : 933.1577
##  ARIMA(2,1,0)(1,0,1)[7]                    : Inf
##  ARIMA(2,1,0)(1,0,1)[7] with drift         : Inf
##  ARIMA(2,1,0)(1,0,2)[7]                    : Inf
##  ARIMA(2,1,0)(1,0,2)[7] with drift         : Inf
##  ARIMA(2,1,0)(2,0,0)[7]                    : 930.6233
##  ARIMA(2,1,0)(2,0,0)[7] with drift         : 932.6229
##  ARIMA(2,1,0)(2,0,1)[7]                    : Inf
##  ARIMA(2,1,0)(2,0,1)[7] with drift         : Inf
##  ARIMA(2,1,1)                              : 926.6461
##  ARIMA(2,1,1)           with drift         : Inf
##  ARIMA(2,1,1)(0,0,1)[7]                    : 920.9339
##  ARIMA(2,1,1)(0,0,1)[7] with drift         : Inf
##  ARIMA(2,1,1)(0,0,2)[7]                    : 919.8506
##  ARIMA(2,1,1)(0,0,2)[7] with drift         : Inf
##  ARIMA(2,1,1)(1,0,0)[7]                    : 917.0336
##  ARIMA(2,1,1)(1,0,0)[7] with drift         : Inf
##  ARIMA(2,1,1)(1,0,1)[7]                    : Inf
##  ARIMA(2,1,1)(1,0,1)[7] with drift         : Inf
##  ARIMA(2,1,1)(2,0,0)[7]                    : 915.1373
##  ARIMA(2,1,1)(2,0,0)[7] with drift         : Inf
##  ARIMA(2,1,2)                              : 928.3411
##  ARIMA(2,1,2)           with drift         : Inf
##  ARIMA(2,1,2)(0,0,1)[7]                    : Inf
##  ARIMA(2,1,2)(0,0,1)[7] with drift         : Inf
##  ARIMA(2,1,2)(1,0,0)[7]                    : 916.9772
##  ARIMA(2,1,2)(1,0,0)[7] with drift         : Inf
##  ARIMA(2,1,3)                              : 927.9445
##  ARIMA(2,1,3)           with drift         : Inf
##  ARIMA(3,1,0)                              : 941.6013
##  ARIMA(3,1,0)           with drift         : 943.5957
##  ARIMA(3,1,0)(0,0,1)[7]                    : 934.2744
##  ARIMA(3,1,0)(0,0,1)[7] with drift         : 936.2742
##  ARIMA(3,1,0)(0,0,2)[7]                    : 933.4882
##  ARIMA(3,1,0)(0,0,2)[7] with drift         : 935.4867
##  ARIMA(3,1,0)(1,0,0)[7]                    : 930.248
##  ARIMA(3,1,0)(1,0,0)[7] with drift         : 932.2476
##  ARIMA(3,1,0)(1,0,1)[7]                    : Inf
##  ARIMA(3,1,0)(1,0,1)[7] with drift         : Inf
##  ARIMA(3,1,0)(2,0,0)[7]                    : 928.6093
##  ARIMA(3,1,0)(2,0,0)[7] with drift         : 930.6034
##  ARIMA(3,1,1)                              : 927.1447
##  ARIMA(3,1,1)           with drift         : 926.7629
##  ARIMA(3,1,1)(0,0,1)[7]                    : 922.13
```

```
##  ARIMA(3,1,1)(0,0,1)[7] with drift         : Inf
##  ARIMA(3,1,1)(1,0,0)[7]                    : 918.6149
##  ARIMA(3,1,1)(1,0,0)[7] with drift         : Inf
##  ARIMA(3,1,2)                              : 927.2523
##  ARIMA(3,1,2)           with drift         : 927.4237
##  ARIMA(4,1,0)                              : 941.885
##  ARIMA(4,1,0)           with drift         : 943.8584
##  ARIMA(4,1,0)(0,0,1)[7]                    : 935.8141
##  ARIMA(4,1,0)(0,0,1)[7] with drift         : 937.8106
##  ARIMA(4,1,0)(1,0,0)[7]                    : 931.9403
##  ARIMA(4,1,0)(1,0,0)[7] with drift         : 933.938
##  ARIMA(4,1,1)                              : 927.7835
##  ARIMA(4,1,1)           with drift         : 927.6555
##  ARIMA(5,1,0)                              : 933.9611
##  ARIMA(5,1,0)           with drift         : 935.8068
##
##
##
##  Best model: ARIMA(0,1,1)(2,0,0)[7]
```

```
## Series: lettuce_train
## ARIMA(0,1,1)(2,0,0)[7]
##
## Coefficients:
##           ma1    sar1    sar2
##       -0.9282  0.2638  0.2617
## s.e.   0.0436  0.1074  0.1120
##
## sigma^2 estimated as 1695:  log likelihood=-452.19
## AIC=912.38   AICc=912.86   BIC=922.29
```

```
# Best model: ARIMA(0,1,1)(2,0,0)[7]  (AIC=912.38)
# Second best:  ARIMA(1,1,1)(2,0,0)[7]  (AIC=913.72)
# Third best: ARIMA(0,1,2)(2,0,0)[7] (AIC=913.83)
```

Based on the output of *auto.arima*(), a couple of models have similar AICs. Now suppose that we choose the three models with the lowest AICs, namely ARIMA(0,1,1)(2,0,0)[7] with AIC=912.38, ARIMA(1,1,1)(2,0,0)[7] with AIC=913.72 AND ARIMA(0,1,2)(2,0,0)[7] with AIC=913.83, as the candidate models that we would like to evaluate further.

```
# three candidate models
lettuce.m1 <- Arima(lettuce_train, order = c(0, 1, 1),
                        seasonal = list(order = c(2, 0, 0), period = 7))
lettuce.m2 <- Arima(lettuce_train, order = c(1, 1, 1),
                        seasonal = list(order = c(2, 0, 0), period = 7))
lettuce.m3 <- Arima(lettuce_train, order = c(0, 1, 2),
                        seasonal = list(order = c(2, 0, 0), period = 7))
```

Now we evaluate the in-sample performance/fit of the model with *accuracy*() function, which summarizes various measures of fitting errors.

A couple of functions are proved to be useful for us to evaluate the in-sample performance/fit of the model. One is accuracy() function, which summarizes various measures of fitting errors. In the post-estimation analysis, we would also like to check out the residual plots, including time series, ACFs and etc, to make sure that there is no warning signal. In particular, residuals shall have a zero mean, constant variance, and distributed symmetrically around mean zero. ACF of any lag greater 0 is expected to be statistically insignificant.

```r
# in-sample one-step forecasts model 1
accuracy(lettuce.m1)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 4.103944 40.24048 31.06843 -0.5328407 11.90586 0.8402411
##                    ACF1
## Training set 0.05766809
```

```r
# in-sample one-step forecasts model 2
accuracy(lettuce.m2)
```

```
##                   ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 4.36677 40.13025 30.81424 -0.4249124 11.82139 0.8333666
##                     ACF1
## Training set -0.02958127
```

```r
# in-sample one-step forecasts model 3
accuracy(lettuce.m3)
```

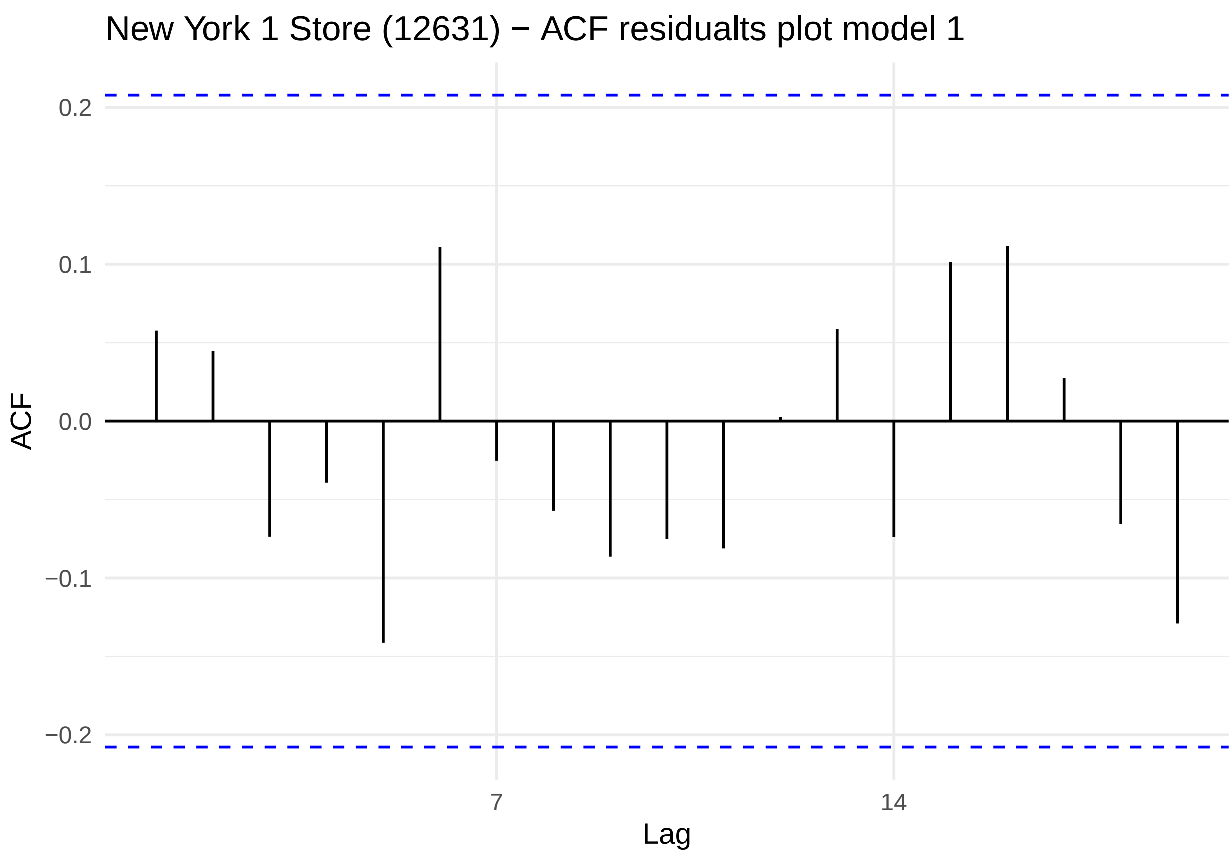
```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 4.317327 40.14598 30.86092 -0.4448752 11.83567 0.8346291
##                     ACF1
## Training set -0.01581229
```

The second model has the lowest RMSE, even though the first has a lowest AIC score.

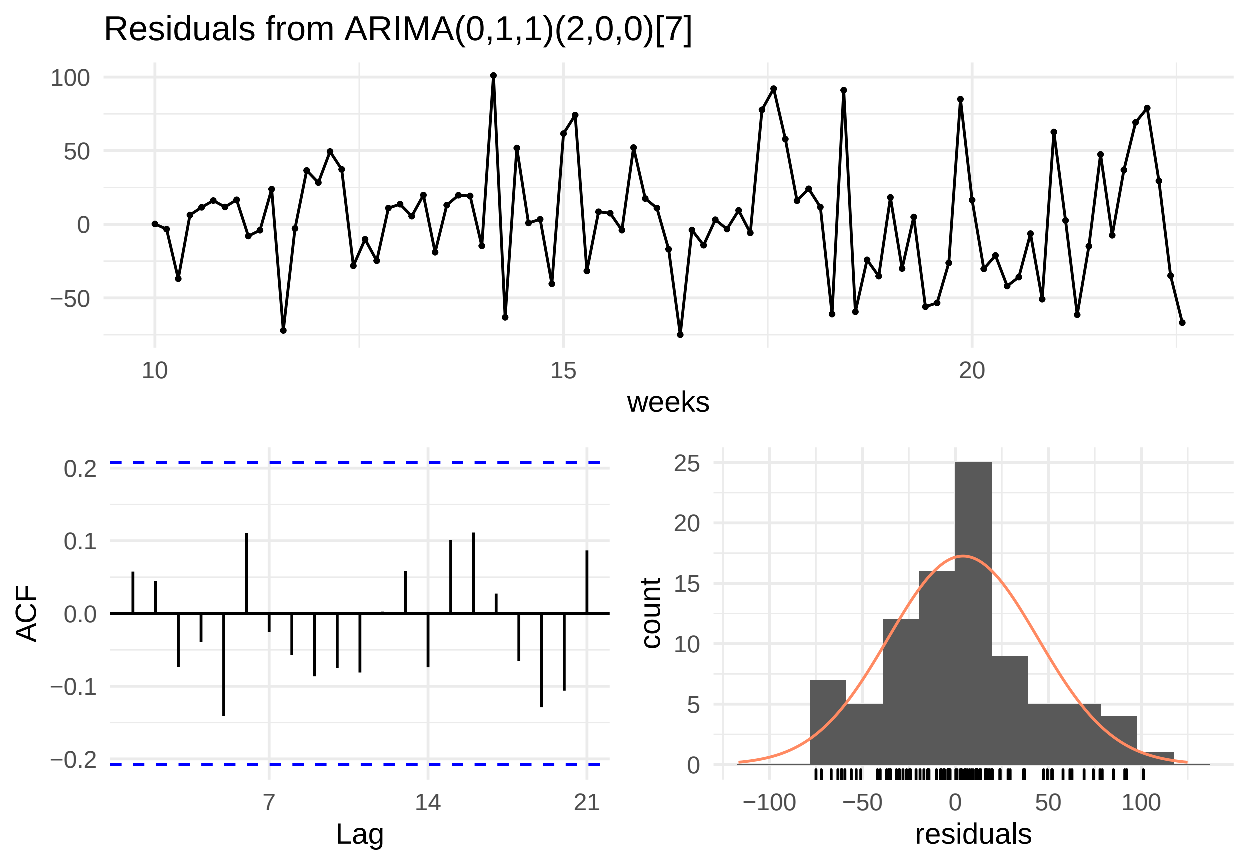
Now we proceed with the residual analysis of the three models.

```r
# residual analysis model 1
autoplot(lettuce.m1$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 1 Store (12631) - Residuals model 1 plot")
```


```
ggAcf(lettuce.m1$residuals) + theme_minimal() +
ggtitle("New York 1 Store (12631) - ACF residualts plot model 1")
```




```r
checkresiduals(lettuce.m1, xlab = "weeks", theme = theme_minimal())
```

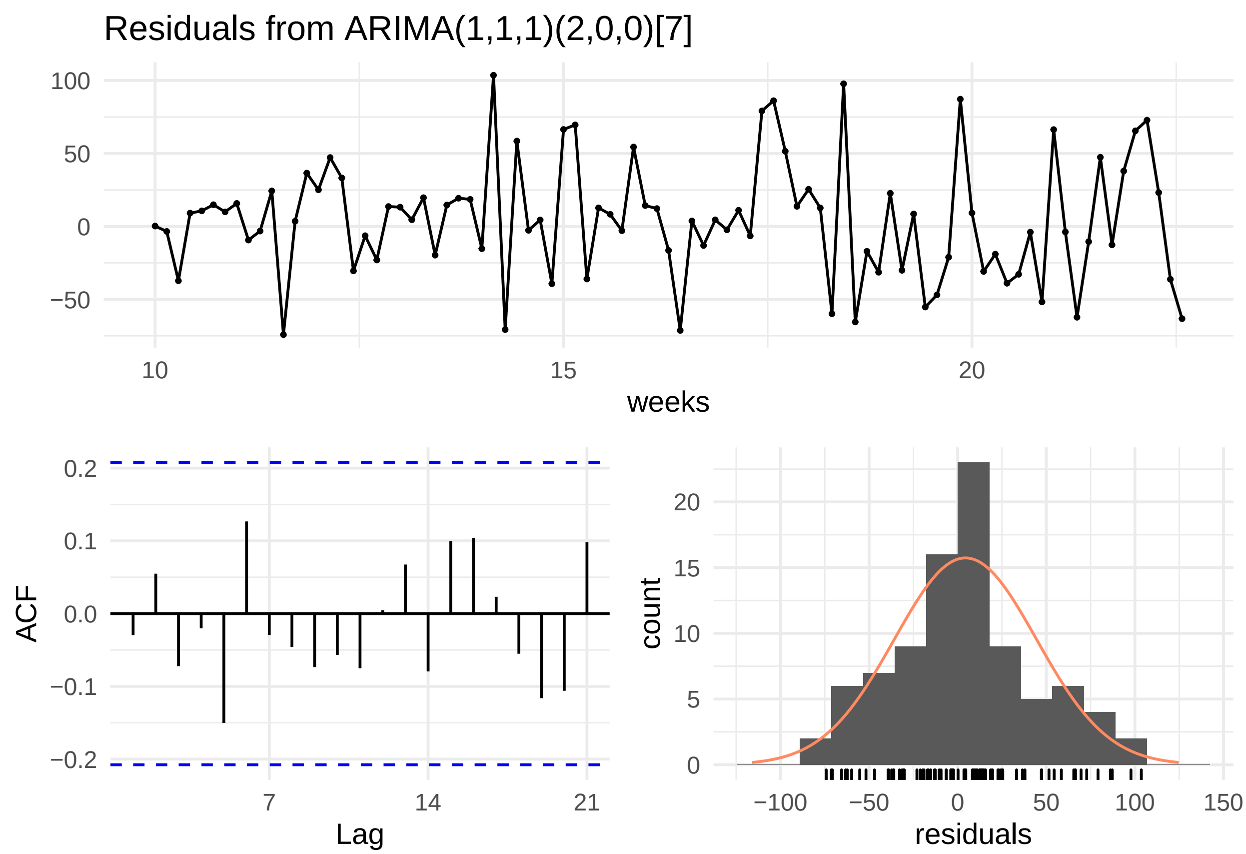



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(2,0,0)[7]
## Q* = 7.6411, df = 11, p-value = 0.745
##
## Model df: 3.   Total lags used: 14
```

```r
# residual analysis model 2
autoplot(lettuce.m2$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 1 Store (12631) - Residuals model 2 plot")
```


```
ggAcf(lettuce.m2$residuals) + theme_minimal() +
ggtitle("New York 1 Store (12631) - ACF residualts plot model 2")
```


```r
checkresiduals(lettuce.m2, xlab = "weeks", theme = theme_minimal())
```

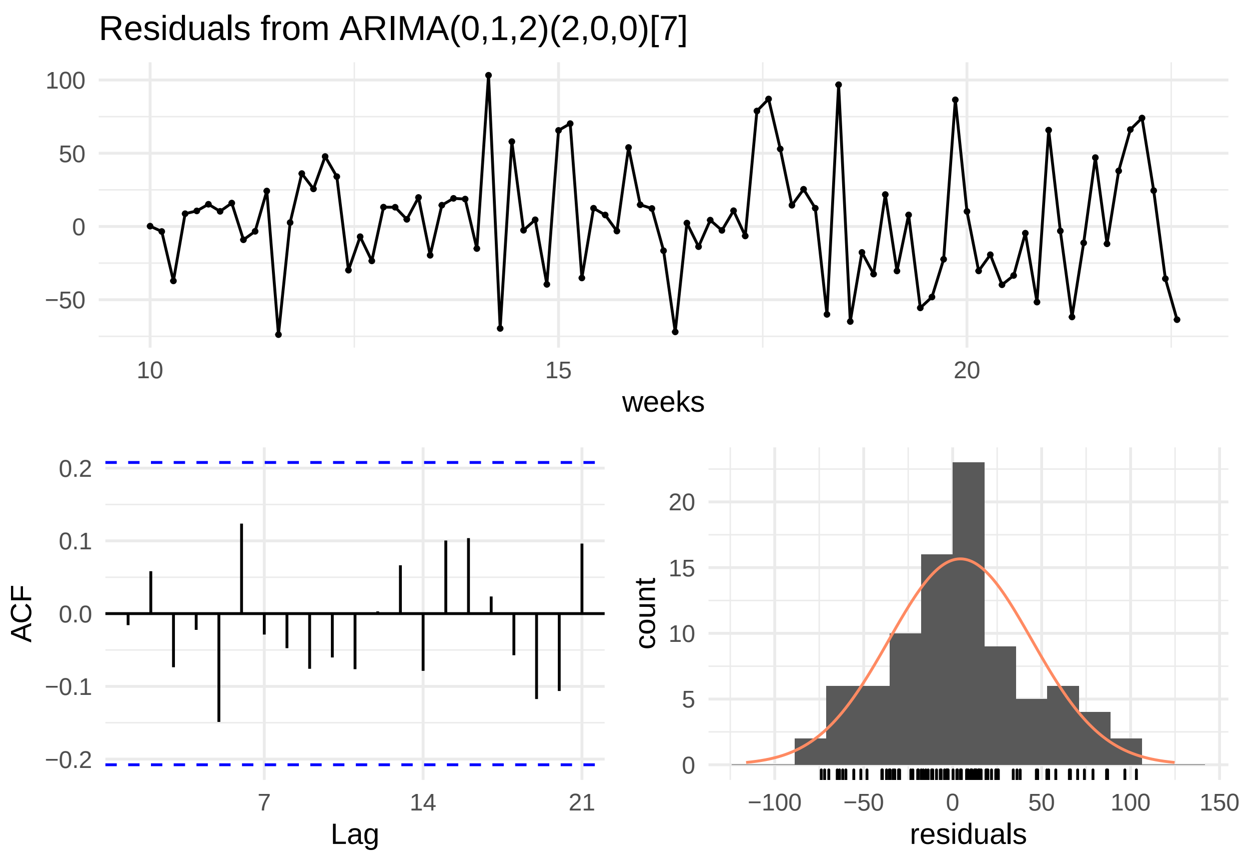



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(2,0,0)[7]
## Q* = 7.5634, df = 10, p-value = 0.6714
##
## Model df: 4.   Total lags used: 14
```

```
# residual analysis model 3
autoplot(lettuce.m3$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 1 Store (12631) - Residuals model 3 plot")
```


```
ggAcf(lettuce.m3$residuals) + theme_minimal() +
ggtitle("New York 1 Store (12631) - ACF residualts plot model 3")
```

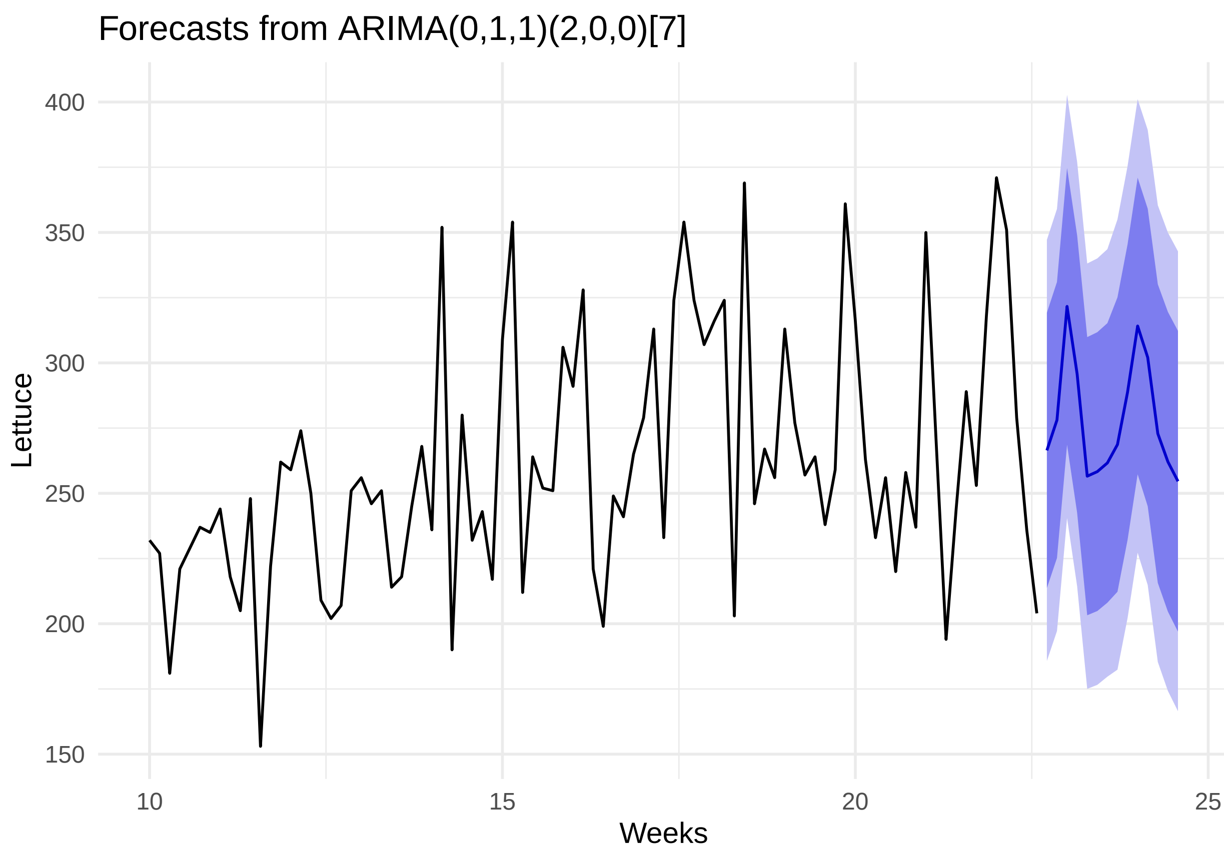

```
checkresiduals(lettuce.m3, xlab = "weeks", theme = theme_minimal())
```



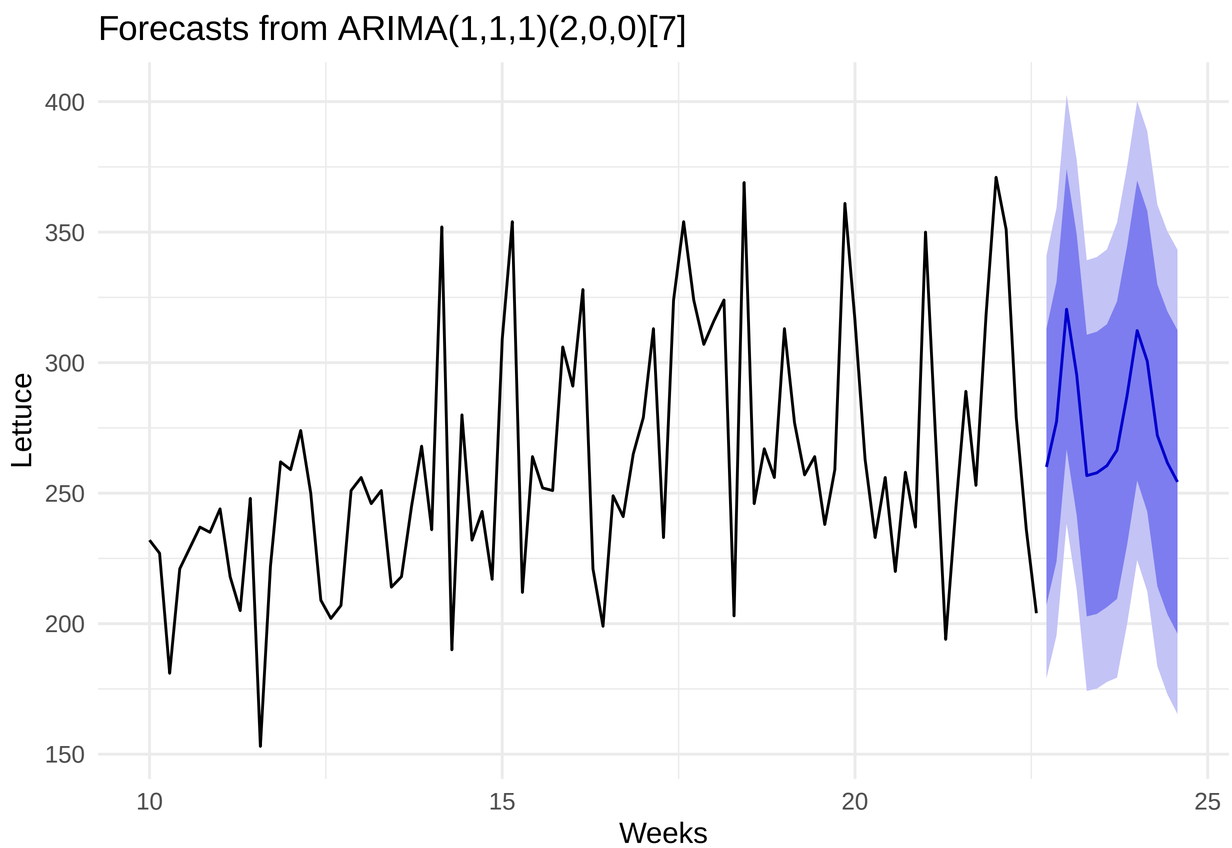

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)(2,0,0)[7]
## Q* = 7.5367, df = 10, p-value = 0.674
##
## Model df: 4.   Total lags used: 14
```

Now we continue with the forecasting part for the three candidate models:

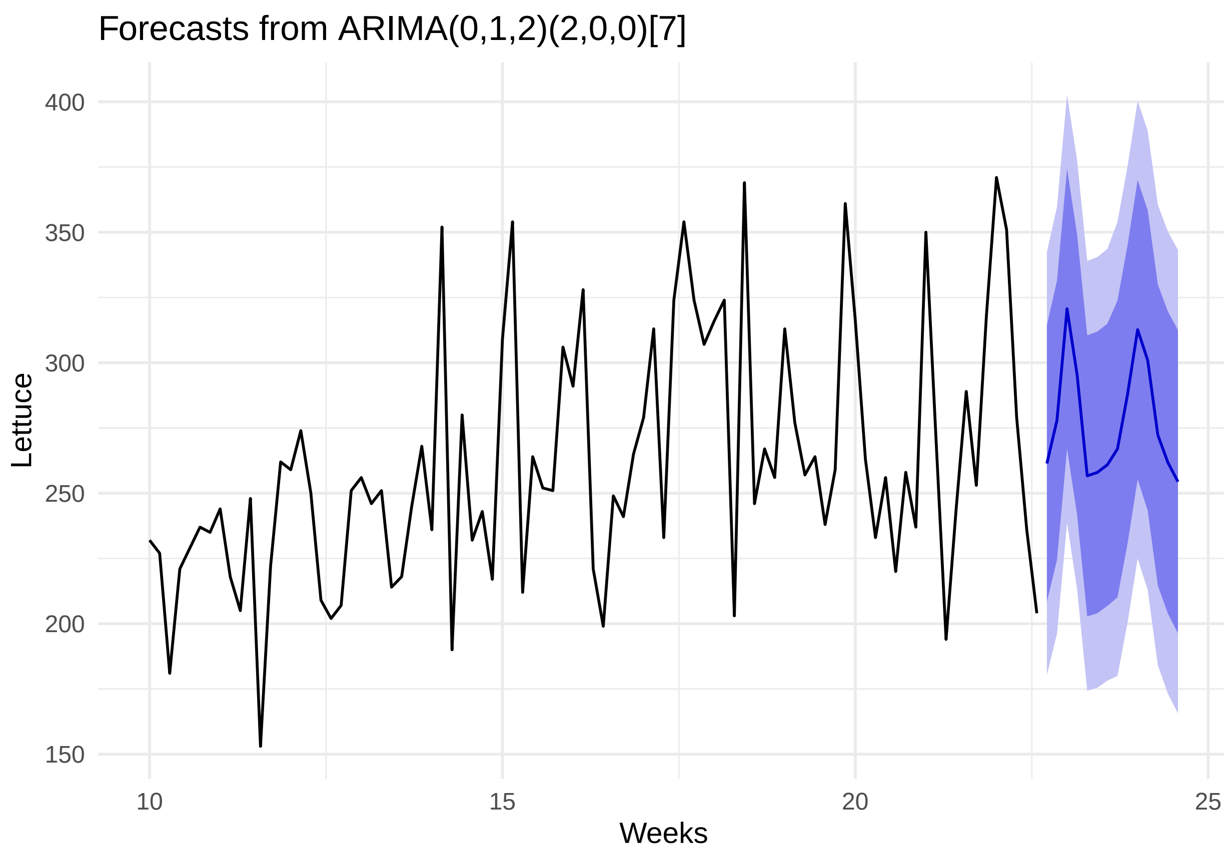
```
#Forecasting part model 1
lettuce.f1 <- forecast(lettuce.m1, h = 14)
autoplot(lettuce.f1, xlab = "Weeks", ylab = "Lettuce") + theme_minimal()
```




```r
#Forecasting part model 2
lettuce.f2 <- forecast(lettuce.m2, h = 14)
autoplot(lettuce.f2, xlab = "Weeks", ylab = "Lettuce") + theme_minimal()
```




```r
#Forecasting part model 3
lettuce.f3 <- forecast(lettuce.m3, h = 14)
autoplot(lettuce.f3, xlab = "Weeks", ylab = "Lettuce") + theme_minimal()
```



Now we need to test how our models performs for test set. Earlier observations are used for training, and more recent observations are used for testing. Suppose we use the first 89 days of data for training and the last 14 for test. Based on auto.arima(), we choose two candidate models with the lowest AICs.

```
### model evaluation
# Apply fitted model to later data
# Accuracy test for candidate model 1
accuracy.m1 <- accuracy(forecast(lettuce.m1, h = 14), lettuce_test)
accuracy.m1
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   4.103944 40.24048 31.06843 -0.5328407 11.90586 0.8402411
## Test set     -12.488880 41.26388 37.45403 -6.8059927 14.08000 1.0129389
##                     ACF1 Theil's U
## Training set 0.057668088        NA
## Test set     0.009455242 0.6520634
```

```
# Accuracy test for candidate model 2
accuracy.m2 <- accuracy(forecast(lettuce.m2, h = 14), lettuce_test)
accuracy.m2
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   4.36677 40.13025 30.81424 -0.4249124 11.82139 0.8333666
## Test set     -11.15152 41.18892 36.99544 -6.3085458 13.84815 1.0005363
##                      ACF1 Theil's U
```

```
## Training set -0.029581270        NA
## Test set      0.006889034 0.6535119
```

```r
# Accuracy test for candidate model 3
accuracy.m3 <- accuracy(forecast(lettuce.m3, h = 14), lettuce_test)
accuracy.m3
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   4.317327 40.14598 30.86092 -0.4448752 11.83567 0.8346291
## Test set     -11.480692 41.22657 37.12523 -6.4329773 13.91183 1.0040464
##                      ACF1 Theil's U
## Training set -0.015812288        NA
## Test set      0.006505855 0.6536673
```

Thus we pick the second model, since it performs better on the test set.

Now we train the second model on the whole date set as follows:

```r
# Training on both train and test set
lettuce.f.both <- Arima(lettuce, order = c(1, 1, 1),
                    seasonal = list(order = c(2, 0, 0), period = 7))
```

Lastly, we forecast lettuce demand for the next 2 weeks.

```r
# Forecast for next 14 days
lettuce.f.final <- forecast(lettuce.f.both, h = 14)
lettuce.f.final
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 24.71429       267.8324 214.6772 320.9876 186.5385 349.1263
## 24.85714       252.9936 199.3167 306.6706 170.9018 335.0855
## 25.00000       293.8467 240.0524 347.6410 211.5754 376.1180
## 25.14286       276.2834 222.3930 330.1737 193.8652 358.7015
## 25.28571       253.1750 199.1905 307.1595 170.6129 335.7372
## 25.42857       266.4303 212.3519 320.5086 183.7246 349.1360
## 25.57143       248.8072 194.6352 302.9793 165.9583 331.6562
## 25.71429       277.1748 221.2675 333.0822 191.6720 362.6777
## 25.85714       254.6642 198.5615 310.7669 168.8625 340.4659
## 26.00000       309.5395 253.3018 365.7773 223.5313 395.5478
## 26.14286       302.7129 246.3450 359.0808 216.5057 388.9202
## 26.28571       252.7133 196.2160 309.2106 166.3081 339.1185
## 26.42857       257.0218 200.3953 313.6482 170.4191 343.6244
## 26.57143       252.8329 196.0776 309.5881 166.0332 339.6325
```

We present our forecast through ARIMA(1,1,1)(2,0,0) model for each of the next 14 days.

```r
forecast_data <- as.data.frame(lettuce.f.final)
next2weeks <- data.frame(day = seq(1, 14))
final_forecast_NewYork1_arima <- cbind(next2weeks, forecast_data$`Point Forecast`)
final_forecast_NewYork1_arima
```

```
##    day forecast_data$`Point Forecast`
## 1    1                      267.8324
## 2    2                      252.9936
## 3    3                      293.8467
## 4    4                      276.2834
## 5    5                      253.1750
## 6    6                      266.4303
## 7    7                      248.8072
## 8    8                      277.1748
## 9    9                      254.6642
## 10  10                      309.5395
## 11  11                      302.7129
## 12  12                      252.7133
## 13  13                      257.0218
## 14  14                      252.8329
```

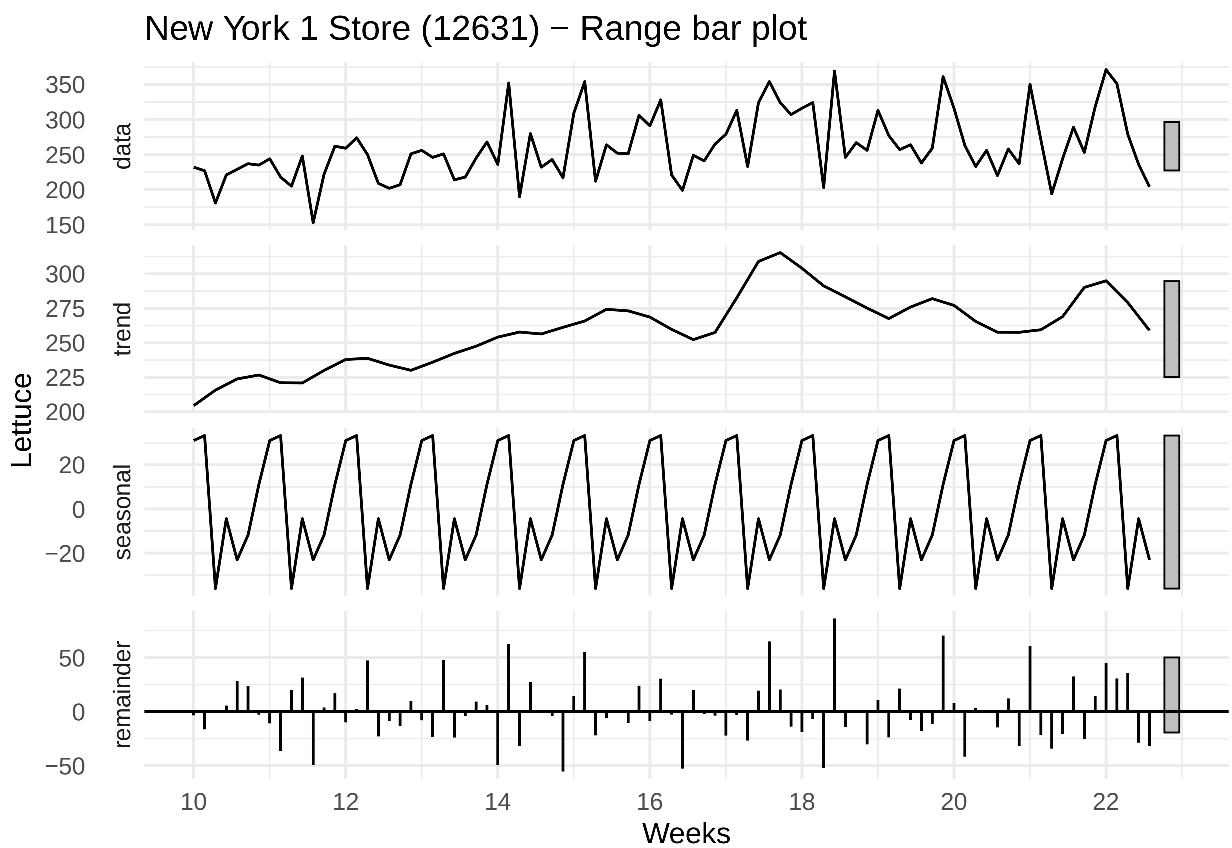
Holt-Winters

Now we will use another model to forecast lettuce demand. Our goal is to pick the model with the most accurate predictions.

We will forecast the lettuce demand for next two weeks using Holt-Winters model.

For time series analysis, the first step is always to visually inspect the time series. In this regard, the stl() function is quite useful. It decomposes the original time series into trend, seasonal factors, and random error terms. The relative importance of different components are indicated by the grey bars in the plots.

```
lettuce_train %>% stl(s.window = "period") %>%
autoplot(xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 1 Store (12631) - Range bar plot")
```



For this data set, the grey bar of the trend panel is larger than that on the orginal time series panel, which indicates that the contribution of the trend component to the variation in the original time series is marginal.

Moreover, the grey bar of the seasonal panel is large, even larger than the grey bar of random error term, which indicates that seasonal component contributes to a small proportion of variations in the time series. In other words, it indicates that there is no seasonality in the data.

With ets(), initial states and smoothing parameters are jointly estimated by maximizing the likelihood function. We need to specify the model in ets() using three letters. The way to approach this is: (1) check out time series plot, and see if there is any trend and seasonality; (2) run ets() with model = "ZZZ", and see whether the best model is consistent with your expectation; (3) if they are consistent, it gives us confidence that our model specification is correct; otherwise try to figure out why there is a discrepancy.

We now use ets function as previously indicated to find our best model:

```
# using ets
lettuce.ets2 <- ets(lettuce_train, model = "ZZZ")
lettuce.ets2
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = lettuce_train, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
```

```
##     phi   = 0.9753
##
##   Initial states:
##     l = 214.8392
##     b = 1.968
##     s = 1.0449 0.9662 0.9145 0.9813 0.853 1.1398
##            1.1003
##
##   sigma:  0.1389
##
##      AIC     AICc      BIC
## 1050.788 1055.641 1083.140
```

Our best model is the ETS(M,Ad,M).

```
# using ets
lettuce.ets <- ets(lettuce_train, model = "MAM", damped = TRUE, ic = 'aic')
lettuce.ets
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = lettuce_train, model = "MAM", damped = TRUE, ic = "aic")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.9753
##
##   Initial states:
##     l = 214.8392
##     b = 1.968
##     s = 1.0449 0.9662 0.9145 0.9813 0.853 1.1398
##            1.1003
##
##   sigma:  0.1389
##
##      AIC     AICc      BIC
## 1050.788 1055.641 1083.140
```

After estimation, we can use accuracy() function to determine in-sample fit and forecast() function to generate forecast.

Similarly with ARIMA model, we use AIC to determine our best model in terms of best in-sample performance.

```
# in-sample one-step forecast
accuracy(lettuce.ets)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -1.077097 34.11242 26.09866 -2.108286 10.08561 0.7058346
##                    ACF1
## Training set 0.06152982
```

We present the in-sample forecast part for the ets model as follows:

```
# best model
lettuce.ets.f <- forecast(lettuce.ets, h = 14)
lettuce.ets.f
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 22.71429       274.6589 225.7799 323.5380 199.9049 349.4130
## 22.85714       297.2379 244.3406 350.1352 216.3385 378.1373
## 23.00000       313.2411 257.4958 368.9863 227.9861 398.4961
## 23.14286       324.6814 266.9002 382.4627 236.3127 413.0502
## 23.28571       243.1538 199.8815 286.4261 176.9745 309.3331
## 23.42857       279.8874 230.0778 329.6969 203.7102 356.0645
## 23.57143       261.0075 214.5579 307.4571 189.9689 332.0461
## 23.71429       275.9276 226.8227 325.0325 200.8282 351.0270
## 23.85714       298.5760 245.4406 351.7115 217.3123 379.8397
## 24.00000       314.6155 258.6256 370.6055 228.9863 400.2447
## 24.14286       326.0700 268.0415 384.0984 237.3231 414.8168
## 24.28571       244.1674 200.7145 287.6202 177.7120 310.6227
## 24.42857       281.0245 231.0124 331.0366 204.5377 357.5114
## 24.57143       262.0411 215.4074 308.6749 190.7210 333.3613
```

After the forecast, we continue with the out of sample accuracy of our best model.

```
# Out of sample accuracy
# best model
accuracy.ets <- accuracy(lettuce.ets.f, lettuce_test)
accuracy.ets
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  -1.077097 34.11242 26.09866 -2.108286 10.08561 0.7058346
## Test set     -19.235015 44.30622 36.85583 -9.276175 14.25979 0.9967607
##                     ACF1 Theil's U
## Training set  0.061529822        NA
## Test set     -0.001611319 0.7245148
```

We now train our best model - ETS(M,Ad,M) on the whole data set as indicated below:

```
# final model
lettuce.ets <- ets(lettuce, model = "MAM", damped = TRUE, ic = 'aic')
lettuce.ets
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = lettuce, model = "MAM", damped = TRUE, ic = "aic")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 0.002
##     phi   = 0.9745
```

```
##
##   Initial states:
##     l = 213.5467
##     b = 1.9142
##     s = 1.028 0.9728 0.9128 0.9858 0.8681 1.1111
##            1.1214
##
##   sigma:  0.1408
##
##      AIC     AICc      BIC
## 1232.106 1236.196 1266.357
```

We now present the out-of-sample forecast for the next 14 days (2 weeks) as seen below:

```
lettuce.ets.f <- forecast(lettuce.ets, h = 14)
lettuce.ets.f
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 24.71429       274.2683 224.7805 323.7561 198.5832 349.9534
## 24.85714       289.9449 237.6285 342.2614 209.9338 369.9560
## 25.00000       316.5410 259.4257 373.6564 229.1907 403.8914
## 25.14286       313.7897 257.1708 370.4086 227.1986 400.3808
## 25.28571       245.1401 200.9080 289.3721 177.4930 312.7872
## 25.42857       278.4584 228.2145 328.7023 201.6170 355.2998
## 25.57143       257.8716 211.3423 304.4009 186.7112 329.0320
## 25.71429       275.0112 225.3891 324.6332 199.1207 350.9016
## 25.85714       290.7099 238.2552 343.1646 210.4873 370.9325
## 26.00000       317.3545 260.0921 374.6169 229.7792 404.9298
## 26.14286       314.5752 257.8143 371.3362 227.7668 401.3836
## 26.28571       245.7378 201.3976 290.0780 177.9254 313.5503
## 26.42857       279.1198 228.7563 329.4834 202.0954 356.1442
## 26.57143       258.4683 211.8310 305.1055 187.1427 329.7938
```

We present our forecast for each of the next 14 days.

```
forecast_data <- as.data.frame(lettuce.ets.f)
next2weeks <- data.frame(day = seq(1, 14))
final_forecast_NewYork1_ets <- cbind(next2weeks, forecast_data$`Point Forecast`)
final_forecast_NewYork1_ets
```

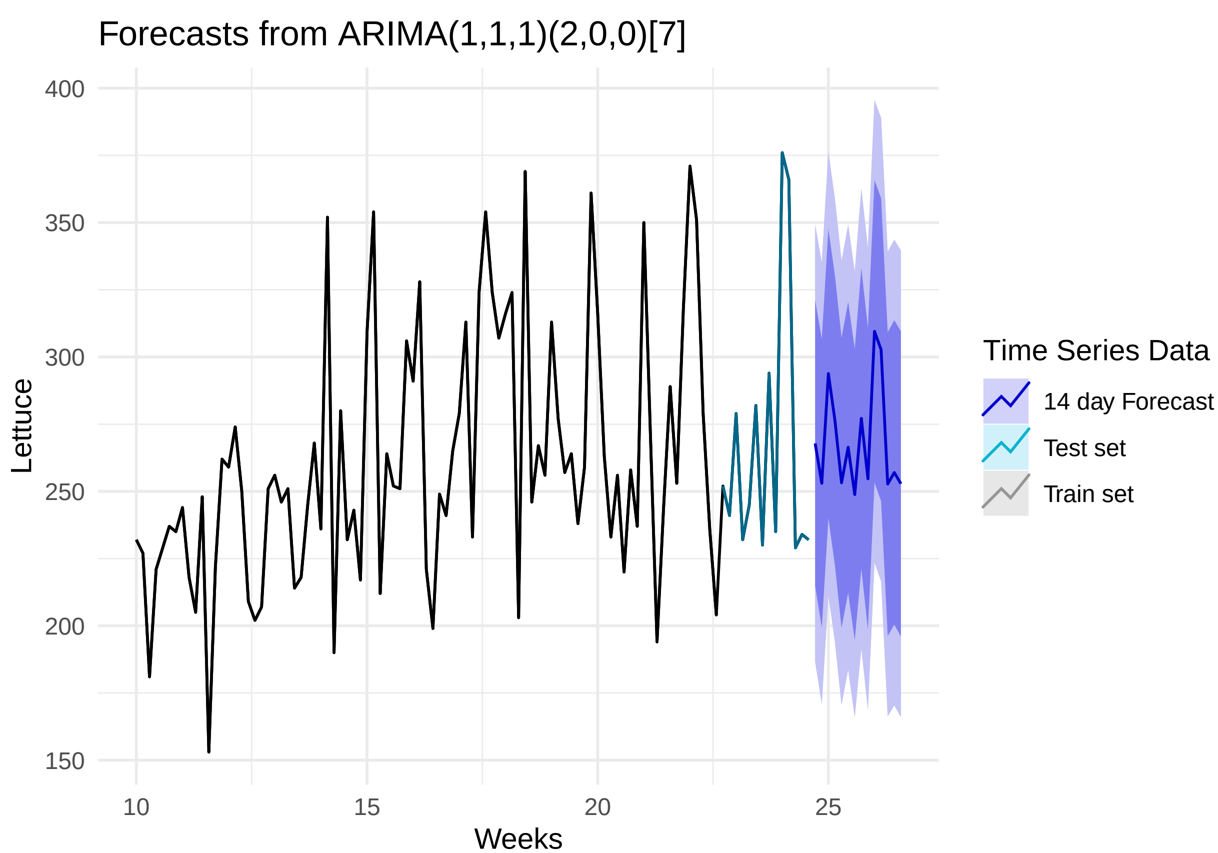
```
##    day forecast_data$`Point Forecast`
## 1    1                       274.2683
## 2    2                       289.9449
## 3    3                       316.5410
## 4    4                       313.7897
## 5    5                       245.1401
## 6    6                       278.4584
## 7    7                       257.8716
## 8    8                       275.0112
## 9    9                       290.7099
## 10  10                       317.3545
## 11  11                       314.5752
## 12  12                       245.7378
## 13  13                       279.1198
## 14  14                       258.4683
```

Comparison

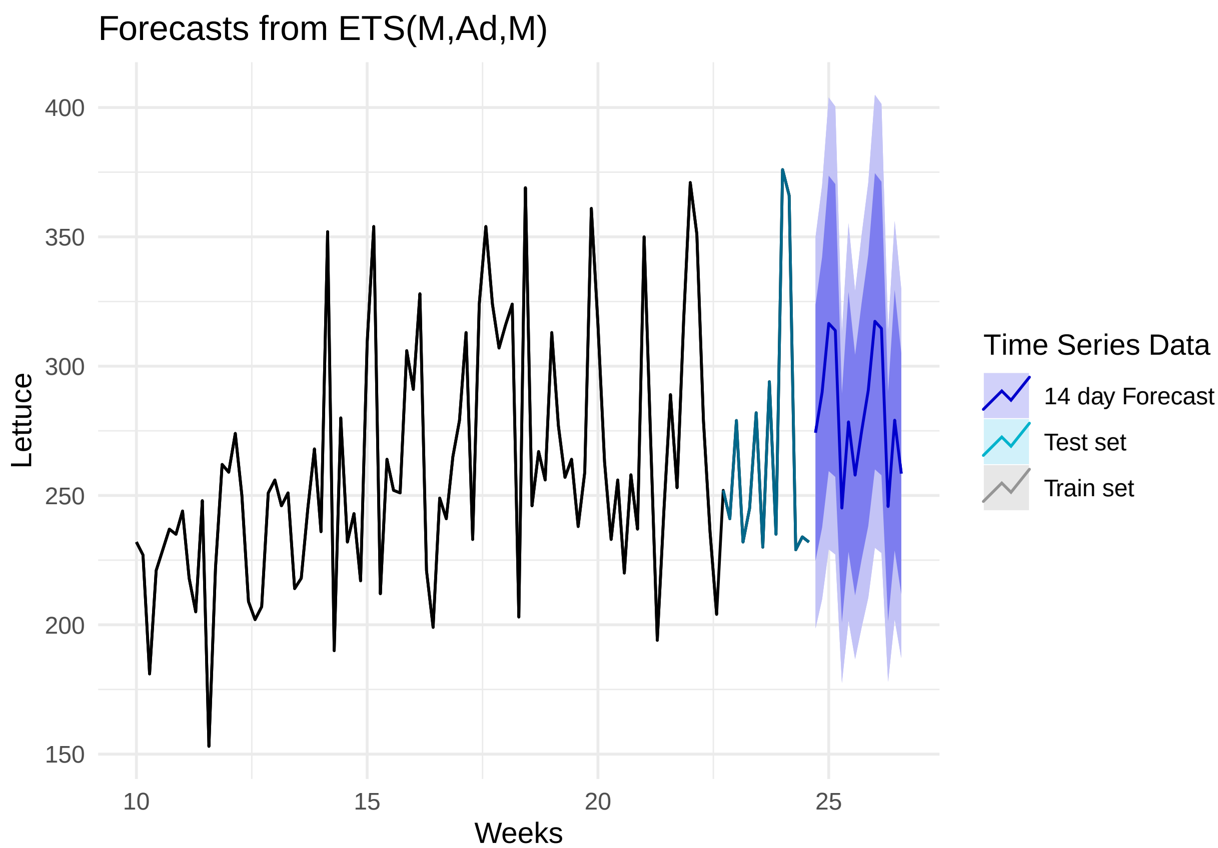
Now we will compare the two best models for New York 1 Store (12631).

We plot time series data for train and test set and also the forecasts from our two models as indicated below:

```
colours <- c("blue", "deepskyblue4", "black")
autoplot(lettuce.f.final, xlab = "Weeks", ylab = "Lettuce") +
  autolayer(lettuce_train, series = "Train set") +
  autolayer(lettuce_test, series = "Test set") +
  autolayer(lettuce.f.final, series = "14 day Forecast") +
  guides(colour = guide_legend(title = "Time Series Data")) +
  scale_colour_manual(values = colours) + theme_minimal()
```




```
autoplot(lettuce.ets.f, xlab = "Weeks", ylab = "Lettuce") +
  autolayer(lettuce_train, series = "Train set") +
  autolayer(lettuce_test, series = "Test set") +
  autolayer(lettuce.ets.f, series = "14 day Forecast") +
  guides(colour = guide_legend(title = "Time Series Data")) +
  scale_colour_manual(values = colours) + theme_minimal()
```



In order to decide which of the two models ARIMA(1,1,1)(2,0,0) or ETS(M,Ad,M) to choose, we will check their RMSE in the test set.

```
# best ets model
# ETS(M,Ad,M)
accuracy.ets
```

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  -1.077097 34.11242 26.09866 -2.108286 10.08561 0.7058346
## Test set     -19.235015 44.30622 36.85583 -9.276175 14.25979 0.9967607
##                      ACF1 Theil's U
## Training set  0.061529822        NA
## Test set     -0.001611319 0.7245148
```

```
# best arima model
# ARIMA(1,1,1)(2,0,0)
accuracy.m2
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   4.36677 40.13025 30.81424 -0.4249124 11.82139 0.8333666
## Test set     -11.15152 41.18892 36.99544 -6.3085458 13.84815 1.0005363
##                      ACF1 Theil's U
## Training set -0.029581270        NA
## Test set      0.006889034 0.6535119
```

We can observe that ARIMA(1,1,1)(2,0,0) has a better (lower) RMSE (41.18892 vs 44.30622) respectively.

Therefore, we choose the ARIMA(1,1,1)(2,0,0) for New York 1 (12631) store.

Hence, our forecast for lettuce demand of next 2 weeks for that store is the following:

```
final_forecast_NewYork1_arima
```

```
##    day forecast_data$`Point Forecast`
## 1    1                       267.8324
## 2    2                       252.9936
## 3    3                       293.8467
## 4    4                       276.2834
## 5    5                       253.1750
## 6    6                       266.4303
## 7    7                       248.8072
## 8    8                       277.1748
## 9    9                       254.6642
## 10  10                       309.5395
## 11  11                       302.7129
## 12  12                       252.7133
## 13  13                       257.0218
## 14  14                       252.8329
```