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 20 March 2015 to 15 June 2015. Each observation includes two values: day pair, and sales in that particular day.

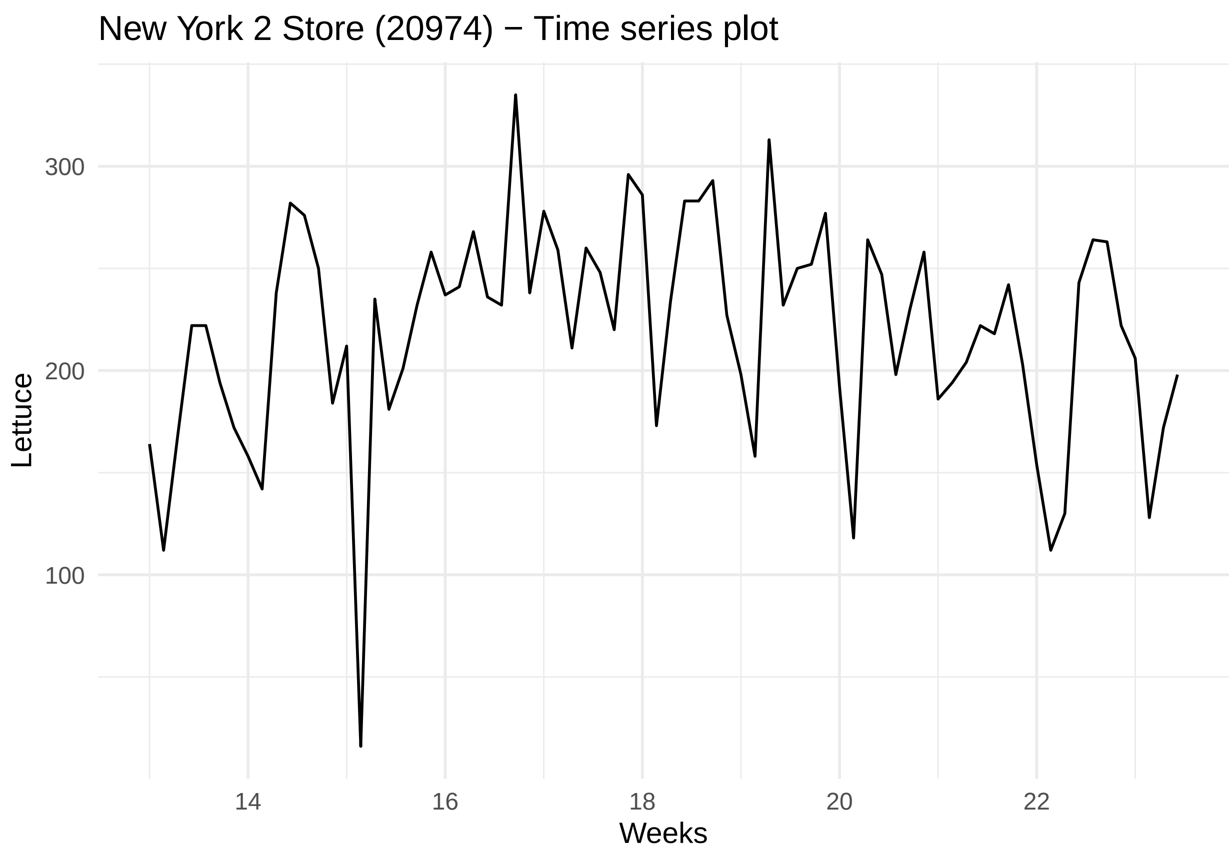
Then, we load and split the data set into train and test set.

```
# read csv file
data <- read.csv(file = "NewYork2_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(12, 2)) # 12th week 2nd day

# split data set into train and test set
lettuce_train <- subset(lettuce, end = 80, start = 7) # ignore first 6 days due to strange data
lettuce_test <- subset(lettuce, start = 81) # last 14 lines-days
```

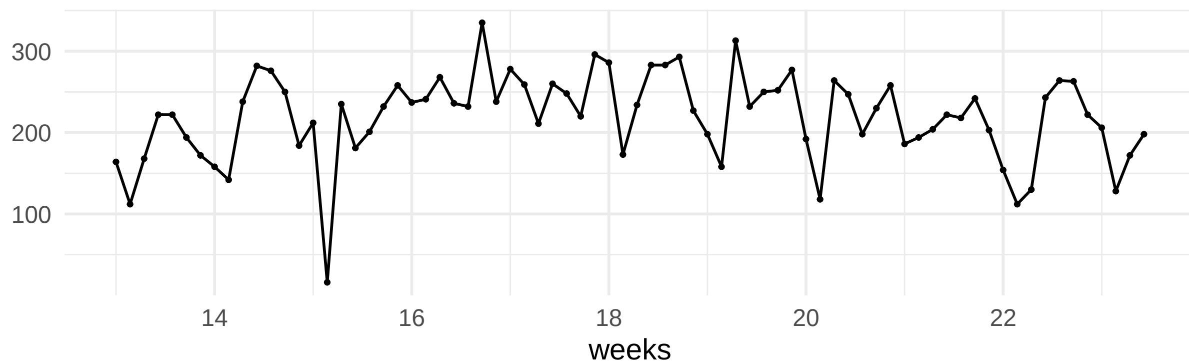
ARIMA

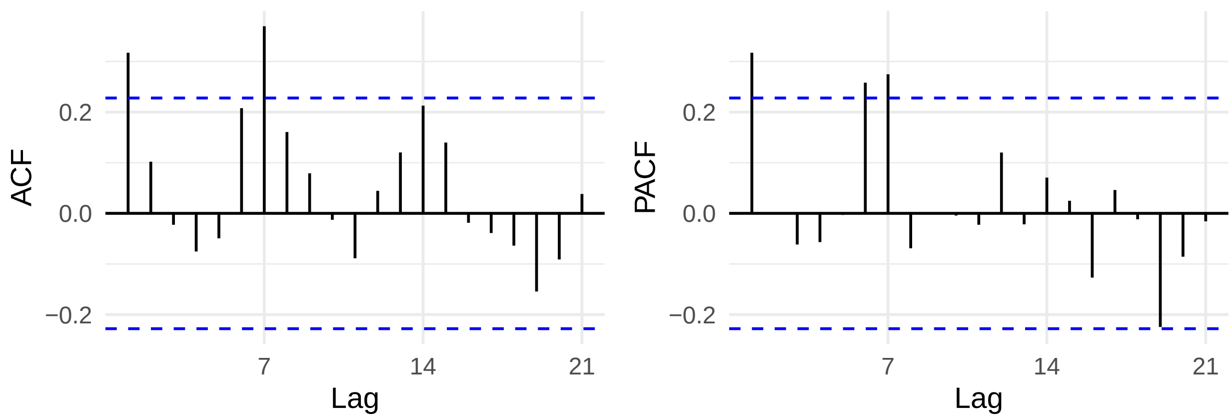
We visually inspect the time series.

```
autoplot(lettuce_train, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 2 Store (20974) - Time series plot")
```




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





We plot the time series, and observe that there is no seasonality and appears to be stationary. We run ADF, PP and KPSS tests to formally test the stationarity of time series and all suggest that the time series is stationary.

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  lettuce_train
## Dickey-Fuller = -3.5721, Lag order = 4, p-value = 0.04182
## 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) = -48.35, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

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

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

```
##
##  KPSS Test for Level Stationarity
##
## data:  lettuce_train
## KPSS Level = 0.25367, Truncation lag parameter = 3, p-value = 0.1
```

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] 0
```

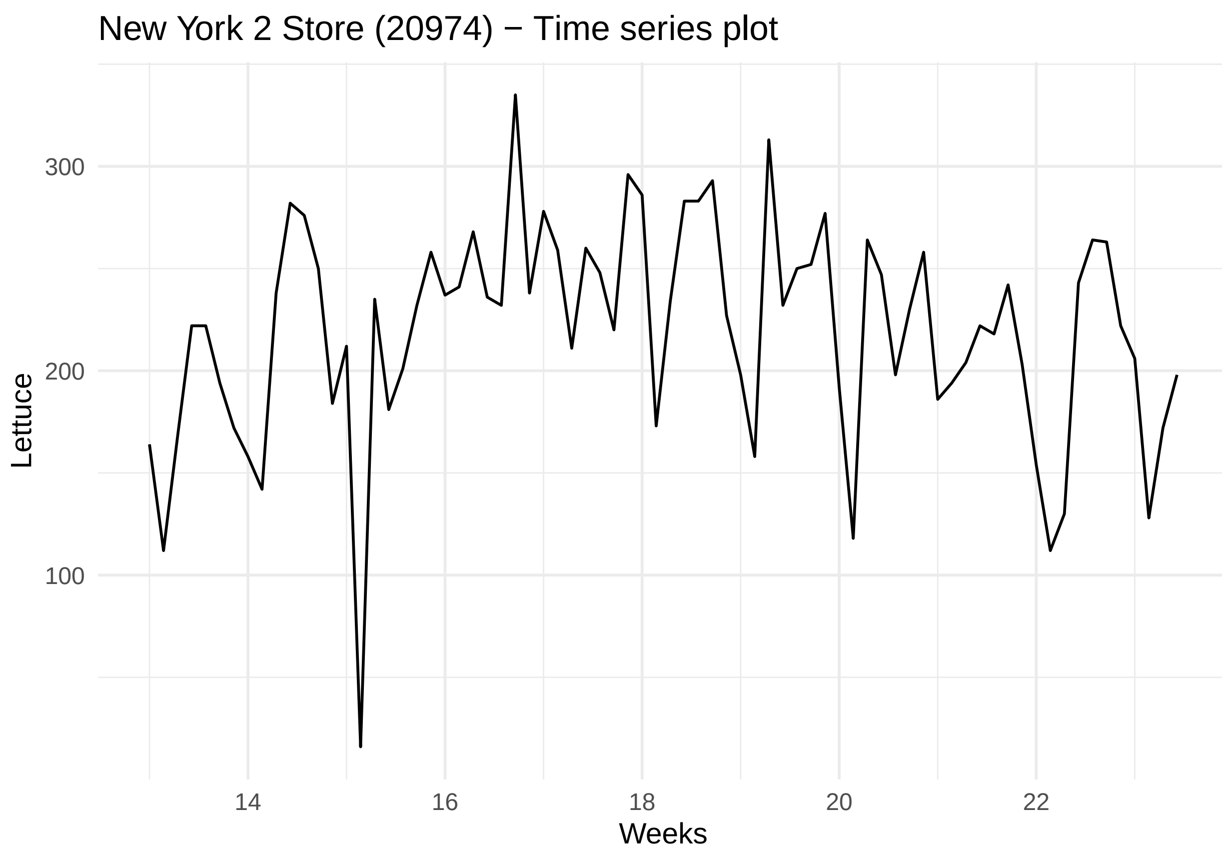
```r
# seasonal stationarity
nsdiffs(lettuce_train)
```

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

We do not need to differentiate.

We plot again the time series to see if stationary.

```r
autoplot(lettuce_train, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 2 Store (20974) - Time series plot")
```



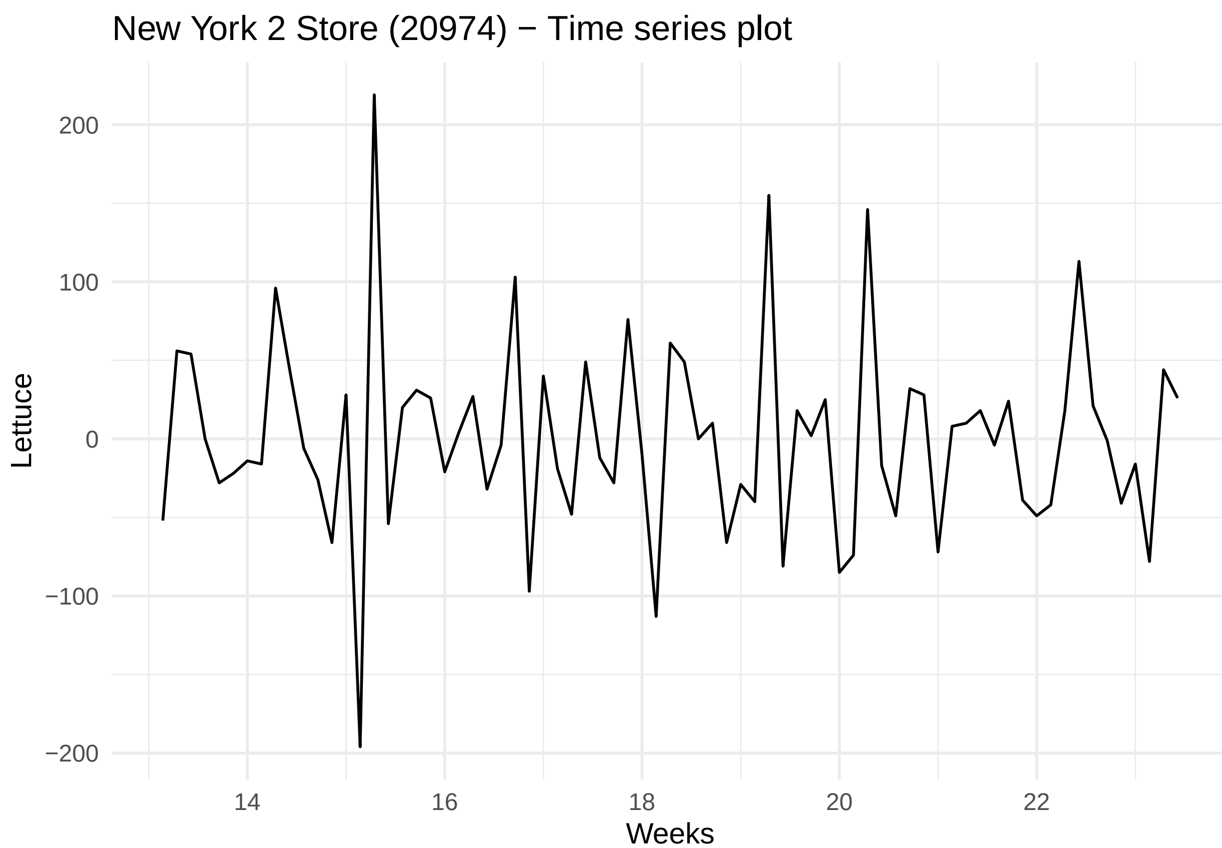
We can easily observe a concave pattern. This forms a trend.

As a result, we have to differentiate as follows:

```
lettuce_train.diff1 <- diff(lettuce_train, differences = 1)
```

We plot again the time series to see if stationary.

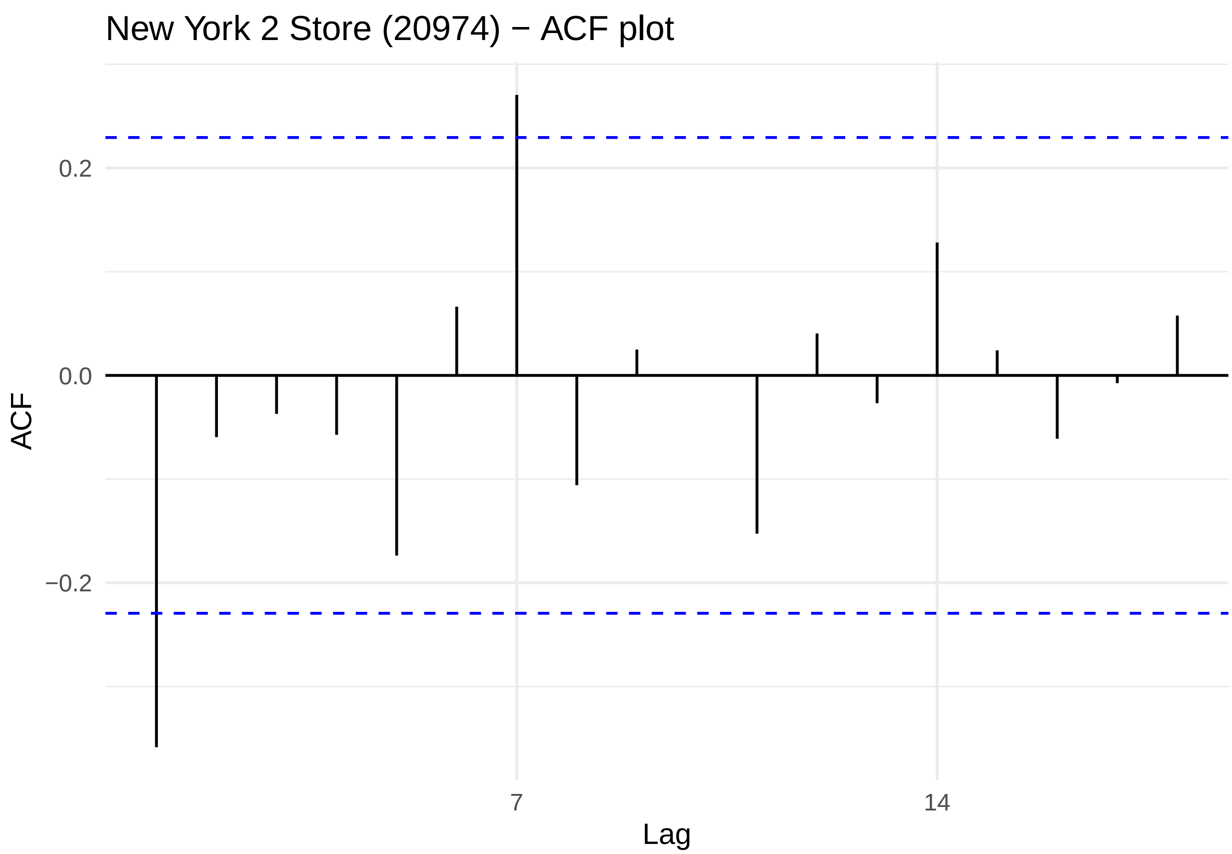
```
autoplot(lettuce_train.diff1, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 2 Store (20974) - Time series plot")
```



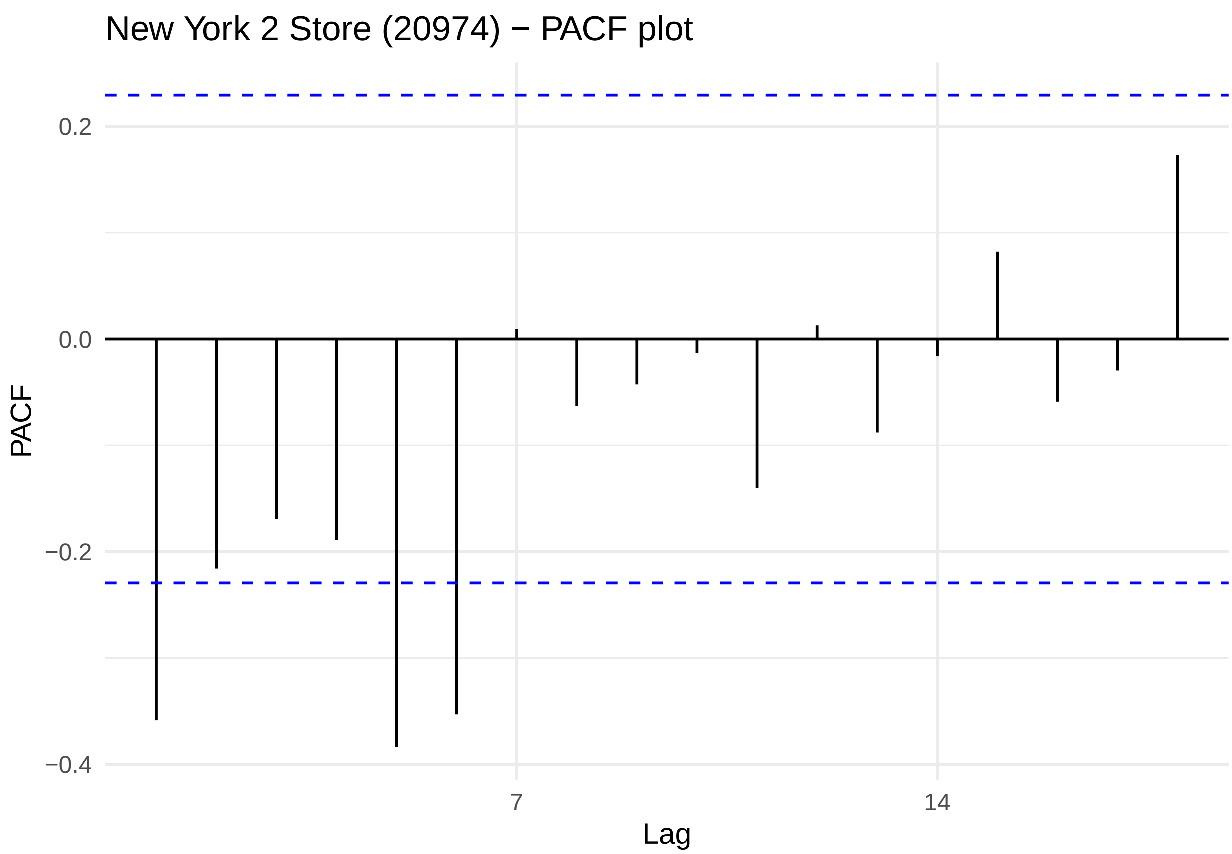
Looks stationary.

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 2 Store (20974) - ACF plot")
```




```r
# pacf plot
ggPacf(lettuce_train.diff1) + theme_minimal() + ggtitle("New York 2 Store (20974) - 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 only first differenting.

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

```
##
##  ARIMA(0,1,0)                           : 813.2385
##  ARIMA(0,1,0)           with drift      : 815.2344
##  ARIMA(0,1,0)(0,0,1)[7]                 : 810.6353
##  ARIMA(0,1,0)(0,0,1)[7] with drift      : 812.6345
##  ARIMA(0,1,0)(0,0,2)[7]                 : 811.7488
##  ARIMA(0,1,0)(0,0,2)[7] with drift      : 813.7482
##  ARIMA(0,1,0)(1,0,0)[7]                 : 809.7859
##  ARIMA(0,1,0)(1,0,0)[7] with drift      : 811.7854
##  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]                 : 811.3921
##  ARIMA(0,1,0)(2,0,0)[7] with drift      : 813.3916
##  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)                              : 790.9536
##  ARIMA(0,1,1)           with drift         : 792.9134
##  ARIMA(0,1,1)(0,0,1)[7]                    : 786.5134
##  ARIMA(0,1,1)(0,0,1)[7] with drift         : 788.4282
##  ARIMA(0,1,1)(0,0,2)[7]                    : 785.0547
##  ARIMA(0,1,1)(0,0,2)[7] with drift         : 787.003
##  ARIMA(0,1,1)(1,0,0)[7]                    : 784.0596
##  ARIMA(0,1,1)(1,0,0)[7] with drift         : 785.9729
##  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]                    : 784.0994
##  ARIMA(0,1,1)(2,0,0)[7] with drift         : 786.0462
##  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)                              : 789.8214
##  ARIMA(0,1,2)           with drift         : 791.7907
##  ARIMA(0,1,2)(0,0,1)[7]                    : 785.9105
##  ARIMA(0,1,2)(0,0,1)[7] with drift         : 787.8476
##  ARIMA(0,1,2)(0,0,2)[7]                    : 785.0438
##  ARIMA(0,1,2)(0,0,2)[7] with drift         : 787.0003
##  ARIMA(0,1,2)(1,0,0)[7]                    : 783.8495
##  ARIMA(0,1,2)(1,0,0)[7] with drift         : 785.7834
##  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]                    : 784.3942
##  ARIMA(0,1,2)(2,0,0)[7] with drift         : 786.3473
##  ARIMA(0,1,2)(2,0,1)[7]                    : Inf
##  ARIMA(0,1,2)(2,0,1)[7] with drift         : Inf
##  ARIMA(0,1,3)                              : 791.6816
##  ARIMA(0,1,3)           with drift         : 793.6516
##  ARIMA(0,1,3)(0,0,1)[7]                    : 787.4216
##  ARIMA(0,1,3)(0,0,1)[7] with drift         : 789.3617
##  ARIMA(0,1,3)(0,0,2)[7]                    : 786.6606
##  ARIMA(0,1,3)(0,0,2)[7] with drift         : 788.6181
##  ARIMA(0,1,3)(1,0,0)[7]                    : 785.2878
##  ARIMA(0,1,3)(1,0,0)[7] with drift         : 787.2256
##  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]                    : 786.0388
##  ARIMA(0,1,3)(2,0,0)[7] with drift         : 787.9935
##  ARIMA(0,1,4)                              : 792.8385
##  ARIMA(0,1,4)           with drift         : 794.7918
##  ARIMA(0,1,4)(0,0,1)[7]                    : 789.4088
##  ARIMA(0,1,4)(0,0,1)[7] with drift         : 791.3501
##  ARIMA(0,1,4)(1,0,0)[7]                    : Inf
```

```
## ARIMA(0,1,4)(1,0,0)[7] with drift         : Inf
## ARIMA(0,1,5)                               : 793.5211
## ARIMA(0,1,5)            with drift         : 795.4626
## ARIMA(1,1,0)                               : 805.2092
## ARIMA(1,1,0)            with drift         : 807.1968
## ARIMA(1,1,0)(0,0,1)[7]                     : 799.9844
## ARIMA(1,1,0)(0,0,1)[7] with drift         : 801.9769
## ARIMA(1,1,0)(0,0,2)[7]                     : 800.4492
## ARIMA(1,1,0)(0,0,2)[7] with drift         : 802.4452
## ARIMA(1,1,0)(1,0,0)[7]                     : 798.2373
## ARIMA(1,1,0)(1,0,0)[7] with drift         : 800.2332
## 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]                     : 799.6011
## ARIMA(1,1,0)(2,0,0)[7] with drift         : 801.5986
## 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)                               : 789.8301
## ARIMA(1,1,1)            with drift         : 791.7999
## ARIMA(1,1,1)(0,0,1)[7]                     : 785.605
## ARIMA(1,1,1)(0,0,1)[7] with drift         : 787.5462
## ARIMA(1,1,1)(0,0,2)[7]                     : 784.7885
## ARIMA(1,1,1)(0,0,2)[7] with drift         : 786.747
## ARIMA(1,1,1)(1,0,0)[7]                     : 783.5071
## ARIMA(1,1,1)(1,0,0)[7] with drift         : 785.4459
## 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]                     : 784.1796
## ARIMA(1,1,1)(2,0,0)[7] with drift         : 786.1346
## ARIMA(1,1,1)(2,0,1)[7]                     : Inf
## ARIMA(1,1,1)(2,0,1)[7] with drift         : Inf
## ARIMA(1,1,2)                               : 791.7503
## ARIMA(1,1,2)            with drift         : 793.7202
## ARIMA(1,1,2)(0,0,1)[7]                     : 787.5884
## ARIMA(1,1,2)(0,0,1)[7] with drift         : 789.5299
## ARIMA(1,1,2)(0,0,2)[7]                     : Inf
## ARIMA(1,1,2)(0,0,2)[7] with drift         : Inf
## ARIMA(1,1,2)(1,0,0)[7]                     : Inf
## 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]                     : 786.1127
## ARIMA(1,1,2)(2,0,0)[7] with drift         : 788.0688
## ARIMA(1,1,3)                               : 793.6015
## ARIMA(1,1,3)            with drift         : 795.571
## ARIMA(1,1,3)(0,0,1)[7]                     : 787.7978
## ARIMA(1,1,3)(0,0,1)[7] with drift         : 789.7134
## ARIMA(1,1,3)(1,0,0)[7]                     : Inf
```

```
## ARIMA(1,1,3)(1,0,0)[7] with drift         : Inf
## ARIMA(1,1,4)                               : 790.5554
## ARIMA(1,1,4)           with drift          : 792.5279
## ARIMA(2,1,0)                               : 803.7339
## ARIMA(2,1,0)           with drift          : 805.7225
## ARIMA(2,1,0)(0,0,1)[7]                     : 798.5767
## ARIMA(2,1,0)(0,0,1)[7] with drift          : 800.5664
## ARIMA(2,1,0)(0,0,2)[7]                     : 798.2416
## ARIMA(2,1,0)(0,0,2)[7] with drift          : 800.2372
## ARIMA(2,1,0)(1,0,0)[7]                     : 796.352
## ARIMA(2,1,0)(1,0,0)[7] with drift          : 798.3457
## 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]                     : 797.2045
## ARIMA(2,1,0)(2,0,0)[7] with drift          : 799.2015
## ARIMA(2,1,0)(2,0,1)[7]                     : Inf
## ARIMA(2,1,0)(2,0,1)[7] with drift          : Inf
## ARIMA(2,1,1)                               : 791.6627
## ARIMA(2,1,1)           with drift          : 793.6323
## ARIMA(2,1,1)(0,0,1)[7]                     : 787.573
## ARIMA(2,1,1)(0,0,1)[7] with drift          : 789.5143
## ARIMA(2,1,1)(0,0,2)[7]                     : Inf
## ARIMA(2,1,1)(0,0,2)[7] with drift          : Inf
## ARIMA(2,1,1)(1,0,0)[7]                     : 785.3313
## ARIMA(2,1,1)(1,0,0)[7] with drift          : 787.2705
## 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]                     : 786.0649
## ARIMA(2,1,1)(2,0,0)[7] with drift          : Inf
## ARIMA(2,1,2)                               : 788.3468
## ARIMA(2,1,2)           with drift          : 790.3279
## 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]                     : Inf
## ARIMA(2,1,2)(1,0,0)[7] with drift          : Inf
## ARIMA(2,1,3)                               : Inf
## ARIMA(2,1,3)           with drift          : Inf
## ARIMA(3,1,0)                               : 803.7635
## ARIMA(3,1,0)           with drift          : 805.7533
## ARIMA(3,1,0)(0,0,1)[7]                     : 798.5321
## ARIMA(3,1,0)(0,0,1)[7] with drift          : 800.5183
## ARIMA(3,1,0)(0,0,2)[7]                     : 798.7036
## ARIMA(3,1,0)(0,0,2)[7] with drift          : 800.6969
## ARIMA(3,1,0)(1,0,0)[7]                     : 796.5931
## ARIMA(3,1,0)(1,0,0)[7] with drift          : 798.5837
## 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]                     : 797.7101
## ARIMA(3,1,0)(2,0,0)[7] with drift          : 799.7051
## ARIMA(3,1,1)                               : 792.3296
## ARIMA(3,1,1)           with drift          : 794.2906
## ARIMA(3,1,1)(0,0,1)[7]                     : 789.0712
```

```
##  ARIMA(3,1,1)(0,0,1)[7] with drift         : 791.0056
##  ARIMA(3,1,1)(1,0,0)[7]                    : 787.1708
##  ARIMA(3,1,1)(1,0,0)[7] with drift         : Inf
##  ARIMA(3,1,2)                              : 790.0768
##  ARIMA(3,1,2)           with drift         : 792.0609
##  ARIMA(4,1,0)                              : 803.2391
##  ARIMA(4,1,0)           with drift         : 805.2291
##  ARIMA(4,1,0)(0,0,1)[7]                    : 799.4988
##  ARIMA(4,1,0)(0,0,1)[7] with drift         : 801.4827
##  ARIMA(4,1,0)(1,0,0)[7]                    : 797.7698
##  ARIMA(4,1,0)(1,0,0)[7] with drift         : 799.7577
##  ARIMA(4,1,1)                              : 791.444
##  ARIMA(4,1,1)           with drift         : 793.3827
##  ARIMA(5,1,0)                              : 793.6175
##  ARIMA(5,1,0)           with drift         : 795.5889
##
##
##
##  Best model: ARIMA(1,1,1)(1,0,0)[7]
```

```
## Series: lettuce_train
## ARIMA(1,1,1)(1,0,0)[7]
##
## Coefficients:
##          ar1      ma1    sar1
##       0.2115  -0.9259  0.3408
## s.e.  0.1354   0.0693  0.1148
##
## sigma^2 estimated as 2442:  log likelihood=-387.75
## AIC=783.51   AICc=784.1   BIC=792.67
```

```
# Best model: ARIMA(1,1,1)(1,0,0)[7]  (AIC=783.51)
# Second best:  ARIMA(0,1,2)(1,0,0)[7]  (AIC=783.85)
# Third best: ARIMA(0,1,1)(1,0,0)[7] (AIC=784.06)
```

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(1,1,1)(1,0,0)[7] with AIC=783.51, ARIMA(0,1,2)(1,0,0)[7] with AIC=783.85 AND ARIMA(0,1,1)(1,0,0)[7] with AIC=784.06, as the candidate models that we would like to evaluate further.

```
# three candidate models
lettuce.m1 <- Arima(lettuce_train, order = c(1, 1, 1),
                        seasonal = list(order = c(1, 0, 0), period = 7))
lettuce.m2 <- Arima(lettuce_train, order = c(0, 1, 2),
                        seasonal = list(order = c(1, 0, 0), period = 7))
lettuce.m3 <- Arima(lettuce_train, order = c(0, 1, 1),
                        seasonal = list(order = c(1, 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        ACF1
## Training set 3.111878 48.05772 37.70491 -16.0591 31.88792 0.8042754 -0.02117438
```

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

```
##                    ME     RMSE    MAE       MPE     MAPE      MASE        ACF1
## Training set 2.848727 48.17716 37.764 -16.37333 32.14221 0.8055358 0.002492967
```

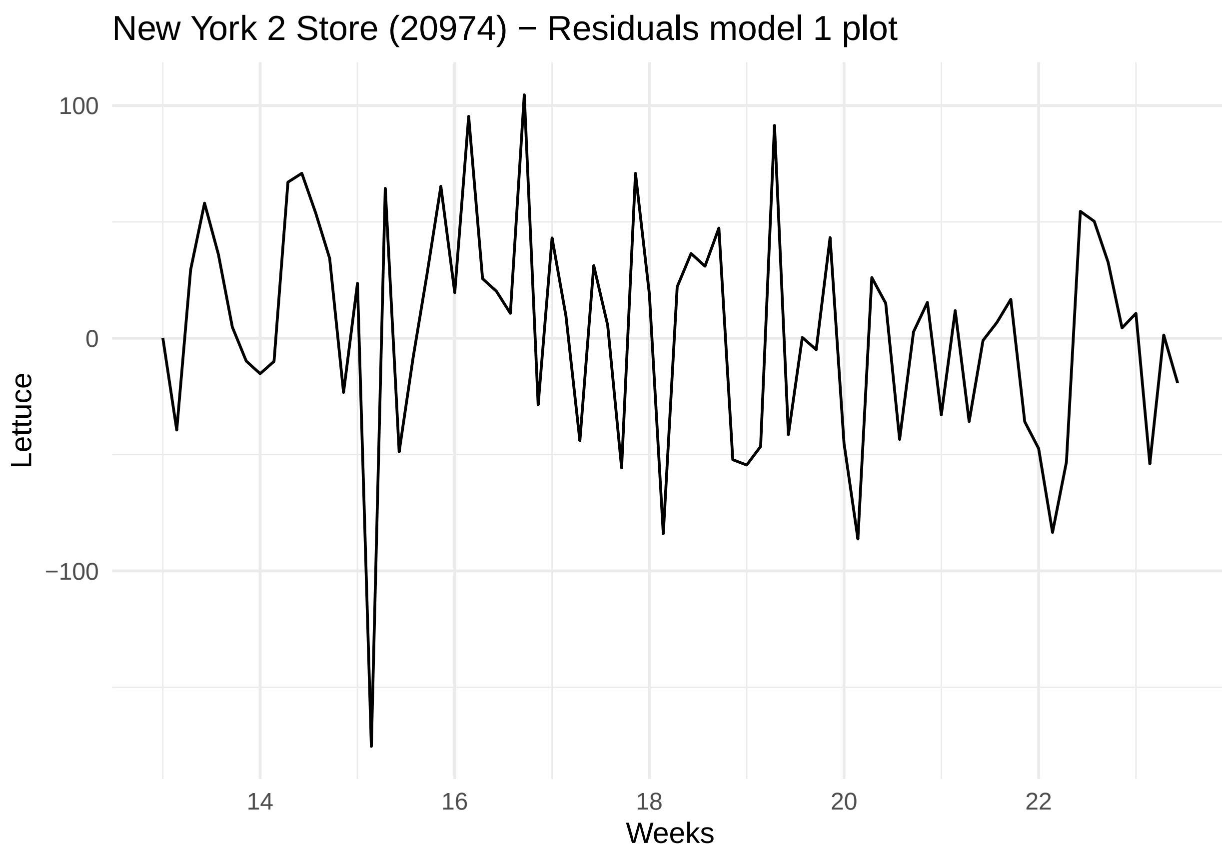
```r
# in-sample one-step forecasts model 3
accuracy(lettuce.m3)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 2.568306 48.86888 38.11537 -16.67426 32.46622 0.8130309 0.1550136
```

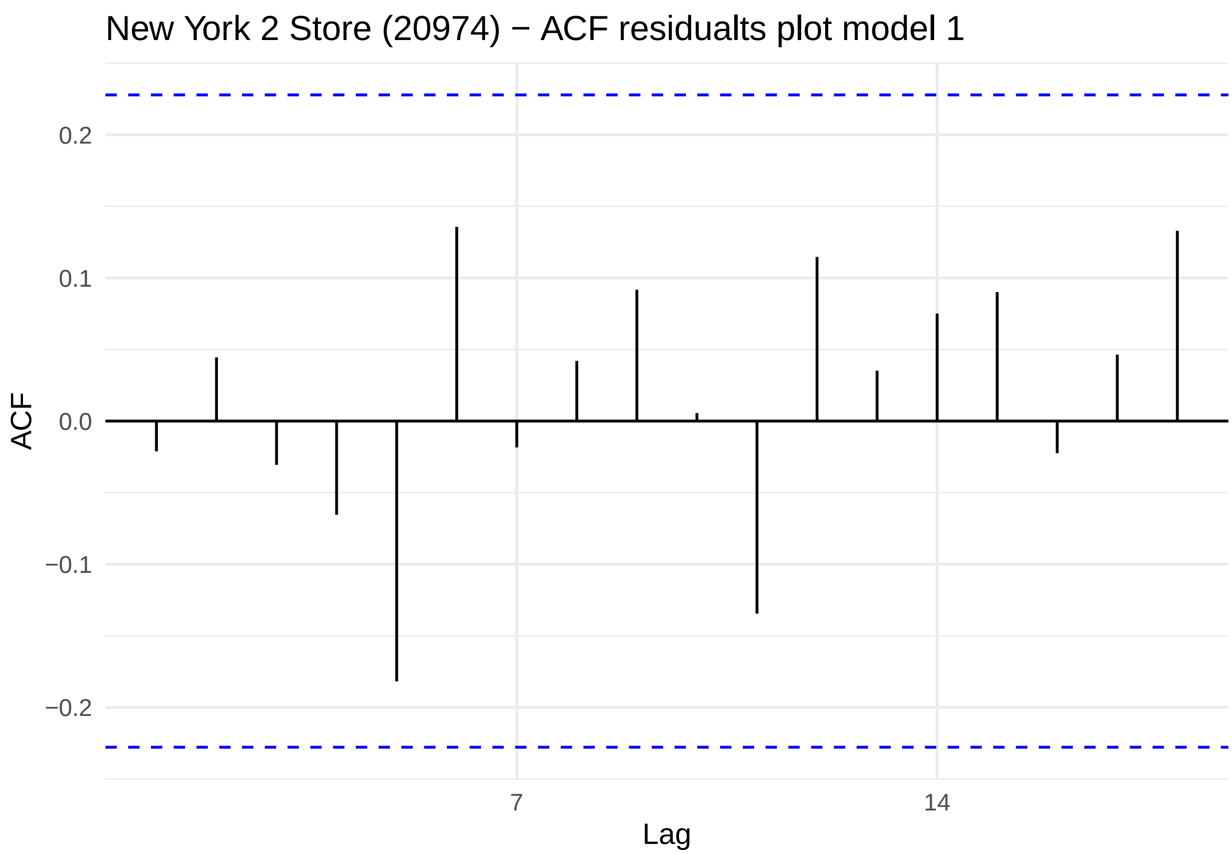
The first model even though it has both the lowest AIC score as well as the lowest RMSE.

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

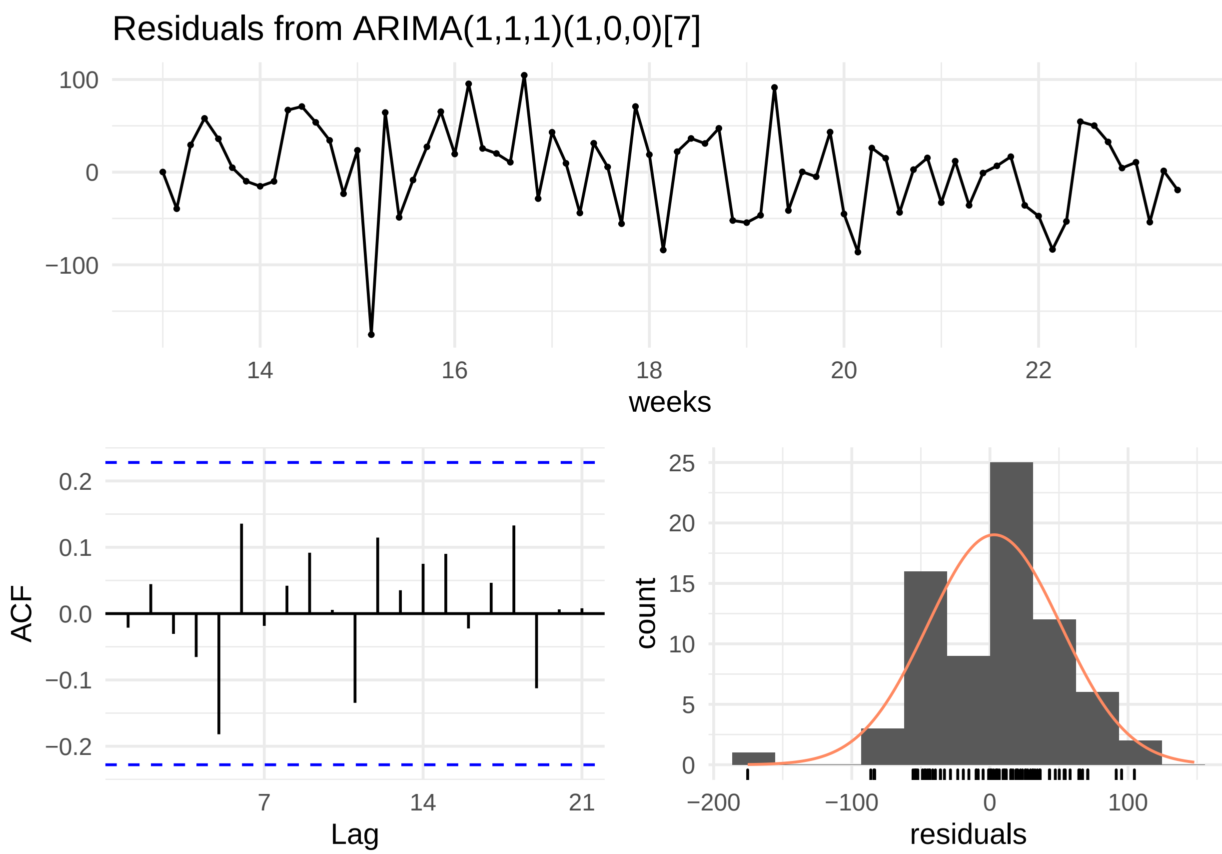
```r
# residual analysis model 1
autoplot(lettuce.m1$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 2 Store (20974) - Residuals model 1 plot")
```




```r
ggAcf(lettuce.m1$residuals) + theme_minimal() +
ggtitle("New York 2 Store (20974) - ACF residualts plot model 1")
```

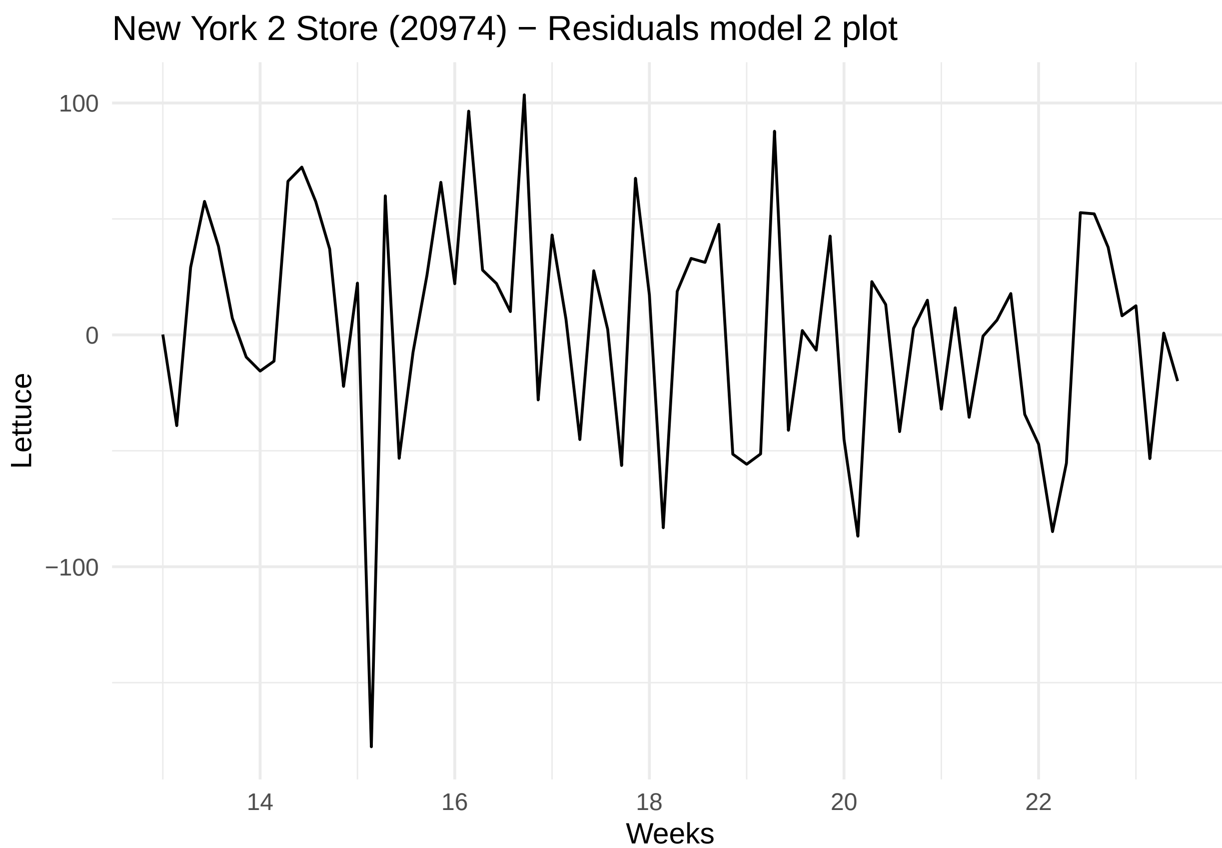



```
checkresiduals(lettuce.m1, xlab = "weeks", theme = theme_minimal())
```

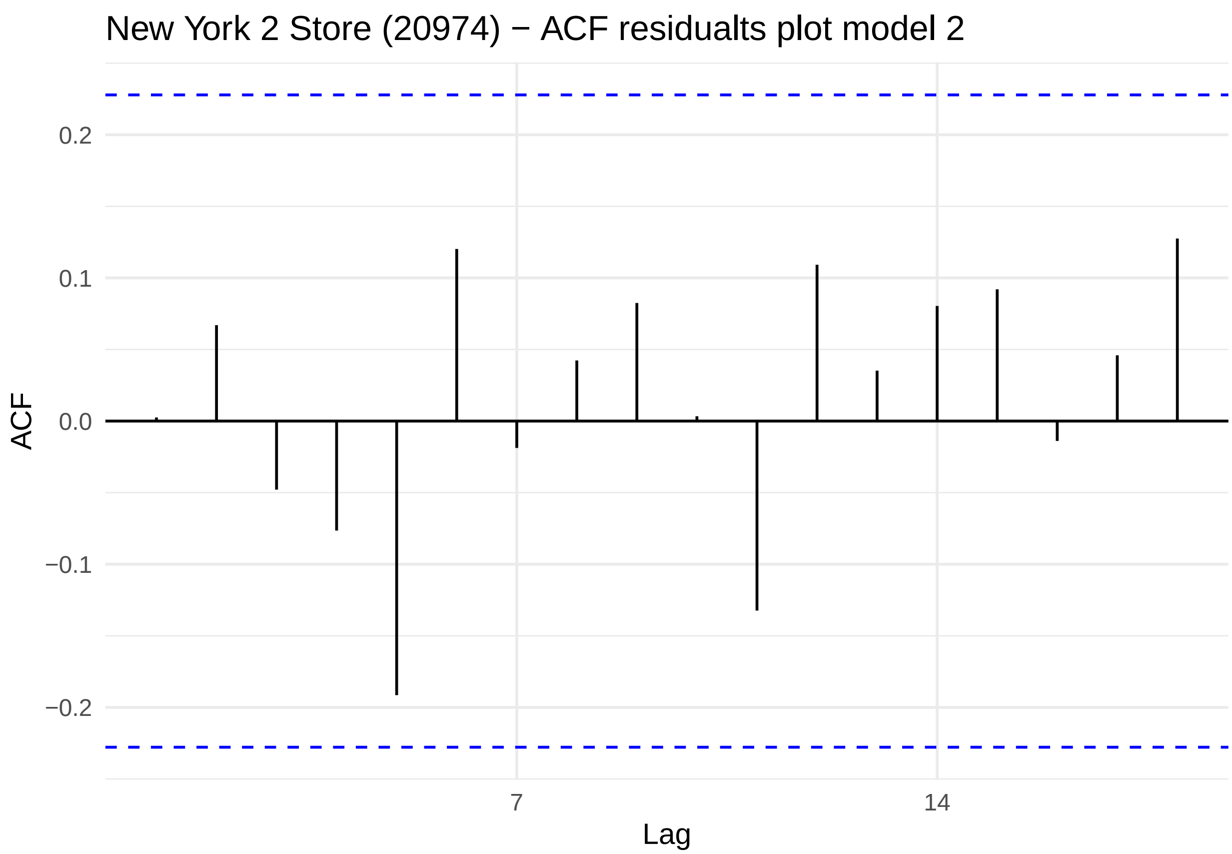



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(1,0,0)[7]
## Q* = 9.185, df = 11, p-value = 0.6048
##
## Model df: 3.   Total lags used: 14
```

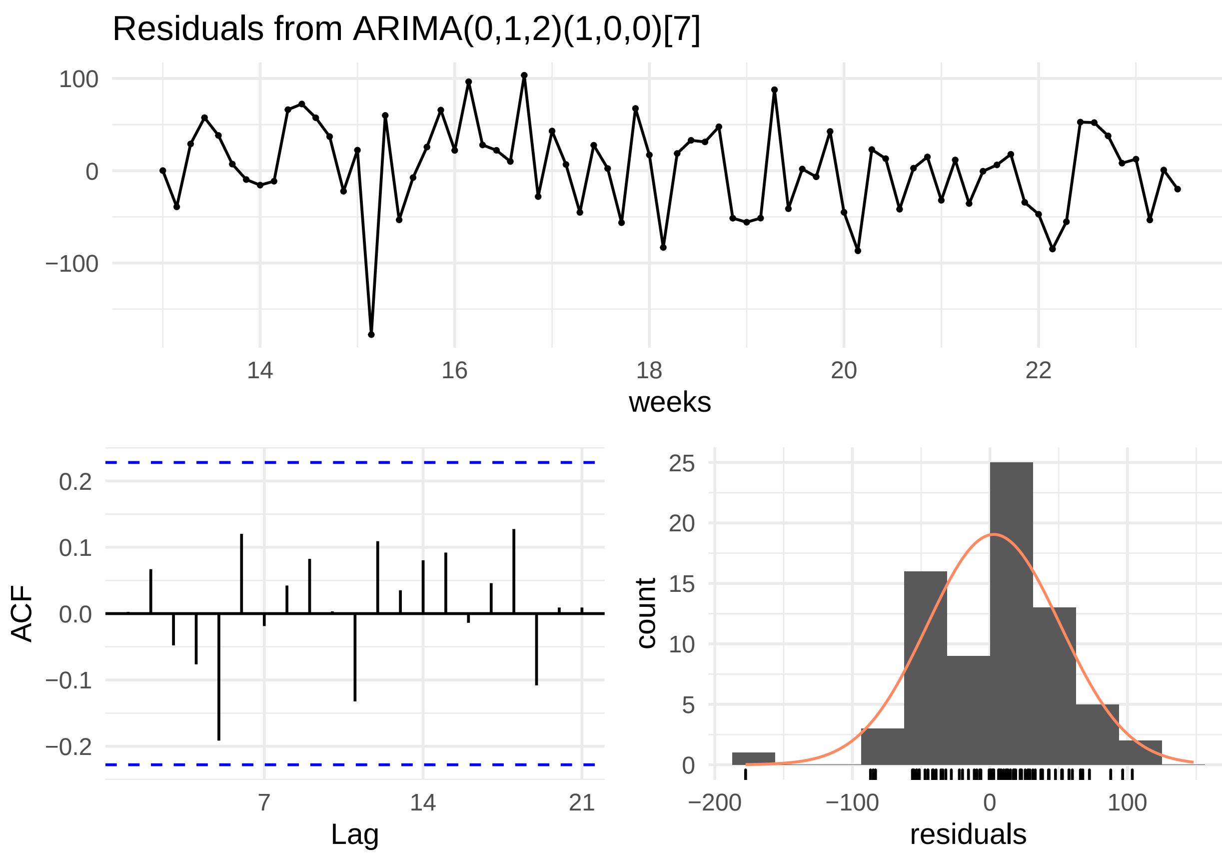
```
# residual analysis model 2
autoplot(lettuce.m2$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 2 Store (20974) - Residuals model 2 plot")
```




```
ggAcf(lettuce.m2$residuals) + theme_minimal() +
ggtitle("New York 2 Store (20974) - ACF residualts plot model 2")
```

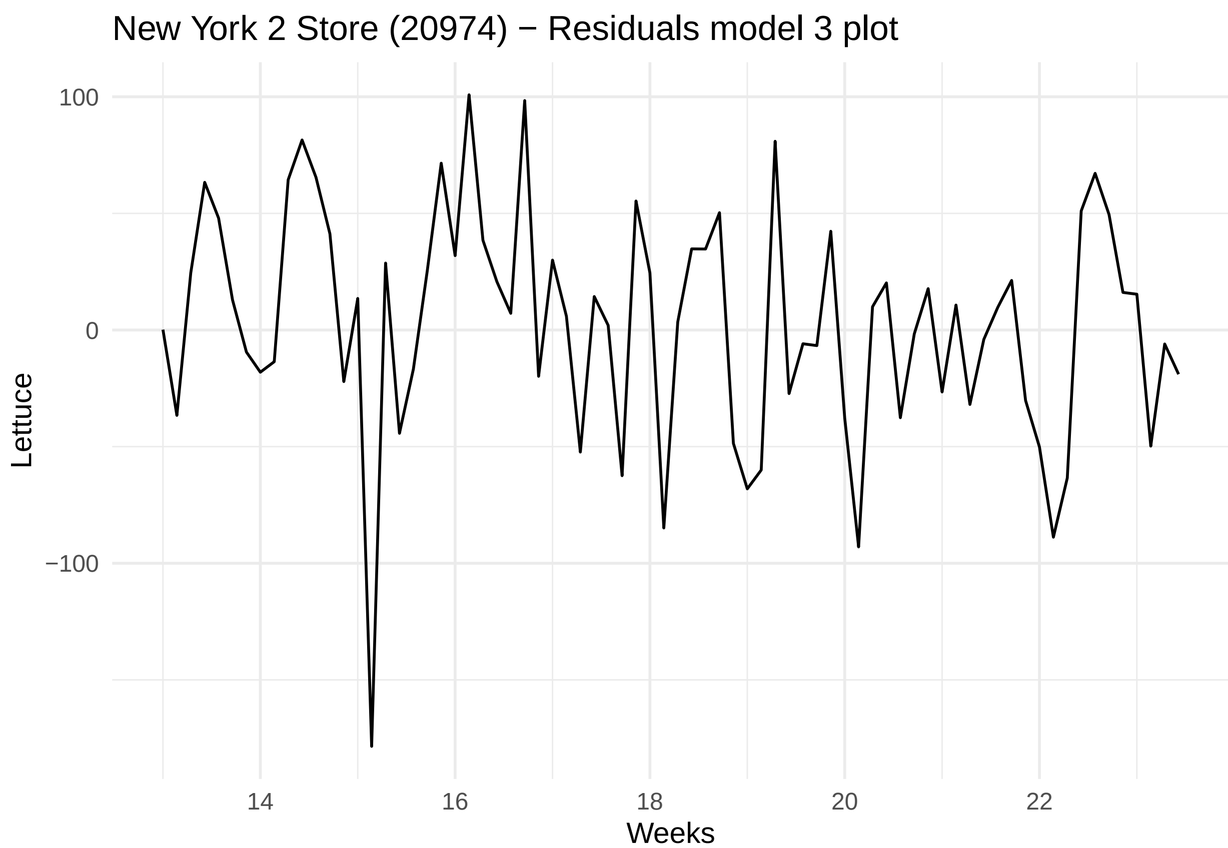



```r
checkresiduals(lettuce.m2, xlab = "weeks", theme = theme_minimal())
```

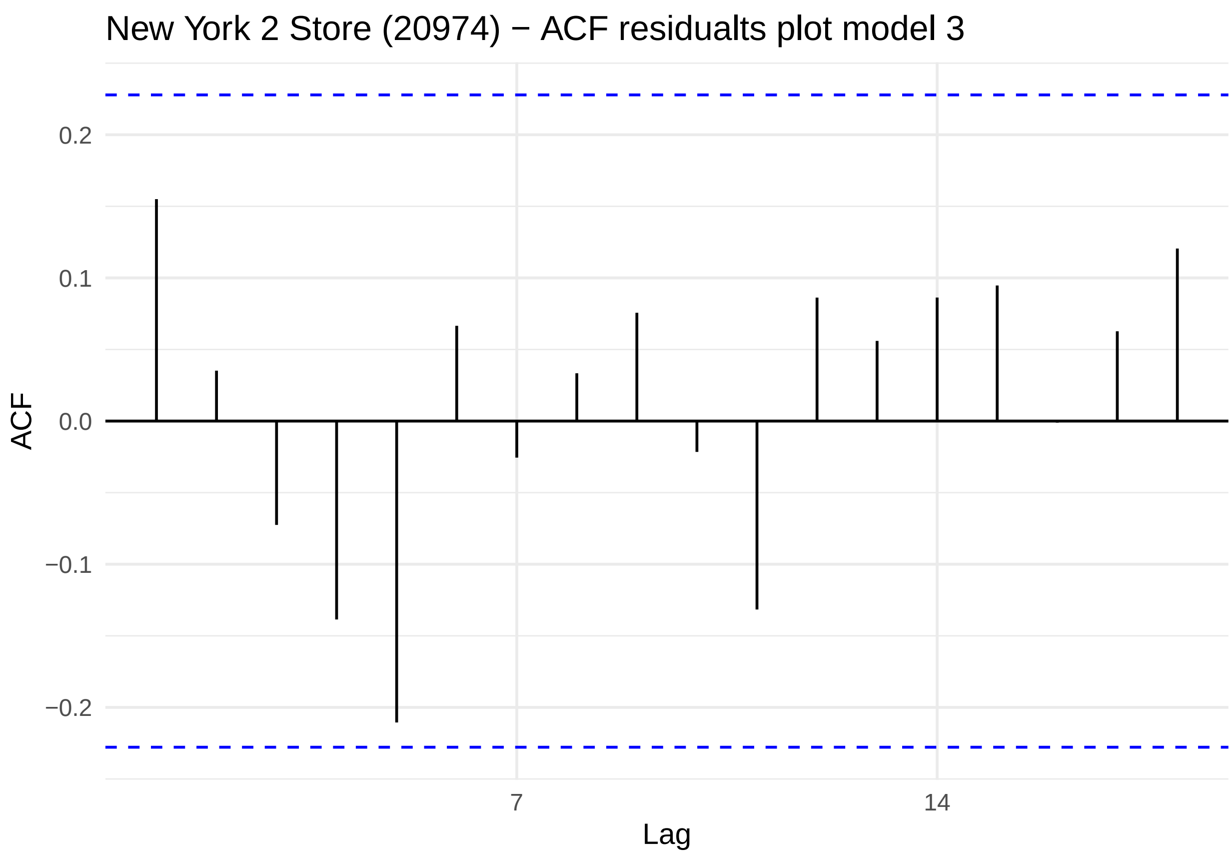



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)(1,0,0)[7]
## Q* = 9.3227, df = 11, p-value = 0.5921
##
## Model df: 3.   Total lags used: 14
```

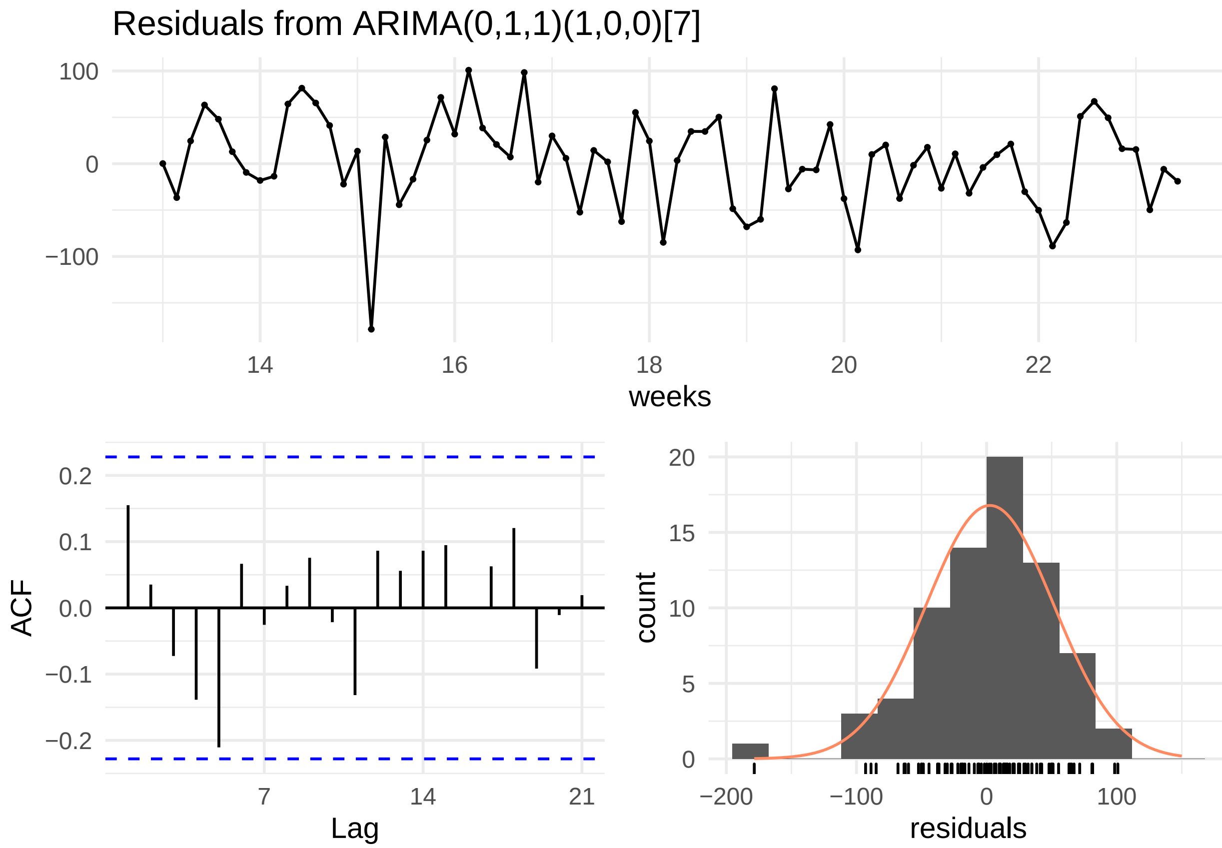
```
# residual analysis model 3
autoplot(lettuce.m3$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("New York 2 Store (20974) - Residuals model 3 plot")
```




```
ggAcf(lettuce.m3$residuals) + theme_minimal() +
ggtitle("New York 2 Store (20974) - ACF residualts plot model 3")
```



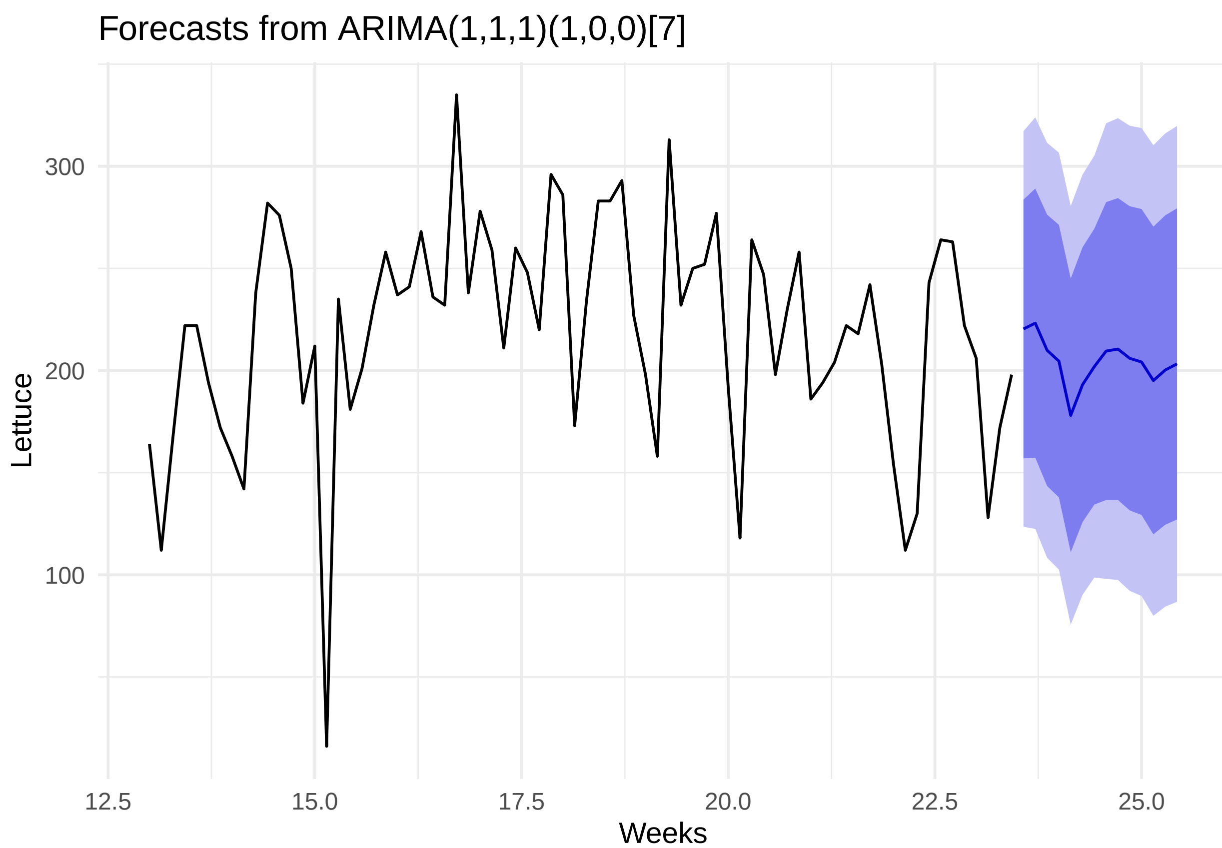

```r
checkresiduals(lettuce.m3, xlab = "weeks", theme = theme_minimal())
```



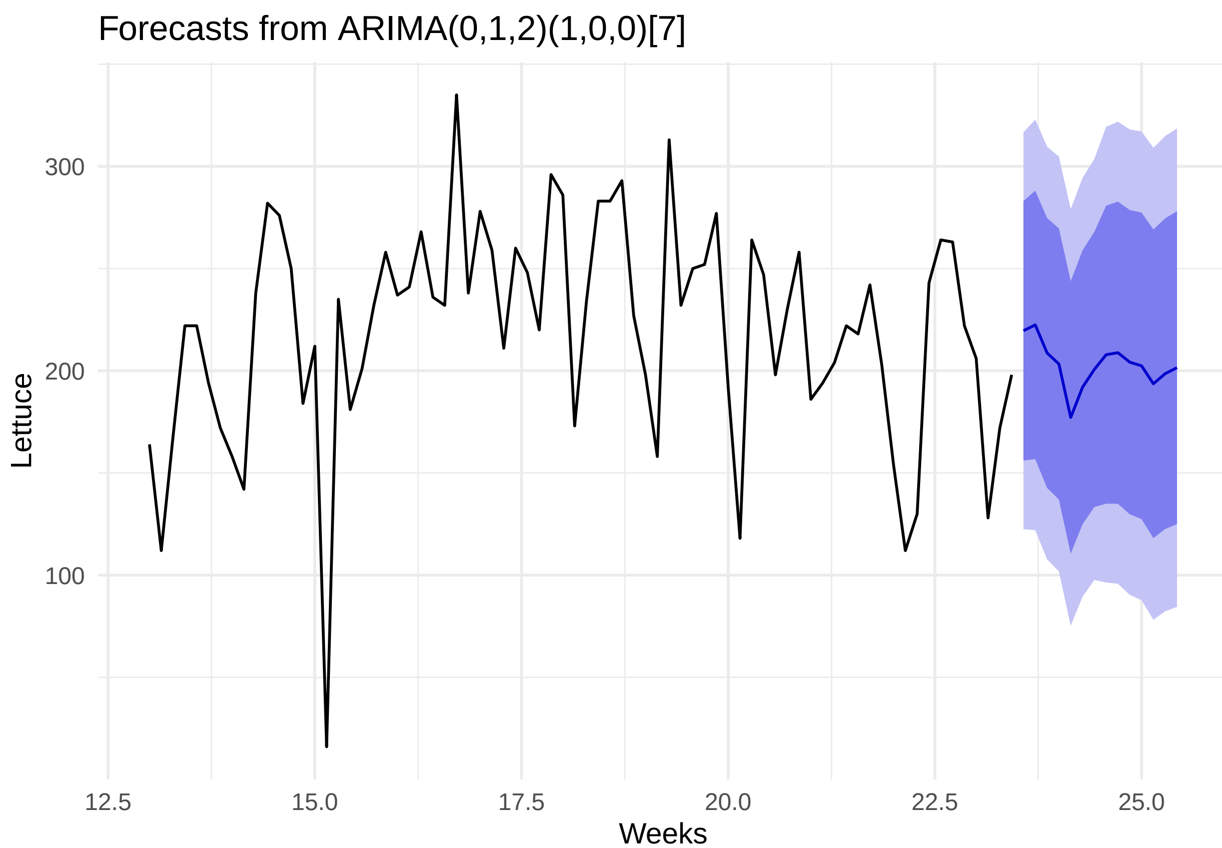

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(1,0,0)[7]
## Q* = 11.78, df = 12, p-value = 0.4635
##
## Model df: 2.   Total lags used: 14
```

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

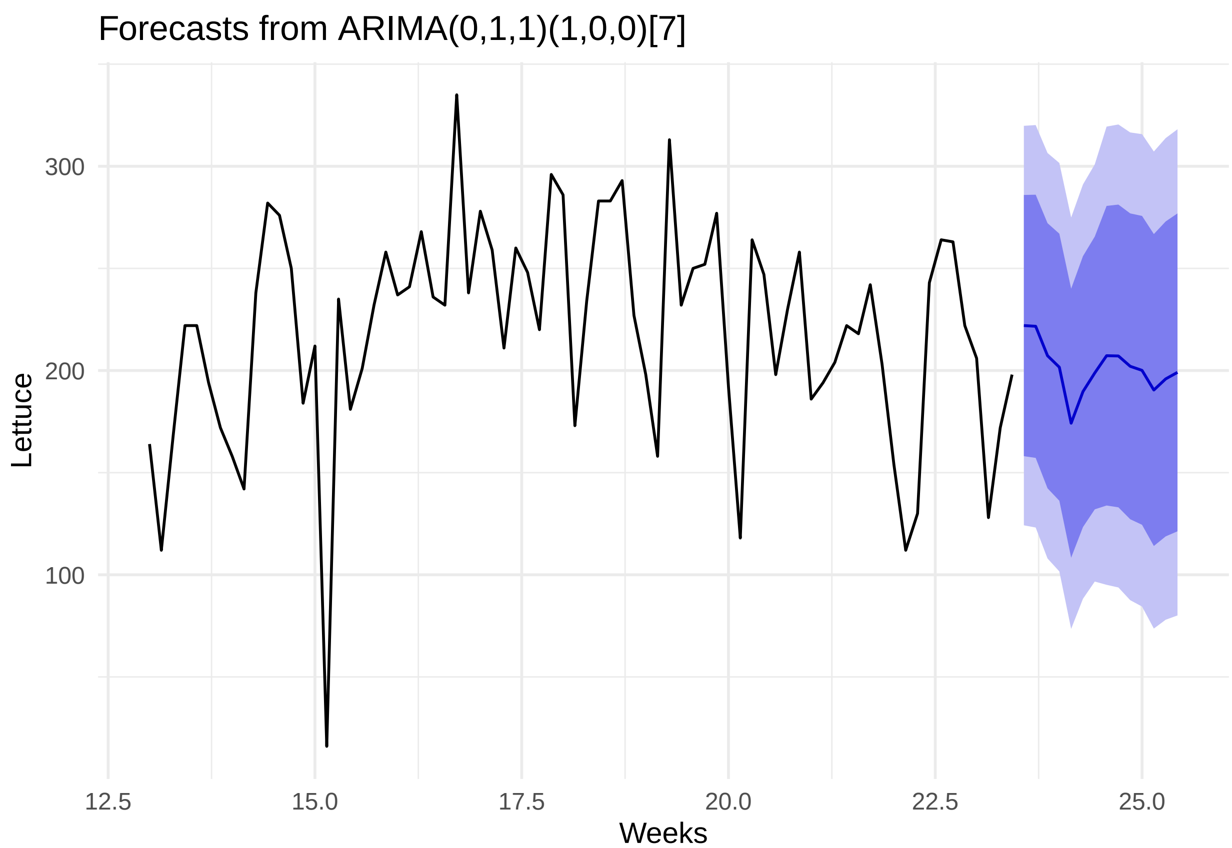
```
#Forecasting part model 1
lettuce.f1 <- forecast(lettuce.m1, h = 14)
autoplot(lettuce.f1, xlab = "Weeks", ylab = "Lettuce") + theme_minimal()
```




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




```
#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 80 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  3.111878 48.05772 37.70491 -16.059101 31.88792 0.8042754
## Test set     19.080288 55.30611 33.94545   4.536817 13.27772 0.7240832
##                     ACF1 Theil's U
## Training set -0.02117438        NA
## Test set     -0.11246459  0.801241
```

```
# 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  2.848727 48.17716 37.76400 -16.373326 32.14221 0.8055358
## Test set     20.453128 55.88913 34.51513   5.176498 13.44700 0.7362348
##                      ACF1 Theil's U
```

```
## Training set  0.002492967        NA
## Test set     -0.108567016 0.8119853
```

```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  2.568306 48.86888 38.11537 -16.67426 32.46622 0.8130309
## Test set     22.125811 56.56734 34.81456   5.99483 13.45120 0.7426220
##                    ACF1 Theil's U
## Training set  0.1550136        NA
## Test set     -0.1130414 0.8256285
```

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

Now we train the first 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(1, 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
## 25.57143       235.5873 164.7033 306.4713 127.17952 343.9950
## 25.71429       248.4292 171.6683 325.1901 131.03351 365.8248
## 25.85714       294.1569 213.8089 374.5050 171.27518 417.0387
## 26.00000       217.3068 133.7452 300.8685  89.51031 345.1034
## 26.14286       230.0948 143.4674 316.7222  97.60964 362.5800
## 26.28571       216.2299 126.6454 305.8144  79.22229 353.2375
## 26.42857       251.4221 158.9756 343.8685 110.03736 392.8068
## 26.57143       240.6110 136.4357 344.7863  81.28861 399.9334
## 26.71429       245.1760 135.8092 354.5428  77.91389 412.4381
## 26.85714       261.4312 147.6837 375.1788  87.46933 435.3931
## 27.00000       234.1127 116.2210 352.0044  53.81288 414.4125
## 27.14286       238.6585 116.7733 360.5438  52.25116 425.0659
## 27.28571       233.7299 107.9792 359.4805  41.41080 426.0489
## 27.42857       246.2399 116.7393 375.7405  48.18579 444.2940
```

We present our forecast through ARIMA(1,1,1)(1,0,0) 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_NewYork2_arima <- cbind(next2weeks, forecast_data$`Point Forecast`)
final_forecast_NewYork2_arima
```

```
##    day forecast_data$`Point Forecast`
## 1    1                      235.5873
## 2    2                      248.4292
## 3    3                      294.1569
## 4    4                      217.3068
## 5    5                      230.0948
## 6    6                      216.2299
## 7    7                      251.4221
## 8    8                      240.6110
## 9    9                      245.1760
## 10  10                      261.4312
## 11  11                      234.1127
## 12  12                      238.6585
## 13  13                      233.7299
## 14  14                      246.2399
```

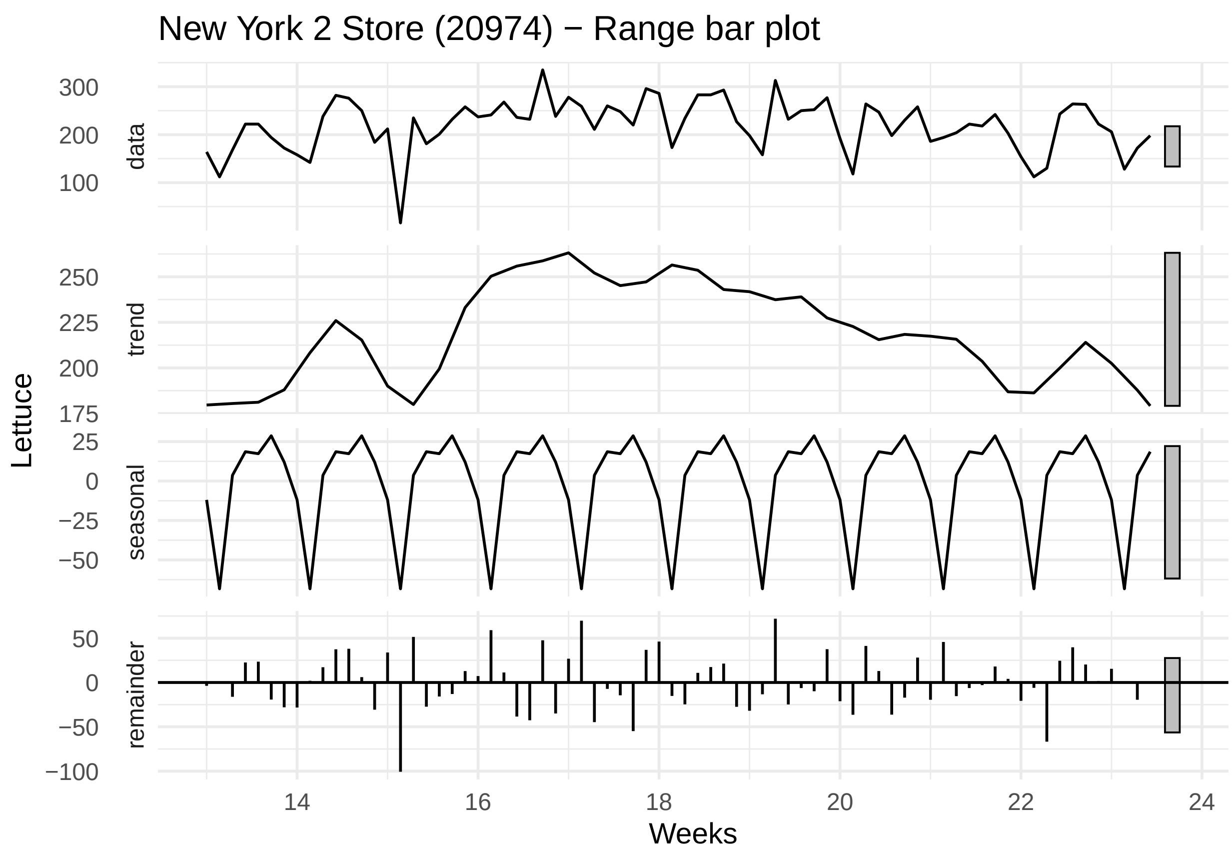
Holt-Winters

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 2 Store (20974) - Range bar plot")
```



For this data set, the grey bar of the trend panel is significantly 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.

The grey bar of the seasonal panel is also large, and larger than the grey bar of random error term, which indicates that the contribution of seasonal factors to the variation in the original time series is marginal too. 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(A,N,A)
##
## Call:
##  ets(y = lettuce_train, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.1905
##     gamma = 1e-04
```

```
##
##   Initial states:
##     l = 209.6333
##     s = 12.0706 29.4123 17.4096 18.8407 4.0938 -65.3266
##            -16.5004
##
##   sigma:  43.4163
##
##      AIC     AICc      BIC
## 886.9883 890.4803 910.0289
```

Our best model is the ETS(A,N,A).

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

```
## ETS(A,N,A)
##
## Call:
##  ets(y = lettuce_train, model = "ANA", ic = "aic")
##
##   Smoothing parameters:
##     alpha = 0.1905
##     gamma = 1e-04
##
##   Initial states:
##     l = 209.6333
##     s = 12.0706 29.4123 17.4096 18.8407 4.0938 -65.3266
##            -16.5004
##
##   sigma:  43.4163
##
##      AIC     AICc      BIC
## 886.9883 890.4803 910.0289
```

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 -0.9560356 40.69057 33.6634 -13.46909 26.18756 0.7180669
##                    ACF1
## Training set 0.08618948
```

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

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

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 23.57143       213.5625 157.92223 269.2027 128.46806 298.6569
## 23.71429       225.5651 168.92369 282.2064 138.93956 312.1905
## 23.85714       208.2230 150.59786 265.8480 120.09298 296.3529
## 24.00000       179.6553 121.06295 238.2476  90.04606 269.2644
## 24.14286       130.8193  71.27544 190.3631  39.75486 221.8837
## 24.28571       200.2432 139.76289 260.7236 107.74653 292.7399
## 24.42857       214.9915 153.58779 276.3952 121.08264 308.9003
## 24.57143       213.5625 151.25019 275.8748 118.26406 308.8609
## 24.71429       225.5651 162.35723 288.7729 128.89704 322.2331
## 24.85714       208.2230 144.13212 272.3138 110.20448 306.2414
## 25.00000       179.6553 114.69340 244.6171  80.30467 279.0058
## 25.14286       130.8193  64.99791 196.6406  30.15420 231.4843
## 25.28571       200.2432 133.57348 266.9130  98.28064 302.2058
## 25.42857       214.9915 147.48298 282.5000 111.74614 318.2368
```

After the forecast, we continue with the in and out of sample accuracy of the two ets models.

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

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.9560356 40.69057 33.66340 -13.46909 26.18756 0.7180669
## Test set     27.2057558 55.99135 33.87778   9.46673 13.38155 0.7226398
##                     ACF1 Theil's U
## Training set  0.08618948        NA
## Test set     -0.29493593 0.8497462
```

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

```r
# final model
lettuce.ets <- ets(lettuce, model = "ANA", ic = 'aic')
lettuce.ets
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = lettuce, model = "ANA", ic = "aic")
##
##   Smoothing parameters:
##     alpha = 0.44
##     gamma = 1e-04
##
##   Initial states:
##     l = 114.9789
##     s = -10.1021 18.4806 24.8543 9.6888 15.7566 -0.3719
```

```
##            -58.3063
##
##   sigma:  53.2174
##
##      AIC     AICc      BIC
## 1184.794 1187.444 1210.227
```

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

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

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 25.57143       239.2185 171.01765 307.4193 134.91430 343.5227
## 25.71429       254.3795 179.86998 328.8891 140.42701 368.3321
## 25.85714       248.0047 167.68037 328.3289 125.15926 370.8500
## 26.00000       219.4262 133.68063 305.1718  88.28965 350.5628
## 26.14286       171.2160  80.37199 262.0599  32.28210 310.1498
## 26.28571       229.1470 133.47598 324.8180  82.83080 375.4632
## 26.42857       245.2863 145.01829 345.5543  91.93962 398.6329
## 26.57143       239.2185 134.55720 343.8798  79.15286 399.2841
## 26.71429       254.3795 145.50206 363.2570  87.86581 420.8933
## 26.85714       248.0047 135.06829 360.9410  75.28340 420.7259
## 27.00000       219.4262 102.57188 336.2806  40.71292 398.1395
## 27.14286       171.2160  50.57079 291.8611 -13.29491 355.7268
## 27.28571       229.1470 104.82654 353.4674  39.01527 419.2787
## 27.42857       245.2863 117.39452 373.1780  49.69272 440.8798
```

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

```r
forecast_data <- as.data.frame(lettuce.ets.f)
next2weeks <- data.frame(day = seq(1, 14))
final_forecast_NewYork2_ets <- cbind(next2weeks, forecast_data$`Point Forecast`)
final_forecast_NewYork2_ets
```

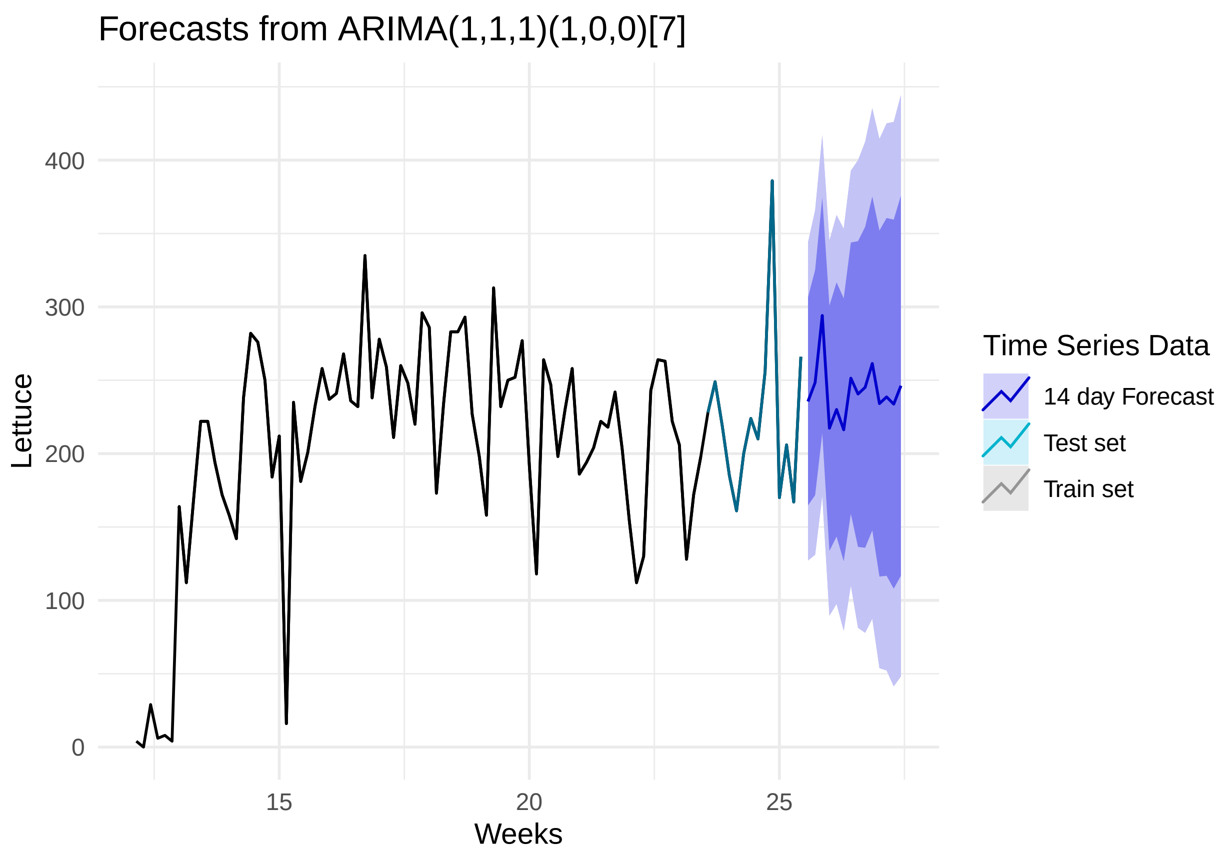
```
##    day forecast_data$`Point Forecast`
## 1    1                       239.2185
## 2    2                       254.3795
## 3    3                       248.0047
## 4    4                       219.4262
## 5    5                       171.2160
## 6    6                       229.1470
## 7    7                       245.2863
## 8    8                       239.2185
## 9    9                       254.3795
## 10  10                       248.0047
## 11  11                       219.4262
## 12  12                       171.2160
## 13  13                       229.1470
## 14  14                       245.2863
```

Comparison

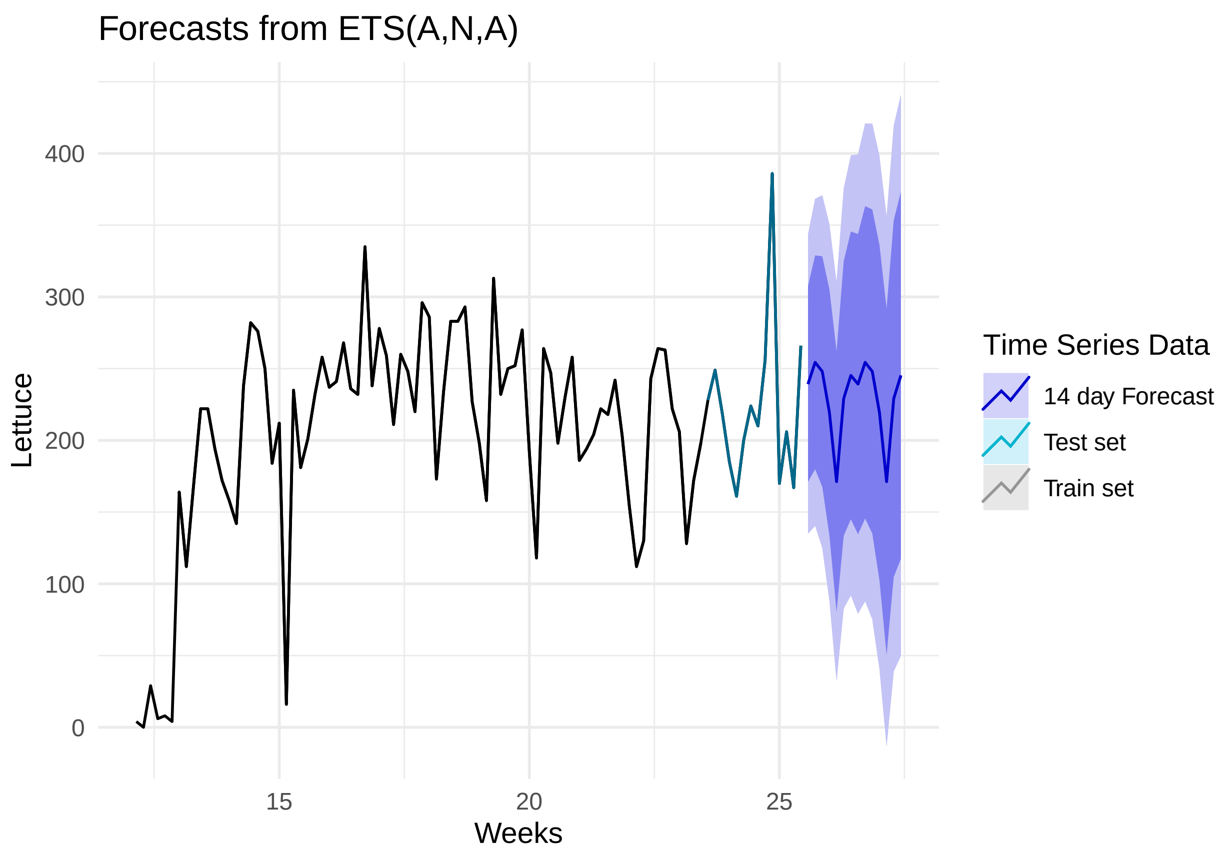
Now we will compare the two best models for New York 2 Store (20974).

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(0,1,2)(1,0,0) or ETS(A,N,A) to choose, we will check their RMSE in the test set.

```
# best ets model
# ETS(A,N,A)
accuracy.ets
```

```
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.9560356 40.69057 33.66340 -13.46909 26.18756 0.7180669
## Test set     27.2057558 55.99135 33.87778   9.46673 13.38155 0.7226398
##                     ACF1 Theil's U
## Training set  0.08618948        NA
## Test set     -0.29493593 0.8497462
```

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

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  2.848727 48.17716 37.76400 -16.373326 32.14221 0.8055358
## Test set     20.453128 55.88913 34.51513   5.176498 13.44700 0.7362348
##                      ACF1 Theil's U
## Training set  0.002492967        NA
## Test set     -0.108567016 0.8119853
```

We can observe that ARIMA(1,1,1)(1,0,0) has a better (lower) RMSE (41.21143 vs 55.99135) respectively. Therefore, we choose the ARIMA(1,1,1)(1,0,0) for New York (20974) store.

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

```
final_forecast_NewYork2_arima
```

```
##    day forecast_data$`Point Forecast`
## 1    1                       235.5873
## 2    2                       248.4292
## 3    3                       294.1569
## 4    4                       217.3068
## 5    5                       230.0948
## 6    6                       216.2299
## 7    7                       251.4221
## 8    8                       240.6110
## 9    9                       245.1760
## 10  10                       261.4312
## 11  11                       234.1127
## 12  12                       238.6585
## 13  13                       233.7299
## 14  14                       246.2399
```