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 California 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.

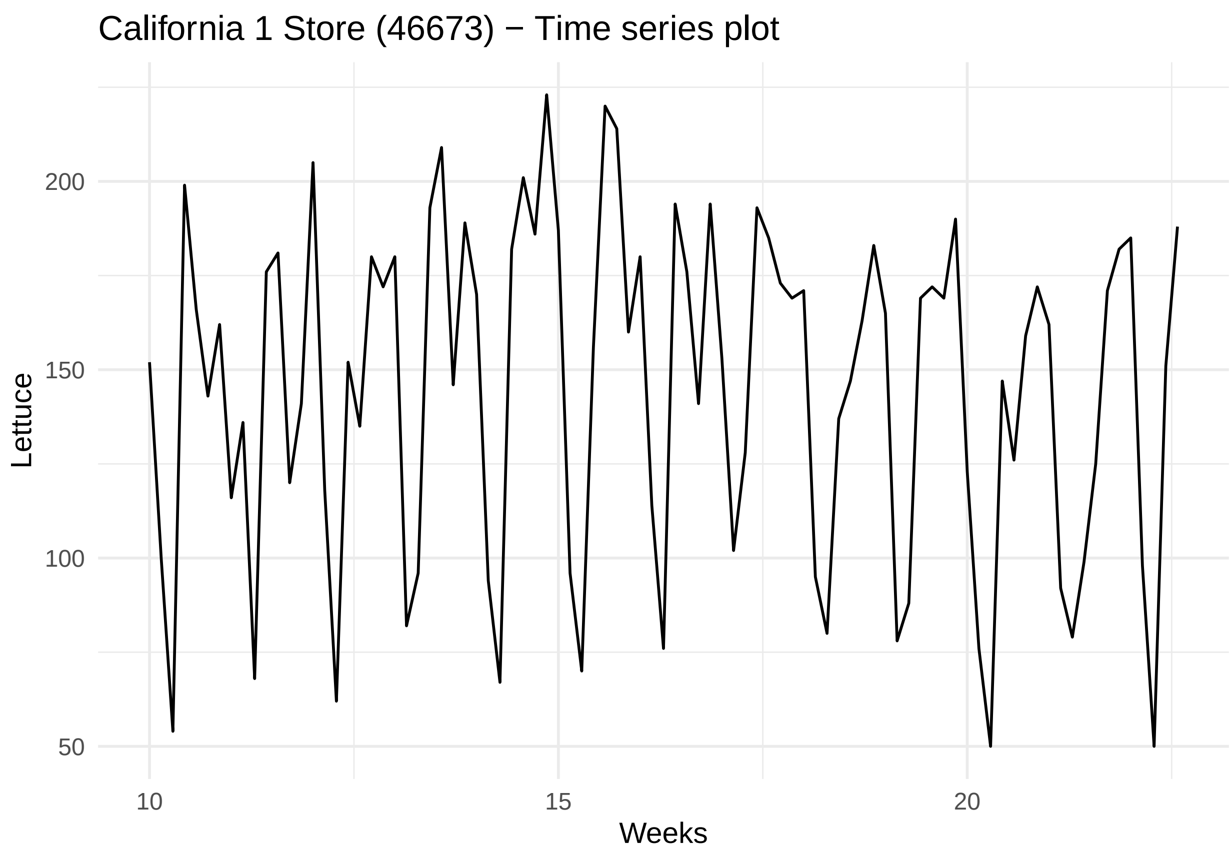
Then, we load and split the data set into train and test set.

```
# read csv file
data <- read.csv(file = "California1_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
```

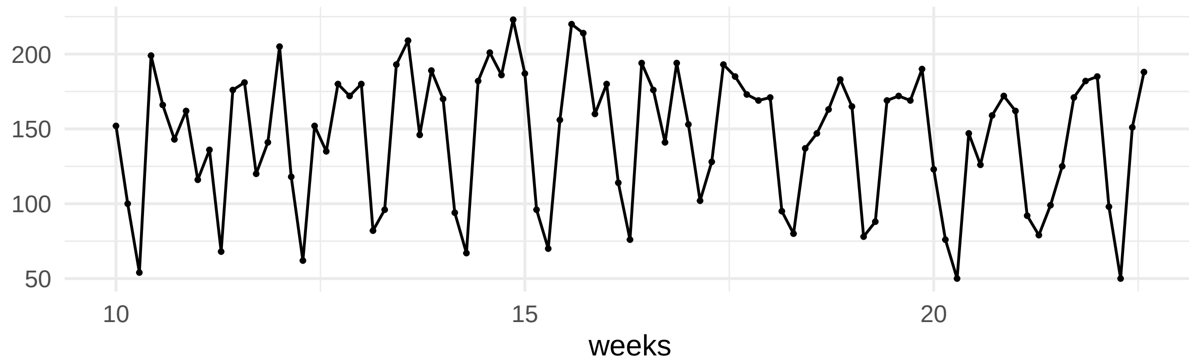
ARIMA

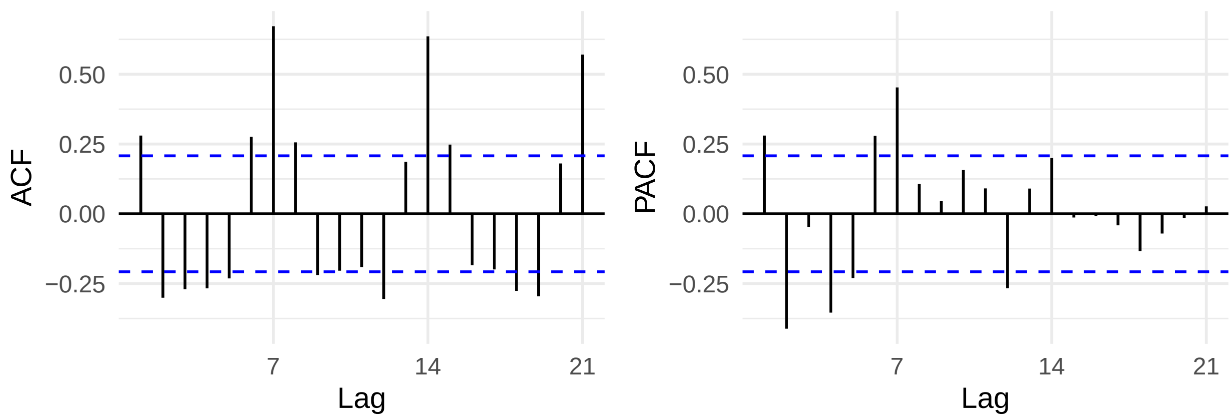
We visually inspect the time series.

```
autoplot(lettuce_train, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 1 Store (46673) - Time series plot")
```




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





Due to the seasonality in time series, it is non-stationary. We can get rid of seasonality by taking first-order difference. We plot the time series after the difference, 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 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 = -7.7984, 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) = -52.779, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

```
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 = 0.15893, 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:

```
ndiffs(lettuce_train)
```

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

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

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

We need to differentiate for seasonality one time.

```
### stationarize time series
# take first order difference
lettuce_train.diff1 <- diff(lettuce_train, differences = 1, lag=7)
```

Check again the tests for stationarity:

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  lettuce_train.diff1
## Dickey-Fuller = -3.613, Lag order = 4, p-value = 0.03748
## alternative hypothesis: stationary
```

```
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) = -74.087, Truncation lag parameter = 3, p-value
## = 0.01
## alternative hypothesis: stationary
```

```
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 = 0.097681, Truncation lag parameter = 3, p-value = 0.1
```

Check again the two automatic functions for stationarity:

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

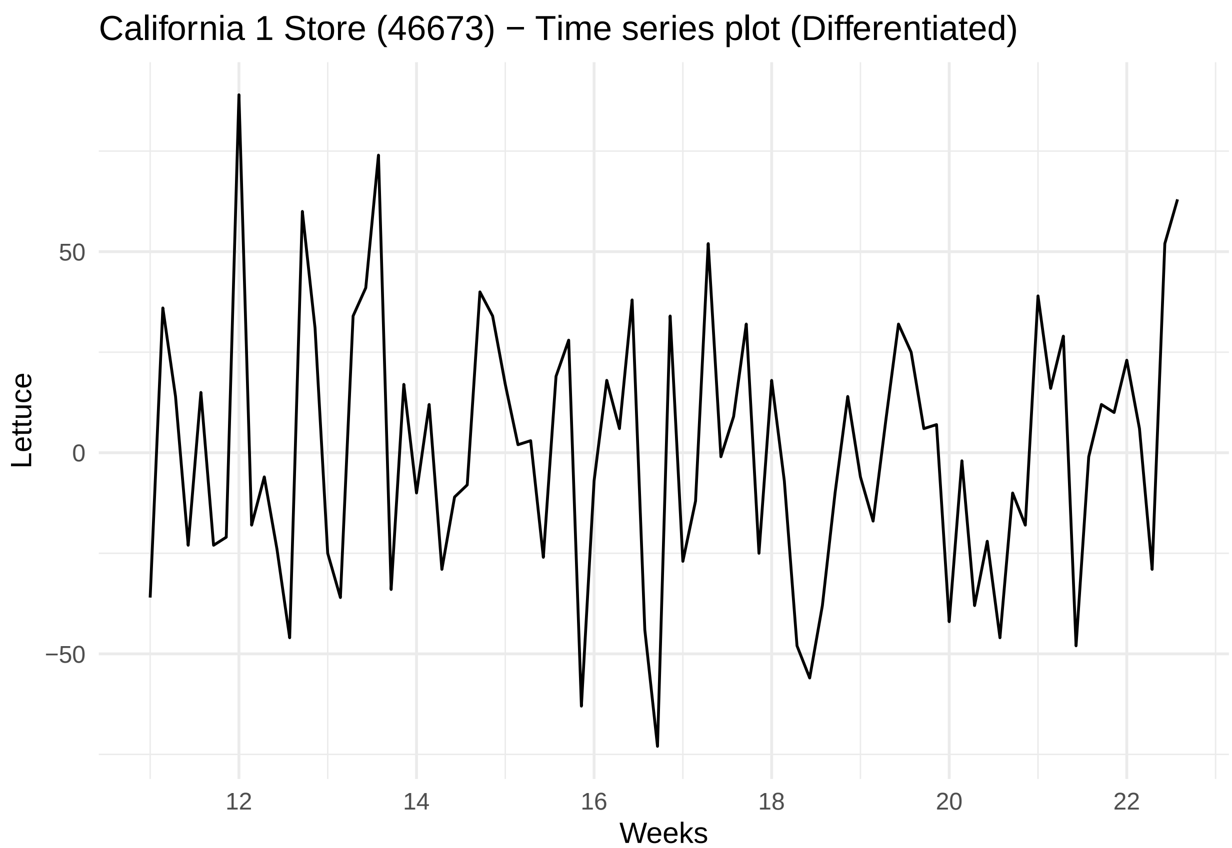
```
## [1] 0
```

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

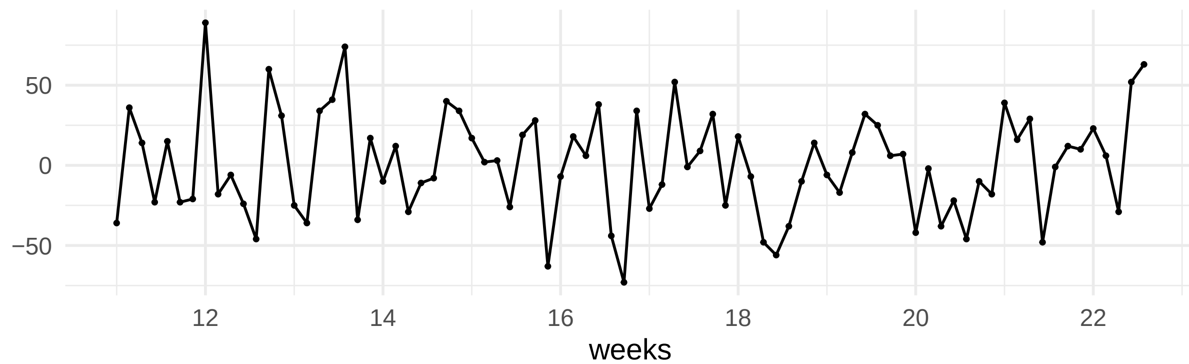
```
## [1] 0
```

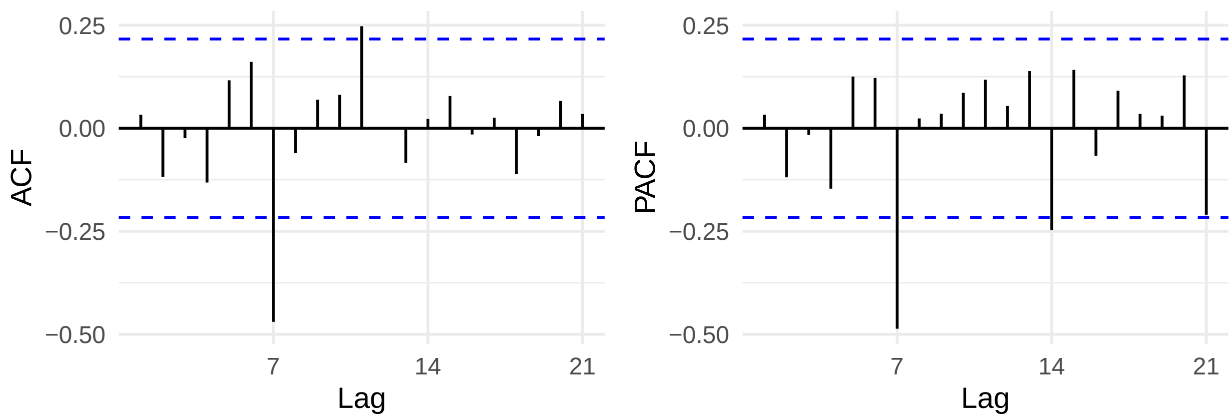
We now visually inspect the differentiated time series.

```
autoplot(lettuce_train.diff1, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 1 Store (46673) - Time series plot (Differentiated)")
```




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

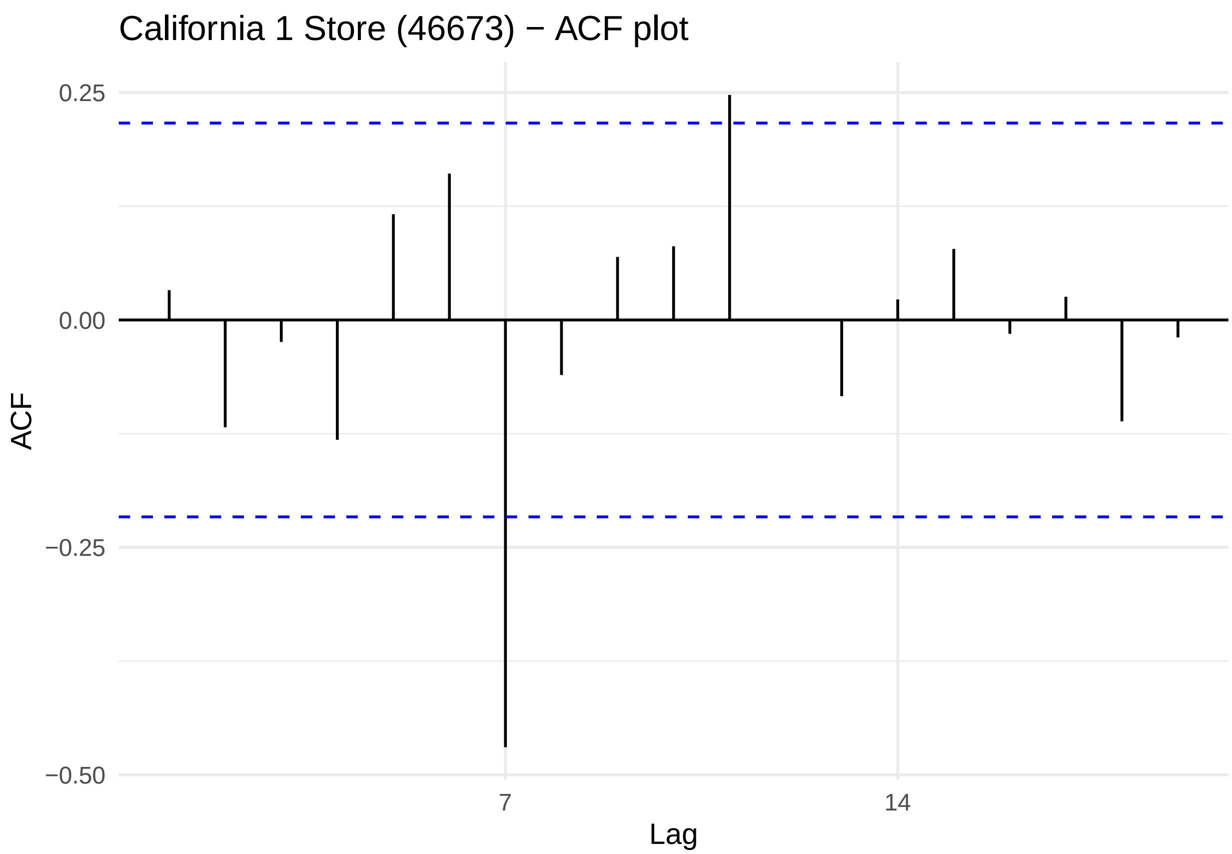




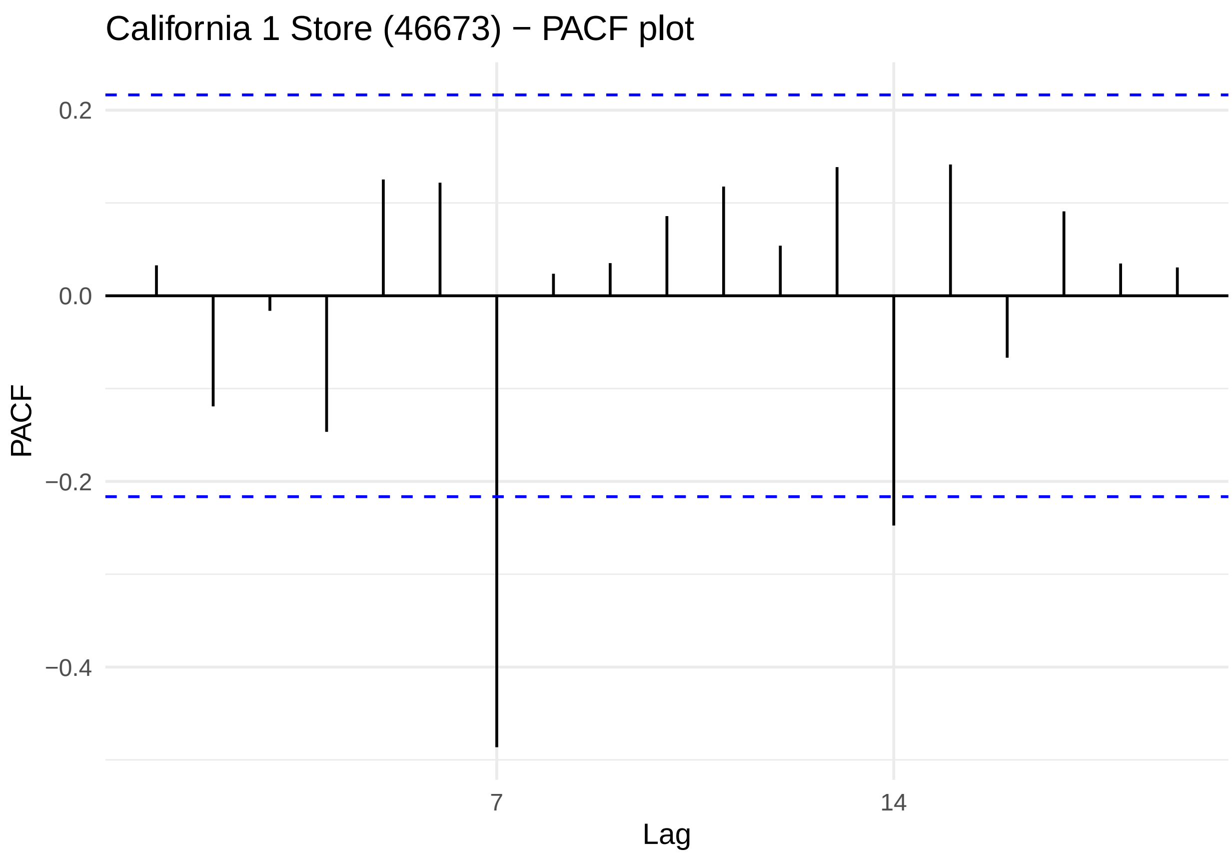
Looks stationary.

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("California 1 Store (46673) - ACF plot")
```




```
# pacf plot
ggPacf(lettuce_train.diff1) + theme_minimal() + ggtitle("California 1 Store (46673) - 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 = 0 and D = 1 since we had no first-differencing but only seasonal-differencing.

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

```
##
##  ARIMA(0,0,0)(0,1,0)[7]                    : 805.6014
##  ARIMA(0,0,0)(0,1,0)[7] with drift         : 807.5736
##  ARIMA(0,0,0)(0,1,1)[7]                    : 776.3348
##  ARIMA(0,0,0)(0,1,1)[7] with drift         : 777.7948
##  ARIMA(0,0,0)(0,1,2)[7]                    : 778.2026
##  ARIMA(0,0,0)(0,1,2)[7] with drift         : 779.51
##  ARIMA(0,0,0)(1,1,0)[7]                    : 785.271
##  ARIMA(0,0,0)(1,1,0)[7] with drift         : 787.2616
##  ARIMA(0,0,0)(1,1,1)[7]                    : 778.1586
##  ARIMA(0,0,0)(1,1,1)[7] with drift         : 779.4319
##  ARIMA(0,0,0)(1,1,2)[7]                    : 780.2928
##  ARIMA(0,0,0)(1,1,2)[7] with drift         : Inf
##  ARIMA(0,0,0)(2,1,0)[7]                    : 780.505
##  ARIMA(0,0,0)(2,1,0)[7] with drift         : 782.4886
##  ARIMA(0,0,0)(2,1,1)[7]                    : Inf
```

```
##  ARIMA(0,0,0)(2,1,1)[7] with drift         : Inf
##  ARIMA(0,0,0)(2,1,2)[7]                    : 781.8725
##  ARIMA(0,0,0)(2,1,2)[7] with drift         : Inf
##  ARIMA(0,0,1)(0,1,0)[7]                    : 807.4772
##  ARIMA(0,0,1)(0,1,0)[7] with drift         : 809.4507
##  ARIMA(0,0,1)(0,1,1)[7]                    : 775.4903
##  ARIMA(0,0,1)(0,1,1)[7] with drift         : 777.0543
##  ARIMA(0,0,1)(0,1,2)[7]                    : 777.4855
##  ARIMA(0,0,1)(0,1,2)[7] with drift         : 779.0522
##  ARIMA(0,0,1)(1,1,0)[7]                    : 785.9265
##  ARIMA(0,0,1)(1,1,0)[7] with drift         : 787.9132
##  ARIMA(0,0,1)(1,1,1)[7]                    : 777.4848
##  ARIMA(0,0,1)(1,1,1)[7] with drift         : 779.052
##  ARIMA(0,0,1)(1,1,2)[7]                    : Inf
##  ARIMA(0,0,1)(1,1,2)[7] with drift         : Inf
##  ARIMA(0,0,1)(2,1,0)[7]                    : 779.3363
##  ARIMA(0,0,1)(2,1,0)[7] with drift         : 781.3269
##  ARIMA(0,0,1)(2,1,1)[7]                    : 779.3704
##  ARIMA(0,0,1)(2,1,1)[7] with drift         : 780.9378
##  ARIMA(0,0,1)(2,1,2)[7]                    : 781.4839
##  ARIMA(0,0,1)(2,1,2)[7] with drift         : 783.0476
##  ARIMA(0,0,2)(0,1,0)[7]                    : 807.8585
##  ARIMA(0,0,2)(0,1,0)[7] with drift         : 809.8421
##  ARIMA(0,0,2)(0,1,1)[7]                    : 777.4827
##  ARIMA(0,0,2)(0,1,1)[7] with drift         : 779.0324
##  ARIMA(0,0,2)(0,1,2)[7]                    : 779.4806
##  ARIMA(0,0,2)(0,1,2)[7] with drift         : 781.0219
##  ARIMA(0,0,2)(1,1,0)[7]                    : 787.7523
##  ARIMA(0,0,2)(1,1,0)[7] with drift         : 789.7431
##  ARIMA(0,0,2)(1,1,1)[7]                    : 779.4803
##  ARIMA(0,0,2)(1,1,1)[7] with drift         : 781.0202
##  ARIMA(0,0,2)(1,1,2)[7]                    : Inf
##  ARIMA(0,0,2)(1,1,2)[7] with drift         : Inf
##  ARIMA(0,0,2)(2,1,0)[7]                    : 781.3324
##  ARIMA(0,0,2)(2,1,0)[7] with drift         : 783.3224
##  ARIMA(0,0,2)(2,1,1)[7]                    : 781.3469
##  ARIMA(0,0,2)(2,1,1)[7] with drift         : 782.8735
##  ARIMA(0,0,3)(0,1,0)[7]                    : 809.8234
##  ARIMA(0,0,3)(0,1,0)[7] with drift         : 811.8086
##  ARIMA(0,0,3)(0,1,1)[7]                    : 779.4683
##  ARIMA(0,0,3)(0,1,1)[7] with drift         : 781.0282
##  ARIMA(0,0,3)(0,1,2)[7]                    : 781.467
##  ARIMA(0,0,3)(0,1,2)[7] with drift         : 783.0172
##  ARIMA(0,0,3)(1,1,0)[7]                    : 789.4395
##  ARIMA(0,0,3)(1,1,0)[7] with drift         : 791.4345
##  ARIMA(0,0,3)(1,1,1)[7]                    : 781.4668
##  ARIMA(0,0,3)(1,1,1)[7] with drift         : 783.0156
##  ARIMA(0,0,3)(2,1,0)[7]                    : 783.3323
##  ARIMA(0,0,3)(2,1,0)[7] with drift         : 785.3224
##  ARIMA(0,0,4)(0,1,0)[7]                    : 806.8108
##  ARIMA(0,0,4)(0,1,0)[7] with drift         : 808.8108
##  ARIMA(0,0,4)(0,1,1)[7]                    : 780.9147
##  ARIMA(0,0,4)(0,1,1)[7] with drift         : 782.51
##  ARIMA(0,0,4)(1,1,0)[7]                    : 790.2486
```

```
##  ARIMA(0,0,4)(1,1,0)[7] with drift         : 792.2461
##  ARIMA(0,0,5)(0,1,0)[7]                    : 804.7502
##  ARIMA(0,0,5)(0,1,0)[7] with drift         : 806.7304
##  ARIMA(1,0,0)(0,1,0)[7]                    : 807.5078
##  ARIMA(1,0,0)(0,1,0)[7] with drift         : 809.4809
##  ARIMA(1,0,0)(0,1,1)[7]                    : 775.6301
##  ARIMA(1,0,0)(0,1,1)[7] with drift         : 777.225
##  ARIMA(1,0,0)(0,1,2)[7]                    : 777.6165
##  ARIMA(1,0,0)(0,1,2)[7] with drift         : 779.225
##  ARIMA(1,0,0)(1,1,0)[7]                    : 786.089
##  ARIMA(1,0,0)(1,1,0)[7] with drift         : 788.0747
##  ARIMA(1,0,0)(1,1,1)[7]                    : 777.6149
##  ARIMA(1,0,0)(1,1,1)[7] with drift         : 779.225
##  ARIMA(1,0,0)(1,1,2)[7]                    : Inf
##  ARIMA(1,0,0)(1,1,2)[7] with drift         : 781.2238
##  ARIMA(1,0,0)(2,1,0)[7]                    : 779.5255
##  ARIMA(1,0,0)(2,1,0)[7] with drift         : 781.5188
##  ARIMA(1,0,0)(2,1,1)[7]                    : 779.5261
##  ARIMA(1,0,0)(2,1,1)[7] with drift         : 781.125
##  ARIMA(1,0,0)(2,1,2)[7]                    : 781.6147
##  ARIMA(1,0,0)(2,1,2)[7] with drift         : 783.221
##  ARIMA(1,0,1)(0,1,0)[7]                    : 808.8825
##  ARIMA(1,0,1)(0,1,0)[7] with drift         : 810.8586
##  ARIMA(1,0,1)(0,1,1)[7]                    : 777.4803
##  ARIMA(1,0,1)(0,1,1)[7] with drift         : 779.028
##  ARIMA(1,0,1)(0,1,2)[7]                    : 779.479
##  ARIMA(1,0,1)(0,1,2)[7] with drift         : 781.0136
##  ARIMA(1,0,1)(1,1,0)[7]                    : 787.8497
##  ARIMA(1,0,1)(1,1,0)[7] with drift         : 789.8379
##  ARIMA(1,0,1)(1,1,1)[7]                    : 779.4788
##  ARIMA(1,0,1)(1,1,1)[7] with drift         : 781.0113
##  ARIMA(1,0,1)(1,1,2)[7]                    : 781.4766
##  ARIMA(1,0,1)(1,1,2)[7] with drift         : 783.0205
##  ARIMA(1,0,1)(2,1,0)[7]                    : 781.3321
##  ARIMA(1,0,1)(2,1,0)[7] with drift         : 783.3221
##  ARIMA(1,0,1)(2,1,1)[7]                    : 781.3449
##  ARIMA(1,0,1)(2,1,1)[7] with drift         : 782.8741
##  ARIMA(1,0,2)(0,1,0)[7]                    : 809.4195
##  ARIMA(1,0,2)(0,1,0)[7] with drift         : 811.4157
##  ARIMA(1,0,2)(0,1,1)[7]                    : 779.4823
##  ARIMA(1,0,2)(0,1,1)[7] with drift         : 781.0318
##  ARIMA(1,0,2)(0,1,2)[7]                    : 781.48
##  ARIMA(1,0,2)(0,1,2)[7] with drift         : 782.988
##  ARIMA(1,0,2)(1,1,0)[7]                    : 789.4938
##  ARIMA(1,0,2)(1,1,0)[7] with drift         : 791.4887
##  ARIMA(1,0,2)(1,1,1)[7]                    : 781.4506
##  ARIMA(1,0,2)(1,1,1)[7] with drift         : 782.986
##  ARIMA(1,0,2)(2,1,0)[7]                    : 782.9369
##  ARIMA(1,0,2)(2,1,0)[7] with drift         : 784.926
##  ARIMA(1,0,3)(0,1,0)[7]                    : 808.5682
##  ARIMA(1,0,3)(0,1,0)[7] with drift         : 810.5637
##  ARIMA(1,0,3)(0,1,1)[7]                    : 781.4141
##  ARIMA(1,0,3)(0,1,1)[7] with drift         : 782.989
##  ARIMA(1,0,3)(1,1,0)[7]                    : 791.315
```

```
## ARIMA(1,0,3)(1,1,0)[7] with drift         : 793.3112
## ARIMA(1,0,4)(0,1,0)[7]                    : 807.0685
## ARIMA(1,0,4)(0,1,0)[7] with drift         : 809.0655
## ARIMA(2,0,0)(0,1,0)[7]                    : 808.2362
## ARIMA(2,0,0)(0,1,0)[7] with drift         : 810.2176
## ARIMA(2,0,0)(0,1,1)[7]                    : 777.4986
## ARIMA(2,0,0)(0,1,1)[7] with drift         : 779.0429
## ARIMA(2,0,0)(0,1,2)[7]                    : 779.4939
## ARIMA(2,0,0)(0,1,2)[7] with drift         : 781.0365
## ARIMA(2,0,0)(1,1,0)[7]                    : 787.5917
## ARIMA(2,0,0)(1,1,0)[7] with drift         : 789.5839
## ARIMA(2,0,0)(1,1,1)[7]                    : 779.4932
## ARIMA(2,0,0)(1,1,1)[7] with drift         : 781.0354
## ARIMA(2,0,0)(1,1,2)[7]                    : 781.4901
## ARIMA(2,0,0)(1,1,2)[7] with drift         : 783.0415
## ARIMA(2,0,0)(2,1,0)[7]                    : 781.3118
## ARIMA(2,0,0)(2,1,0)[7] with drift         : 783.3009
## ARIMA(2,0,0)(2,1,1)[7]                    : 781.338
## ARIMA(2,0,0)(2,1,1)[7] with drift         : 782.8527
## ARIMA(2,0,1)(0,1,0)[7]                    : 809.5714
## ARIMA(2,0,1)(0,1,0)[7] with drift         : 811.5672
## ARIMA(2,0,1)(0,1,1)[7]                    : 779.4998
## ARIMA(2,0,1)(0,1,1)[7] with drift         : 781.0408
## ARIMA(2,0,1)(0,1,2)[7]                    : 781.4952
## ARIMA(2,0,1)(0,1,2)[7] with drift         : 783.0347
## ARIMA(2,0,1)(1,1,0)[7]                    : 789.2182
## ARIMA(2,0,1)(1,1,0)[7] with drift         : 791.2149
## ARIMA(2,0,1)(1,1,1)[7]                    : 781.4335
## ARIMA(2,0,1)(1,1,1)[7] with drift         : 782.983
## ARIMA(2,0,1)(2,1,0)[7]                    : 783.3088
## ARIMA(2,0,1)(2,1,0)[7] with drift         : 785.2993
## ARIMA(2,0,2)(0,1,0)[7]                    : 811.1701
## ARIMA(2,0,2)(0,1,0)[7] with drift         : 813.1666
## ARIMA(2,0,2)(0,1,1)[7]                    : 777.6733
## ARIMA(2,0,2)(0,1,1)[7] with drift         : 779.1775
## ARIMA(2,0,2)(1,1,0)[7]                    : Inf
## ARIMA(2,0,2)(1,1,0)[7] with drift         : Inf
## ARIMA(2,0,3)(0,1,0)[7]                    : 813.1684
## ARIMA(2,0,3)(0,1,0)[7] with drift         : 815.165
## ARIMA(3,0,0)(0,1,0)[7]                    : 810.2056
## ARIMA(3,0,0)(0,1,0)[7] with drift         : 812.1879
## ARIMA(3,0,0)(0,1,1)[7]                    : 779.4983
## ARIMA(3,0,0)(0,1,1)[7] with drift         : 781.0415
## ARIMA(3,0,0)(0,1,2)[7]                    : 781.4938
## ARIMA(3,0,0)(0,1,2)[7] with drift         : 783.0355
## ARIMA(3,0,0)(1,1,0)[7]                    : 788.9664
## ARIMA(3,0,0)(1,1,0)[7] with drift         : 790.9635
## ARIMA(3,0,0)(1,1,1)[7]                    : 781.4931
## ARIMA(3,0,0)(1,1,1)[7] with drift         : 783.0344
## ARIMA(3,0,0)(2,1,0)[7]                    : 783.2566
## ARIMA(3,0,0)(2,1,0)[7] with drift         : 785.2435
## ARIMA(3,0,1)(0,1,0)[7]                    : Inf
## ARIMA(3,0,1)(0,1,0)[7] with drift         : Inf
## ARIMA(3,0,1)(0,1,1)[7]                    : 781.4343
```

```
##  ARIMA(3,0,1)(0,1,1)[7] with drift         : 782.9956
##  ARIMA(3,0,1)(1,1,0)[7]                    : 790.8717
##  ARIMA(3,0,1)(1,1,0)[7] with drift         : 792.8698
##  ARIMA(3,0,2)(0,1,0)[7]                    : 813.167
##  ARIMA(3,0,2)(0,1,0)[7] with drift         : 815.1636
##  ARIMA(4,0,0)(0,1,0)[7]                    : 810.3548
##  ARIMA(4,0,0)(0,1,0)[7] with drift         : 812.3416
##  ARIMA(4,0,0)(0,1,1)[7]                    : 781.4421
##  ARIMA(4,0,0)(0,1,1)[7] with drift         : 782.9646
##  ARIMA(4,0,0)(1,1,0)[7]                    : 790.5389
##  ARIMA(4,0,0)(1,1,0)[7] with drift         : 792.538
##  ARIMA(4,0,1)(0,1,0)[7]                    : 812.0322
##  ARIMA(4,0,1)(0,1,0)[7] with drift         : 814.0178
##  ARIMA(5,0,0)(0,1,0)[7]                    : 811.0776
##  ARIMA(5,0,0)(0,1,0)[7] with drift         : 813.0551
##
##
##
##  Best model: ARIMA(0,0,1)(0,1,1)[7]
```

```
## Series: lettuce_train
## ARIMA(0,0,1)(0,1,1)[7]
##
## Coefficients:
##          ma1     sma1
##       0.1931  -0.7812
## s.e.  0.1115   0.1152
##
## sigma^2 estimated as 658.6:  log likelihood=-384.75
## AIC=775.49   AICc=775.8   BIC=782.71
```

```r
# Best model: ARIMA(0,0,1)(0,1,1)[7]  (AIC=775.49)
# Second best:  ARIMA(1,0,0)(0,1,1)[7]  (AIC=775.63)
# Third best: ARIMA(0,0,0)(0,1,1)[7] (AIC=776.33)
```

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(0,0,1)(0,1,1)[7] with AIC=775.49, ARIMA(1,0,0)(0,1,1)[7] with AIC=775.63 AND ARIMA(0,0,0)(0,1,1)[7] with AIC=776.33, as the candidate models that we would like to evaluate further.

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

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

```
##                       ME     RMSE      MAE     MPE     MAPE      MASE
## Training set 0.002828693 24.33113 17.78837 -2.7052 13.37245 0.6687969
##                     ACF1
## Training set 0.003568804
```

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

```
##                      ME   RMSE      MAE       MPE    MAPE      MASE      ACF1
## Training set 0.02288093 24.347 17.82533 -2.726029 13.3699 0.6701867 0.0132016
```

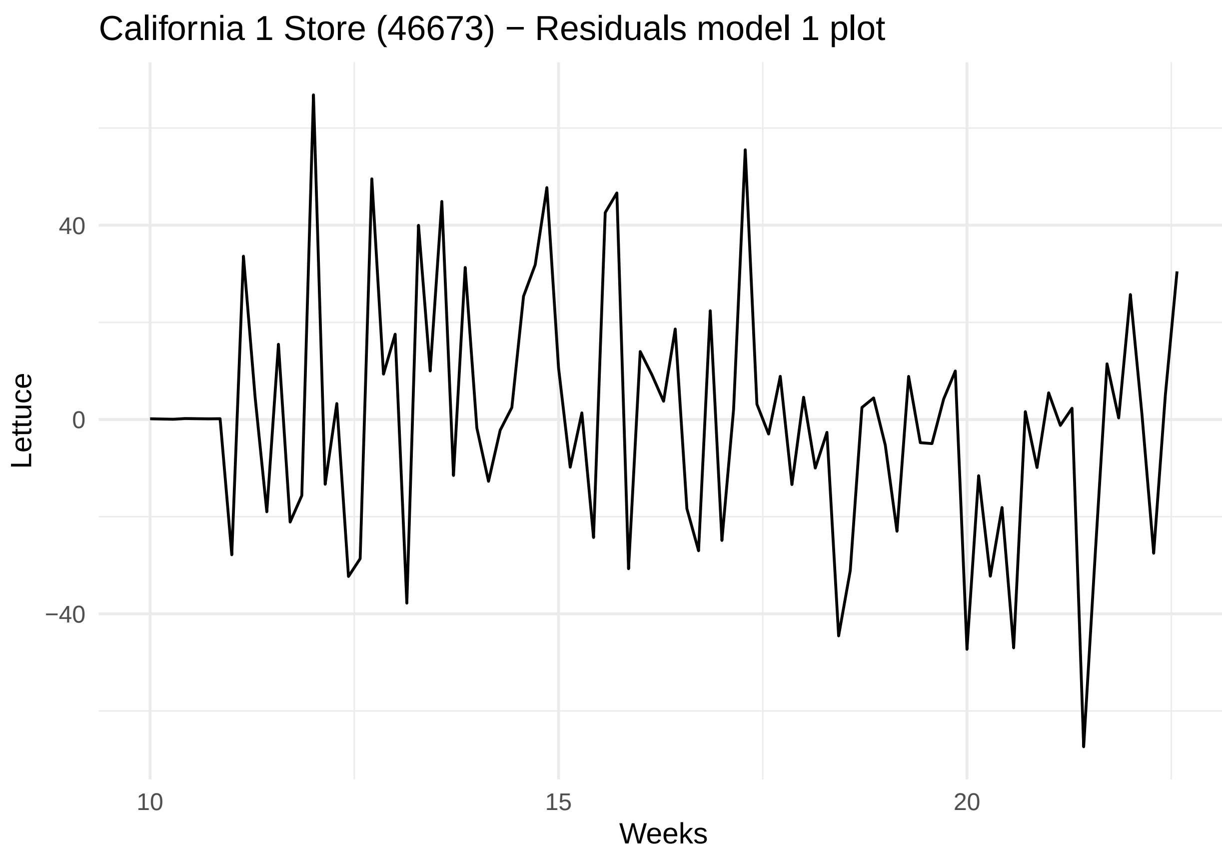
```
# in-sample one-step forecasts model 3
accuracy(lettuce.m3)
```

```
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.06320554 24.89233 18.27833 -2.935965 13.78152 0.6872184
##                   ACF1
## Training set 0.1811863
```

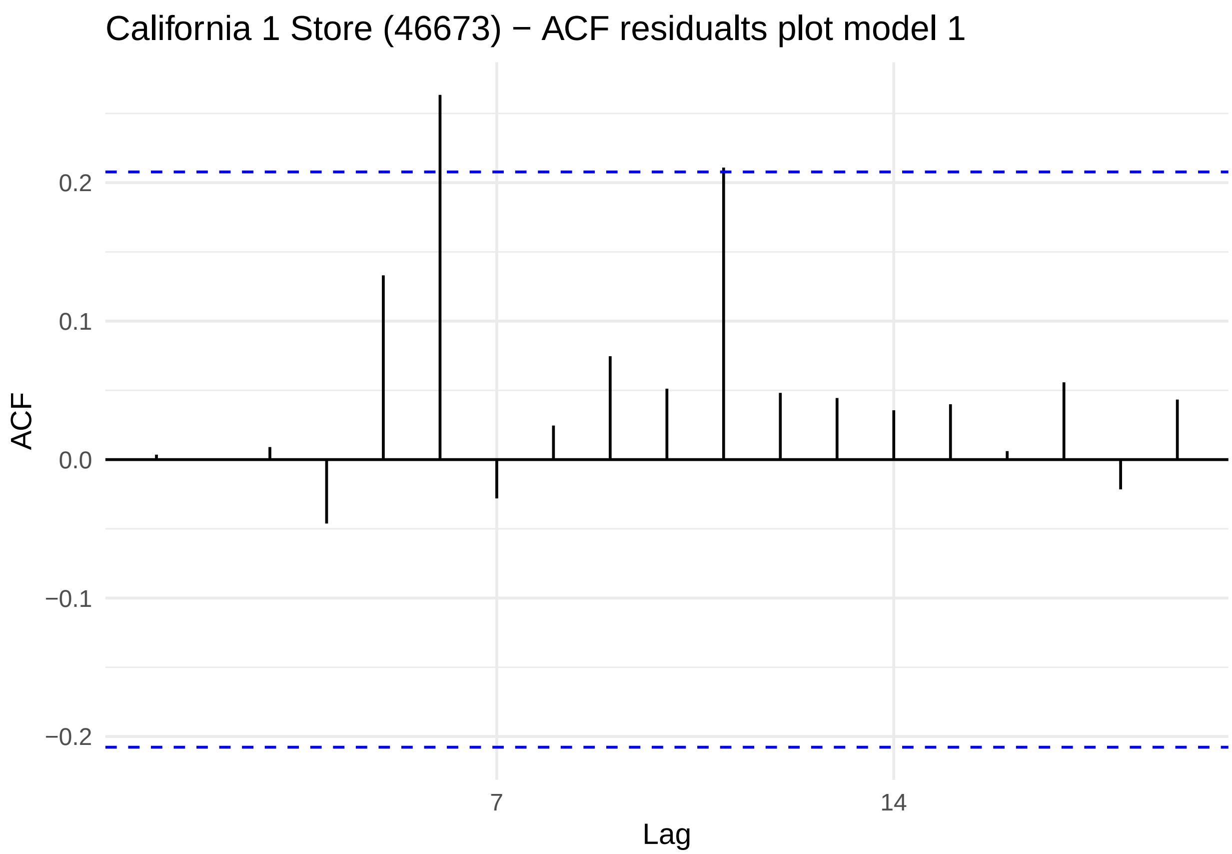
The first model even though it has both the lowest AIC score as well as the lowest RMSE.

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

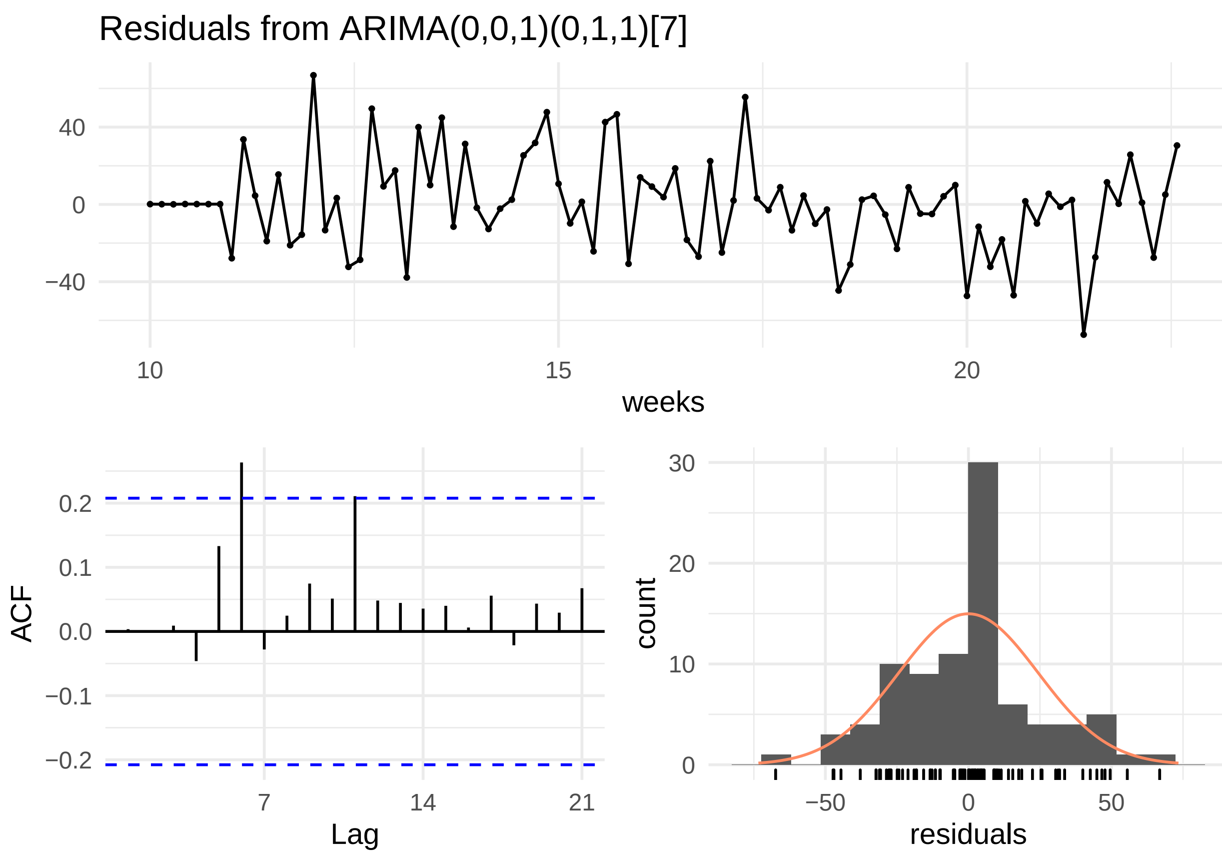
```
# residual analysis model 1
autoplot(lettuce.m1$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 1 Store (46673) - Residuals model 1 plot")
```




```
ggAcf(lettuce.m1$residuals) + theme_minimal() +
ggtitle("California 1 Store (46673) - ACF residualts plot model 1")
```

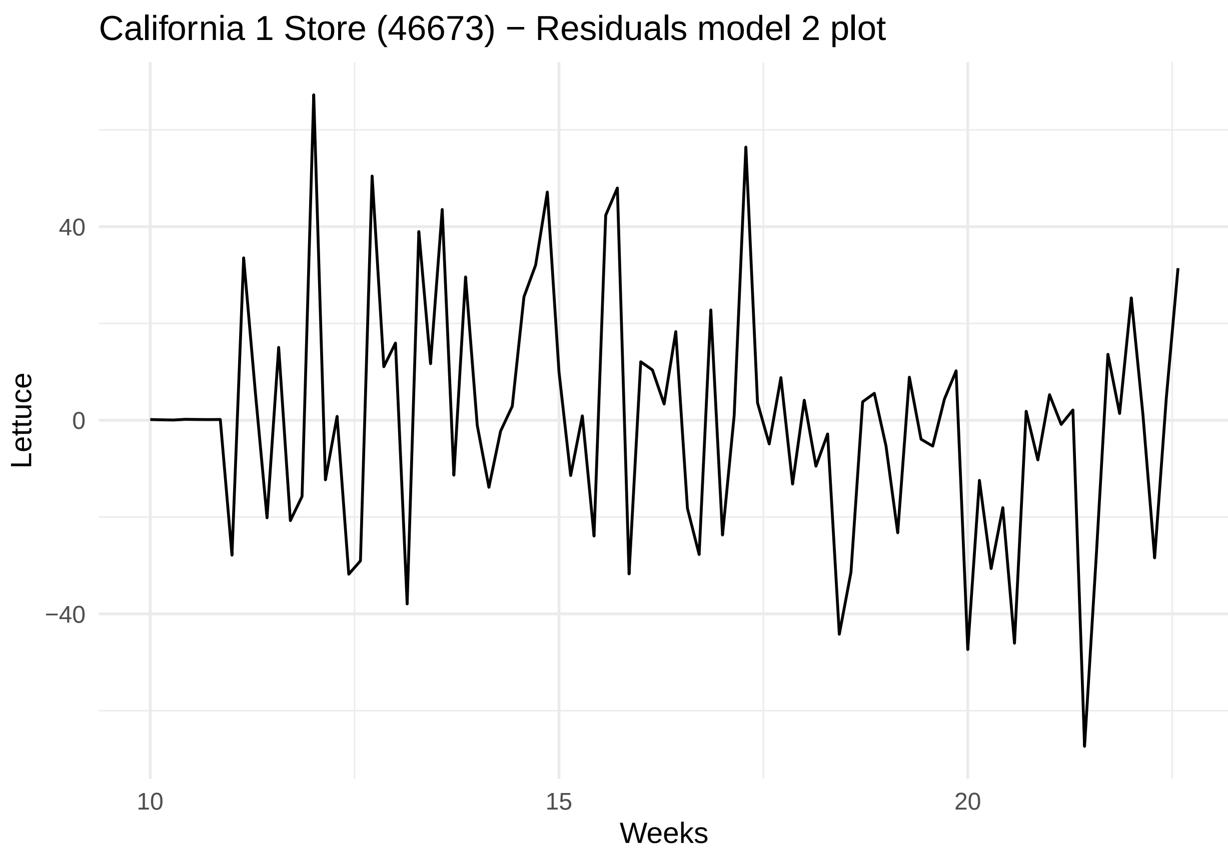



```r
checkresiduals(lettuce.m1, xlab = "weeks", theme = theme_minimal())
```

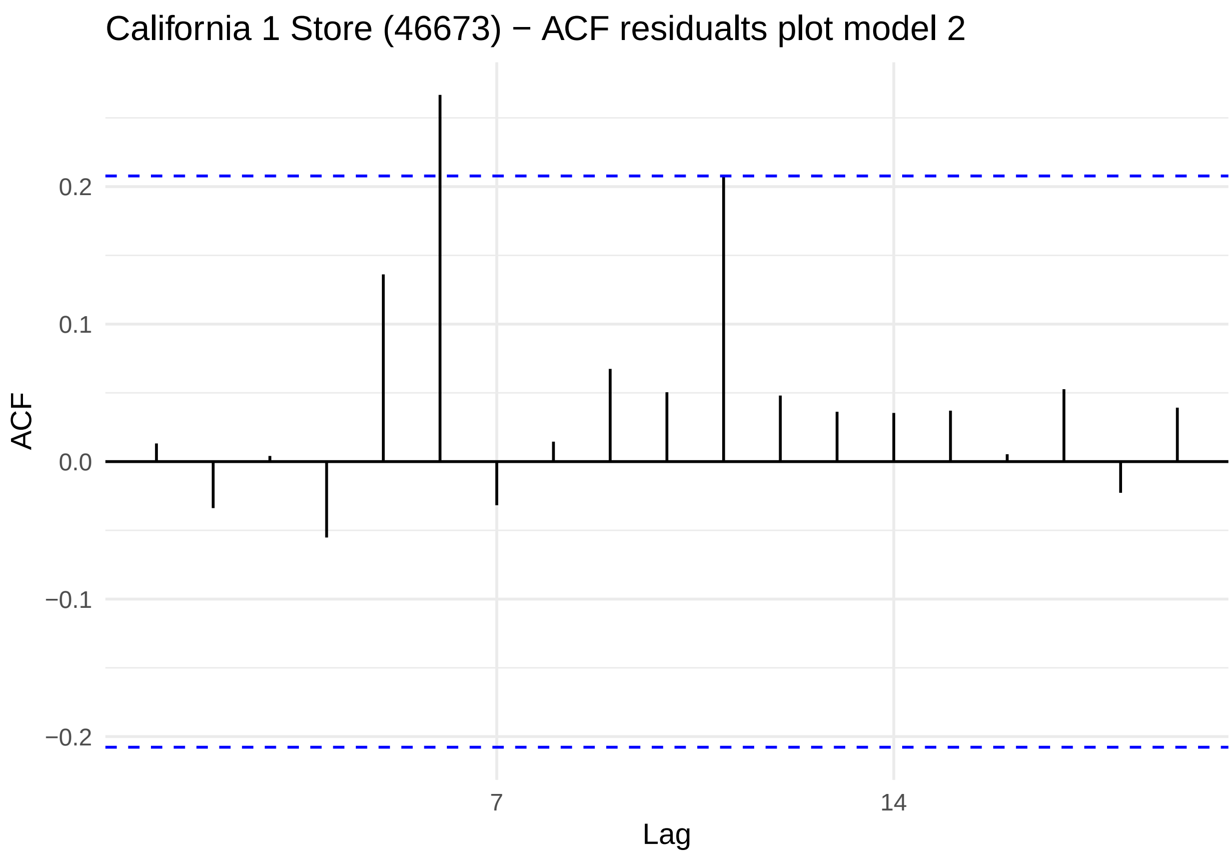



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1)(0,1,1)[7]
## Q* = 14.87, df = 12, p-value = 0.2486
##
## Model df: 2.   Total lags used: 14
```

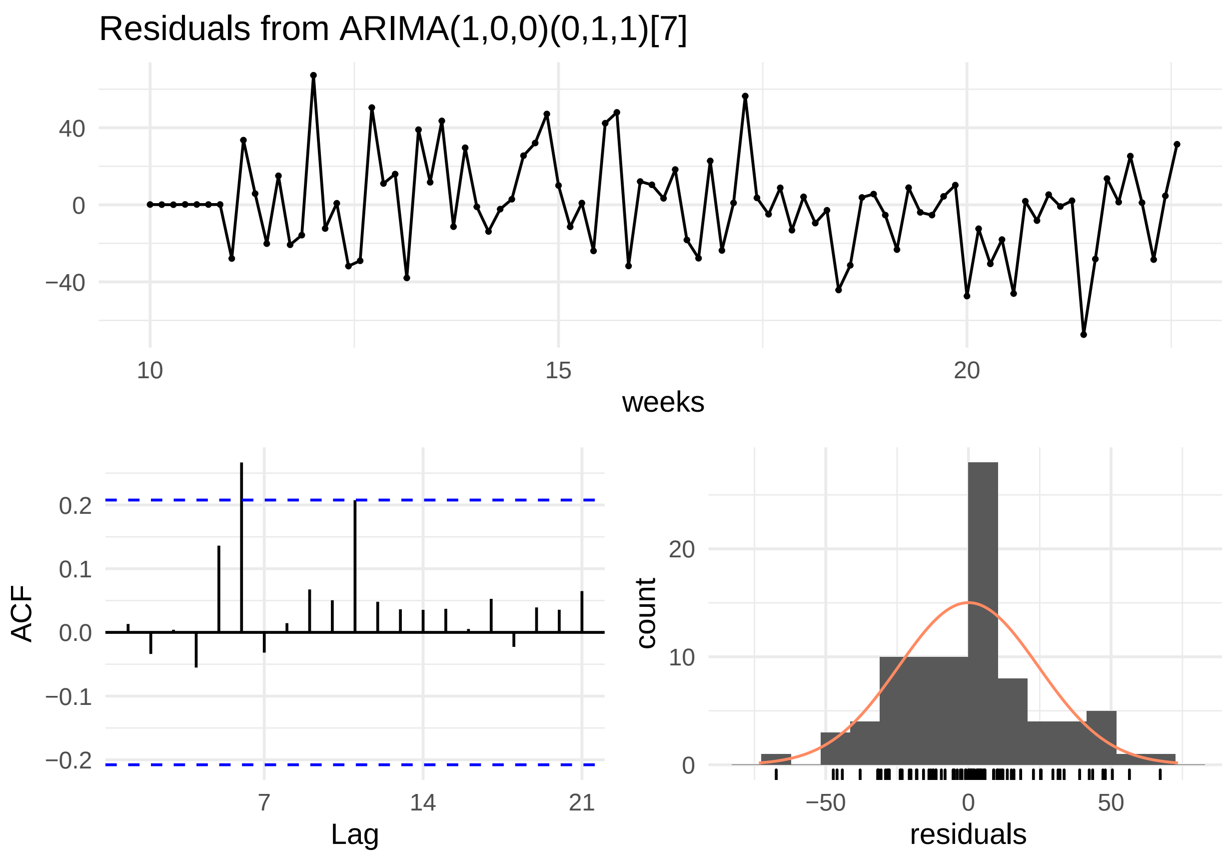
```
# residual analysis model 2
autoplot(lettuce.m2$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 1 Store (46673) - Residuals model 2 plot")
```




```
ggAcf(lettuce.m2$residuals) + theme_minimal() +
ggtitle("California 1 Store (46673) - ACF residualts plot model 2")
```

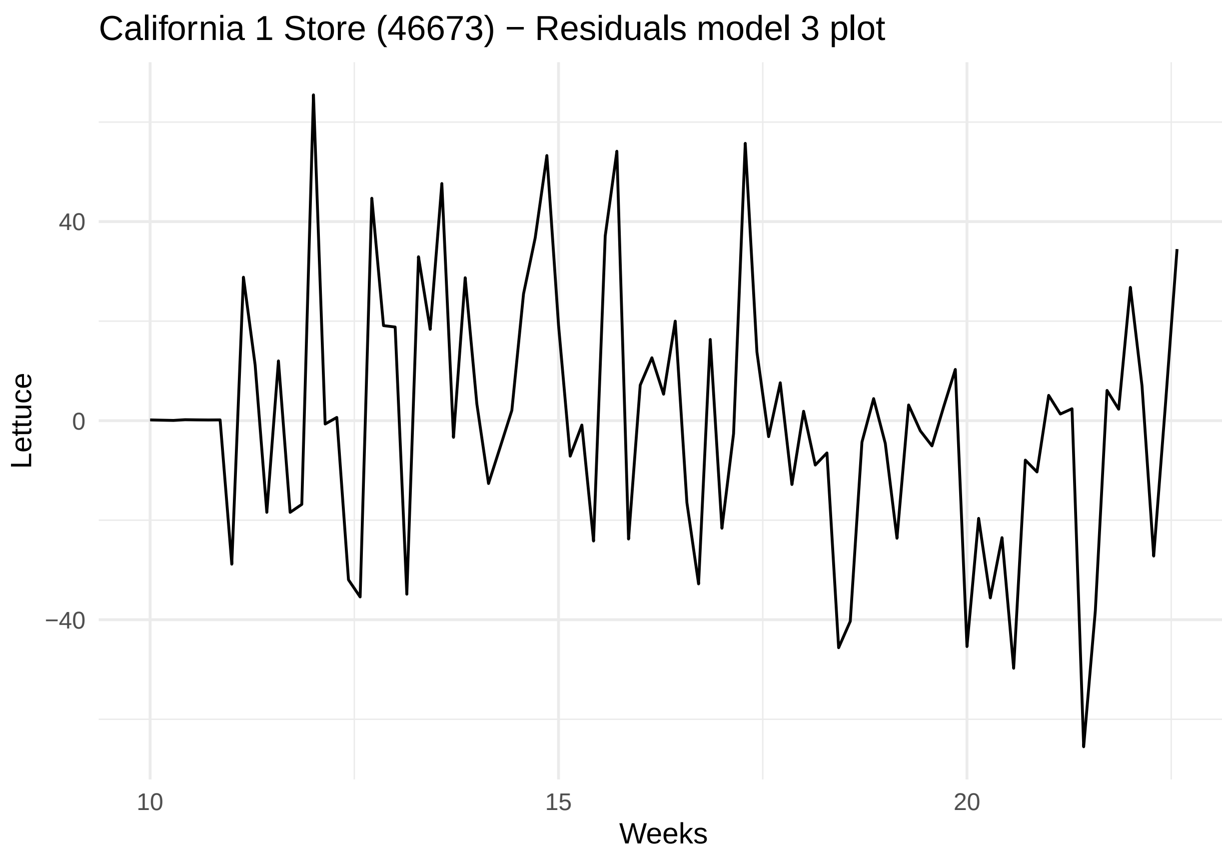



```r
checkresiduals(lettuce.m2, xlab = "weeks", theme = theme_minimal())
```

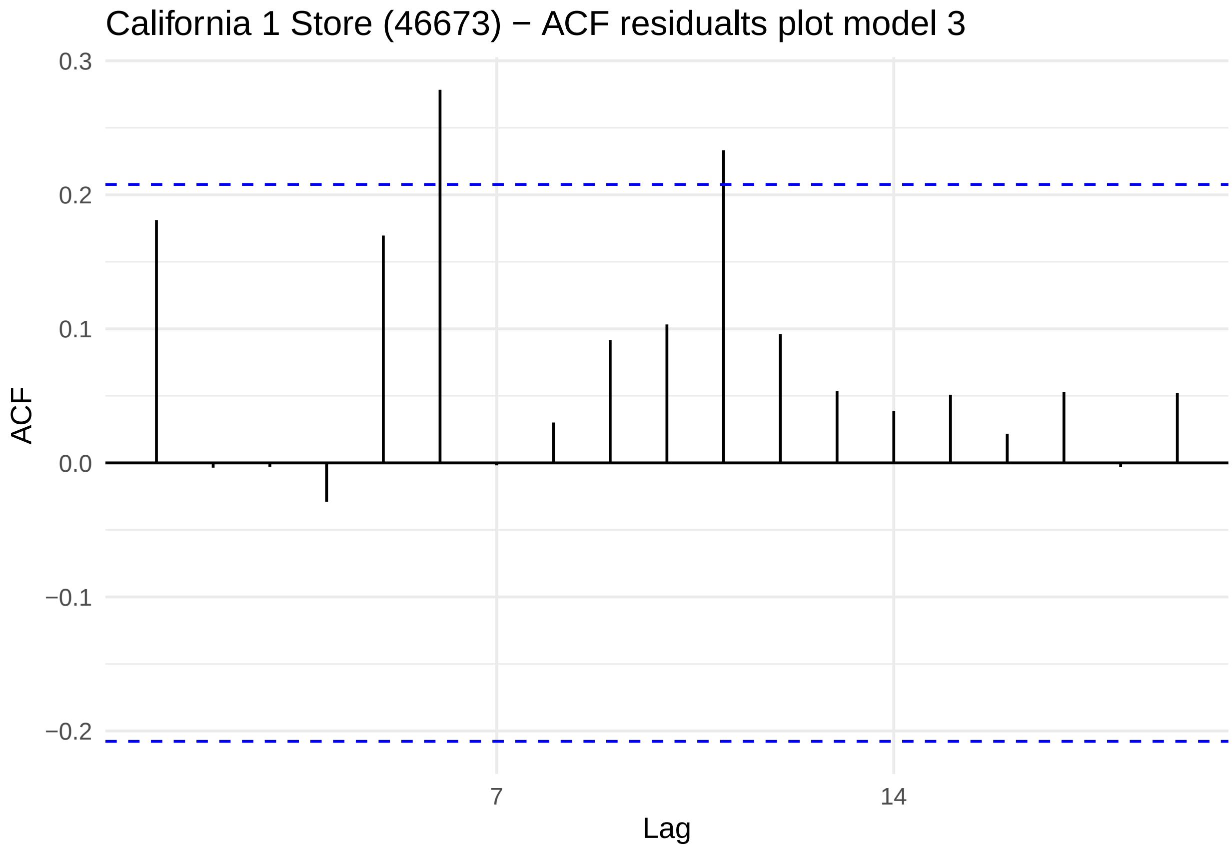



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,1)[7]
## Q* = 14.974, df = 12, p-value = 0.2428
##
## Model df: 2.   Total lags used: 14
```

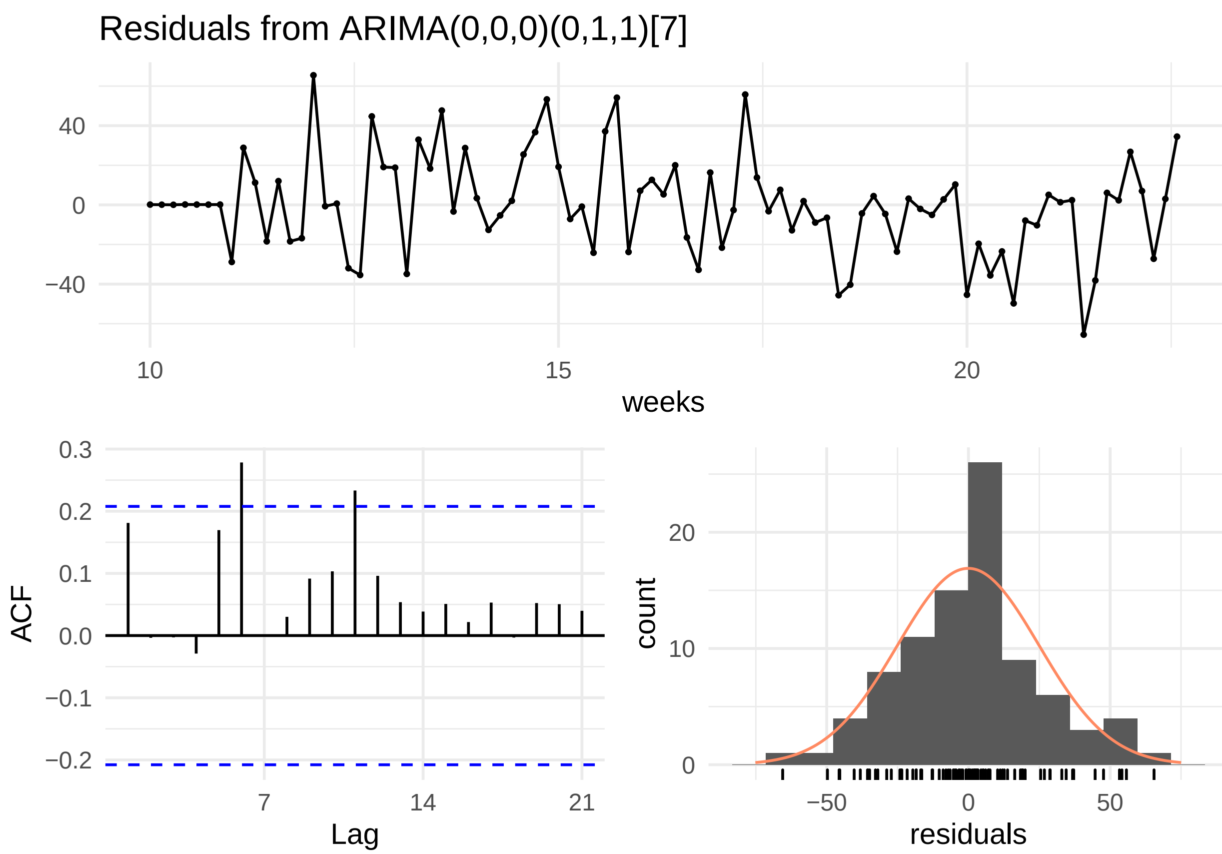
```
# residual analysis model 3
autoplot(lettuce.m3$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 1 Store (46673) - Residuals model 3 plot")
```




```
ggAcf(lettuce.m3$residuals) + theme_minimal() +
ggtitle("California 1 Store (46673) - ACF residualts plot model 3")
```



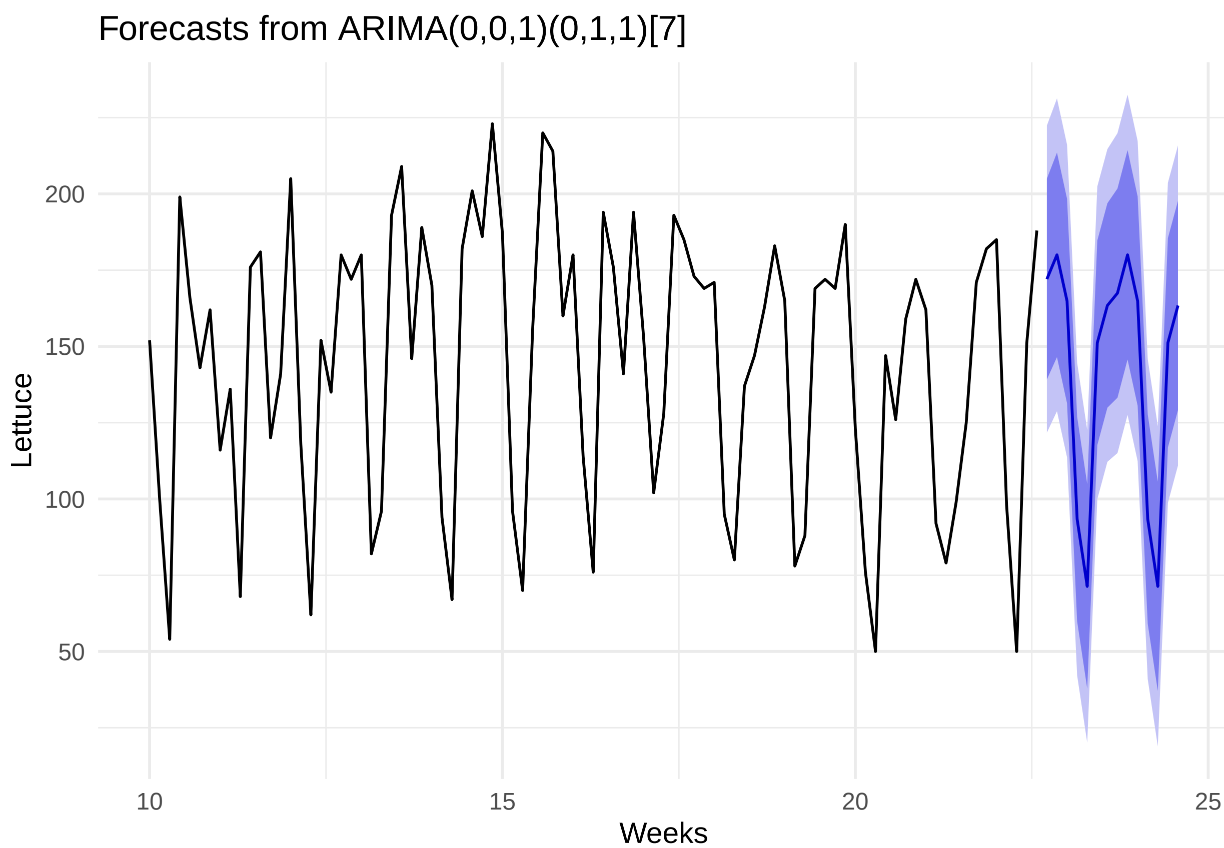

```r
checkresiduals(lettuce.m3, xlab = "weeks", theme = theme_minimal())
```



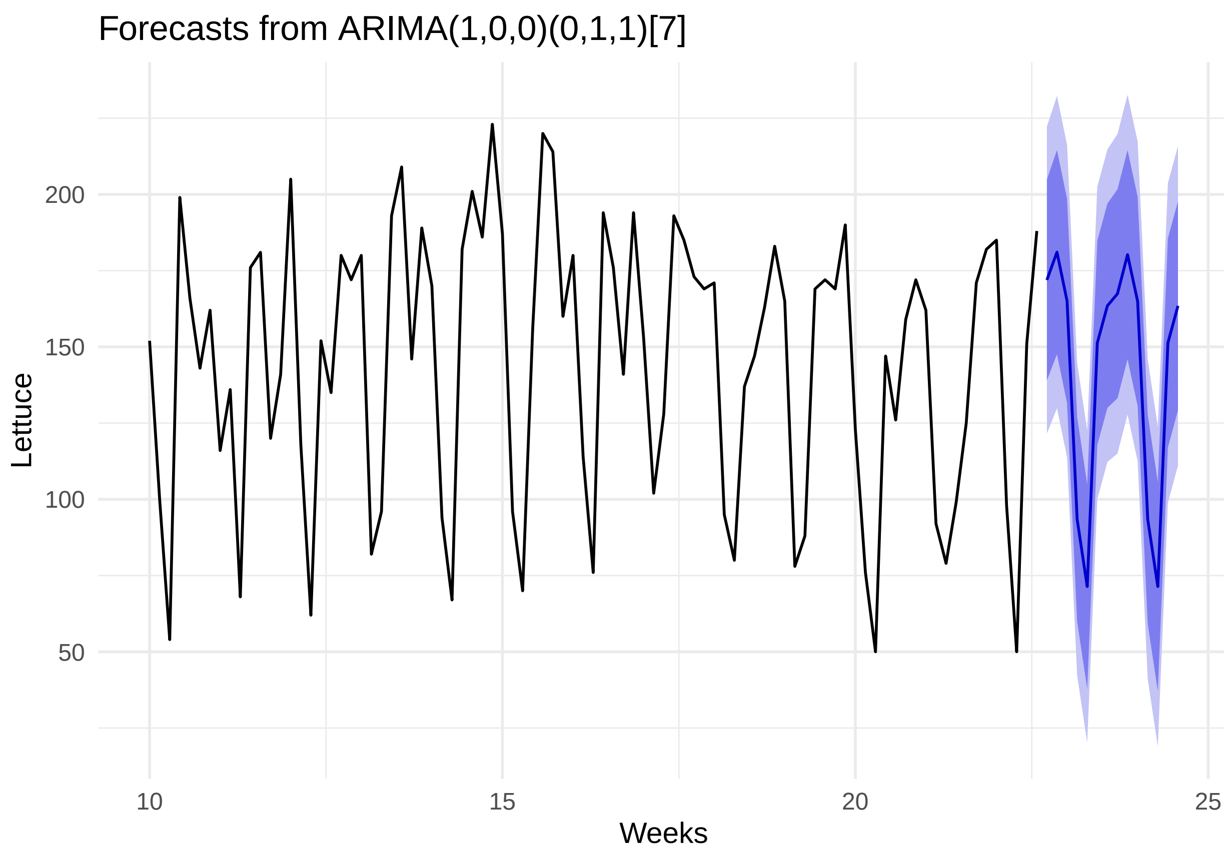

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0)(0,1,1)[7]
## Q* = 22.566, df = 13, p-value = 0.04719
##
## Model df: 1.   Total lags used: 14
```

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

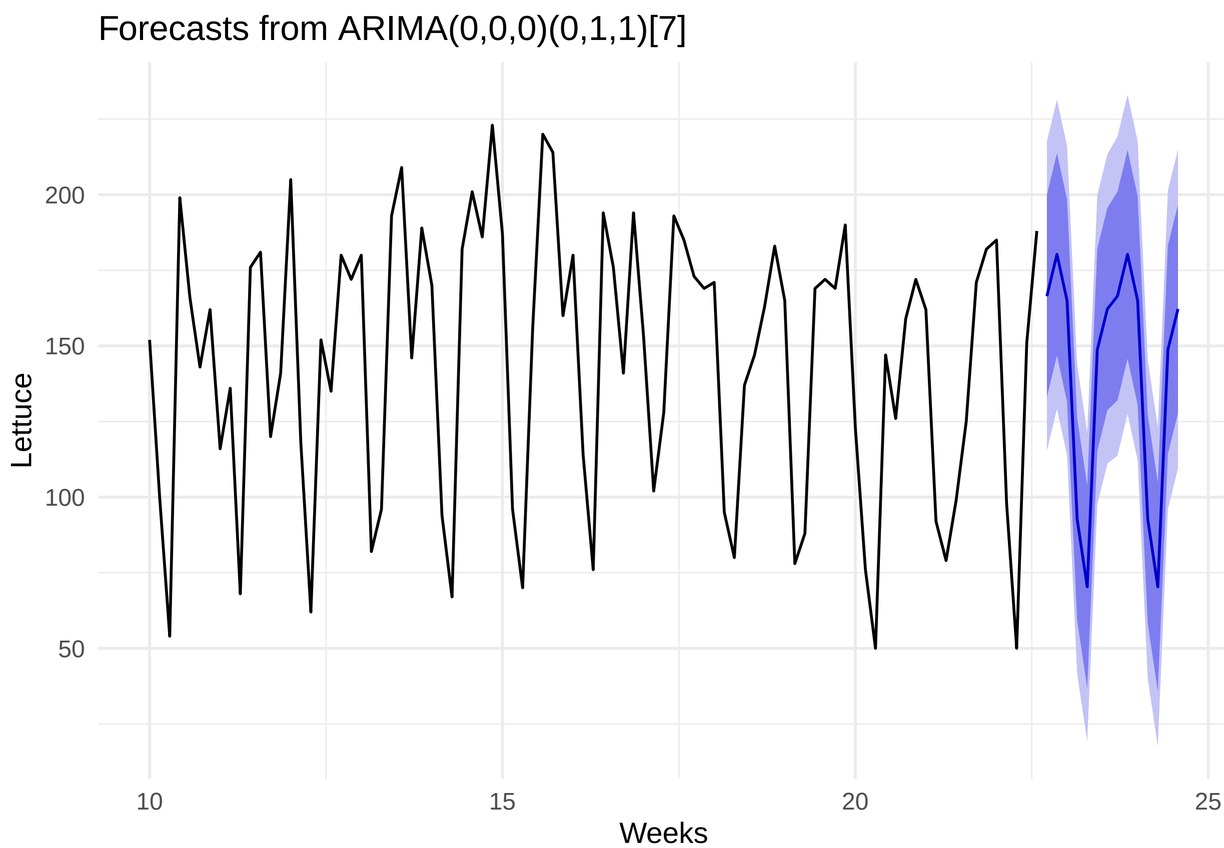
```
#Forecasting part model 1
lettuce.f1 <- forecast(lettuce.m1, h = 14)
autoplot(lettuce.f1, xlab = "Weeks", ylab = "Lettuce") + theme_minimal()
```




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




```
#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 0.002828693 24.33113 17.78837 -2.70520 13.37245 0.6687969
## Test set     7.493254846 41.08078 30.72942  2.31373 19.62275 1.1553471
##                     ACF1 Theil's U
## Training set 0.003568804        NA
## Test set     0.171116665 0.6433336
```

```
# 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 0.02288093 24.34700 17.82533 -2.726029 13.36990 0.6701867
## Test set     7.36559725 41.09264 30.77366  2.230649 19.64924 1.1570106
##                    ACF1 Theil's U
```

```
## Training set 0.0132016        NA
## Test set     0.1722622 0.6432981
```

```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 -0.06320554 24.89233 18.27833 -2.935965 13.78152 0.6872184
## Test set      8.67456799 41.49454 30.70063  3.132855 19.59288 1.1542648
##                   ACF1 Theil's U
## Training set 0.1811863        NA
## Test set     0.1803610 0.6557006
```

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(0, 0, 1),
                    seasonal = list(order = c(0, 1, 1), 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      175.90960 141.15504 210.6642 122.75709 229.0621
## 24.85714      173.09271 137.65046 208.5350 118.88847 227.2970
## 25.00000      164.46665 129.10022 199.8331 110.37836 218.5549
## 25.14286      100.73332  65.36689 136.0998  46.64503 154.8216
## 25.28571       76.66666  41.30022 112.0331  22.57836 130.7550
## 25.42857      168.66665 133.30022 204.0331 114.57836 222.7549
## 25.57143      176.53332 141.16688 211.8998 122.44502 230.6216
## 25.71429      161.06062 125.61838 196.5029 106.85638 215.2649
## 25.85714      173.09271 137.65046 208.5350 118.88847 227.2970
## 26.00000      164.46665 129.10022 199.8331 110.37836 218.5549
## 26.14286      100.73332  65.36689 136.0998  46.64503 154.8216
## 26.28571       76.66666  41.30022 112.0331  22.57836 130.7550
## 26.42857      168.66665 133.30022 204.0331 114.57836 222.7549
## 26.57143      176.53332 141.16688 211.8998 122.44502 230.6216
```

We present our forecast through ARIMA(0,0,1)(0,1,1) 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_California1_arima <- cbind(next2weeks, forecast_data$`Point Forecast`)
final_forecast_California1_arima
```

```
##    day forecast_data$`Point Forecast`
## 1    1                     175.90960
## 2    2                     173.09271
## 3    3                     164.46665
## 4    4                     100.73332
## 5    5                      76.66666
## 6    6                     168.66665
## 7    7                     176.53332
## 8    8                     161.06062
## 9    9                     173.09271
## 10  10                     164.46665
## 11  11                     100.73332
## 12  12                      76.66666
## 13  13                     168.66665
## 14  14                     176.53332
```

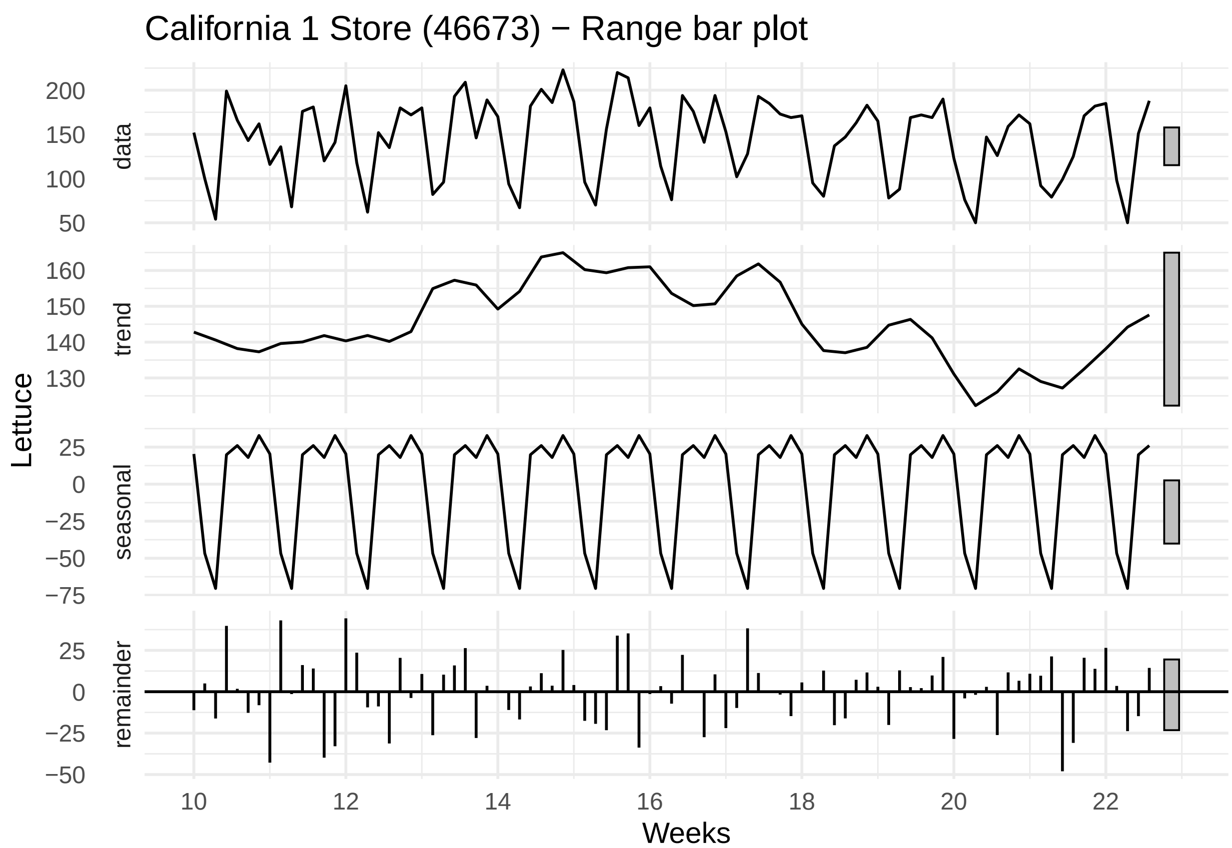
Holt-Winters

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("California 1 Store (46673) - 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.

On the other hand, the grey bar of the seasonal panel is small, even smaller than the grey bar of random error term, which indicates that seasonal component contributes to a great proportion of variations in the time series. In other words, it indicates that there is strong 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 = 0.0946
##     gamma = 1e-04
```

```
##
##   Initial states:
##     l = 146.0165
##     s = 32.6444 19.1329 25.5759 20.8428 -70.5411 -47.1212
##            19.4663
##
##   sigma:  24.3637
##
##      AIC     AICc      BIC
##  978.3710  981.1915 1003.2574
```

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 = 0.0946
##     gamma = 1e-04
##
##   Initial states:
##     l = 146.0165
##     s = 32.6444 19.1329 25.5759 20.8428 -70.5411 -47.1212
##            19.4663
##
##   sigma:  24.3637
##
##      AIC     AICc      BIC
##  978.3710  981.1915 1003.2574
```

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      ACF1
## Training set -1.014639 23.09898 18.47706 -3.708049 14.26531 0.6946899 0.0879864
```

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      156.60619 125.38288 187.82950 108.85426 204.3581
## 22.85714      170.11891 138.75626 201.48156 122.15388 218.0839
## 23.00000      156.94131 125.43993 188.44268 108.76411 205.1185
## 23.14286       90.35365  58.71416 121.99314  41.96522 138.7421
## 23.28571       66.93300  35.15599  98.71000  18.33426 115.5317
## 23.42857      158.31624 126.40231 190.23017 109.50810 207.1244
## 23.57143      163.05173 131.00118 195.10229 114.03465 212.0688
## 23.71429      156.60619 124.41988 188.79250 107.38148 205.8309
## 23.85714      170.11891 137.79741 202.44041 120.68744 219.5504
## 24.00000      156.94131 124.48518 189.39743 107.30395 206.5787
## 24.14286       90.35365  57.76345 122.94384  40.51124 140.1960
## 24.28571       66.93300  34.20928  99.65671  16.88639 116.9796
## 24.42857      158.31624 125.45955 191.17293 108.06627 208.5662
## 24.57143      163.05173 130.06232 196.04115 112.59878 213.5047
```

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

```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 -1.014639 23.09898 18.47706 -3.708049 14.26531 0.6946899
## Test set     12.025567 38.53718 28.55525  5.920676 18.50333 1.0736040
##                    ACF1 Theil's U
## Training set 0.08798640        NA
## Test set     0.07795779  0.608142
```

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 = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 145.9345
##     s = 27.1516 13.8629 29.8353 24.1902 -69.0184 -44.6583
##            18.6367
##
##   sigma:  26.6343
##
##      AIC     AICc      BIC
## 1164.093 1166.484 1190.440
```

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      159.79583 125.66254 193.9291 107.59347 211.9982
## 24.85714      173.08458 138.95130 207.2179 120.88223 225.2869
## 25.00000      164.56924 130.43595 198.7025 112.36688 216.7716
## 25.14286      101.27350  67.14021 135.4068  49.07114 153.4758
## 25.28571       76.91392  42.78064 111.0472  24.71157 129.1163
## 25.42857      170.12068 135.98740 204.2540 117.91833 222.3230
## 25.57143      175.76910 141.63581 209.9024 123.56674 227.9715
## 25.71429      159.79583 125.66254 193.9291 107.59347 211.9982
## 25.85714      173.08458 138.95130 207.2179 120.88223 225.2869
## 26.00000      164.56924 130.43595 198.7025 112.36688 216.7716
## 26.14286      101.27350  67.14021 135.4068  49.07114 153.4759
## 26.28571       76.91392  42.78064 111.0472  24.71156 129.1163
## 26.42857      170.12068 135.98740 204.2540 117.91833 222.3230
## 26.57143      175.76910 141.63581 209.9024 123.56674 227.9715
```

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_California1_ets <- cbind(next2weeks, forecast_data$`Point Forecast`)
final_forecast_California1_ets
```

```
##    day forecast_data$`Point Forecast`
## 1    1                      159.79583
## 2    2                      173.08458
## 3    3                      164.56924
## 4    4                      101.27350
## 5    5                       76.91392
## 6    6                      170.12068
## 7    7                      175.76910
## 8    8                      159.79583
## 9    9                      173.08458
## 10  10                      164.56924
## 11  11                      101.27350
## 12  12                       76.91392
## 13  13                      170.12068
## 14  14                      175.76910
```

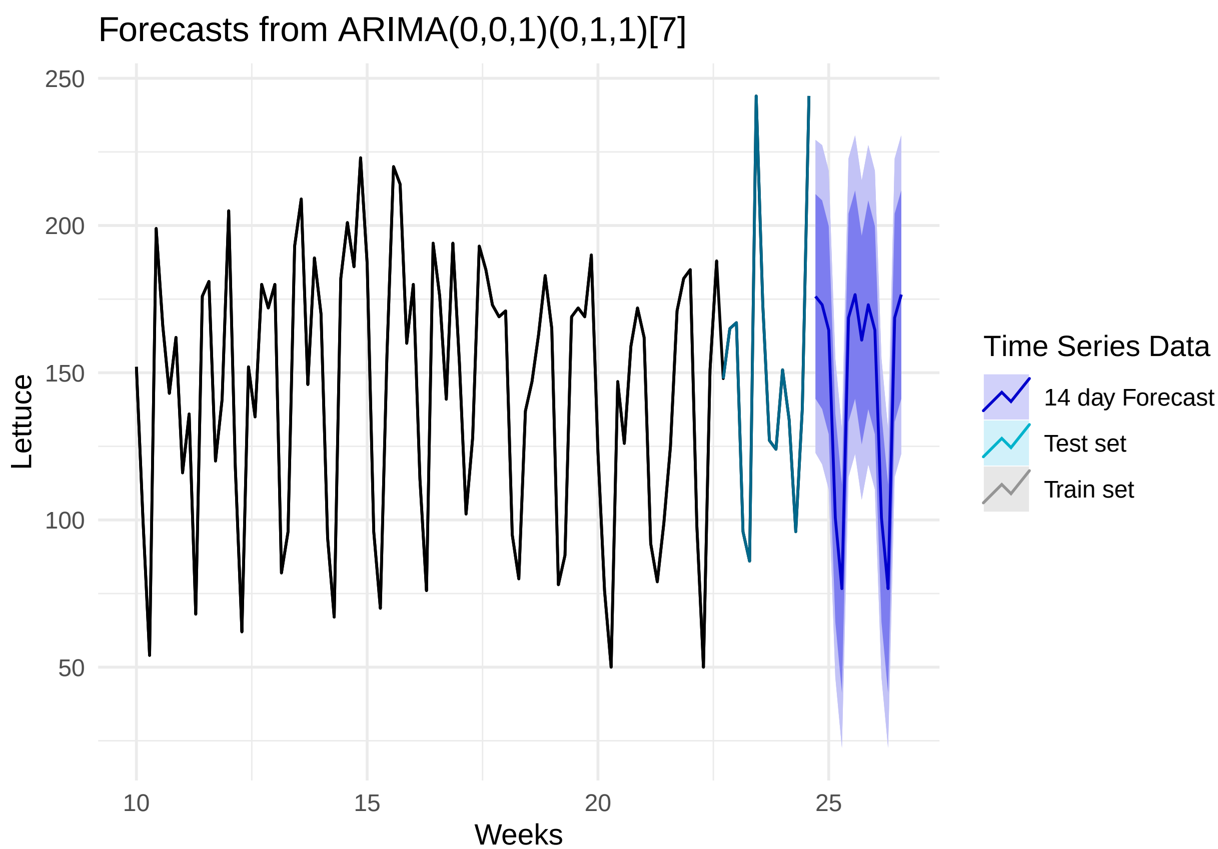
Comparison

Now we will compare the two best models for California1 Store (46673).

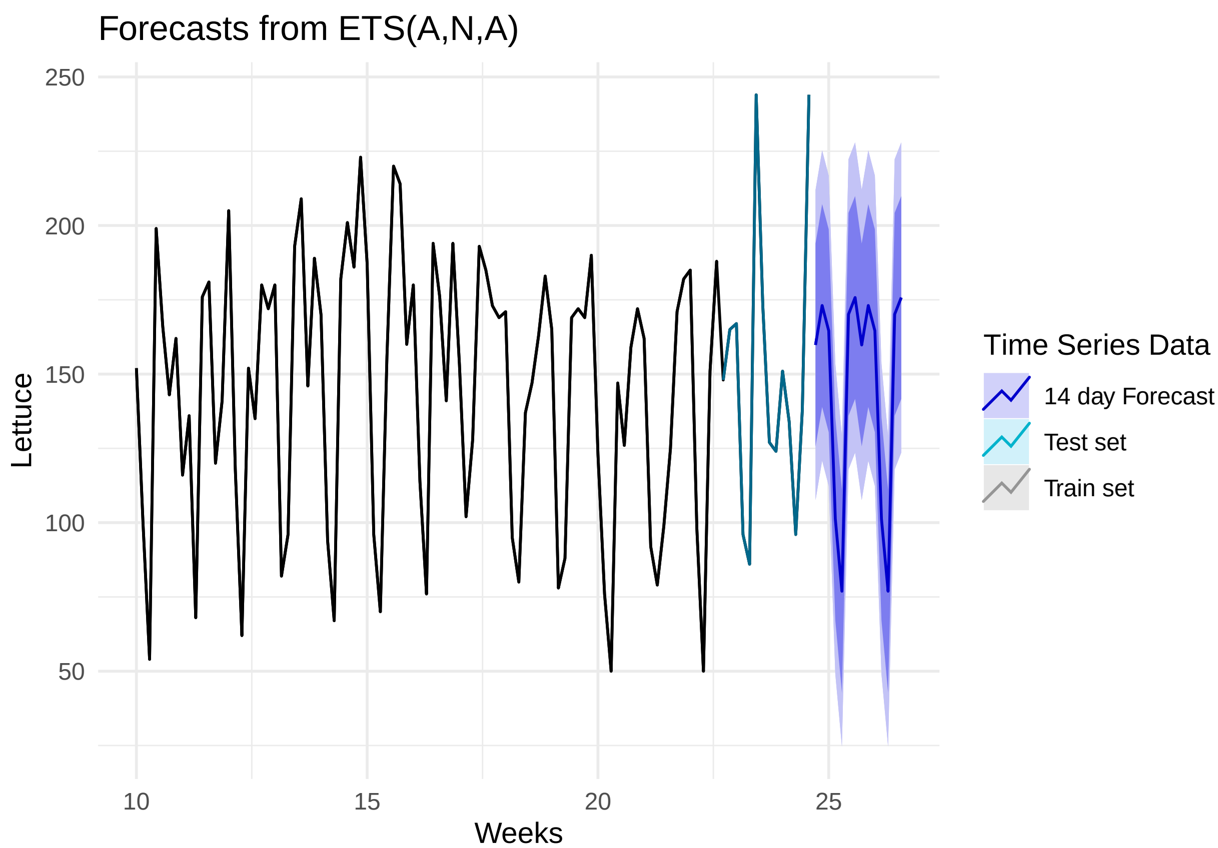
We plot time series data for train and test set and also the forecasts from our two models as indicated below:

```
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(0,0,1)(0,1,1) 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 -1.014639 23.09898 18.47706 -3.708049 14.26531 0.6946899
## Test set     12.025567 38.53718 28.55525  5.920676 18.50333 1.0736040
##                    ACF1 Theil's U
## Training set 0.08798640        NA
## Test set     0.07795779  0.608142
```

```
# best arima model
# ARIMA(0,0,1)(0,1,1)
accuracy.m1
```

```
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.002828693 24.33113 17.78837 -2.70520 13.37245 0.6687969
## Test set     7.493254846 41.08078 30.72942  2.31373 19.62275 1.1553471
##                     ACF1 Theil's U
## Training set 0.003568804        NA
## Test set     0.171116665 0.6433336
```

We can observe that ETS(A,N,A) has a better (lower) RMSE (38.53718 vs 41.08078) respectively.

Therefore, we choose the ETS(A,N,A) for California1 (46673) store.

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

```
final_forecast_California1_ets
```

```
##    day forecast_data$`Point Forecast`
## 1    1                      159.79583
## 2    2                      173.08458
## 3    3                      164.56924
## 4    4                      101.27350
## 5    5                       76.91392
## 6    6                      170.12068
## 7    7                      175.76910
## 8    8                      159.79583
## 9    9                      173.08458
## 10  10                      164.56924
## 11  11                      101.27350
## 12  12                       76.91392
## 13  13                      170.12068
## 14  14                      175.76910
```