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

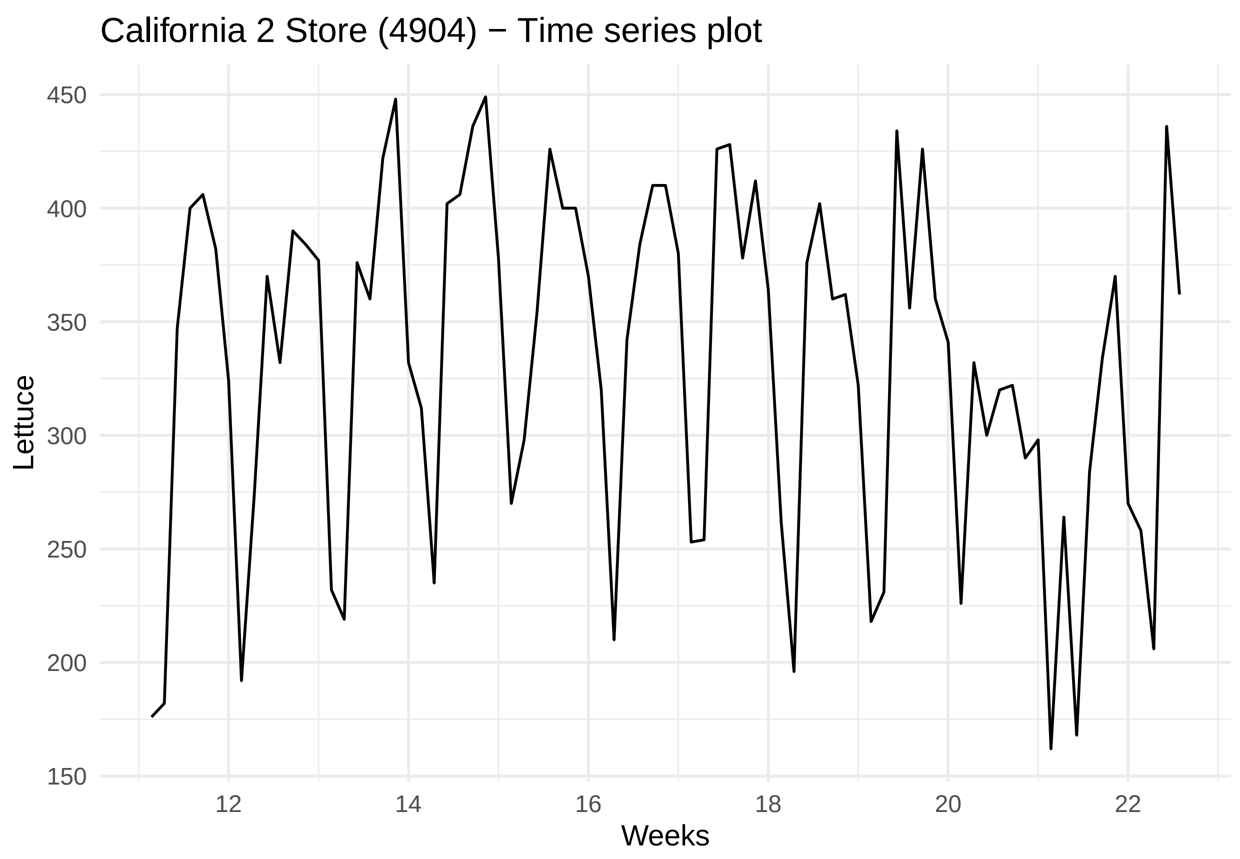
Then, we load and split the data set into train and test set.

```
# read csv file
data <- read.csv(file = "California2_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(11, 2)) # 11th week 2nd day

# split data set into train and test set
lettuce_train <- subset(lettuce, end = 81)
lettuce_test <- subset(lettuce, start = 82) # last 14 lines-days
```

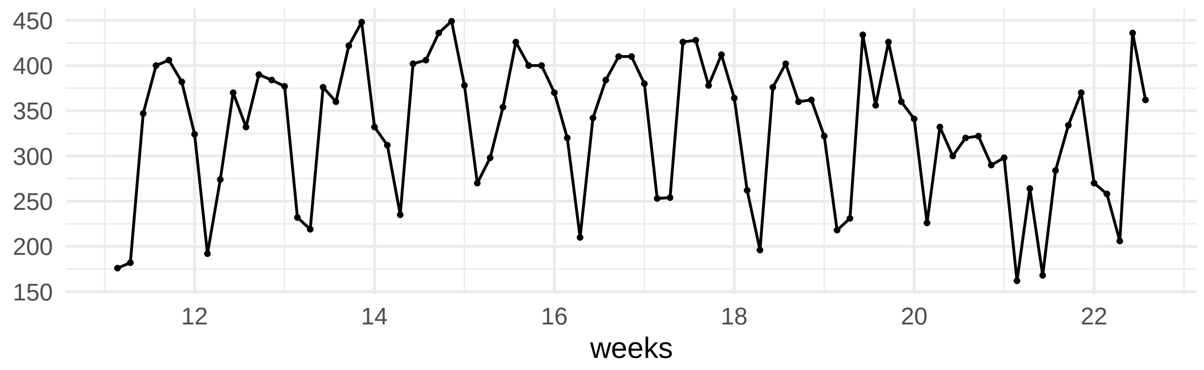
ARIMA

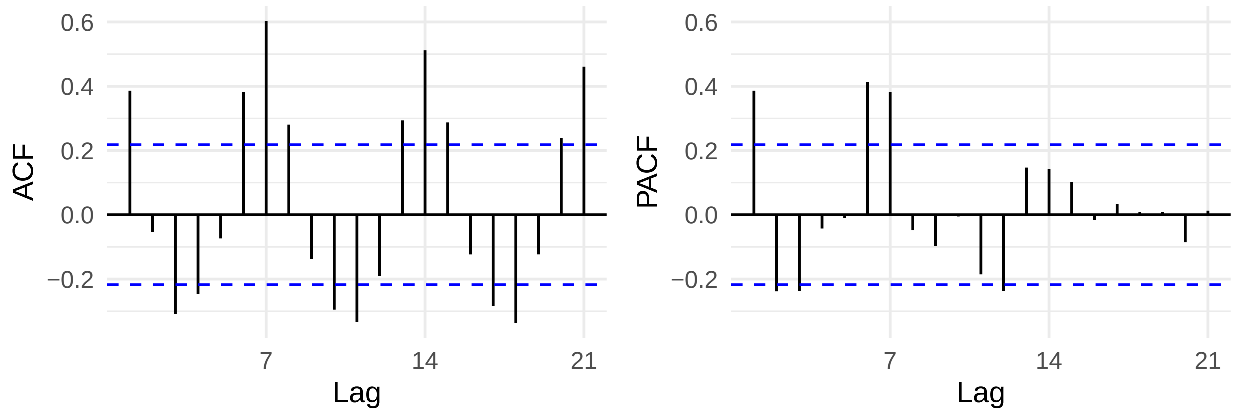
We visually inspect the time series.

```
autoplot(lettuce_train, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 2 Store (4904) - 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.

```
# 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 = -5.1903, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
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) = -44.411, 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 = 0.28583, 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] 1
```

We need to differentiate for seasonality one time.

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

Check again the tests for stationarity:

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  lettuce_train.diff1
## Dickey-Fuller = -0.98038, Lag order = 4, p-value = 0.9349
## 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) = -75.208, 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 = 0.15501, Truncation lag parameter = 3, p-value = 0.1
```

Check again the two automatic functions for stationarity:

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

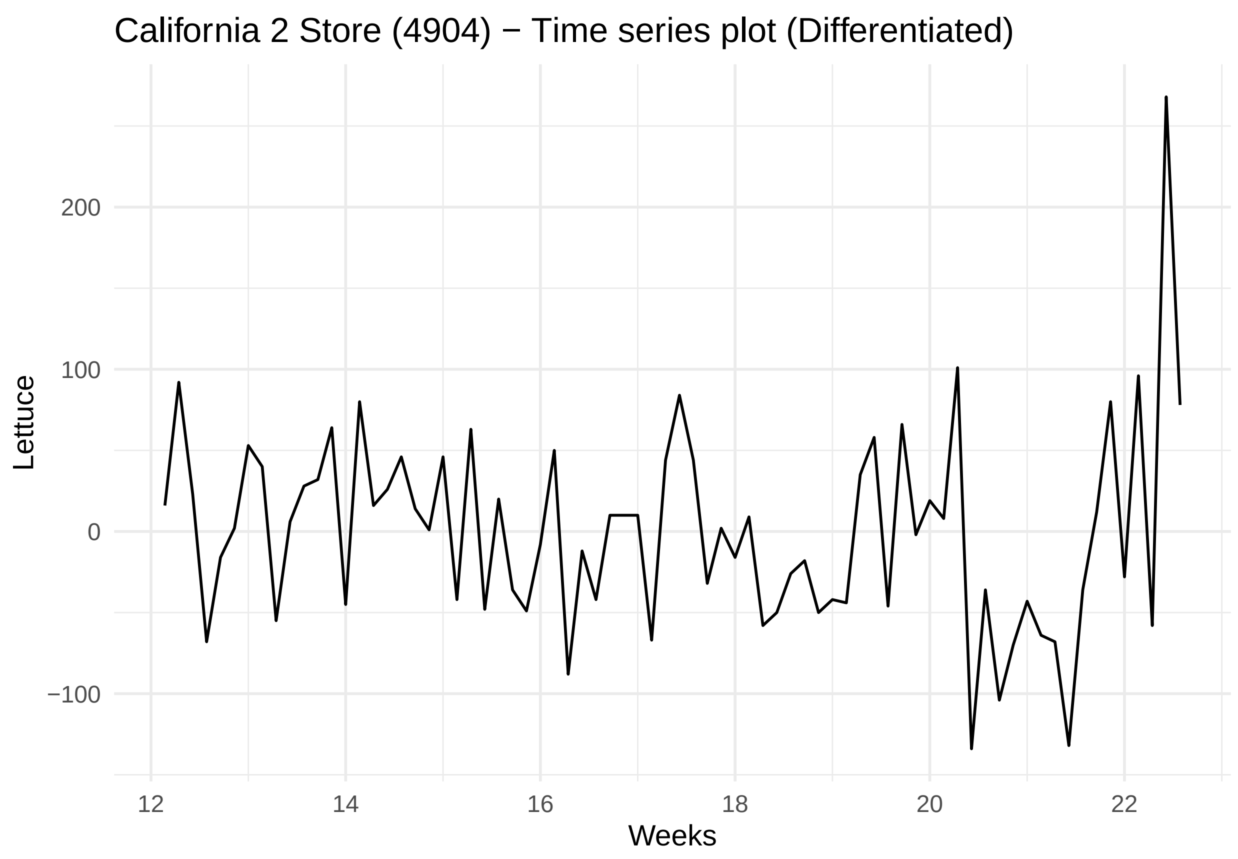
```
## [1] 0
```

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

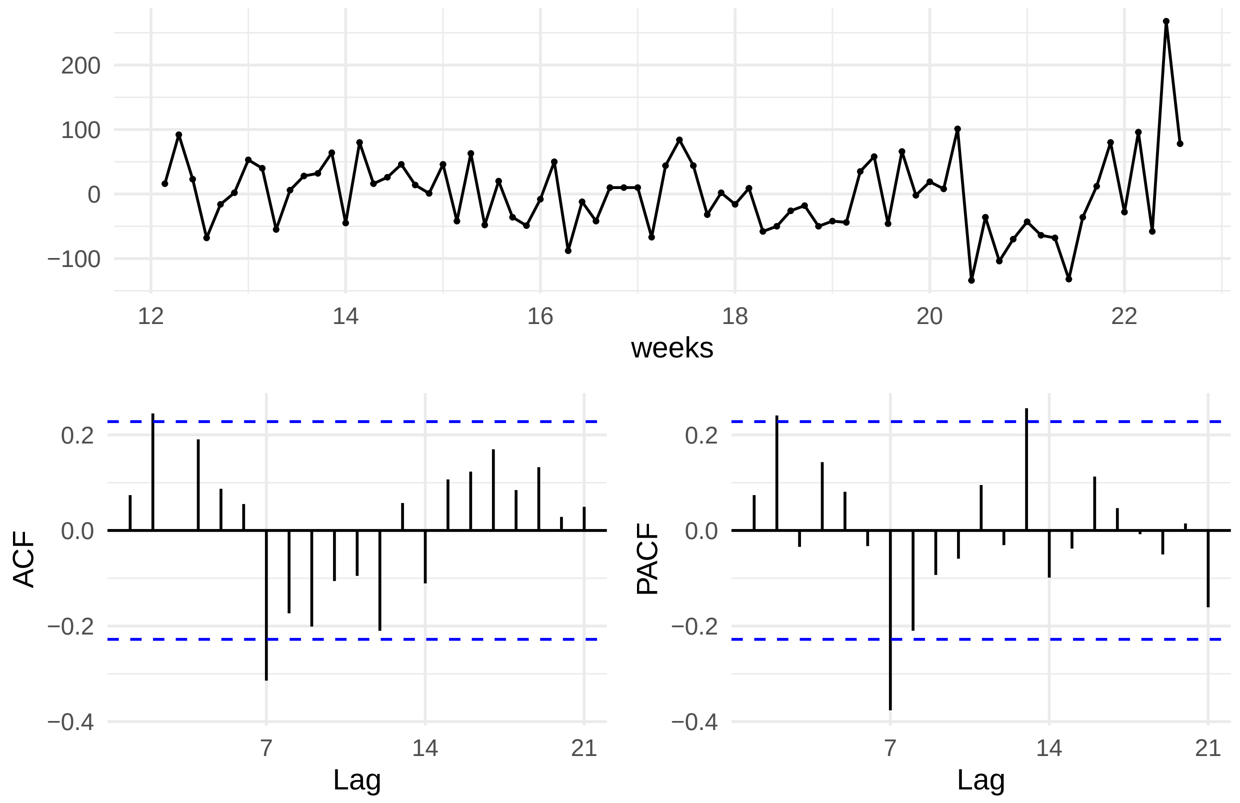
```
## [1] 0
```

We now visually inspect the differentiated time series.

```
autoplot(lettuce_train.diff1, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 2 Store (4904) - Time series plot (Differentiated)")
```



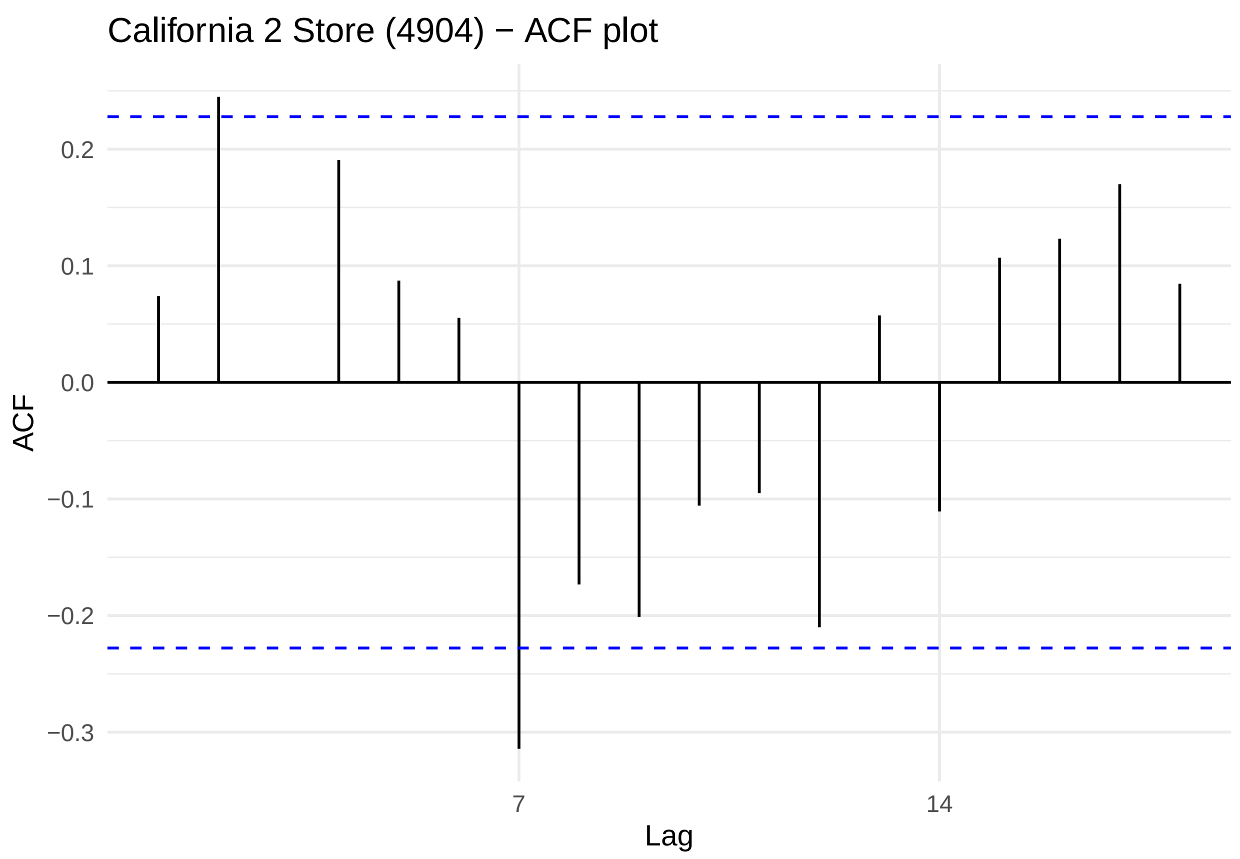
```r
ggtsdisplay(lettuce_train.diff1, xlab = "weeks", theme = theme_minimal())
```



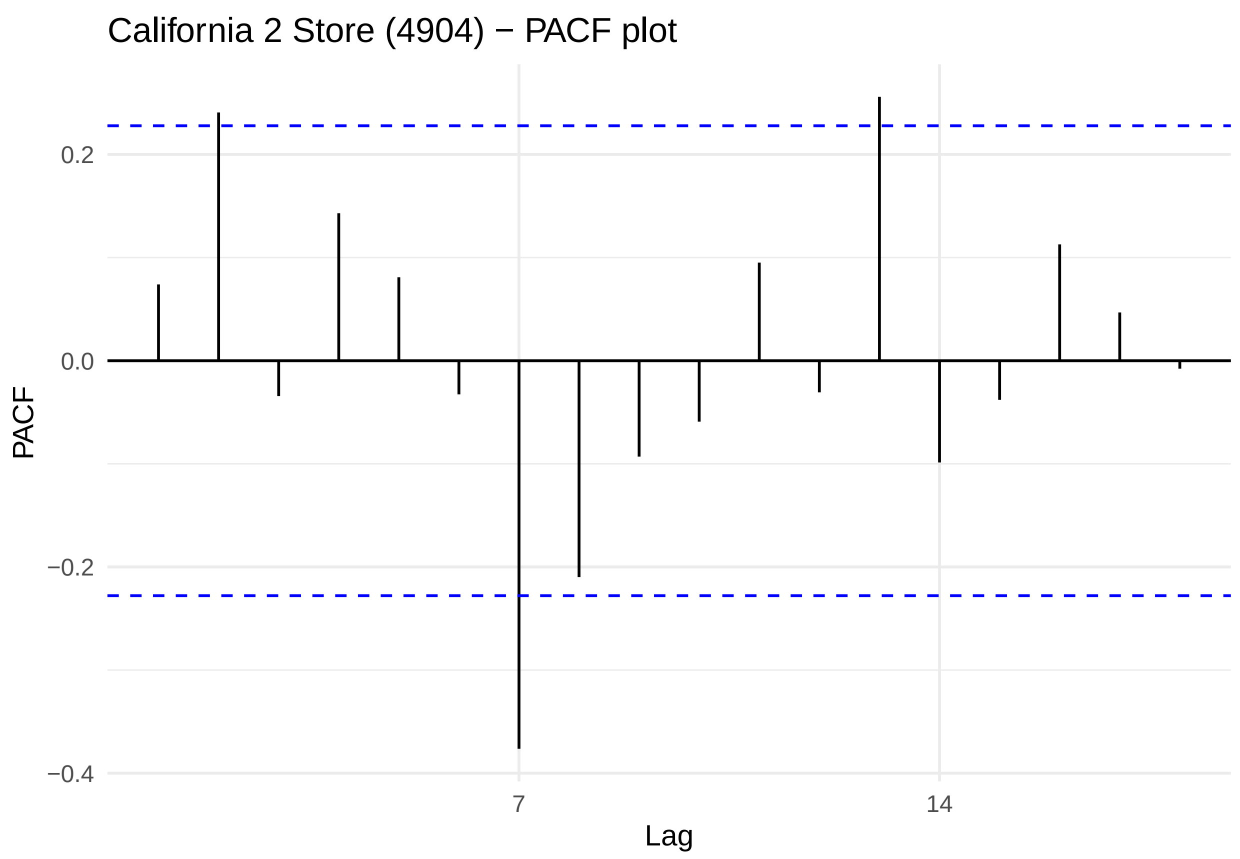
Looks stationary.

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("California 2 Store (4904) - ACF plot")
```



```
# pacf plot
ggPacf(lettuce_train.diff1) + theme_minimal() + ggtitle("California 2 Store (4904) - 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]                    : 821.5322
##  ARIMA(0,0,0)(0,1,0)[7] with drift         : 823.5309
##  ARIMA(0,0,0)(0,1,1)[7]                    : 804.2917
##  ARIMA(0,0,0)(0,1,1)[7] with drift         : 803.4415
##  ARIMA(0,0,0)(0,1,2)[7]                    : Inf
##  ARIMA(0,0,0)(0,1,2)[7] with drift         : Inf
##  ARIMA(0,0,0)(1,1,0)[7]                    : 811.866
##  ARIMA(0,0,0)(1,1,0)[7] with drift         : 813.6334
##  ARIMA(0,0,0)(1,1,1)[7]                    : Inf
##  ARIMA(0,0,0)(1,1,1)[7] with drift         : Inf
##  ARIMA(0,0,0)(1,1,2)[7]                    : 805.7436
##  ARIMA(0,0,0)(1,1,2)[7] with drift         : Inf
##  ARIMA(0,0,0)(2,1,0)[7]                    : 807.9309
##  ARIMA(0,0,0)(2,1,0)[7] with drift         : 809.4067
##  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]                    : Inf
##  ARIMA(0,0,0)(2,1,2)[7] with drift         : Inf
##  ARIMA(0,0,1)(0,1,0)[7]                    : 823.2593
##  ARIMA(0,0,1)(0,1,0)[7] with drift         : 825.2576
##  ARIMA(0,0,1)(0,1,1)[7]                    : 804.7957
##  ARIMA(0,0,1)(0,1,1)[7] with drift         : Inf
##  ARIMA(0,0,1)(0,1,2)[7]                    : Inf
##  ARIMA(0,0,1)(0,1,2)[7] with drift         : Inf
##  ARIMA(0,0,1)(1,1,0)[7]                    : 813.7948
##  ARIMA(0,0,1)(1,1,0)[7] with drift         : 815.5793
##  ARIMA(0,0,1)(1,1,1)[7]                    : Inf
##  ARIMA(0,0,1)(1,1,1)[7] with drift         : Inf
##  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]                    : 809.1625
##  ARIMA(0,0,1)(2,1,0)[7] with drift         : 810.7236
##  ARIMA(0,0,1)(2,1,1)[7]                    : Inf
##  ARIMA(0,0,1)(2,1,1)[7] with drift         : Inf
##  ARIMA(0,0,1)(2,1,2)[7]                    : Inf
##  ARIMA(0,0,1)(2,1,2)[7] with drift         : Inf
##  ARIMA(0,0,2)(0,1,0)[7]                    : 819.7539
##  ARIMA(0,0,2)(0,1,0)[7] with drift         : 821.7104
##  ARIMA(0,0,2)(0,1,1)[7]                    : 800.4323
##  ARIMA(0,0,2)(0,1,1)[7] with drift         : Inf
##  ARIMA(0,0,2)(0,1,2)[7]                    : Inf
##  ARIMA(0,0,2)(0,1,2)[7] with drift         : Inf
##  ARIMA(0,0,2)(1,1,0)[7]                    : 811.3836
##  ARIMA(0,0,2)(1,1,0)[7] with drift         : 813.3429
##  ARIMA(0,0,2)(1,1,1)[7]                    : Inf
##  ARIMA(0,0,2)(1,1,1)[7] with drift         : Inf
##  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]                    : 806.7005
##  ARIMA(0,0,2)(2,1,0)[7] with drift         : 808.5373
##  ARIMA(0,0,2)(2,1,1)[7]                    : Inf
##  ARIMA(0,0,2)(2,1,1)[7] with drift         : Inf
##  ARIMA(0,0,3)(0,1,0)[7]                    : 820.761
##  ARIMA(0,0,3)(0,1,0)[7] with drift         : 822.7283
##  ARIMA(0,0,3)(0,1,1)[7]                    : 801.9803
##  ARIMA(0,0,3)(0,1,1)[7] with drift         : Inf
##  ARIMA(0,0,3)(0,1,2)[7]                    : Inf
##  ARIMA(0,0,3)(0,1,2)[7] with drift         : Inf
##  ARIMA(0,0,3)(1,1,0)[7]                    : 813.2566
##  ARIMA(0,0,3)(1,1,0)[7] with drift         : 815.2093
##  ARIMA(0,0,3)(1,1,1)[7]                    : Inf
##  ARIMA(0,0,3)(1,1,1)[7] with drift         : Inf
##  ARIMA(0,0,3)(2,1,0)[7]                    : 808.6235
##  ARIMA(0,0,3)(2,1,0)[7] with drift         : 810.4687
##  ARIMA(0,0,4)(0,1,0)[7]                    : 815.1303
##  ARIMA(0,0,4)(0,1,0)[7] with drift         : 817.0265
##  ARIMA(0,0,4)(0,1,1)[7]                    : 800.1768
##  ARIMA(0,0,4)(0,1,1)[7] with drift         : Inf
##  ARIMA(0,0,4)(1,1,0)[7]                    : 808.7009
##  ARIMA(0,0,4)(1,1,0)[7] with drift         : 810.6985
##  ARIMA(0,0,5)(0,1,0)[7]                    : 810.6515
##  ARIMA(0,0,5)(0,1,0)[7] with drift         : 812.4925
##  ARIMA(1,0,0)(0,1,0)[7]                    : 823.123
##  ARIMA(1,0,0)(0,1,0)[7] with drift         : 825.1209
##  ARIMA(1,0,0)(0,1,1)[7]                    : 803.4738
##  ARIMA(1,0,0)(0,1,1)[7] with drift         : Inf
##  ARIMA(1,0,0)(0,1,2)[7]                    : Inf
##  ARIMA(1,0,0)(0,1,2)[7] with drift         : Inf
##  ARIMA(1,0,0)(1,1,0)[7]                    : 813.7587
##  ARIMA(1,0,0)(1,1,0)[7] with drift         : 815.5525
##  ARIMA(1,0,0)(1,1,1)[7]                    : Inf
##  ARIMA(1,0,0)(1,1,1)[7] with drift         : Inf
##  ARIMA(1,0,0)(1,1,2)[7]                    : Inf
##  ARIMA(1,0,0)(1,1,2)[7] with drift         : Inf
##  ARIMA(1,0,0)(2,1,0)[7]                    : 808.7056
##  ARIMA(1,0,0)(2,1,0)[7] with drift         : 810.3328
##  ARIMA(1,0,0)(2,1,1)[7]                    : Inf
##  ARIMA(1,0,0)(2,1,1)[7] with drift         : Inf
##  ARIMA(1,0,0)(2,1,2)[7]                    : Inf
##  ARIMA(1,0,0)(2,1,2)[7] with drift         : Inf
##  ARIMA(1,0,1)(0,1,0)[7]                    : 822.7868
##  ARIMA(1,0,1)(0,1,0)[7] with drift         : 824.7338
##  ARIMA(1,0,1)(0,1,1)[7]                    : Inf
##  ARIMA(1,0,1)(0,1,1)[7] with drift         : Inf
##  ARIMA(1,0,1)(0,1,2)[7]                    : Inf
##  ARIMA(1,0,1)(0,1,2)[7] with drift         : Inf
##  ARIMA(1,0,1)(1,1,0)[7]                    : 812.7861
##  ARIMA(1,0,1)(1,1,0)[7] with drift         : 814.7813
##  ARIMA(1,0,1)(1,1,1)[7]                    : Inf
##  ARIMA(1,0,1)(1,1,1)[7] with drift         : Inf
##  ARIMA(1,0,1)(1,1,2)[7]                    : Inf
##  ARIMA(1,0,1)(1,1,2)[7] with drift         : Inf
##  ARIMA(1,0,1)(2,1,0)[7]                    : 804.4291
##  ARIMA(1,0,1)(2,1,0)[7] with drift         : 806.4279
##  ARIMA(1,0,1)(2,1,1)[7]                    : Inf
##  ARIMA(1,0,1)(2,1,1)[7] with drift         : Inf
##  ARIMA(1,0,2)(0,1,0)[7]                    : 820.7395
##  ARIMA(1,0,2)(0,1,0)[7] with drift         : 822.7017
##  ARIMA(1,0,2)(0,1,1)[7]                    : Inf
##  ARIMA(1,0,2)(0,1,1)[7] with drift         : Inf
##  ARIMA(1,0,2)(0,1,2)[7]                    : Inf
##  ARIMA(1,0,2)(0,1,2)[7] with drift         : Inf
##  ARIMA(1,0,2)(1,1,0)[7]                    : 812.6489
##  ARIMA(1,0,2)(1,1,0)[7] with drift         : 814.5506
##  ARIMA(1,0,2)(1,1,1)[7]                    : Inf
##  ARIMA(1,0,2)(1,1,1)[7] with drift         : Inf
##  ARIMA(1,0,2)(2,1,0)[7]                    : 803.7958
##  ARIMA(1,0,2)(2,1,0)[7] with drift         : 805.7785
##  ARIMA(1,0,3)(0,1,0)[7]                    : 822.7249
##  ARIMA(1,0,3)(0,1,0)[7] with drift         : 824.6894
##  ARIMA(1,0,3)(0,1,1)[7]                    : Inf
##  ARIMA(1,0,3)(0,1,1)[7] with drift         : Inf
##  ARIMA(1,0,3)(1,1,0)[7]                    : 814.0892
##  ARIMA(1,0,3)(1,1,0)[7] with drift         : 816.0381
##  ARIMA(1,0,4)(0,1,0)[7]                    : 809.7097
##  ARIMA(1,0,4)(0,1,0)[7] with drift         : 811.4297
##  ARIMA(2,0,0)(0,1,0)[7]                    : 818.6198
##  ARIMA(2,0,0)(0,1,0)[7] with drift         : 820.5045
##  ARIMA(2,0,0)(0,1,1)[7]                    : Inf
##  ARIMA(2,0,0)(0,1,1)[7] with drift         : Inf
##  ARIMA(2,0,0)(0,1,2)[7]                    : Inf
##  ARIMA(2,0,0)(0,1,2)[7] with drift         : Inf
##  ARIMA(2,0,0)(1,1,0)[7]                    : 809.7259
##  ARIMA(2,0,0)(1,1,0)[7] with drift         : 811.7238
##  ARIMA(2,0,0)(1,1,1)[7]                    : Inf
##  ARIMA(2,0,0)(1,1,1)[7] with drift         : Inf
##  ARIMA(2,0,0)(1,1,2)[7]                    : Inf
##  ARIMA(2,0,0)(1,1,2)[7] with drift         : Inf
##  ARIMA(2,0,0)(2,1,0)[7]                    : 804.327
##  ARIMA(2,0,0)(2,1,0)[7] with drift         : 806.2977
##  ARIMA(2,0,0)(2,1,1)[7]                    : Inf
##  ARIMA(2,0,0)(2,1,1)[7] with drift         : Inf
##  ARIMA(2,0,1)(0,1,0)[7]                    : 819.8515
##  ARIMA(2,0,1)(0,1,0)[7] with drift         : 821.7936
##  ARIMA(2,0,1)(0,1,1)[7]                    : Inf
##  ARIMA(2,0,1)(0,1,1)[7] with drift         : Inf
##  ARIMA(2,0,1)(0,1,2)[7]                    : Inf
##  ARIMA(2,0,1)(0,1,2)[7] with drift         : Inf
##  ARIMA(2,0,1)(1,1,0)[7]                    : 811.276
##  ARIMA(2,0,1)(1,1,0)[7] with drift         : 813.2674
##  ARIMA(2,0,1)(1,1,1)[7]                    : Inf
##  ARIMA(2,0,1)(1,1,1)[7] with drift         : Inf
##  ARIMA(2,0,1)(2,1,0)[7]                    : 804.0774
##  ARIMA(2,0,1)(2,1,0)[7] with drift         : 806.0656
##  ARIMA(2,0,2)(0,1,0)[7]                    : 820.9584
##  ARIMA(2,0,2)(0,1,0)[7] with drift         : 822.8177
##  ARIMA(2,0,2)(0,1,1)[7]                    : Inf
##  ARIMA(2,0,2)(0,1,1)[7] with drift         : Inf
##  ARIMA(2,0,2)(1,1,0)[7]                    : 811.7578
##  ARIMA(2,0,2)(1,1,0)[7] with drift         : 813.7493
##  ARIMA(2,0,3)(0,1,0)[7]                    : Inf
##  ARIMA(2,0,3)(0,1,0)[7] with drift         : Inf
##  ARIMA(3,0,0)(0,1,0)[7]                    : 820.4527
##  ARIMA(3,0,0)(0,1,0)[7] with drift         : 822.3605
##  ARIMA(3,0,0)(0,1,1)[7]                    : Inf
##  ARIMA(3,0,0)(0,1,1)[7] with drift         : Inf
##  ARIMA(3,0,0)(0,1,2)[7]                    : Inf
##  ARIMA(3,0,0)(0,1,2)[7] with drift         : Inf
##  ARIMA(3,0,0)(1,1,0)[7]                    : 811.7259
##  ARIMA(3,0,0)(1,1,0)[7] with drift         : 813.7238
##  ARIMA(3,0,0)(1,1,1)[7]                    : Inf
##  ARIMA(3,0,0)(1,1,1)[7] with drift         : Inf
##  ARIMA(3,0,0)(2,1,0)[7]                    : 805.9991
##  ARIMA(3,0,0)(2,1,0)[7] with drift         : 807.9869
##  ARIMA(3,0,1)(0,1,0)[7]                    : 821.4908
##  ARIMA(3,0,1)(0,1,0)[7] with drift         : 823.4058
##  ARIMA(3,0,1)(0,1,1)[7]                    : Inf
##  ARIMA(3,0,1)(0,1,1)[7] with drift         : Inf
##  ARIMA(3,0,1)(1,1,0)[7]                    : 813.0845
##  ARIMA(3,0,1)(1,1,0)[7] with drift         : 815.0774
##  ARIMA(3,0,2)(0,1,0)[7]                    : Inf
##  ARIMA(3,0,2)(0,1,0)[7] with drift         : Inf
##  ARIMA(4,0,0)(0,1,0)[7]                    : 820.2613
##  ARIMA(4,0,0)(0,1,0)[7] with drift         : 822.0925
##  ARIMA(4,0,0)(0,1,1)[7]                    : Inf
##  ARIMA(4,0,0)(0,1,1)[7] with drift         : Inf
##  ARIMA(4,0,0)(1,1,0)[7]                    : 810.8025
##  ARIMA(4,0,0)(1,1,0)[7] with drift         : 812.7895
##  ARIMA(4,0,1)(0,1,0)[7]                    : 822.0442
##  ARIMA(4,0,1)(0,1,0)[7] with drift         : 823.8563
##  ARIMA(5,0,0)(0,1,0)[7]                    : 821.7107
##  ARIMA(5,0,0)(0,1,0)[7] with drift         : 823.5025
##
##
##
##  Best model: ARIMA(0,0,4)(0,1,1)[7]
```

```
## Series: lettuce_train
## ARIMA(0,0,4)(0,1,1)[7]
##
## Coefficients:
##          ma1     ma2     ma3     ma4     sma1
##       0.0689  0.2087  0.0177  0.2860  -0.8260
## s.e.  0.1842  0.1461  0.1137  0.1694   0.2124
##
## sigma^2 estimated as 2370:  log likelihood=-394.09
## AIC=800.18   AICc=801.43   BIC=814
```

```
# Best model: ARIMA(0,0,4)(0,1,1)[7]  (AIC=800.18)
# Second best:  ARIMA(0,0,2)(0,1,1)[7]  (AIC=800.43)
# Third best: ARIMA(0,0,3)(0,1,1)[7] (AIC=801.98)
```

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(0,0,4)(0,1,1)[7] with AIC=800.18, ARIMA(0,0,2)(0,1,1)[7] with AIC=800.43 AND ARIMA(0,0,3)(0,1,1)[7] with AIC=801.98, as the candidate models that we would like to evaluate further.

```
# three candidate models
lettuce.m1 <- Arima(lettuce_train, order = c(0, 0, 4),
                       seasonal = list(order = c(0, 1, 1), period = 7))
lettuce.m2 <- Arima(lettuce_train, order = c(0, 0, 2),
                       seasonal = list(order = c(0, 1, 1), period = 7))
lettuce.m3 <- Arima(lettuce_train, order = c(0, 0, 3),
                       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       ACF1
## Training set -0.8700088 44.92879 32.33449 -1.577557 10.8242 0.686586 0.04671246
```

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

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -1.495663 46.23796 33.24954 -2.132859 11.26877 0.7060161
##                    ACF1
## Training set 0.02332537
```

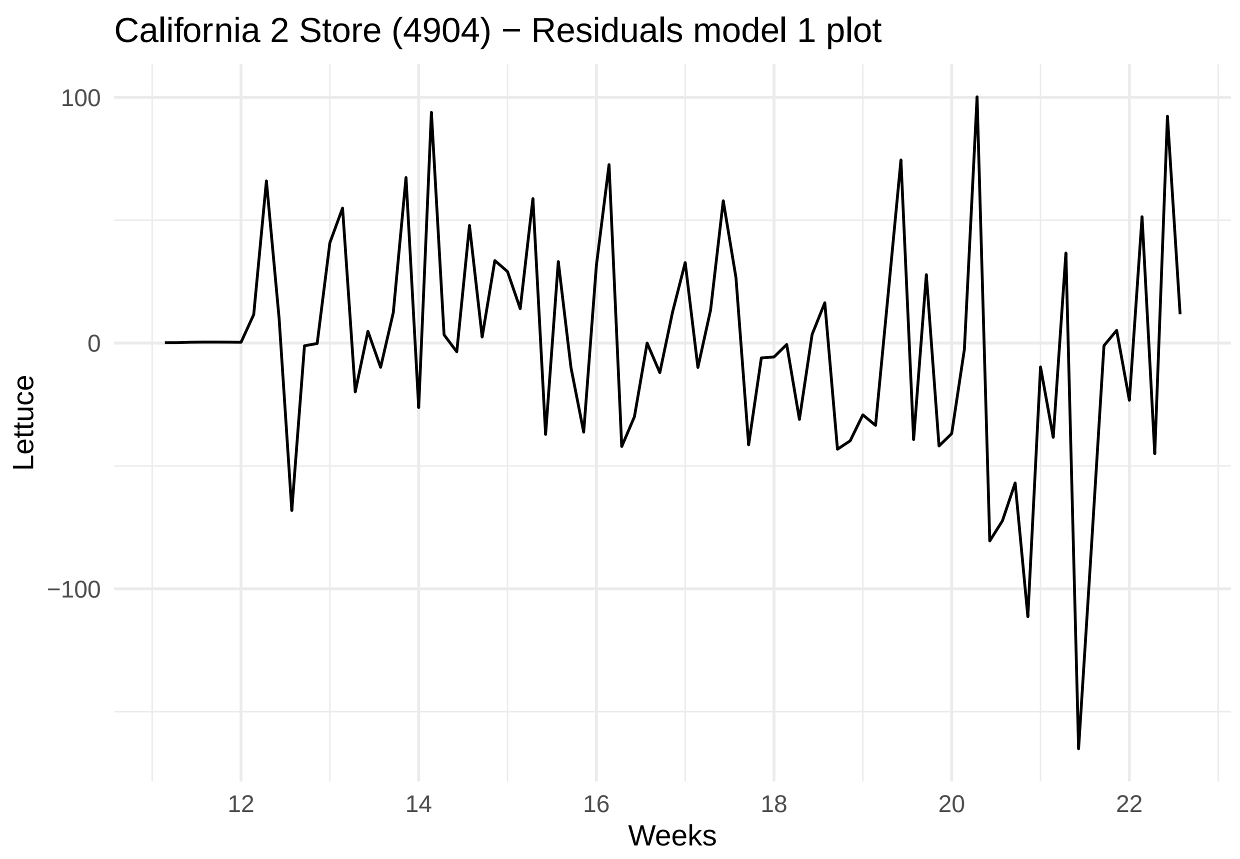
```
# in-sample one-step forecasts model 3
accuracy(lettuce.m3)
```

```
##                     ME     RMSE      MAE       MPE    MAPE      MASE       ACF1
## Training set -1.223225 45.53461 32.85346 -2.012476 11.1098 0.6976058 0.01349593
```

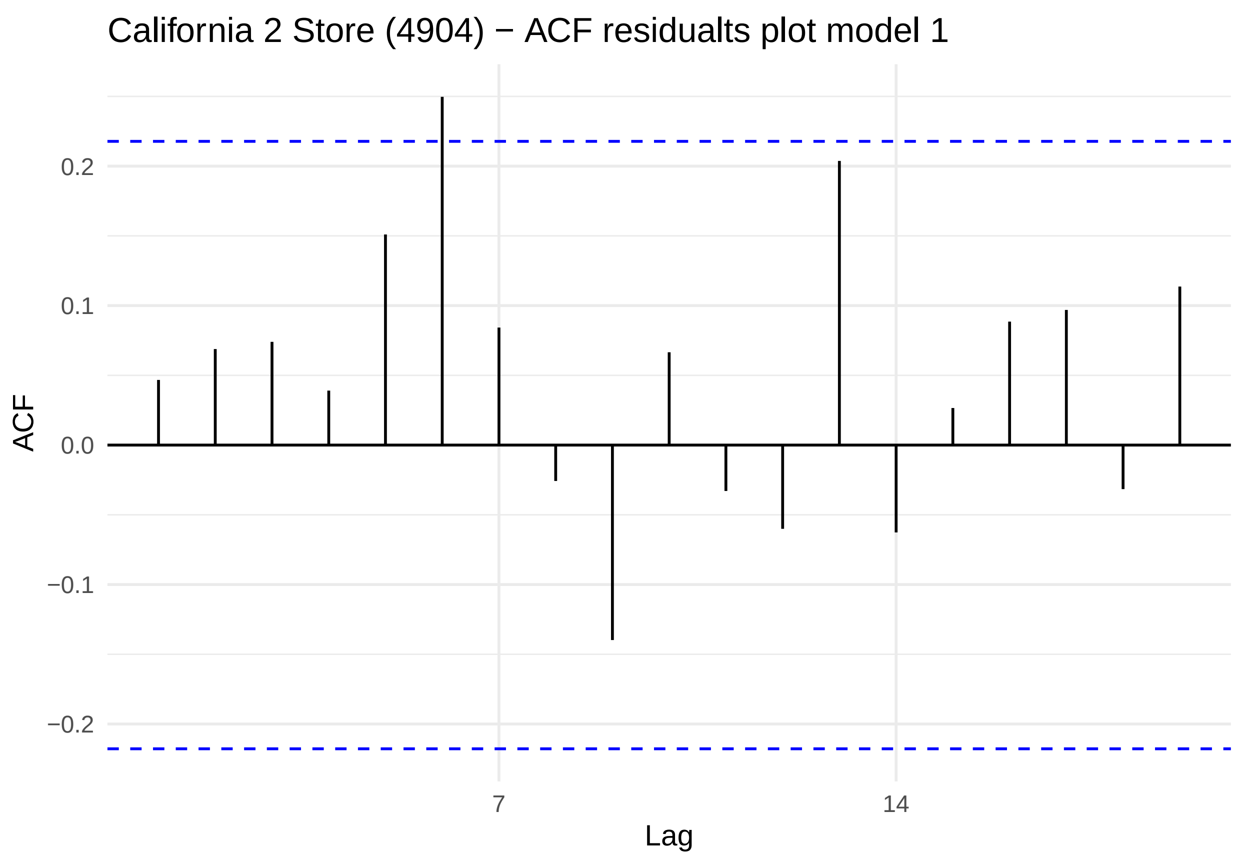
The first model even though it has both the lowest AIC score as well as the lowest RMSE.

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

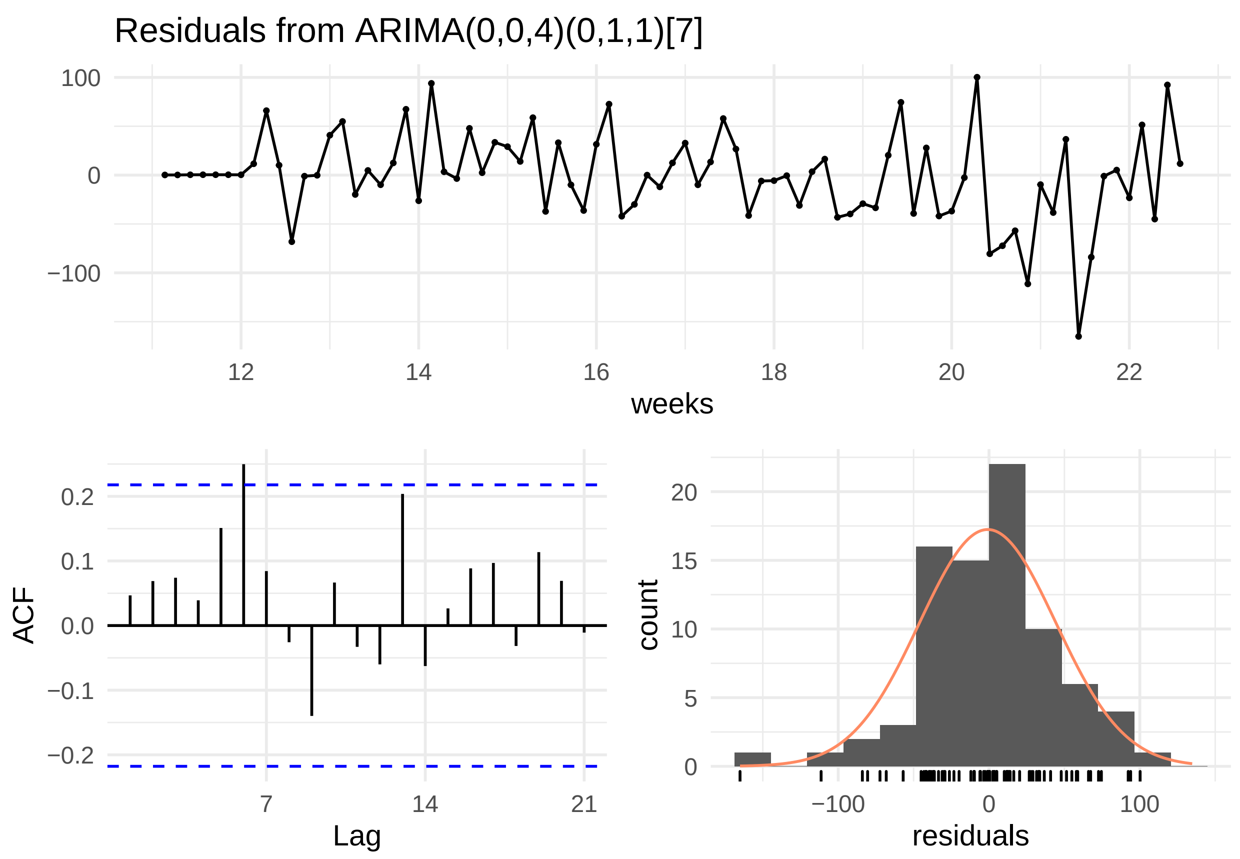
```
# residual analysis model 1
autoplot(lettuce.m1$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 2 Store (4904) - Residuals model 1 plot")
```



```
ggAcf(lettuce.m1$residuals) + theme_minimal() +
ggtitle("California 2 Store (4904) - ACF residualts plot model 1")
```

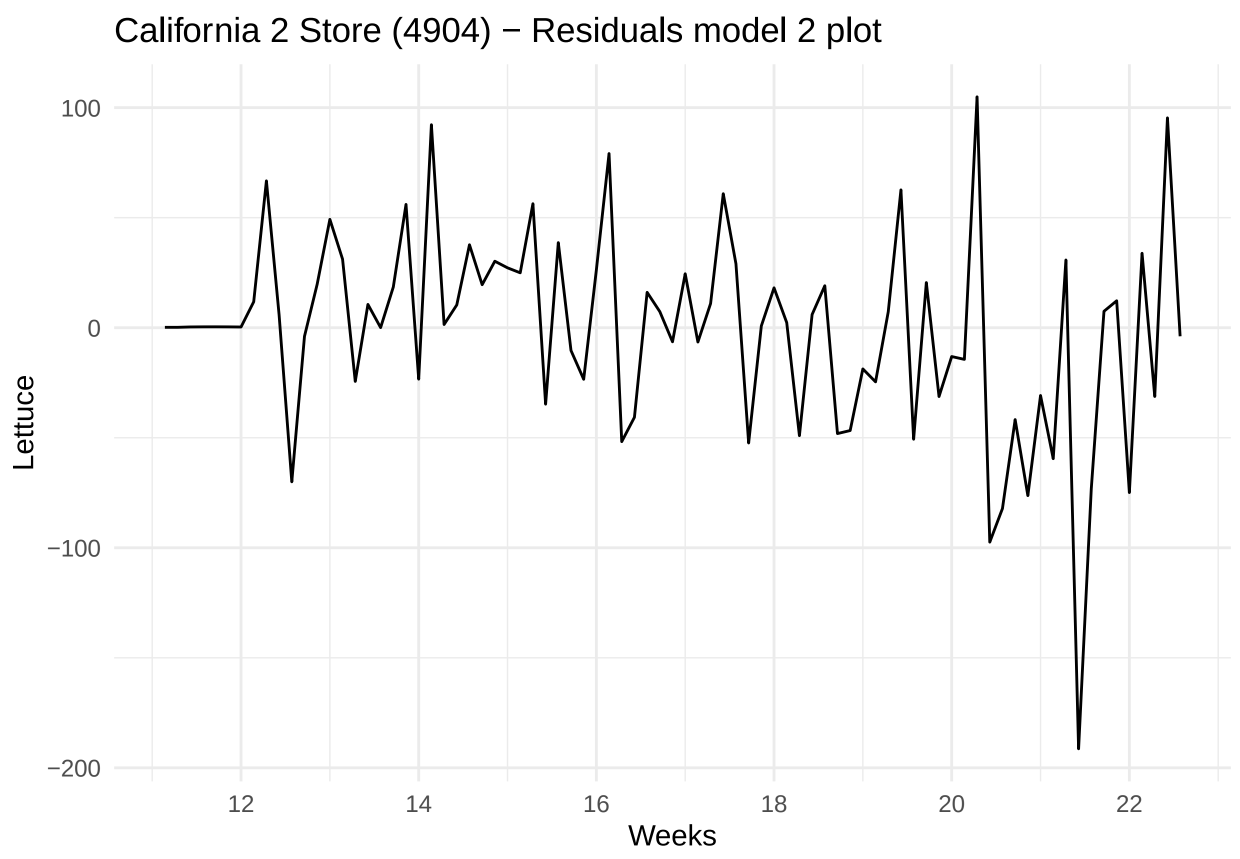


```r
checkresiduals(lettuce.m1, xlab = "weeks", theme = theme_minimal())
```

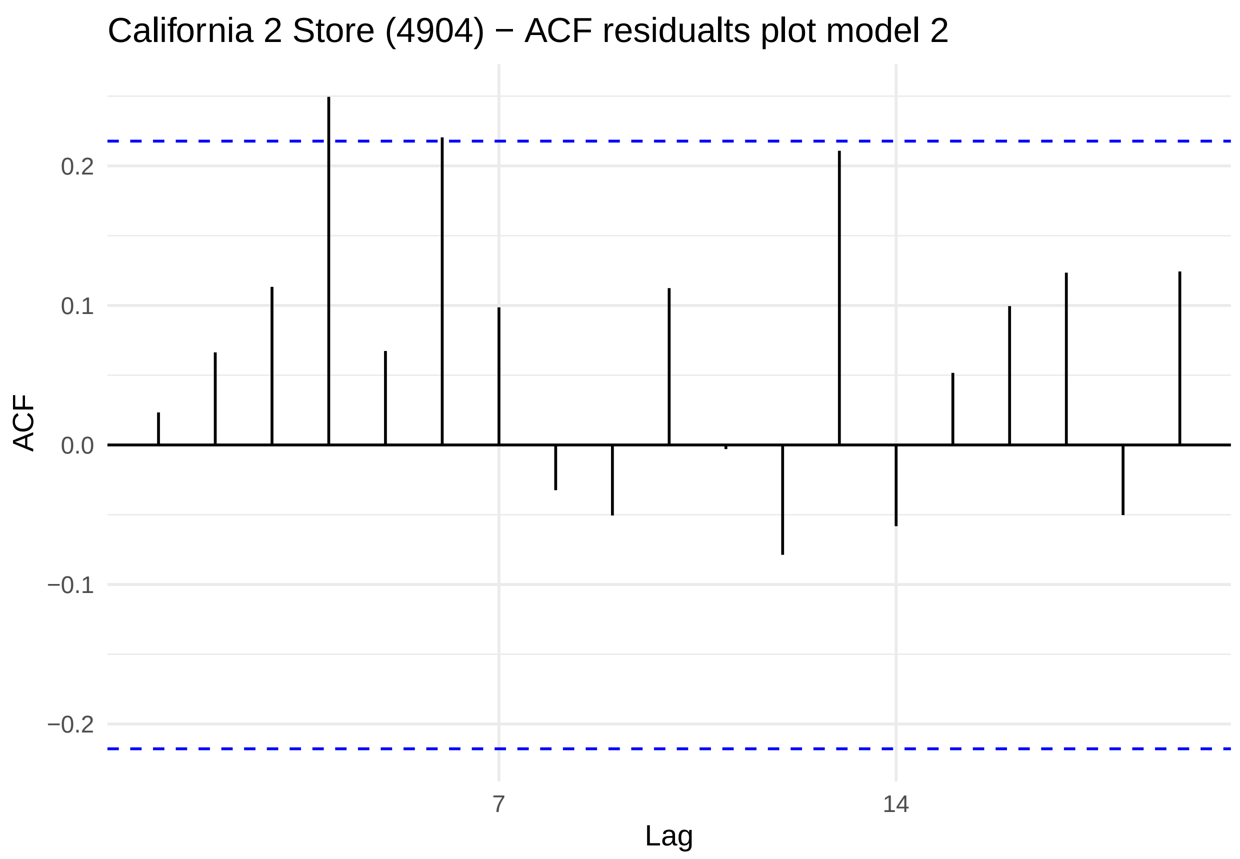


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,4)(0,1,1)[7]
## Q* = 16.702, df = 9, p-value = 0.0536
##
## Model df: 5.   Total lags used: 14
```

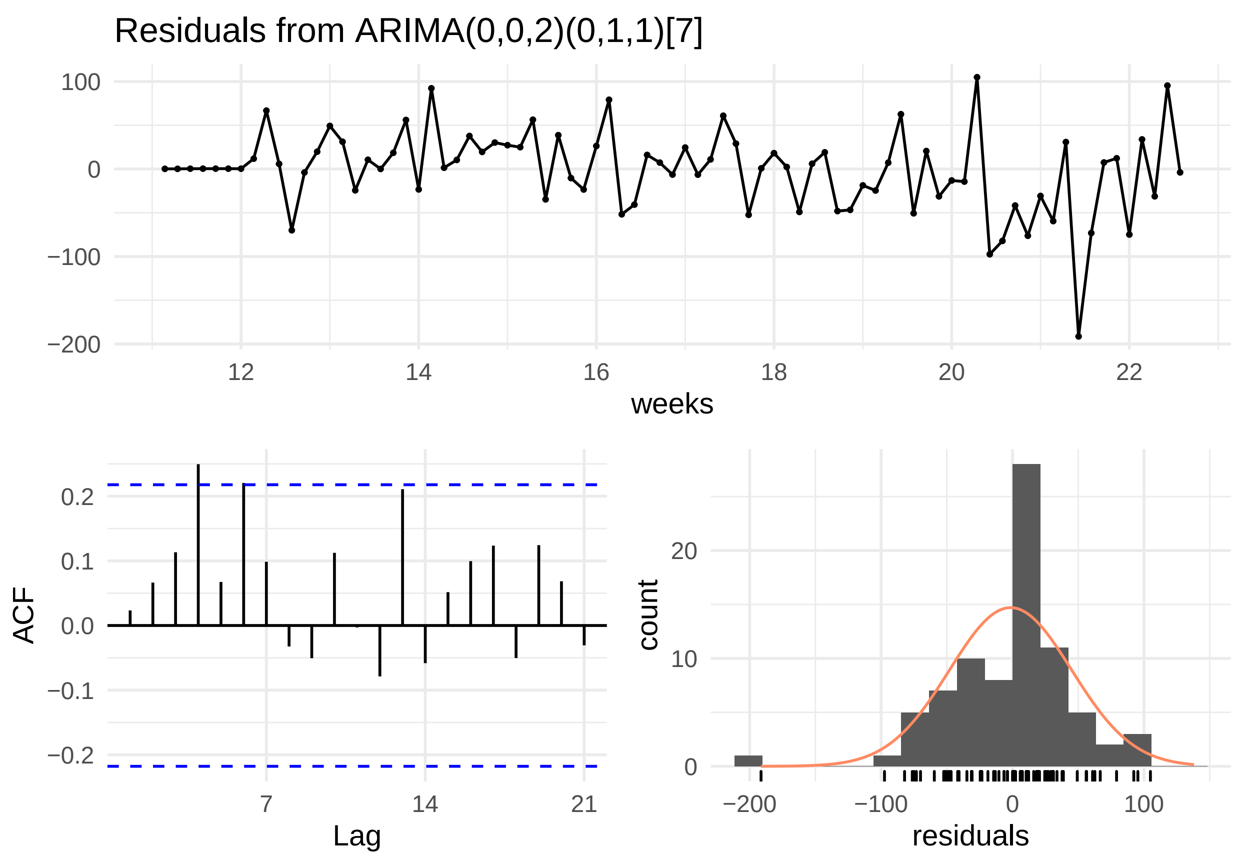
```
# residual analysis model 2
autoplot(lettuce.m2$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 2 Store (4904) - Residuals model 2 plot")
```



```
ggAcf(lettuce.m2$residuals) + theme_minimal() +
ggtitle("California 2 Store (4904) - ACF residualts plot model 2")
```

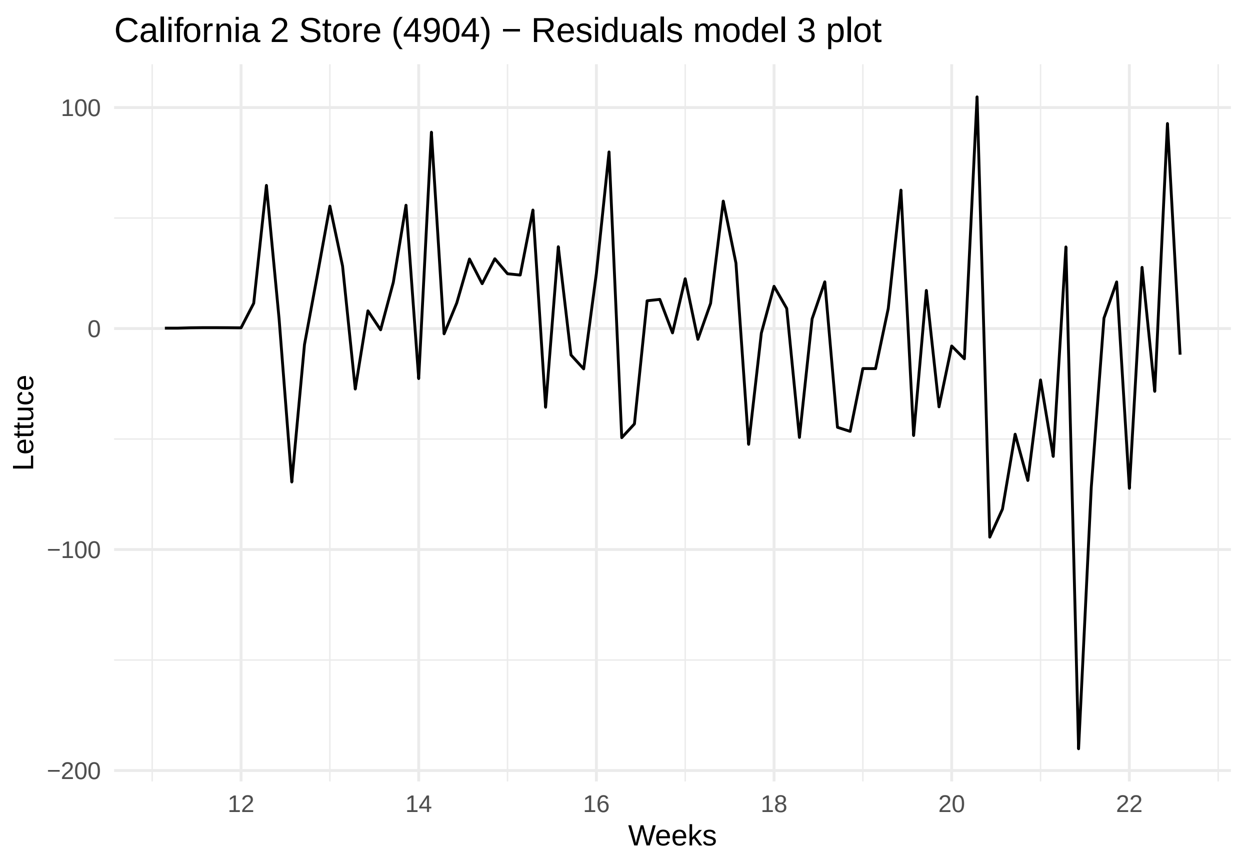


```r
checkresiduals(lettuce.m2, xlab = "weeks", theme = theme_minimal())
```

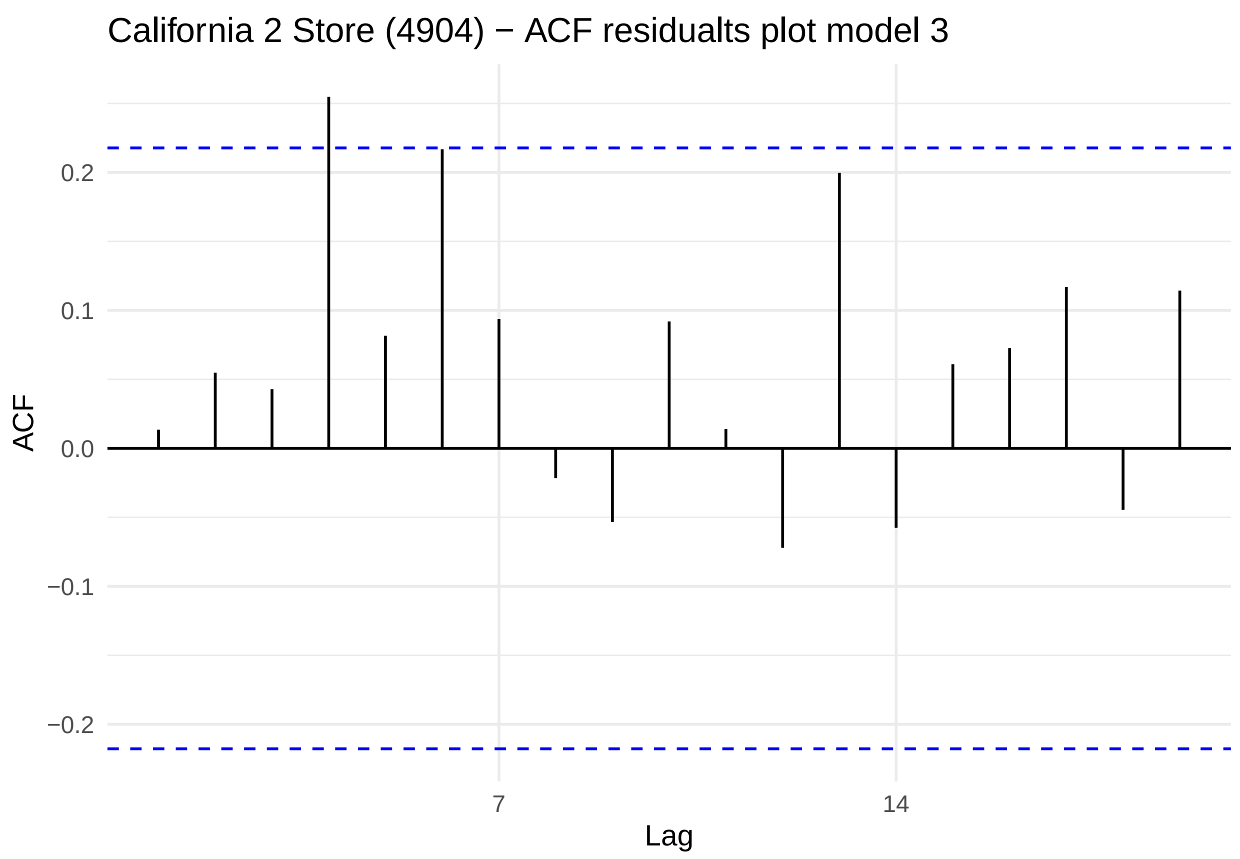


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,2)(0,1,1)[7]
## Q* = 19.478, df = 11, p-value = 0.05303
##
## Model df: 3.   Total lags used: 14
```

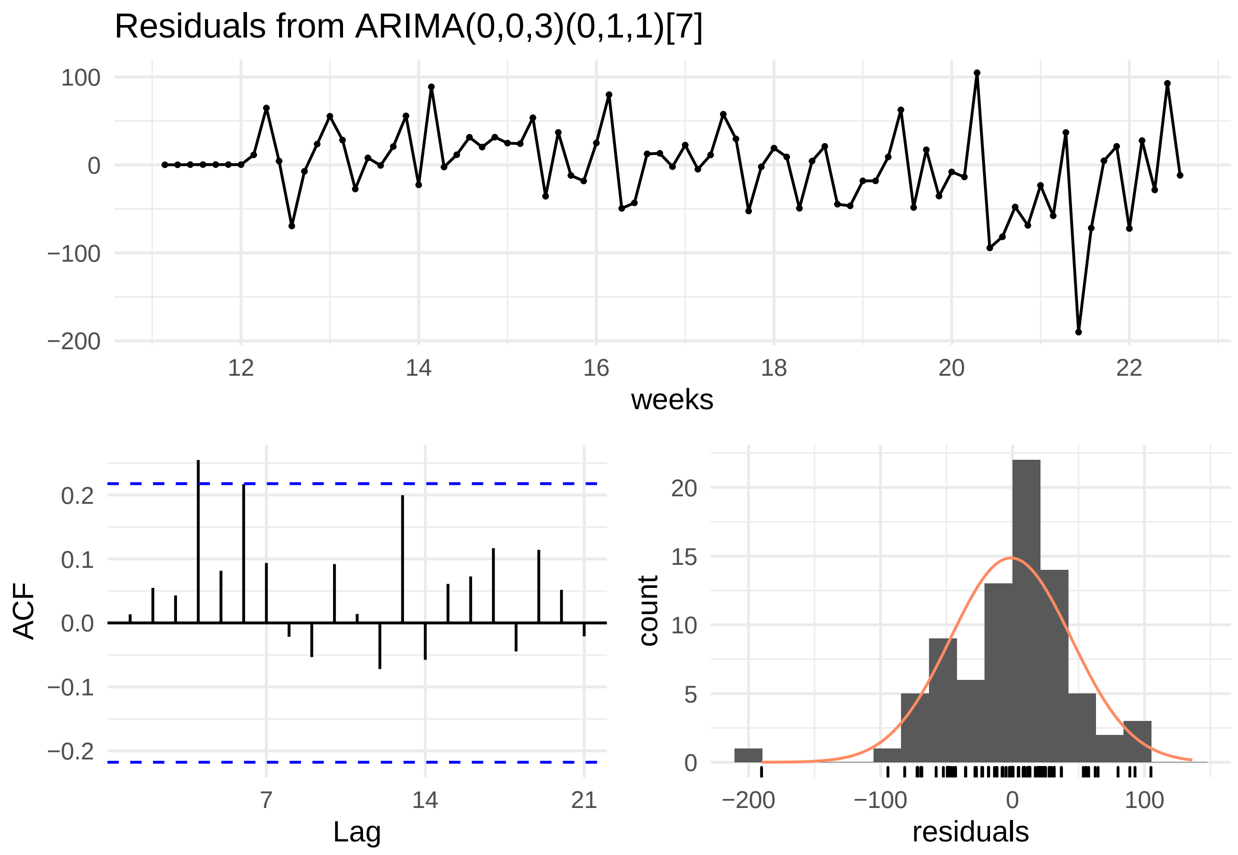
```
# residual analysis model 3
autoplot(lettuce.m3$residuals, xlab = "Weeks", ylab = "Lettuce") + theme_minimal() +
ggtitle("California 2 Store (4904) - Residuals model 3 plot")
```



```r
ggAcf(lettuce.m3$residuals) + theme_minimal() +
ggtitle("California 2 Store (4904) - ACF residualts plot model 3")
```



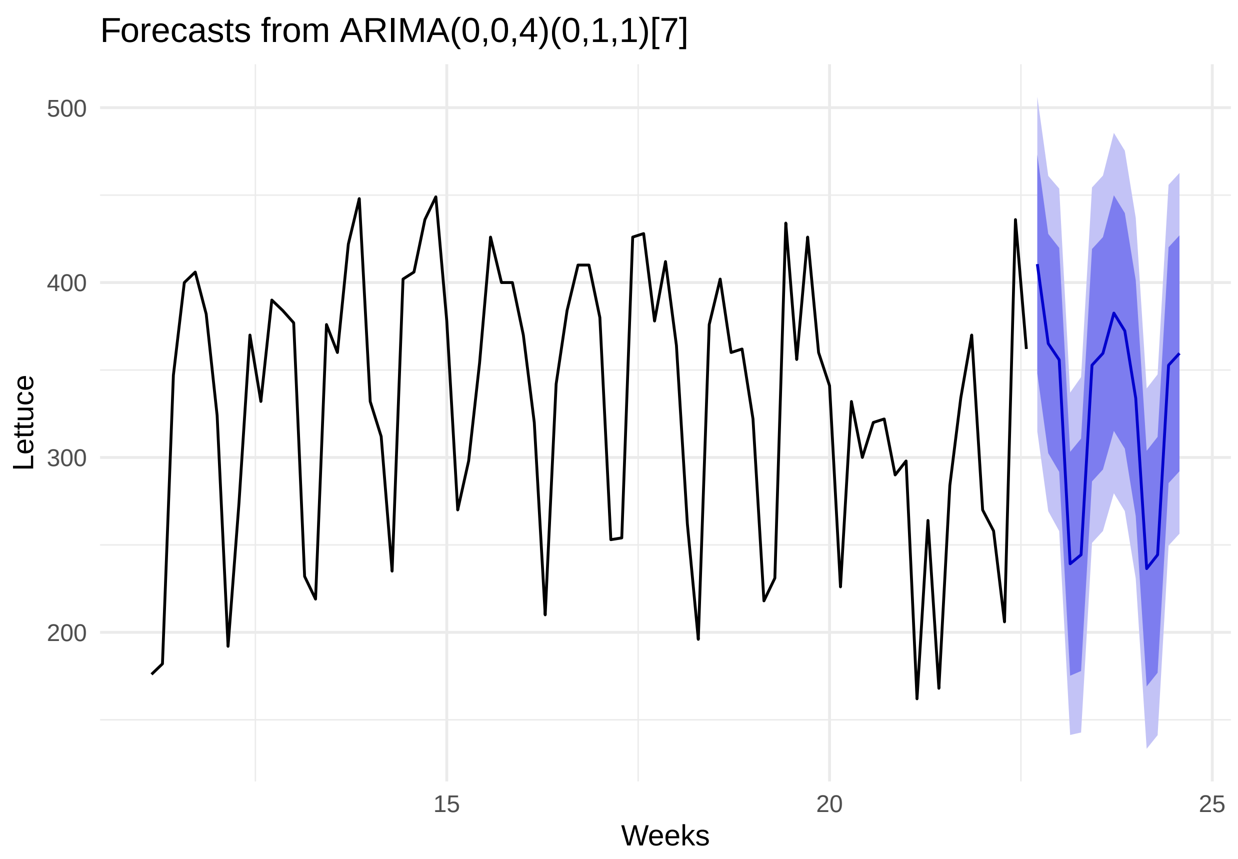
```r
checkresiduals(lettuce.m3, xlab = "weeks", theme = theme_minimal())
```



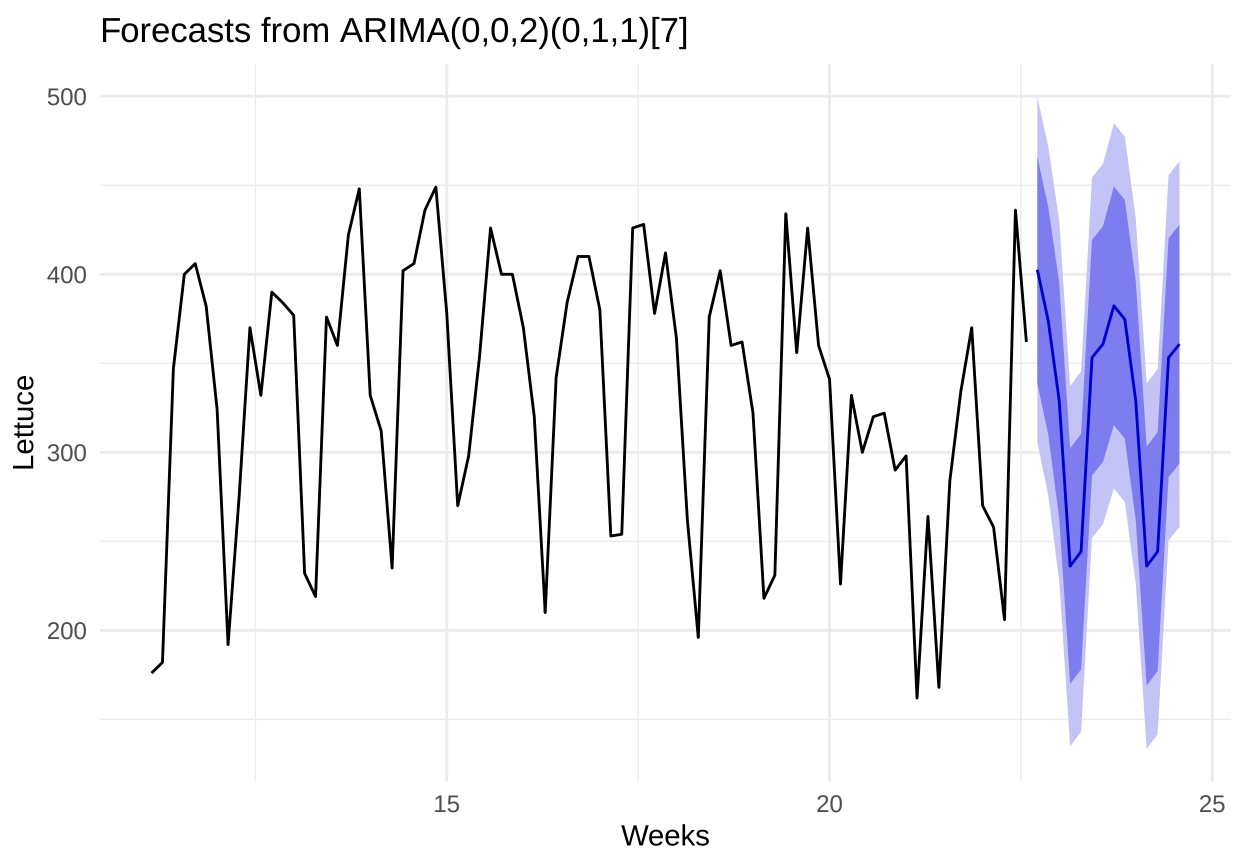
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,3)(0,1,1)[7]
## Q* = 17.609, df = 10, p-value = 0.06192
##
## Model df: 4.   Total lags used: 14
```

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

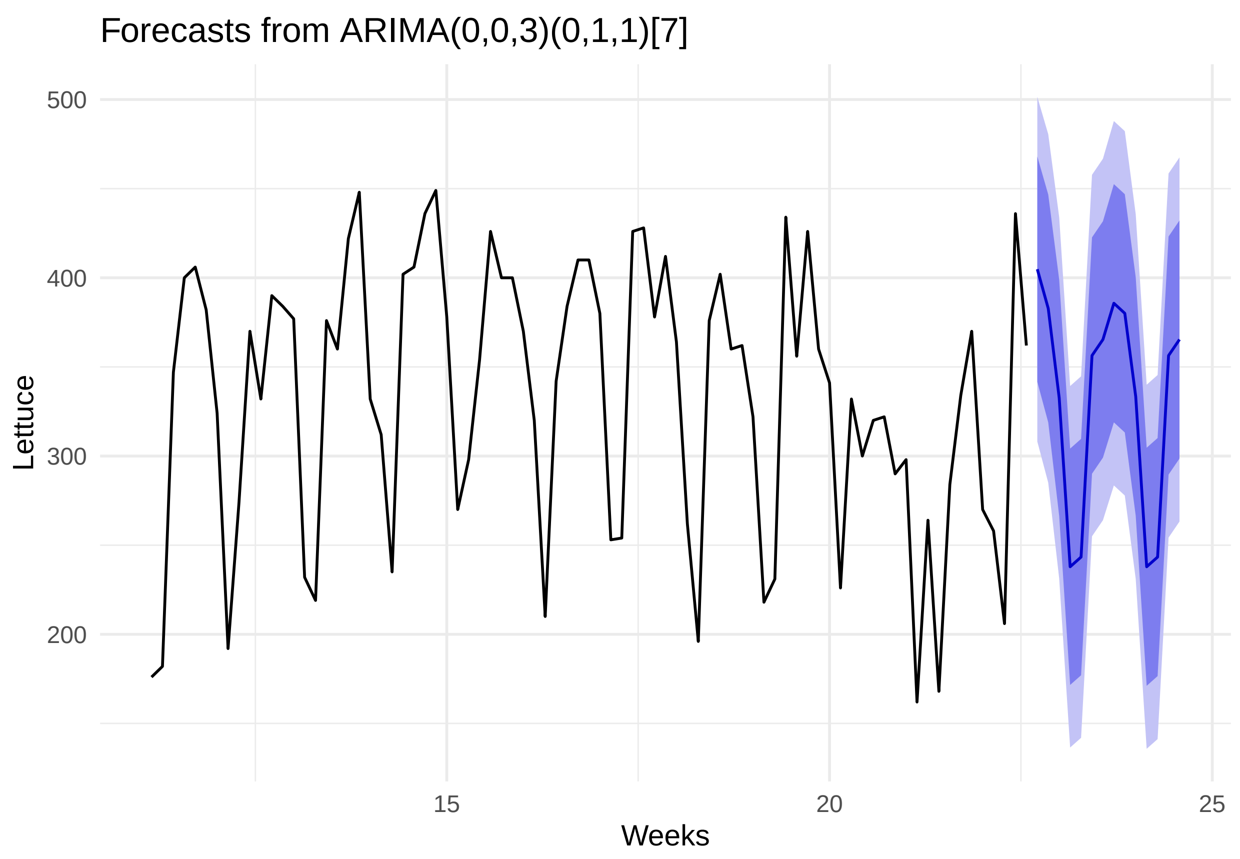
```
#Forecasting part model 1
lettuce.f1 <- forecast(lettuce.m1, h = 14)
autoplot(lettuce.f1, xlab = "Weeks", ylab = "Lettuce") + theme_minimal()
```



```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 81 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  -0.8700088 44.92879 32.33449 -1.577557 10.82420 0.6865860
## Test set     -24.2285979 41.64471 34.94981 -8.726629 11.57019 0.7421192
##                    ACF1 Theil's U
## Training set 0.04671246        NA
## Test set     0.15125519 0.4528834
```

```
# 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  -1.495663 46.23796 33.24954 -2.132859 11.26877 0.7060161
## Test set     -22.155255 41.21143 33.39216 -8.072531 11.11847 0.7090444
##                    ACF1 Theil's U
## Training set 0.02332537        NA
## Test set     0.13330175 0.4483704
```

```
# 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  -1.223225 45.53461 32.85346 -2.012476 11.1098 0.6976058
## Test set     -25.376320 43.84325 35.76735 -9.079674 11.8735 0.7594789
##                    ACF1 Theil's U
## Training set 0.01349593        NA
## Test set     0.15472062 0.4661476
```

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:

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

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

```
# 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       362.1033 300.6988 423.5077 268.1932 456.0133
## 24.85714       340.2753 278.3077 402.2428 245.5041 435.0465
## 25.00000       306.4619 242.4717 370.4522 208.5973 404.3266
## 25.14286       233.7681 169.7816 297.7547 135.9092 331.6271
## 25.28571       235.1361 171.1495 299.1227 137.2771 332.9951
## 25.42857       361.2778 297.2913 425.2644 263.4189 459.1368
## 25.57143       342.9025 278.9159 406.8890 245.0435 440.7615
## 25.71429       365.0649 299.3351 430.7946 264.5399 465.5898
## 25.85714       340.5518 274.7906 406.3130 239.9787 441.1248
## 26.00000       306.4619 240.5847 372.3391 205.7115 407.2124
## 26.14286       233.7681 167.8945 299.6417 133.0232 334.5131
## 26.28571       235.1361 169.2625 301.0097 134.3911 335.8811
## 26.42857       361.2778 295.4042 427.1514 260.5329 462.0228
## 26.57143       342.9025 277.0289 408.7761 242.1575 443.6475
```

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

```
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                       362.1033
## 2    2                       340.2753
## 3    3                       306.4619
## 4    4                       233.7681
## 5    5                       235.1361
## 6    6                       361.2778
## 7    7                       342.9025
## 8    8                       365.0649
## 9    9                       340.5518
## 10  10                       306.4619
## 11  11                       233.7681
## 12  12                       235.1361
## 13  13                       361.2778
## 14  14                       342.9025
```

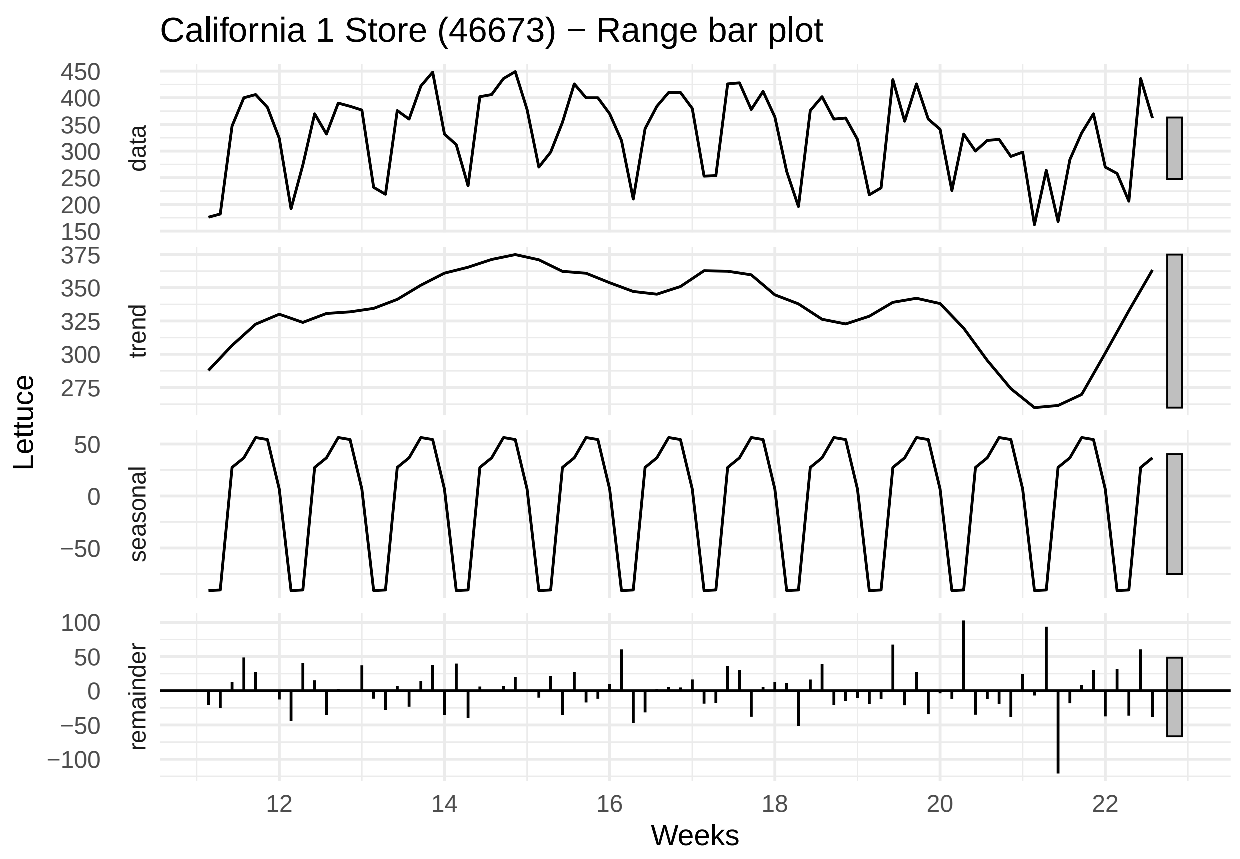
Holt-Winters

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.

```r
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.

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. In other words, it indicates that there is small 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 = 0.194
##     gamma = 1e-04
```

```
##
##   Initial states:
##     l = 329.2286
##     s = 6.3379 52.1586 56.1676 42.0506 29.5549 -91.1832
##            -95.0864
##
##   sigma:  45.6581
##
##       AIC      AICc       BIC
##  985.4413  988.5842 1009.3858
```

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 = 0.194
##     gamma = 1e-04
##
##   Initial states:
##     l = 329.2286
##     s = 6.3379 52.1586 56.1676 42.0506 29.5549 -91.1832
##            -95.0864
##
##   sigma:  45.6581
##
##       AIC      AICc       BIC
##  985.4413  988.5842 1009.3858
```

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 -0.9194867 43.04688 32.02198 -2.351651 10.99932 0.6799502
##                     ACF1
## Training set -0.06676219
```

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
## 22.71429       370.9474 312.4342 429.4606 281.4592 460.4357
## 22.85714       366.9414 307.3374 426.5453 275.7850 458.0978
## 23.00000       321.1197 260.4446 381.7948 228.3251 413.9142
## 23.14286       219.6958 157.9682 281.4235 125.2915 314.1001
## 23.28571       223.5961 160.8336 286.3587 127.6091 319.5832
## 23.42857       344.3326 280.5520 408.1133 246.7885 441.8767
## 23.57143       356.8269 292.0431 421.6107 257.7486 455.9051
## 23.71429       370.9474 305.1768 436.7180 270.3599 471.5349
## 23.85714       366.9414 300.1985 433.6842 264.8670 469.0158
## 24.00000       321.1197 253.4185 388.8208 217.5797 424.6596
## 24.14286       219.6958 151.0498 288.3419 114.7108 324.6809
## 24.28571       223.5961 154.0180 293.1743 117.1856 330.0067
## 24.42857       344.3326 273.8348 414.8305 236.5155 452.1498
## 24.57143       356.8269 285.4202 428.2336 247.6197 466.0340
```

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 -0.9194867 43.04688 32.02198 -2.351651 10.999318 0.6799502
## Test set     -9.7799936 37.75467 28.39140 -3.682129  9.266691 0.6028589
##                     ACF1 Theil's U
## Training set -0.06676219        NA
## Test set      0.13061792 0.4259769
```

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 = 0.1774
##     gamma = 1e-04
##
##   Initial states:
##     l = 326.7586
##     s = 6.7708 45.7372 57.1993 34.777 34.4967 -89.1465
##            -89.8345
##
##   sigma:  44.3439
##
##      AIC     AICc      BIC
## 1163.638 1166.257 1189.177
```

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
## 24.71429       359.2945 302.4655 416.1235 272.3820 446.2069
## 24.85714       347.8354 290.1187 405.5521 259.5654 436.1055
## 25.00000       308.8660 250.2751 367.4569 219.2589 398.4731
## 25.14286       212.2648 152.8125 271.7171 121.3403 303.1892
## 25.28571       212.9511 152.6498 273.2525 120.7282 305.1741
## 25.42857       336.5967 275.4580 397.7353 243.0932 430.1002
## 25.57143       336.8785 274.9129 398.8440 242.1103 431.6466
## 25.71429       359.2945 296.5138 422.0751 263.2798 455.3092
## 25.85714       347.8354 284.2501 411.4207 250.5901 445.0807
## 26.00000       308.8660 244.4861 373.2459 210.4055 407.3265
## 26.14286       212.2648 147.1000 277.4295 112.6038 311.9257
## 26.28571       212.9511 147.0108 278.8915 112.1041 313.7982
## 26.42857       336.5967 269.8898 403.3035 234.5773 438.6160
## 26.57143       336.8785 269.4129 404.3440 233.6988 440.0581
```

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_California2_ets <- cbind(next2weeks, forecast_data$`Point Forecast`)
final_forecast_California2_ets
```

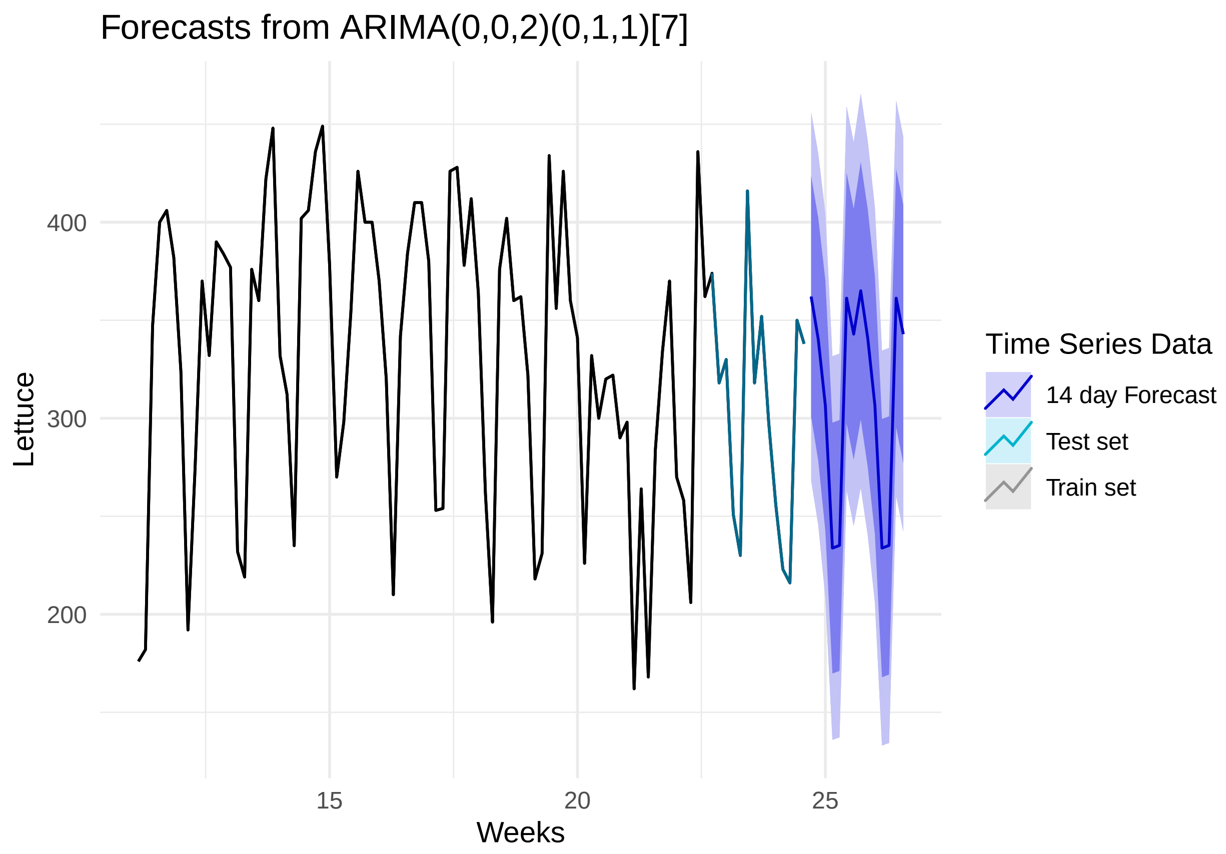
```
##    day forecast_data$`Point Forecast`
## 1    1                       359.2945
## 2    2                       347.8354
## 3    3                       308.8660
## 4    4                       212.2648
## 5    5                       212.9511
## 6    6                       336.5967
## 7    7                       336.8785
## 8    8                       359.2945
## 9    9                       347.8354
## 10  10                       308.8660
## 11  11                       212.2648
## 12  12                       212.9511
## 13  13                       336.5967
## 14  14                       336.8785
```

Comparison

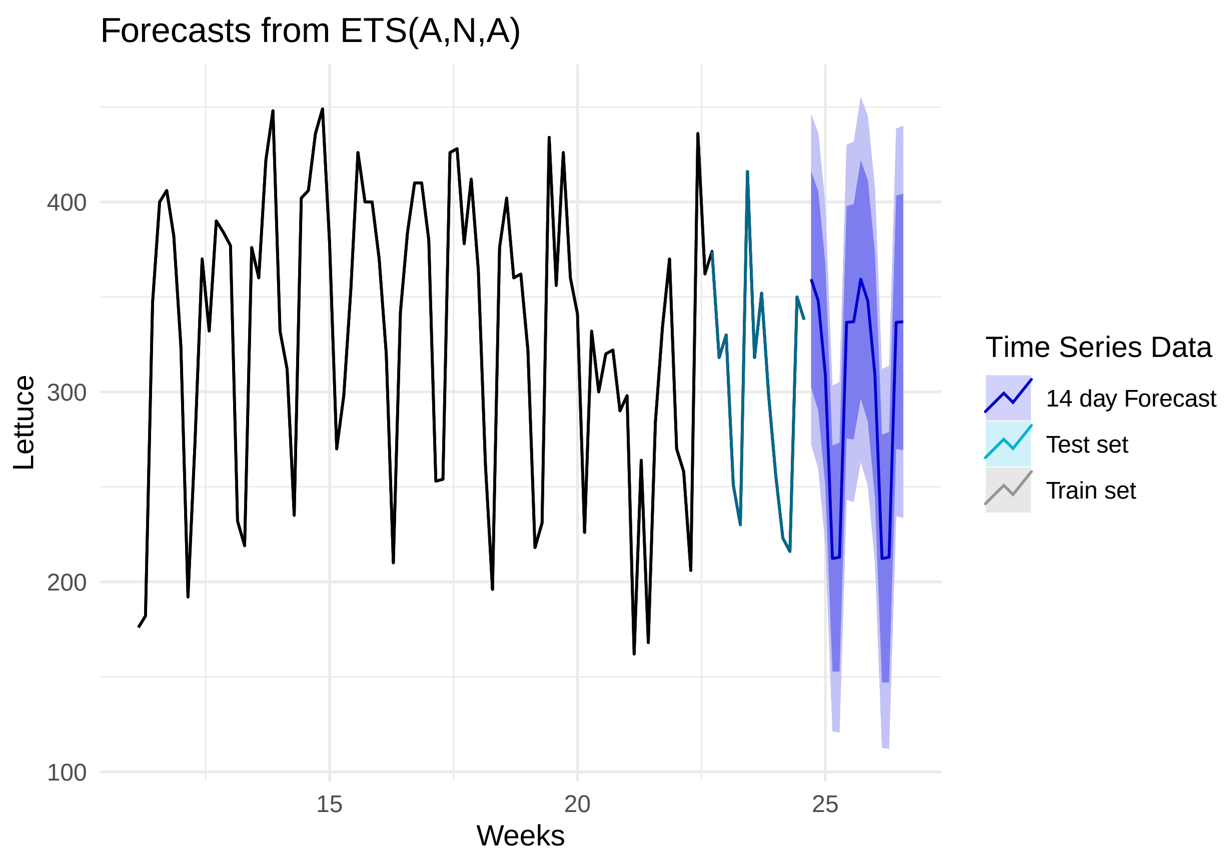
Now we will compare the two best models for California 2 Store (4904).

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

```r
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()
```



```r
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(0,0,2)(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 -0.9194867 43.04688 32.02198 -2.351651 10.999318 0.6799502
## Test set     -9.7799936 37.75467 28.39140 -3.682129  9.266691 0.6028589
##                     ACF1 Theil's U
## Training set -0.06676219        NA
## Test set      0.13061792 0.4259769
```

```
# best arima model
# ARIMA(0,0,2)(0,1,1)
accuracy.m1
```

```
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  -0.8700088 44.92879 32.33449 -1.577557 10.82420 0.6865860
## Test set     -24.2285979 41.64471 34.94981 -8.726629 11.57019 0.7421192
##                    ACF1 Theil's U
## Training set 0.04671246        NA
## Test set     0.15125519 0.4528834
```

We can observe that ETS(A,N,A) has a better (lower) RMSE (37.75467 vs 41.64471) respectively.

Therefore, we choose the ETS(A,N,A) for California2 (4904) store.

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

```
final_forecast_California2_ets
```

```
##    day forecast_data$`Point Forecast`
## 1    1                       359.2945
## 2    2                       347.8354
## 3    3                       308.8660
## 4    4                       212.2648
## 5    5                       212.9511
## 6    6                       336.5967
## 7    7                       336.8785
## 8    8                       359.2945
## 9    9                       347.8354
## 10  10                       308.8660
## 11  11                       212.2648
## 12  12                       212.9511
## 13  13                       336.5967
## 14  14                       336.8785
```