Tomato Sales Prediction Using ARIMA Model

Sharmin Hossain

19 December 2020

1 Introduction

In retail industry, if the sales quantity is predicted properly, half of the difficult work seems to be done. Failing to predict the sales properly can leads to multiple difficulties such as losing orders, overstocking inventory, customer dissatisfaction due to overwhelming complaints etc.

In this project, I have tried to apply a time series analysis over a series of data to understand how with time, sales quantity per month changes. I have applied ARIMA method and Seasonal Naïve method to understand which model will fit the data properly.

After that, With ARIMA method, got a point forecast as well as prediction interval for future sales. To understand how the model works out with real data, I have tested some prediction interval with original sales value.

The motivation to work on this project came from the company I work for. For perishable, we have to go through a number of issues everyday such as damaged product, under stocked product. overstocked products, customer complaints, overpricing products etc. Most of the issues can be solved by estimating the correct prediction of sales. That's why i started to take interest on one product and have interest to work on multiple ones if this works properly.

2 Scientific Review

I have read multiple papers to understand this model behaviour. Among them, this one from Reutlingen University seemed most relevant for my purpose.[6]

Their paper name is **Sales Prediction with Parametrized Time Series Analysis**. This paper tried to find out a sales prediction using their historical sales value and daily sales prices. They have used ARIMA model and F_r method to compare the accuracy of the model.

First they tried to use default ARIMA model to get their desired output. But the disappointing result of 47 percent accuracy motivated them to modify their work. In their modified version they tried to assume that the future price is casually influenced and should not be treated as stochastic variable. They also tried to filter out cyclic behavior from the "white noise" in the case of low volume sales.

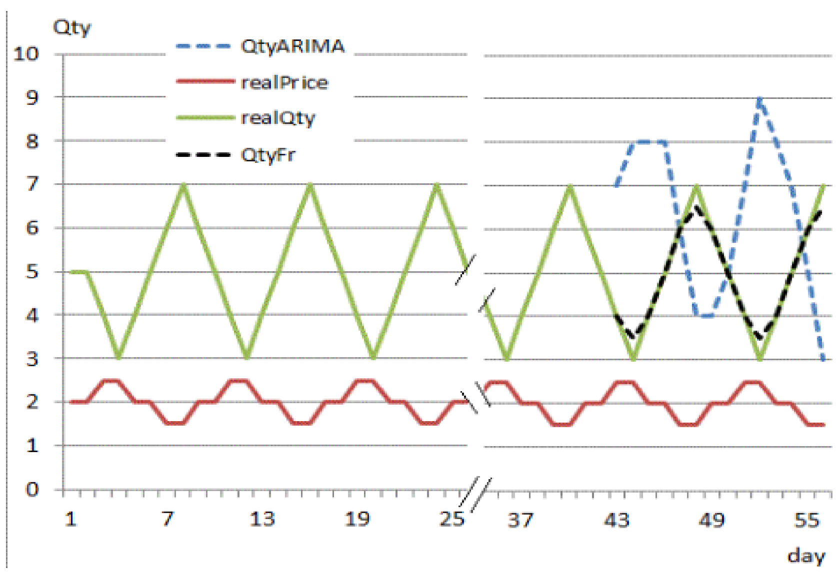


Figure 1: Forecast of Synthetic time series with delayed price-sales dependency

3 Data And Methodology

Data Collection: I have collected the data from the company I am working as a data analyst. After collecting the data, I have modified it to avoid the real sales value. I have compared two methods and then decided on a analysis which method to use to fit my data set.

Seasonal Naïve Method: For naïve forecasts, we simply set all forecasts to be the value of the last observation. A similar method is useful for highly seasonal data. In this case, we set each forecast to be equal to the last observed value from the same season of the year (e.g., the same month of the previous year). Formally, the forecast for time

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$$

This looks more complicated than it really is. For example, with monthly data, the forecast for all future February values is equal to the last observed February value. With quarterly data, the forecast of all future Q2 values is equal to the last observed Q2 value (where Q2 means the second quarter). Similar rules apply for other months and quarters, and for other seasonal periods.[7]

ARIMA Method: ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.
An ARIMA model is characterized by 3 terms: p, d, q, where, p is the order of the AR term and q is the order of the MA term and d is the number of differencing required to make the time series stationary

'p' is the order of the 'Auto Regressive' (AR) term. It refers to the number of lags of Y to be used as predictors. And 'q' is the order of the 'Moving Average' (MA) term. It refers to the number of lagged forecast errors that should go into the ARIMA Model.

A pure Auto Regressive (AR only) model is one where Y_t depends only on its own lags. That is, Y_t is a function of the 'lags of Y_t'.

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$$

where, Y_{t-1} is the lag1 of the series, β_1 is the coefficient of lag1 that the model estimates and α is the intercept term, also estimated by the model.

Likewise a pure Moving Average (MA only) model is one where Y_t depends only on the lagged forecast errors.

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

where the error terms are the errors of the auto regressive models of the respective lags. The errors E_t and E_{t-1} are the errors from the following equations :

$$Y_t = \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_0 Y_0 + \epsilon_t Y_{t-1} = \beta_1 Y_{t-2} + \beta_2 Y_{t-3} + \ldots + \beta_0 Y_0 + \epsilon_{t-1}$$

An ARIMA model is one where the time series was differenced at least once to make it stationary and combine the AR and the MA terms. So the equation becomes:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \ldots + \phi_q \epsilon_{t-q}$$

ARIMA model in words:

Predicted Y_t = Constant + Linear combination Lags of Y (upto p lags) + Linear Combination of Lagged forecast errors (upto q lags) [1]

4 Working Procedure

- Loaded the CSV file ("Tomato5") into R Studio.Checked the summary of the data set.
- Loaded the package "fpp2" for using forecast, ggplot2 packages.
- Converted the data set into time series data frame for further analysis.
- Plotted the initial graph based on this time series value.
- As there could be some positive upward trend, to remove this trend, i simply made the data set flat/stationary so that trend does not effect my data anymore. So instead of looking at the raw data we were going to look at the differences in change in sales for each month.
- Then plotted the modified data sets again to see how it looks like now.
- After that, i plotted a seasonal chart to see for each year, if there were any pattern for each month. Evidently, for each year, in winter, the sales trend lines were kind of similar.
- Then i plotted a sub series plot to see the changes in mean for each month. Similarly, in winter, mean values for each year were almost same.
- After the initial analysis, i first tried our data-set with "Seasonal Naive Method" to see if its fitted with our data properly. The residual standard deviation for this method was 456.9902.
- Then we tried out ARIMA model. It gave us a residual standard deviation of 403.3646.
- As ARIMA model seemed to be better than Seasonal Naive method here, I used this for further calculation. I used the forecast formula and plotted the forecast value for next 24 months.
- The forecast output gave us two types of results, one was point forecast and another one was prediction interval based on 80 percent and 95 percent prediction interval.

- Lastly, to check how good my model works, I tried to predict some of the existing months data using historical values. [2][3][4]

5 Result

By Using the ARIMA Model, We predicted the future sales value to identify the possible patterns in the coming days.

If i only depend on point forecast, the model did not give a very good prediction there.

But in 95 percent prediction interval, it matched with every month of 2020. (Except for April 2020)

Though the prediction interval distance was very high, it worked well.

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       2200.752 1965.244 2436.259 1840.5741 2560.929
## Feb 2020       2461.146 2084.469 2837.823 1885.0689 3037.224
## Mar 2020       2638.349 2160.534 3116.164 1907.5945 3369.104
## Apr 2020       2573.680 2012.673 3134.687 1715.6940 3431.666
## May 2020       2338.253 1704.888 2971.617 1369.6057 3306.900
## Jun 2020       2189.398 1491.134 2887.661 1121.4958 3257.299
## Jul 2020       2024.009 1266.386 2781.633  865.3241 3182.695
## Aug 2020       2094.998 1282.339 2907.657  852.1429 3337.853
## Sep 2020       2138.197 1273.999 3002.394  816.5211 3459.872
## Oct 2020       2151.263 1238.433 3064.093  755.2101 3547.315
## Nov 2020       2241.560 1282.560 3200.559  774.8969 3708.223
```

```
#Testing With Original Data

original_2020<-tail(Y,11)
print(original_2020)
```

//E:/NDSU/NDSU_ Classes/Applied Regression/Project/Final Project/TomatoProject.html 16/17

9/2020 TomatoProject.R

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov
## 2020 2280 2320 2170 4930 3330 3300 3090 2670 3170 3480 3700
```

Figure 2: Comparison with Predicted value vs Original Value

6 Future Research

It is crucial to predicting the sales amounts as close as to the actual sales amounts for enterprises to increase their profits. Unless an accurate forecasting model is built, cash flow problems are inevitable. Therefore, building this kind of prediction models for sales forecasting has a high priority for the organizations.[5]

As the prediction interval distance is too high, i need to add more variables to make my ARIMA model perfect.

I can use daily price change, marketing budget, holiday/festival data set, weather data set to get a better prediction interval so that it can help predicting other products properly too.

7 Reference

1. https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/
2. Time Series Forecasting Example in RStudio
3. ARIMA modeling in R
4. Time Series In R — Time Series Forecasting
5. Bohanec, M., Borštnar, M. K., Robnik-Šikonja, M. (2017). Explaining machine learning models in sales predictions. Expert Systems with Applications, 71, 416-428
6. Sales Prediction with Parametrized Time Series Analysis
7. Some Simple Forecasting Methods

TomatoProject.R

HP

2020-12-19

```
# Read CSV File

my_data <- read.csv("E:/NDSU/NDSU_ Classes/Applied Regression/Project/Final Project/Tomato5.csv"
)

#See the Summary of the data

summary(my_data)
```

```
##      Date           Sales.Per.Day
##  Length:83          Min.   :  20
##  Class :character   1st Qu.: 545
##  Mode  :character   Median :1420
##                     Mean   :1434
##                     3rd Qu.:2075
##                     Max.   :4930
```

```
#Check out the library
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 4.0.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## -- Attaching packages ---------------------------------------------- fpp2 2.4 --
```

```
## v ggplot2   3.3.2     v fma       2.4
## v forecast  8.13      v expsmooth 2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.0.3
```

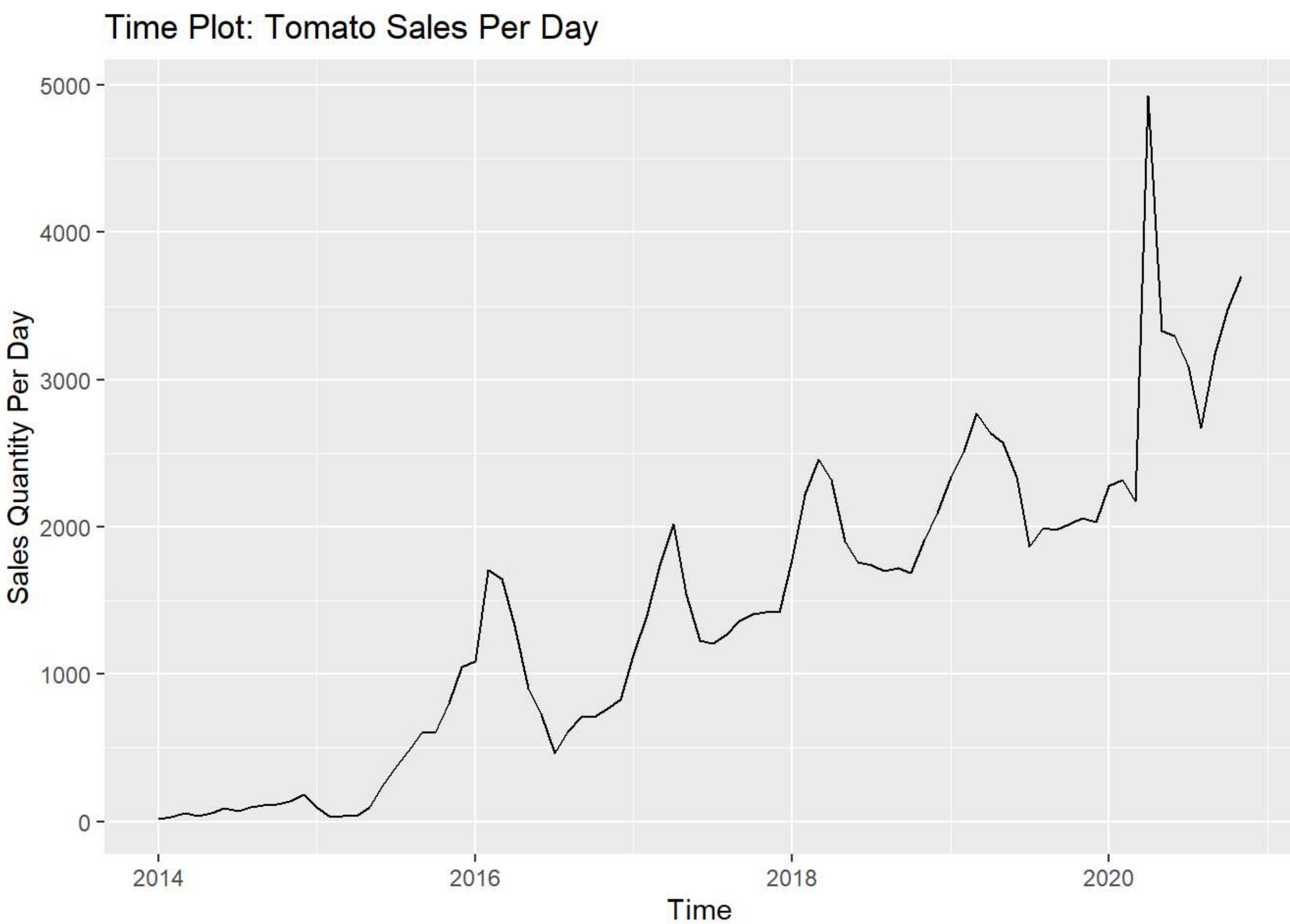
```
## Warning: package 'forecast' was built under R version 4.0.3
```

```
## Warning: package 'fma' was built under R version 4.0.3
```

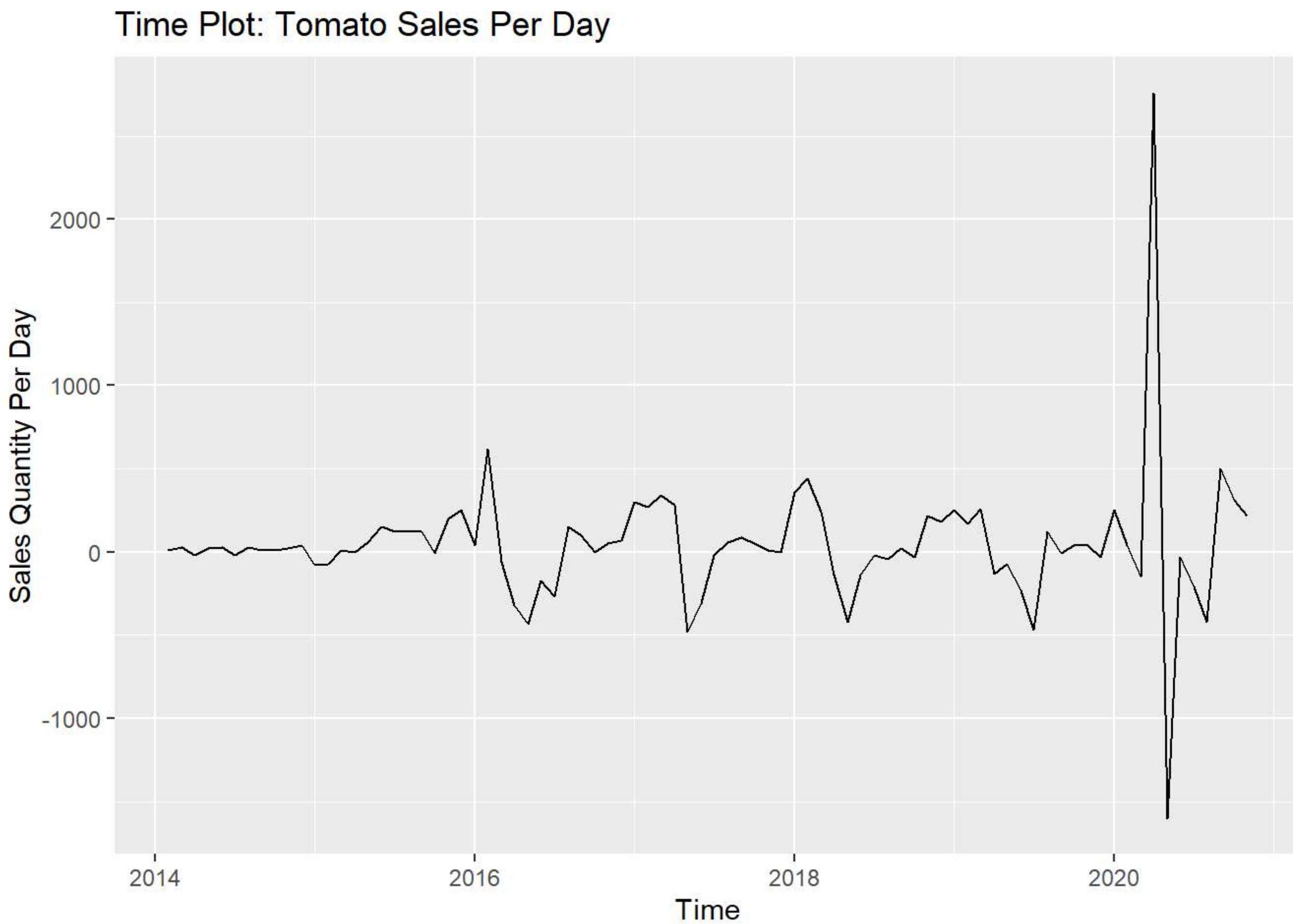
```
## Warning: package 'expsmooth' was built under R version 4.0.3
```

```
##
```

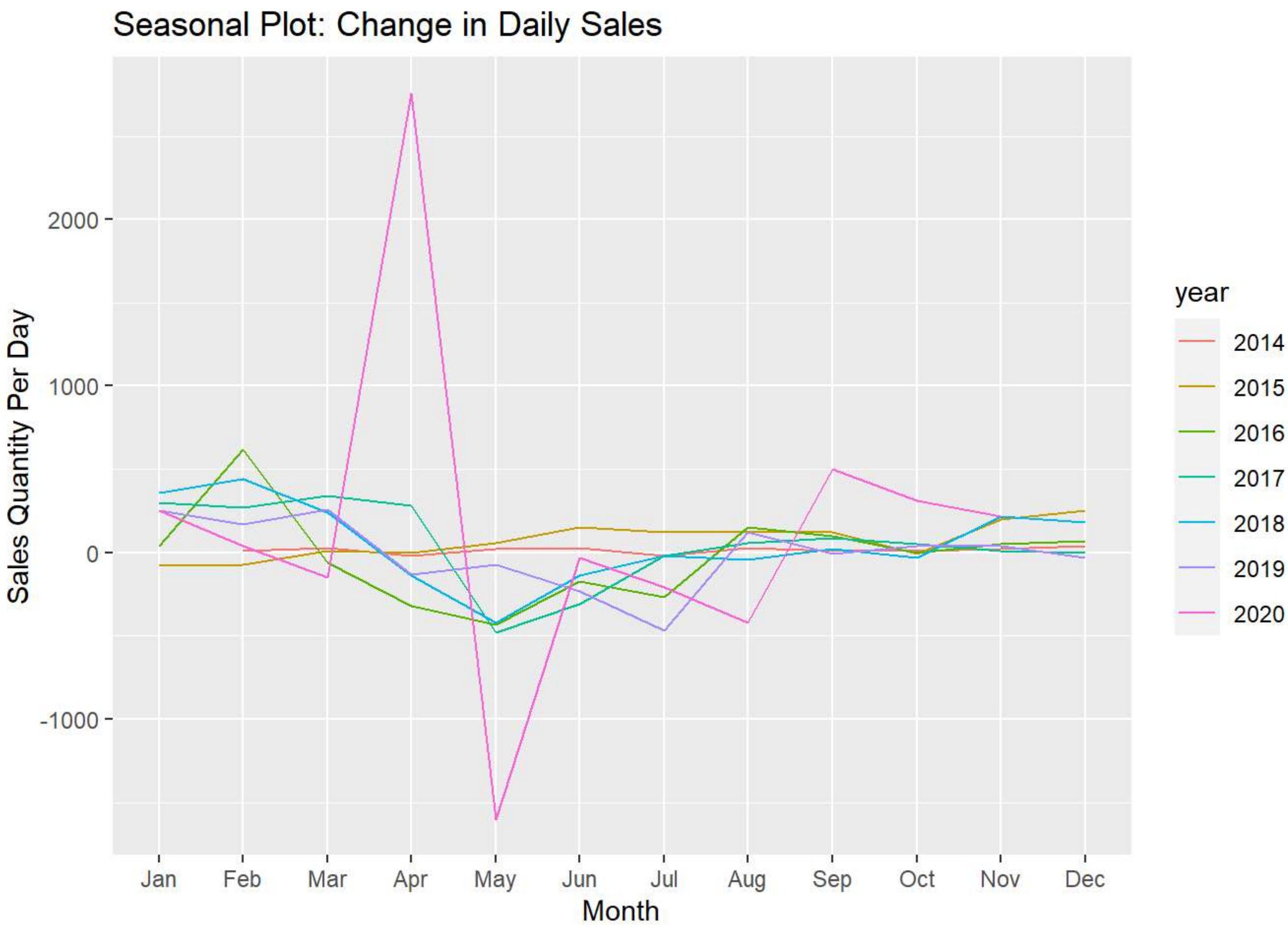
```
#Convert the data frame into time series data

Y<-ts(my_data[,2],start=c(2014,1),frequency = 12)

#Plot the main time series v alues

autoplot(Y)+ggtitle("Time Plot: Tomato Sales Per Day")+
  ylab("Sales Quantity Per Day")
```



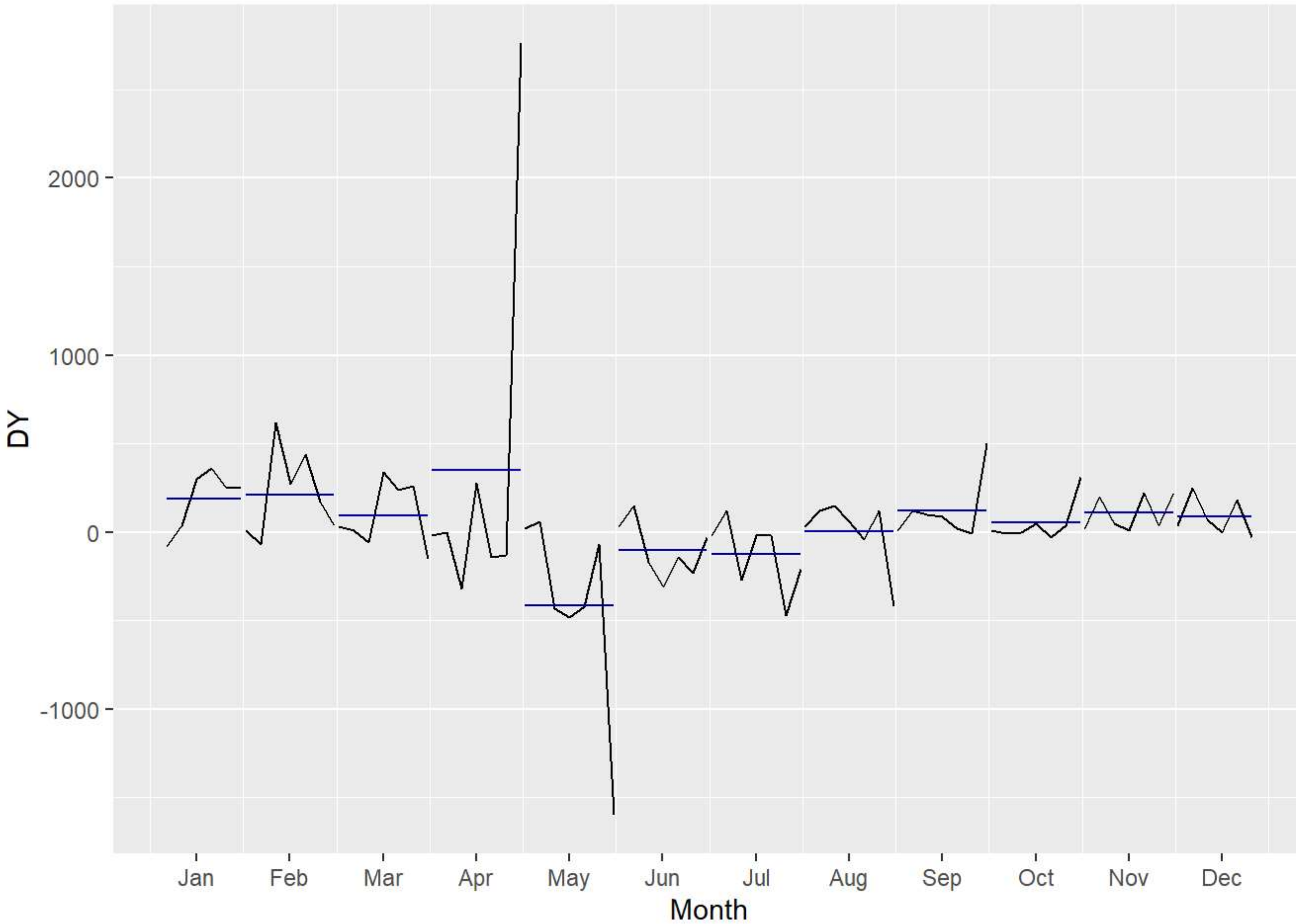

```
#Differencing the Y value

DY<-diff(Y)

#Plot the modified time series values

autoplot(DY)+ggtitle("Time Plot: Tomato Sales Per Day")+
  ylab("Sales Quantity Per Day")
```




```
#Plot a seasonal chart

ggseasonplot(DY)+
  ggtitle("Seasonal Plot: Change in Daily Sales")+
  ylab("Sales Quantity Per Day")
```




```
#Plot a subseries chart

ggsubseriesplot(DY)
```

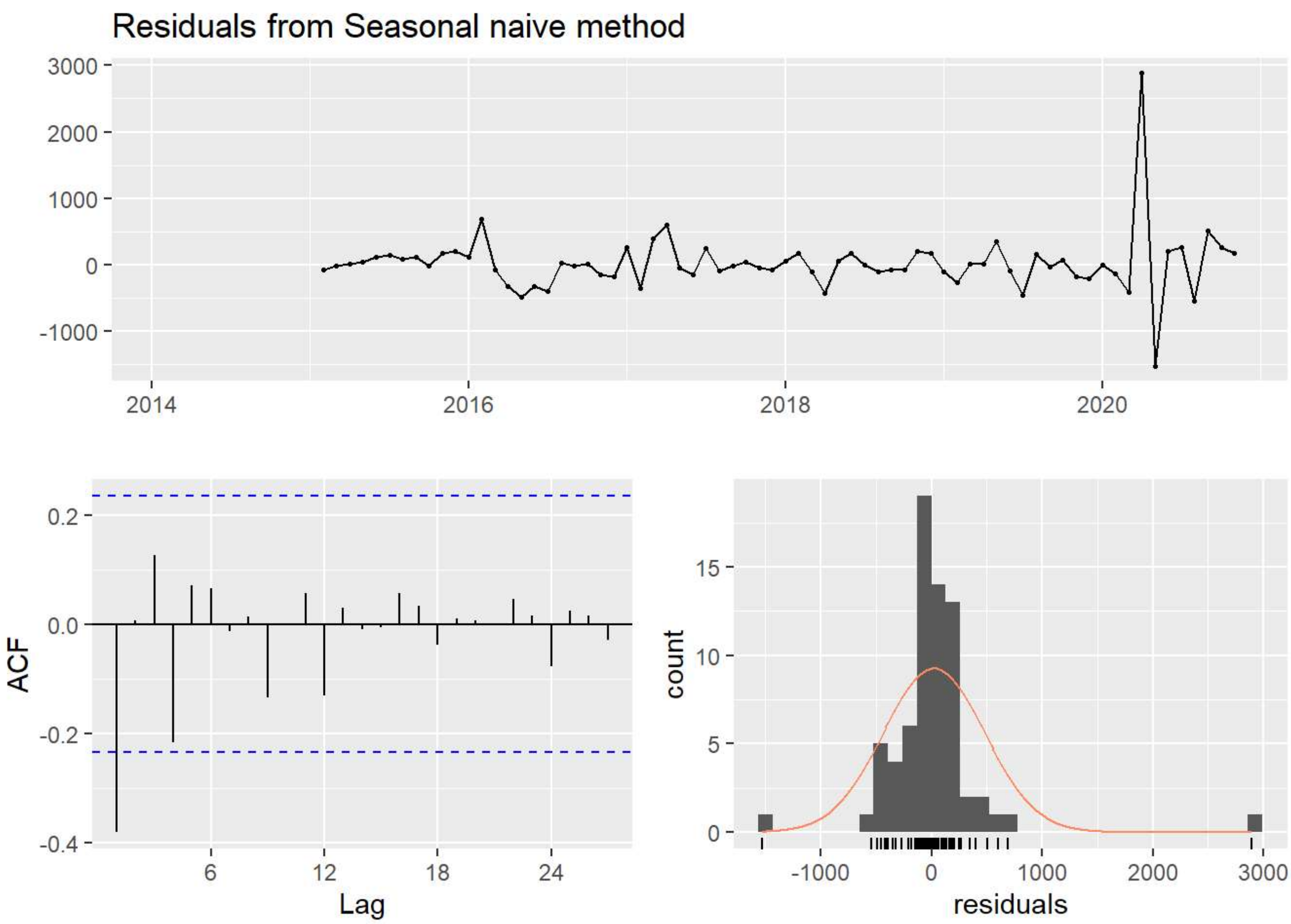



```
#Seasonal Naive Method as our benchmark #Residual sd: 45.699
#y_t=y_{t-s}+e_t

fit<-snaive(DY)
print(summary(fit))
```

```
##
## Forecast method: Seasonal naive method
##
## Model Information:
## Call: snaive(y = DY)
##
## Residual sd: 456.9902
##
## Error measures:
##                    ME     RMSE      MAE MPE MAPE MASE      ACF1
## Training set 22.28571 456.9902 237.4286 NaN  Inf    1 -0.381392
##
## Forecasts:
##          Point Forecast       Lo 80      Hi 80      Lo 95     Hi 95
## Dec 2020            -30  -615.65645   555.6564  -925.6842  865.6842
## Jan 2021            250  -335.65645   835.6564  -645.6842 1145.6842
## Feb 2021             40  -545.65645   625.6564  -855.6842  935.6842
## Mar 2021           -150  -735.65645   435.6564 -1045.6842  745.6842
## Apr 2021           2760  2174.34355  3345.6564  1864.3158 3655.6842
## May 2021          -1600 -2185.65645 -1014.3436 -2495.6842 -704.3158
## Jun 2021            -30  -615.65645   555.6564  -925.6842  865.6842
## Jul 2021           -210  -795.65645   375.6564 -1105.6842  685.6842
## Aug 2021           -420 -1005.65645   165.6564 -1315.6842  475.6842
## Sep 2021            500   -85.65645  1085.6564  -395.6842 1395.6842
## Oct 2021            310  -275.65645   895.6564  -585.6842 1205.6842
## Nov 2021            220  -365.65645   805.6564  -675.6842 1115.6842
## Dec 2021            -30  -858.24329   798.2433 -1296.6888 1236.6888
## Jan 2022            250  -578.24329  1078.2433 -1016.6888 1516.6888
## Feb 2022             40  -788.24329   868.2433 -1226.6888 1306.6888
## Mar 2022           -150  -978.24329   678.2433 -1416.6888 1116.6888
## Apr 2022           2760  1931.75671  3588.2433  1493.3112 4026.6888
## May 2022          -1600 -2428.24329  -771.7567 -2866.6888 -333.3112
## Jun 2022            -30  -858.24329   798.2433 -1296.6888 1236.6888
## Jul 2022           -210 -1038.24329   618.2433 -1476.6888 1056.6888
## Aug 2022           -420 -1248.24329   408.2433 -1686.6888  846.6888
## Sep 2022            500  -328.24329  1328.2433  -766.6888 1766.6888
## Oct 2022            310  -518.24329  1138.2433  -956.6888 1576.6888
## Nov 2022            220  -608.24329  1048.2433 -1046.6888 1486.6888
##          Point Forecast       Lo 80      Hi 80      Lo 95     Hi 95
## Dec 2020            -30  -615.65645   555.6564  -925.6842  865.6842
## Jan 2021            250  -335.65645   835.6564  -645.6842 1145.6842
## Feb 2021             40  -545.65645   625.6564  -855.6842  935.6842
## Mar 2021           -150  -735.65645   435.6564 -1045.6842  745.6842
## Apr 2021           2760  2174.34355  3345.6564  1864.3158 3655.6842
## May 2021          -1600 -2185.65645 -1014.3436 -2495.6842 -704.3158
## Jun 2021            -30  -615.65645   555.6564  -925.6842  865.6842
## Jul 2021           -210  -795.65645   375.6564 -1105.6842  685.6842
## Aug 2021           -420 -1005.65645   165.6564 -1315.6842  475.6842
## Sep 2021            500   -85.65645  1085.6564  -395.6842 1395.6842
## Oct 2021            310  -275.65645   895.6564  -585.6842 1205.6842
## Nov 2021            220  -365.65645   805.6564  -675.6842 1115.6842
## Dec 2021            -30  -858.24329   798.2433 -1296.6888 1236.6888
## Jan 2022            250  -578.24329  1078.2433 -1016.6888 1516.6888
```

```
## Feb 2022            40  -788.24329   868.2433 -1226.6888 1306.6888
## Mar 2022          -150  -978.24329   678.2433 -1416.6888 1116.6888
## Apr 2022          2760  1931.75671  3588.2433  1493.3112 4026.6888
## May 2022         -1600 -2428.24329  -771.7567 -2866.6888 -333.3112
## Jun 2022           -30  -858.24329   798.2433 -1296.6888 1236.6888
## Jul 2022          -210 -1038.24329   618.2433 -1476.6888 1056.6888
## Aug 2022          -420 -1248.24329   408.2433 -1686.6888  846.6888
## Sep 2022           500  -328.24329  1328.2433  -766.6888 1766.6888
## Oct 2022           310  -518.24329  1138.2433  -956.6888 1576.6888
## Nov 2022           220  -608.24329  1048.2433 -1046.6888 1486.6888
```

```
checkresiduals(fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 19.838, df = 16, p-value = 0.2276
##
## Model df: 0.   Total lags used: 16
```

```
#Fit on ARIMA Model

fit_arima<-auto.arima(Y,d=1,D=1,stepwise = FALSE,approximation = FALSE,trace=TRUE)
```

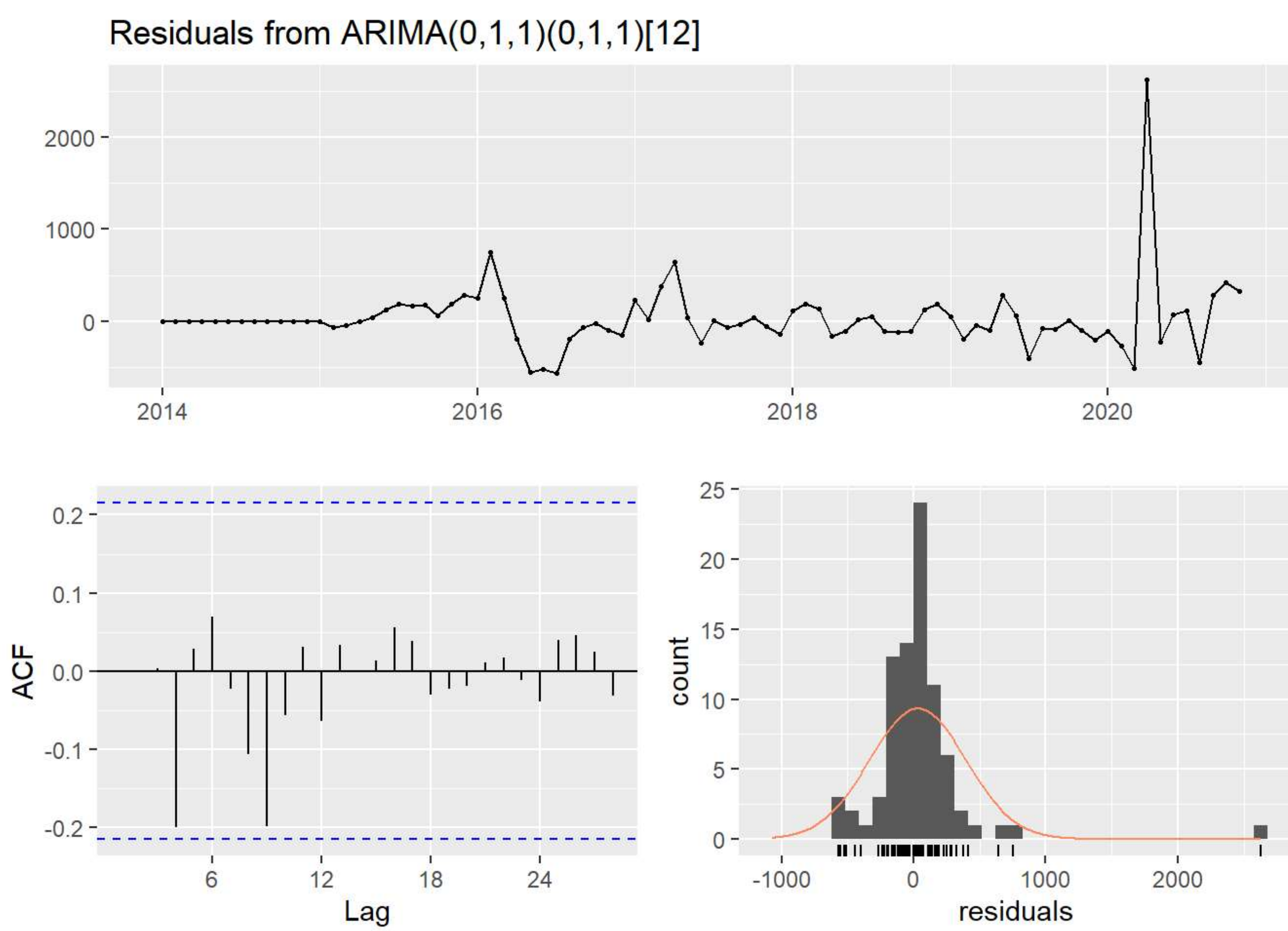
```
##
##  ARIMA(0,1,0)(0,1,0)[12]                    : 1058.163
##  ARIMA(0,1,0)(0,1,1)[12]                    : 1055.437
##  ARIMA(0,1,0)(0,1,2)[12]                    : 1057.277
##  ARIMA(0,1,0)(1,1,0)[12]                    : 1056.817
##  ARIMA(0,1,0)(1,1,1)[12]                    : 1057.51
##  ARIMA(0,1,0)(1,1,2)[12]                    : 1059.262
##  ARIMA(0,1,0)(2,1,0)[12]                    : 1055.746
##  ARIMA(0,1,0)(2,1,1)[12]                    : Inf
##  ARIMA(0,1,0)(2,1,2)[12]                    : Inf
##  ARIMA(0,1,1)(0,1,0)[12]                    : 1048.205
##  ARIMA(0,1,1)(0,1,1)[12]                    : 1045.49
##  ARIMA(0,1,1)(0,1,2)[12]                    : 1047.594
##  ARIMA(0,1,1)(1,1,0)[12]                    : 1046.676
##  ARIMA(0,1,1)(1,1,1)[12]                    : 1047.695
##  ARIMA(0,1,1)(1,1,2)[12]                    : Inf
##  ARIMA(0,1,1)(2,1,0)[12]                    : 1045.821
##  ARIMA(0,1,1)(2,1,1)[12]                    : 1046.245
##  ARIMA(0,1,1)(2,1,2)[12]                    : 1048.633
##  ARIMA(0,1,2)(0,1,0)[12]                    : 1050.324
##  ARIMA(0,1,2)(0,1,1)[12]                    : 1047.704
##  ARIMA(0,1,2)(0,1,2)[12]                    : 1049.823
##  ARIMA(0,1,2)(1,1,0)[12]                    : 1048.902
##  ARIMA(0,1,2)(1,1,1)[12]                    : 1049.962
##  ARIMA(0,1,2)(1,1,2)[12]                    : 1052.05
##  ARIMA(0,1,2)(2,1,0)[12]                    : 1048.09
##  ARIMA(0,1,2)(2,1,1)[12]                    : 1048.635
##  ARIMA(0,1,3)(0,1,0)[12]                    : 1052.554
##  ARIMA(0,1,3)(0,1,1)[12]                    : 1049.701
##  ARIMA(0,1,3)(0,1,2)[12]                    : 1051.709
##  ARIMA(0,1,3)(1,1,0)[12]                    : 1051.01
##  ARIMA(0,1,3)(1,1,1)[12]                    : 1051.98
##  ARIMA(0,1,3)(2,1,0)[12]                    : 1049.677
##  ARIMA(0,1,4)(0,1,0)[12]                    : Inf
##  ARIMA(0,1,4)(0,1,1)[12]                    : Inf
##  ARIMA(0,1,4)(1,1,0)[12]                    : Inf
##  ARIMA(0,1,5)(0,1,0)[12]                    : Inf
##  ARIMA(1,1,0)(0,1,0)[12]                    : 1049.598
##  ARIMA(1,1,0)(0,1,1)[12]                    : 1047.438
##  ARIMA(1,1,0)(0,1,2)[12]                    : 1049.634
##  ARIMA(1,1,0)(1,1,0)[12]                    : 1048.681
##  ARIMA(1,1,0)(1,1,1)[12]                    : 1049.675
##  ARIMA(1,1,0)(1,1,2)[12]                    : Inf
##  ARIMA(1,1,0)(2,1,0)[12]                    : 1047.582
##  ARIMA(1,1,0)(2,1,1)[12]                    : 1047.805
##  ARIMA(1,1,0)(2,1,2)[12]                    : 1050.195
##  ARIMA(1,1,1)(0,1,0)[12]                    : 1050.324
##  ARIMA(1,1,1)(0,1,1)[12]                    : 1047.682
##  ARIMA(1,1,1)(0,1,2)[12]                    : Inf
##  ARIMA(1,1,1)(1,1,0)[12]                    : 1048.892
##  ARIMA(1,1,1)(1,1,1)[12]                    : Inf
##  ARIMA(1,1,1)(1,1,2)[12]                    : Inf
##  ARIMA(1,1,1)(2,1,0)[12]                    : Inf
```

```
##  ARIMA(1,1,1)(2,1,1)[12]                    : 1048.343
##  ARIMA(1,1,2)(0,1,0)[12]                    : 1052.574
##  ARIMA(1,1,2)(0,1,1)[12]                    : 1049.636
##  ARIMA(1,1,2)(0,1,2)[12]                    : Inf
##  ARIMA(1,1,2)(1,1,0)[12]                    : 1050.751
##  ARIMA(1,1,2)(1,1,1)[12]                    : Inf
##  ARIMA(1,1,2)(2,1,0)[12]                    : Inf
##  ARIMA(1,1,3)(0,1,0)[12]                    : 1053.972
##  ARIMA(1,1,3)(0,1,1)[12]                    : Inf
##  ARIMA(1,1,3)(1,1,0)[12]                    : 1053.14
##  ARIMA(1,1,4)(0,1,0)[12]                    : Inf
##  ARIMA(2,1,0)(0,1,0)[12]                    : 1049.993
##  ARIMA(2,1,0)(0,1,1)[12]                    : 1048
##  ARIMA(2,1,0)(0,1,2)[12]                    : 1050.295
##  ARIMA(2,1,0)(1,1,0)[12]                    : 1049.123
##  ARIMA(2,1,0)(1,1,1)[12]                    : 1050.314
##  ARIMA(2,1,0)(1,1,2)[12]                    : Inf
##  ARIMA(2,1,0)(2,1,0)[12]                    : 1048.55
##  ARIMA(2,1,0)(2,1,1)[12]                    : 1049.092
##  ARIMA(2,1,1)(0,1,0)[12]                    : 1051.34
##  ARIMA(2,1,1)(0,1,1)[12]                    : Inf
##  ARIMA(2,1,1)(0,1,2)[12]                    : Inf
##  ARIMA(2,1,1)(1,1,0)[12]                    : Inf
##  ARIMA(2,1,1)(1,1,1)[12]                    : Inf
##  ARIMA(2,1,1)(2,1,0)[12]                    : Inf
##  ARIMA(2,1,2)(0,1,0)[12]                    : 1053.558
##  ARIMA(2,1,2)(0,1,1)[12]                    : Inf
##  ARIMA(2,1,2)(1,1,0)[12]                    : Inf
##  ARIMA(2,1,3)(0,1,0)[12]                    : 1055.236
##  ARIMA(3,1,0)(0,1,0)[12]                    : 1051.835
##  ARIMA(3,1,0)(0,1,1)[12]                    : 1050.262
##  ARIMA(3,1,0)(0,1,2)[12]                    : 1052.65
##  ARIMA(3,1,0)(1,1,0)[12]                    : 1051.371
##  ARIMA(3,1,0)(1,1,1)[12]                    : 1052.656
##  ARIMA(3,1,0)(2,1,0)[12]                    : 1050.918
##  ARIMA(3,1,1)(0,1,0)[12]                    : 1053.633
##  ARIMA(3,1,1)(0,1,1)[12]                    : Inf
##  ARIMA(3,1,1)(1,1,0)[12]                    : Inf
##  ARIMA(3,1,2)(0,1,0)[12]                    : Inf
##  ARIMA(4,1,0)(0,1,0)[12]                    : 1052.497
##  ARIMA(4,1,0)(0,1,1)[12]                    : 1050
##  ARIMA(4,1,0)(1,1,0)[12]                    : 1051.35
##  ARIMA(4,1,1)(0,1,0)[12]                    : 1054.7
##  ARIMA(5,1,0)(0,1,0)[12]                    : 1054.525
## 
## 
## 
##  Best model: ARIMA(0,1,1)(0,1,1)[12]
```

```
print(summary(fit_arima))
```

```
## Series: Y
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.4452  -0.4181
## s.e.   0.1164   0.1709
##
## sigma^2 estimated as 162703:  log likelihood=-519.56
## AIC=1045.13   AICc=1045.49   BIC=1051.87
##
## Training set error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 30.57467 365.1016 183.916 -3.745538 17.46485 0.3396107
##                      ACF1
## Training set -0.001002148
##                    ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 30.57467 365.1016 183.916 -3.745538 17.46485 0.3396107
##                      ACF1
## Training set -0.001002148
```

```
checkresiduals(fit_arima)
```

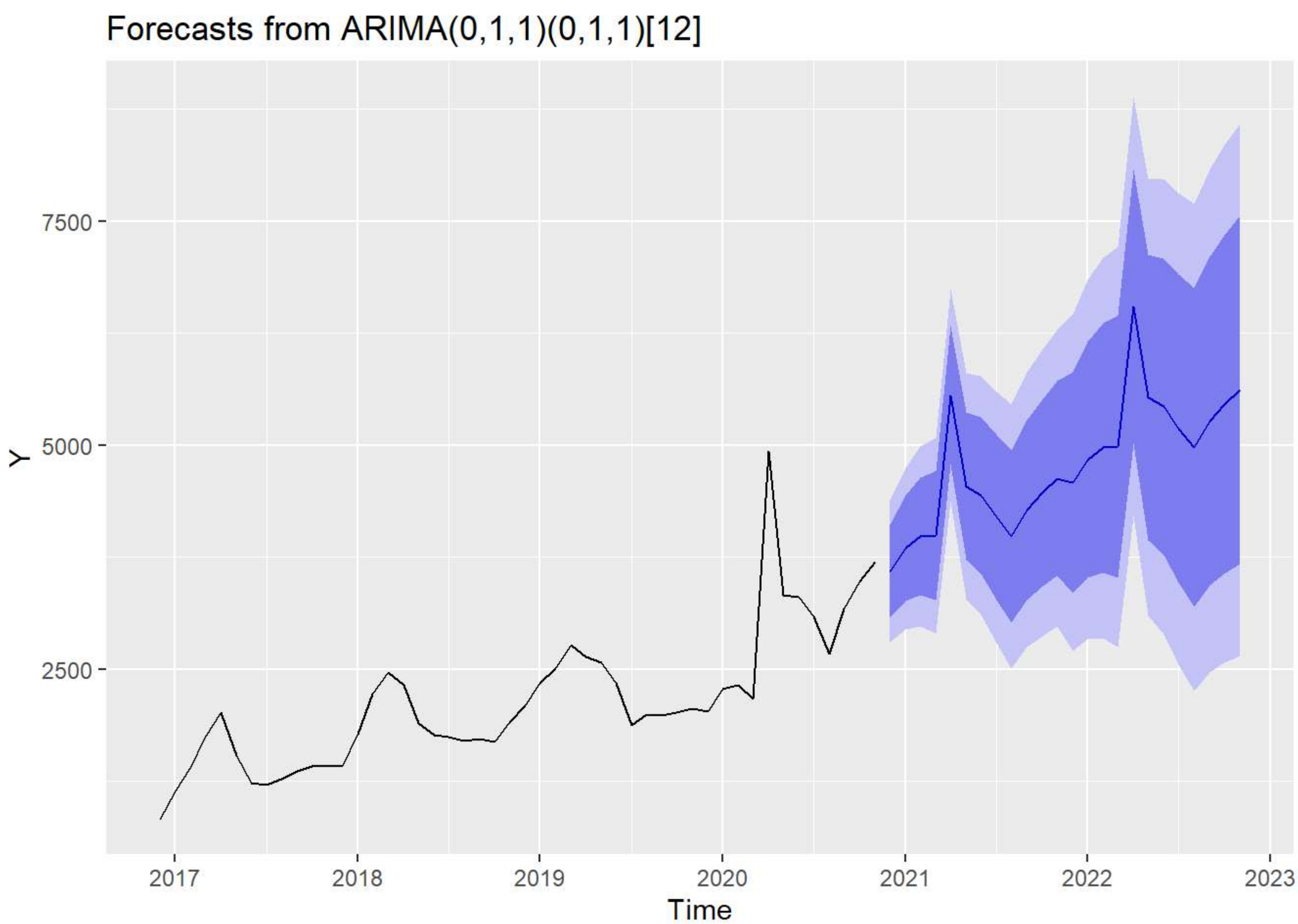


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(0,1,1)[12]
## Q* = 10.395, df = 15, p-value = 0.7942
##
## Model df: 2.   Total lags used: 17
```

```
sqrt(162703)
```

```
## [1] 403.3646
```

```
#Forecasting on ARIMA Model

fcst<-forecast(fit_arima,h=24)
autoplot(fcst,include=48)
```




```
print(summary(fcst))
```

```
##
## Forecast method: ARIMA(0,1,1)(0,1,1)[12]
##
## Model Information:
## Series: Y
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.4452  -0.4181
## s.e.   0.1164   0.1709
##
## sigma^2 estimated as 162703:  log likelihood=-519.56
## AIC=1045.13   AICc=1045.49   BIC=1051.87
##
## Error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 30.57467 365.1016 183.916 -3.745538 17.46485 0.3396107
##                      ACF1
## Training set -0.001002148
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Dec 2020       3591.388 3074.449 4108.328 2800.798 4381.979
## Jan 2021       3847.541 3256.381 4438.701 2943.440 4751.643
## Feb 2021       3979.102 3322.053 4636.151 2974.232 4983.972
## Mar 2021       3993.106 3276.198 4710.015 2896.690 5089.523
## Apr 2021       5559.518 4787.377 6331.658 4378.631 6740.404
## May 2021       4540.993 3717.316 5364.670 3281.287 5800.699
## Jun 2021       4438.388 3566.214 5310.562 3104.513 5772.263
## Jul 2021       4194.946 3276.833 5113.059 2790.813 5599.078
## Aug 2021       3981.996 3020.136 4943.856 2510.958 5453.034
## Sep 2021       4279.213 3275.511 5282.915 2744.183 5814.243
## Oct 2021       4468.395 3424.526 5512.263 2871.935 6064.854
## Nov 2021       4631.550 3549.005 5714.096 2975.940 6287.161
## Dec 2021       4582.788 3351.041 5814.536 2698.993 6466.583
## Jan 2022       4838.941 3526.300 6151.582 2831.430 6846.453
## Feb 2022       4970.502 3581.673 6359.331 2846.471 7094.533
## Mar 2022       4984.506 3523.457 6445.556 2750.024 7218.989
## Apr 2022       6550.917 5021.053 8080.782 4211.192 8890.643
## May 2022       5532.393 3936.678 7128.107 3091.958 7972.827
## Jun 2022       5429.788 3770.835 7088.740 2892.639 7966.937
## Jul 2022       5186.346 3466.478 6906.213 2556.036 7816.655
## Aug 2022       4973.396 3194.699 6752.093 2253.114 7693.678
## Sep 2022       5270.613 3434.971 7106.255 2463.241 8077.985
## Oct 2022       5459.794 3568.921 7350.667 2567.954 8351.635
## Nov 2022       5622.950 3678.414 7567.486 2649.039 8596.861
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Dec 2020       3591.388 3074.449 4108.328 2800.798 4381.979
## Jan 2021       3847.541 3256.381 4438.701 2943.440 4751.643
## Feb 2021       3979.102 3322.053 4636.151 2974.232 4983.972
## Mar 2021       3993.106 3276.198 4710.015 2896.690 5089.523
## Apr 2021       5559.518 4787.377 6331.658 4378.631 6740.404
```

```
## May 2021       4540.993 3717.316 5364.670 3281.287 5800.699
## Jun 2021       4438.388 3566.214 5310.562 3104.513 5772.263
## Jul 2021       4194.946 3276.833 5113.059 2790.813 5599.078
## Aug 2021       3981.996 3020.136 4943.856 2510.958 5453.034
## Sep 2021       4279.213 3275.511 5282.915 2744.183 5814.243
## Oct 2021       4468.395 3424.526 5512.263 2871.935 6064.854
## Nov 2021       4631.550 3549.005 5714.096 2975.940 6287.161
## Dec 2021       4582.788 3351.041 5814.536 2698.993 6466.583
## Jan 2022       4838.941 3526.300 6151.582 2831.430 6846.453
## Feb 2022       4970.502 3581.673 6359.331 2846.471 7094.533
## Mar 2022       4984.506 3523.457 6445.556 2750.024 7218.989
## Apr 2022       6550.917 5021.053 8080.782 4211.192 8890.643
## May 2022       5532.393 3936.678 7128.107 3091.958 7972.827
## Jun 2022       5429.788 3770.835 7088.740 2892.639 7966.937
## Jul 2022       5186.346 3466.478 6906.213 2556.036 7816.655
## Aug 2022       4973.396 3194.699 6752.093 2253.114 7693.678
## Sep 2022       5270.613 3434.971 7106.255 2463.241 8077.985
## Oct 2022       5459.794 3568.921 7350.667 2567.954 8351.635
## Nov 2022       5622.950 3678.414 7567.486 2649.039 8596.861
```

```
#Check with Original vs Predicted Value

YTest<-ts(my_data[,2],frequency=12,start=c(2014,1),end=c(2019,12))
fitTest<-auto.arima(YTest,d=1,D=1,stepwise=FALSE,approximation = FALSE,trace=TRUE)
```

```
##
##  ARIMA(0,1,0)(0,1,0)[12]                    : 808.3935
##  ARIMA(0,1,0)(0,1,1)[12]                    : 795.9805
##  ARIMA(0,1,0)(0,1,2)[12]                    : Inf
##  ARIMA(0,1,0)(1,1,0)[12]                    : 799.3952
##  ARIMA(0,1,0)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,0)(1,1,2)[12]                    : Inf
##  ARIMA(0,1,0)(2,1,0)[12]                    : 799.3141
##  ARIMA(0,1,0)(2,1,1)[12]                    : Inf
##  ARIMA(0,1,0)(2,1,2)[12]                    : Inf
##  ARIMA(0,1,1)(0,1,0)[12]                    : 808.2001
##  ARIMA(0,1,1)(0,1,1)[12]                    : 794.7672
##  ARIMA(0,1,1)(0,1,2)[12]                    : Inf
##  ARIMA(0,1,1)(1,1,0)[12]                    : 799.4682
##  ARIMA(0,1,1)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,1)(1,1,2)[12]                    : Inf
##  ARIMA(0,1,1)(2,1,0)[12]                    : 798.6091
##  ARIMA(0,1,1)(2,1,1)[12]                    : Inf
##  ARIMA(0,1,1)(2,1,2)[12]                    : Inf
##  ARIMA(0,1,2)(0,1,0)[12]                    : 809.5609
##  ARIMA(0,1,2)(0,1,1)[12]                    : 797.0715
##  ARIMA(0,1,2)(0,1,2)[12]                    : Inf
##  ARIMA(0,1,2)(1,1,0)[12]                    : 801.3493
##  ARIMA(0,1,2)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,2)(1,1,2)[12]                    : Inf
##  ARIMA(0,1,2)(2,1,0)[12]                    : 800.9482
##  ARIMA(0,1,2)(2,1,1)[12]                    : Inf
##  ARIMA(0,1,3)(0,1,0)[12]                    : 811.8644
##  ARIMA(0,1,3)(0,1,1)[12]                    : 799.4181
##  ARIMA(0,1,3)(0,1,2)[12]                    : Inf
##  ARIMA(0,1,3)(1,1,0)[12]                    : 803.7328
##  ARIMA(0,1,3)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,3)(2,1,0)[12]                    : 803.2117
##  ARIMA(0,1,4)(0,1,0)[12]                    : 813.9531
##  ARIMA(0,1,4)(0,1,1)[12]                    : Inf
##  ARIMA(0,1,4)(1,1,0)[12]                    : Inf
##  ARIMA(0,1,5)(0,1,0)[12]                    : Inf
##  ARIMA(1,1,0)(0,1,0)[12]                    : 808.7955
##  ARIMA(1,1,0)(0,1,1)[12]                    : 795.0395
##  ARIMA(1,1,0)(0,1,2)[12]                    : Inf
##  ARIMA(1,1,0)(1,1,0)[12]                    : 799.8518
##  ARIMA(1,1,0)(1,1,1)[12]                    : Inf
##  ARIMA(1,1,0)(1,1,2)[12]                    : Inf
##  ARIMA(1,1,0)(2,1,0)[12]                    : 798.9885
##  ARIMA(1,1,0)(2,1,1)[12]                    : Inf
##  ARIMA(1,1,0)(2,1,2)[12]                    : Inf
##  ARIMA(1,1,1)(0,1,0)[12]                    : 809.6018
##  ARIMA(1,1,1)(0,1,1)[12]                    : 797.0715
##  ARIMA(1,1,1)(0,1,2)[12]                    : Inf
##  ARIMA(1,1,1)(1,1,0)[12]                    : 801.3309
##  ARIMA(1,1,1)(1,1,1)[12]                    : Inf
##  ARIMA(1,1,1)(1,1,2)[12]                    : Inf
##  ARIMA(1,1,1)(2,1,0)[12]                    : 800.9665
```

```
##  ARIMA(1,1,1)(2,1,1)[12]                     : Inf
##  ARIMA(1,1,2)(0,1,0)[12]                     : 811.8635
##  ARIMA(1,1,2)(0,1,1)[12]                     : Inf
##  ARIMA(1,1,2)(0,1,2)[12]                     : Inf
##  ARIMA(1,1,2)(1,1,0)[12]                     : 803.7221
##  ARIMA(1,1,2)(1,1,1)[12]                     : Inf
##  ARIMA(1,1,2)(2,1,0)[12]                     : Inf
##  ARIMA(1,1,3)(0,1,0)[12]                     : Inf
##  ARIMA(1,1,3)(0,1,1)[12]                     : Inf
##  ARIMA(1,1,3)(1,1,0)[12]                     : Inf
##  ARIMA(1,1,4)(0,1,0)[12]                     : Inf
##  ARIMA(2,1,0)(0,1,0)[12]                     : 809.8791
##  ARIMA(2,1,0)(0,1,1)[12]                     : 796.9422
##  ARIMA(2,1,0)(0,1,2)[12]                     : Inf
##  ARIMA(2,1,0)(1,1,0)[12]                     : 801.4879
##  ARIMA(2,1,0)(1,1,1)[12]                     : Inf
##  ARIMA(2,1,0)(1,1,2)[12]                     : Inf
##  ARIMA(2,1,0)(2,1,0)[12]                     : 800.7361
##  ARIMA(2,1,0)(2,1,1)[12]                     : Inf
##  ARIMA(2,1,1)(0,1,0)[12]                     : 811.8761
##  ARIMA(2,1,1)(0,1,1)[12]                     : Inf
##  ARIMA(2,1,1)(0,1,2)[12]                     : Inf
##  ARIMA(2,1,1)(1,1,0)[12]                     : Inf
##  ARIMA(2,1,1)(1,1,1)[12]                     : Inf
##  ARIMA(2,1,1)(2,1,0)[12]                     : Inf
##  ARIMA(2,1,2)(0,1,0)[12]                     : Inf
##  ARIMA(2,1,2)(0,1,1)[12]                     : Inf
##  ARIMA(2,1,2)(1,1,0)[12]                     : Inf
##  ARIMA(2,1,3)(0,1,0)[12]                     : Inf
##  ARIMA(3,1,0)(0,1,0)[12]                     : 812.1188
##  ARIMA(3,1,0)(0,1,1)[12]                     : 798.8604
##  ARIMA(3,1,0)(0,1,2)[12]                     : Inf
##  ARIMA(3,1,0)(1,1,0)[12]                     : 803.8553
##  ARIMA(3,1,0)(1,1,1)[12]                     : Inf
##  ARIMA(3,1,0)(2,1,0)[12]                     : 802.6333
##  ARIMA(3,1,1)(0,1,0)[12]                     : Inf
##  ARIMA(3,1,1)(0,1,1)[12]                     : Inf
##  ARIMA(3,1,1)(1,1,0)[12]                     : Inf
##  ARIMA(3,1,2)(0,1,0)[12]                     : Inf
##  ARIMA(4,1,0)(0,1,0)[12]                     : 813.9395
##  ARIMA(4,1,0)(0,1,1)[12]                     : 799.9092
##  ARIMA(4,1,0)(1,1,0)[12]                     : 804.6628
##  ARIMA(4,1,1)(0,1,0)[12]                     : 816.3976
##  ARIMA(5,1,0)(0,1,0)[12]                     : 816.3737
##
##
##
##  Best model: ARIMA(0,1,1)(0,1,1)[12]
```

```
ftest<-forecast(fitTest,h=11)
print(summary(ftest))
```

```
##
## Forecast method: ARIMA(0,1,1)(0,1,1)[12]
##
## Model Information:
## Series: YTest
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##          ma1     sma1
##       0.2540  -0.7301
## s.e.  0.1294   0.2615
##
## sigma^2 estimated as 33097:  log likelihood=-394.17
## AIC=794.33   AICc=794.77   BIC=800.56
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 0.783362 161.8688 110.6011 -1.825139 12.66811 0.2421922 0.01522368
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       2200.752 1965.244 2436.259 1840.5741 2560.929
## Feb 2020       2461.146 2084.469 2837.823 1885.0689 3037.224
## Mar 2020       2638.349 2160.534 3116.164 1907.5945 3369.104
## Apr 2020       2573.680 2012.673 3134.687 1715.6940 3431.666
## May 2020       2338.253 1704.888 2971.617 1369.6057 3306.900
## Jun 2020       2189.398 1491.134 2887.661 1121.4958 3257.299
## Jul 2020       2024.009 1266.386 2781.633  865.3241 3182.695
## Aug 2020       2094.998 1282.339 2907.657  852.1429 3337.853
## Sep 2020       2138.197 1273.999 3002.394  816.5211 3459.872
## Oct 2020       2151.263 1238.433 3064.093  755.2101 3547.315
## Nov 2020       2241.560 1282.560 3200.559  774.8969 3708.223
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       2200.752 1965.244 2436.259 1840.5741 2560.929
## Feb 2020       2461.146 2084.469 2837.823 1885.0689 3037.224
## Mar 2020       2638.349 2160.534 3116.164 1907.5945 3369.104
## Apr 2020       2573.680 2012.673 3134.687 1715.6940 3431.666
## May 2020       2338.253 1704.888 2971.617 1369.6057 3306.900
## Jun 2020       2189.398 1491.134 2887.661 1121.4958 3257.299
## Jul 2020       2024.009 1266.386 2781.633  865.3241 3182.695
## Aug 2020       2094.998 1282.339 2907.657  852.1429 3337.853
## Sep 2020       2138.197 1273.999 3002.394  816.5211 3459.872
## Oct 2020       2151.263 1238.433 3064.093  755.2101 3547.315
## Nov 2020       2241.560 1282.560 3200.559  774.8969 3708.223
```

```
#Testing With Original Data

original_2020<-tail(Y,11)
print(original_2020)
```

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov
## 2020 2280 2320 2170 4930 3330 3300 3090 2670 3170 3480 3700
```