Sales Forecast with ARIMA

Forecast has been always a crucial and challenging task for a business. Given analysis is intended to explore and evaluate ARIMA forecasting techniques, its accuracy in prediction and application.

1. Profiling Data

Data set is derived prior to the analysis from a database and contains historical sales data for the department in interest.

Data set variables:

Yr - Year;
Mo_no - Month;
TotalUniqShipTos - Monthly count of accounts invoiced;
TotalInv - Monthly sales invoices;

```
##       Yr Mo_No      Date TotalUniqShipTos   TotalInv
## 1   2021     2  2/1/2021             7215    4540000
## 2   2021     1  1/1/2021             7641 5059810.67
## 3   2020    12 12/1/2020             7006    4351108
## ...  ...   ...      <NA>              ...        ...
## 59  2016     4  4/1/2016             5403 2508213.51
## 60  2016     3  3/1/2016             6362 2926410.14
## 61  2016     2  2/1/2016             5753 2745165.31
## 62  2016     1  1/1/2016             5112 2442804.13
```

While exploring variable types, I see that Yr, Mo_no and Date variables need to be transformed into a categorical data type:

```
##
##  No. of observations =  62
##   Variable         Class           Description
## 1 Yr               integer
## 2 Mo_No            integer
## 3 Date             character
## 4 TotalUniqShipTos integer
## 5 TotalInv         numeric
```

Next I sort data by date from latest to earliest. This is to assure that training set will consist of later months and validation set will contain the most recent months.

```
dataset <- dataset[order(dataset$Date, dataset$Mo_No), ]
```

Missing Values Check:

Data set does not have any missing values, so I can move on.

```
any(is.na(dataset))
```

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

Final look at the data.

```r
attach(dataset)
des(dataset)
```

```
##
##  No. of observations =  62
##   Variable         Class           Description
## 1 Yr               factor
## 2 Mo_No            factor
## 3 Date             Date
## 4 TotalUniqShipTos integer
## 5 TotalInv         numeric
```

```r
headTail(dataset, 6)
```

```
##       Yr Mo_No       Date TotalUniqShipTos   TotalInv
## 62  2016     1 2016-01-01             5112 2442804.13
## 61  2016     2 2016-02-01             5753 2745165.31
## 60  2016     3 2016-03-01             6362 2926410.14
## 59  2016     4 2016-04-01             5403 2508213.51
## 58  2016     5 2016-05-01             5295 2496541.34
## 57  2016     6 2016-06-01             6046 2810085.05
## ... <NA>  <NA>       <NA>              ...        ...
## 4   2020    11 2020-11-01             7125    4459344
## 3   2020    12 2020-12-01             7006    4351108
## 2   2021     1 2021-01-01             7641 5059810.67
## 1   2021     2 2021-02-01             7215    4540000
```

2. Exploring Data Set

Side Note:

The current data set consists of 5 years of data (2016-2020). Important to mention that April and May of 2020 were outliers caused by pandemic. Revenue during those months did not take its natural course of flow but rather were impacted by rare event. While synthesizing this data set, I replaced sales values for April and May with values derived from pre-pandemic forecast, which allowed me to still include the those months in my analysis and utilize them in the model.

Both Fig 1 and Fig 2 show the steady exponential trend of sales and served locations. However, from looking at the data I see that 2016 year is not quite in a line with he pattern observed in the later years. Considering required at least 3 years for training the model and latest 12 month (1 year) for validating the model, I decide to exclude 2016 year from this analysis and utilize records 13 to 61 that represent 2017-2020 years.

Going forward I *train* the model on previous 36 months (3 year) - Jan, 2017 to Jan, 2020 (inclusive) and *test* the model on last 12 months - Feb 2020 to Jan 2021 (inclusive).

Fig.1 Sales Over Time

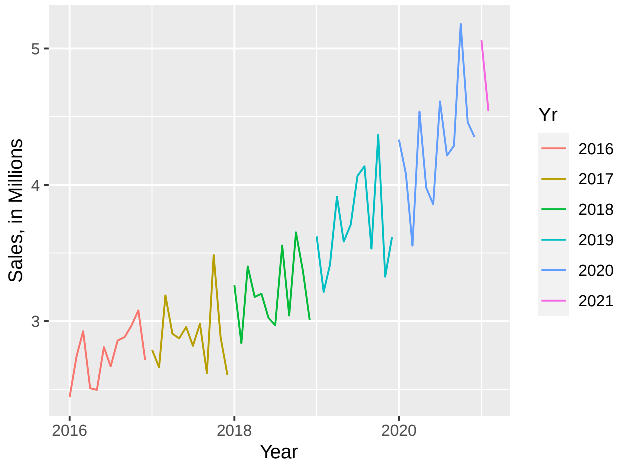
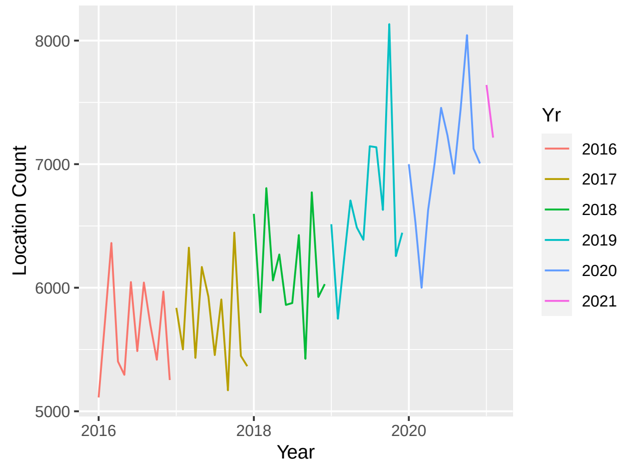


Fig.2 Locations Served Over Time

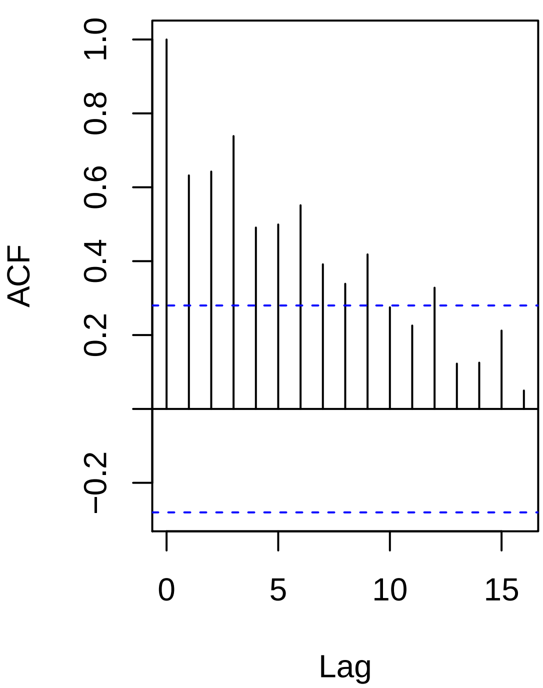


3. Evaluating Time Series

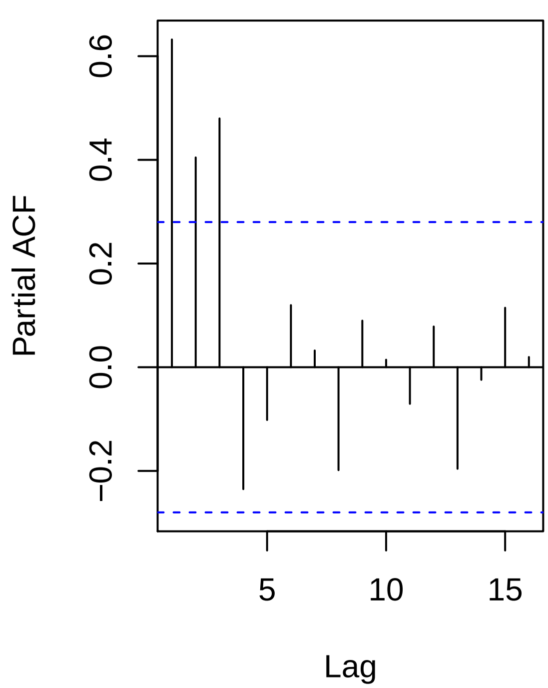
Since this is a time series data, it is necessary to determine whether the data is **stationary**.

ACF plot shows the significant correlation between current time series and previous lags. Typically, such assumption is appropriate when time series, represented by vertical lines, cross the level of significance (blue dashed line), just as demonstrated below. Also, observation suggests presence of correlation between the last 3 terms, which assumes the lag equal to 3. Since time series drop gradually on ACF graph, there is an assumption of non-stationary data behavior. (Versus, if there was a sudden drop it would point on the stationary data and zero correlation between lags.)
PACF plot also suggests lag equal to 3, because of the first three vertical lines, that are above the 95% of significance level.

ACF for Sales



PACF for Sales



To verify above assumptions, adf.test() is performed. Null hypothesis is defined as data is non-stationary. Since p-value is more then assumed 0.05, we fail to reject null hypothesis and have enough evidence to assume that the data is non-stationary. Adf.test() also recommends lag of 3, which confirms above evaluation from the plots.

```
tseries::adf.test(TotalInv[13:61], alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  TotalInv[13:61]
## Dickey-Fuller = -2.6275, Lag order = 3, p-value =
## 0.3231
## alternative hypothesis: stationary
```

4. Fitting AUTO.ARIMA

Next I define the training data set as time series and apply auto.arima() algorithm to find fitted model that appear to be described as (0,1,1)(0,1,0)[12].
Aiken value is 672.30, which sets the benchmark for choosing the best fitted model.

```
ts <- ts(dataset$TotalInv[13:49], frequency = 12)
ts1 <- window(ts)
(arima1 <- auto.arima(ts1, D = 1))
```

```
## Series: ts1
## ARIMA(0,1,1)(0,1,0)[12]
##
## Coefficients:
##           ma1
##       -0.8114
## s.e.   0.1198
##
## sigma^2 estimated as 7.243e+10:  log likelihood=-334.15
## AIC=672.3   AICc=672.87   BIC=674.66
```

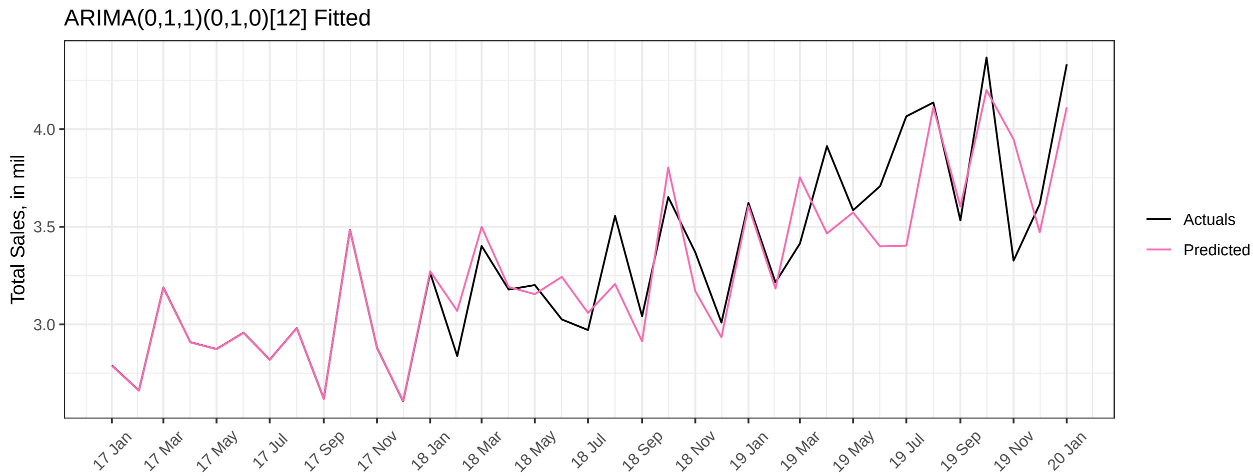
5. Forecast of the next 12 months using AUTO.ARIMA(0,1,1)(0,1,0)[12] (Option 1)

By using auto.arima() in a **for loop** with iterator 12 , I intend to forecast 12 month sales (validation set) by predicting one month at a time, while including previously predicted month in training set. By the time **for loop** is finished, I will have predicted sales for the 12 months straight.

```
train_index <- 49
n_total <- nrow(dataset[13:61, ])
dataset_train1 <- (dataset[13:(train_index), ])  # training 1-49
dataset_test <- dataset[(train_index + 1):61, ]  # testing 49-61
predicted <- numeric((n_total + 12 - train_index))  #61-49=12
for (i in 1:(n_total - train_index + 12)) {
    dataset_train <- dataset[13:(train_index - 1 + i), ]
    arima.model <- auto.arima(ts(dataset_train$TotalInv))
    pred <- forecast(arima.model, 1)
    predicted[i] <- pred$mean
}
```

Next, I calculate error rates of the fitted values for Jan, 2017 to Jan, 2020 and output them into a table and on a plot:

```
## # A tibble: 6 x 4
##   Date       Actuals Predicted `Pred.Error Rate`
##   <date>       <dbl>     <dbl>             <dbl>
## 1 2019-08-01 4135940.  4112642.              0.01
## 2 2019-09-01 3532631.  3603197.              0.02
## 3 2019-10-01 4366414.  4200143.              0.04
## 4 2019-11-01 3326330.  3948279.              0.19
## 5 2019-12-01 3615882.  3471552.              0.04
## 6 2020-01-01 4331877.  4111353.              0.05
```



6. Adding Number of Locations as a Regressor to the ARIMA model

I considered to include one more variable into a model - *TotalUniqShipTos - Count of the Served Accounts*.

AIC of model with additional variable is 684.58, while AIC of the model without it is 672.30. Apparently, previous model is better.
I make a decision to move forward without adding this additional variable.

```
(arima2 <- auto.arima(ts1, D = 1, xreg = cbind(as.numeric(dataset$TotalUniqShipTos[13:49]))))
```

```
## Series: ts1
## Regression with ARIMA(0,0,0)(1,1,0)[12] errors
##
## Coefficients:
##          sar1      drift      xreg
##       -0.4589  20521.522  384.2810
## s.e.   0.2256   3911.615   84.2086
##
## sigma^2 estimated as 3.366e+10:  log likelihood=-338.29
## AIC=684.58   AICc=686.58   BIC=689.45
```

```
arima1
```

```
## Series: ts1
## ARIMA(0,1,1)(0,1,0)[12]
##
## Coefficients:
##           ma1
```

```
##       -0.8114
## s.e.   0.1198
##
## sigma^2 estimated as 7.243e+10:  log likelihood=-334.15
## AIC=672.3   AICc=672.87   BIC=674.66
```

6. Forecasting 12 Months Sales (Defining Seasonality)

Previously, I predicted 12 months of validation set without considering seasons in the pattern. This time I am using AUTO.ARIMA algorithm with defined seasonality attribute (*seasonal = TRUE*). Also, I decide to use built-in **forecast** function instead of **for loop** approach.

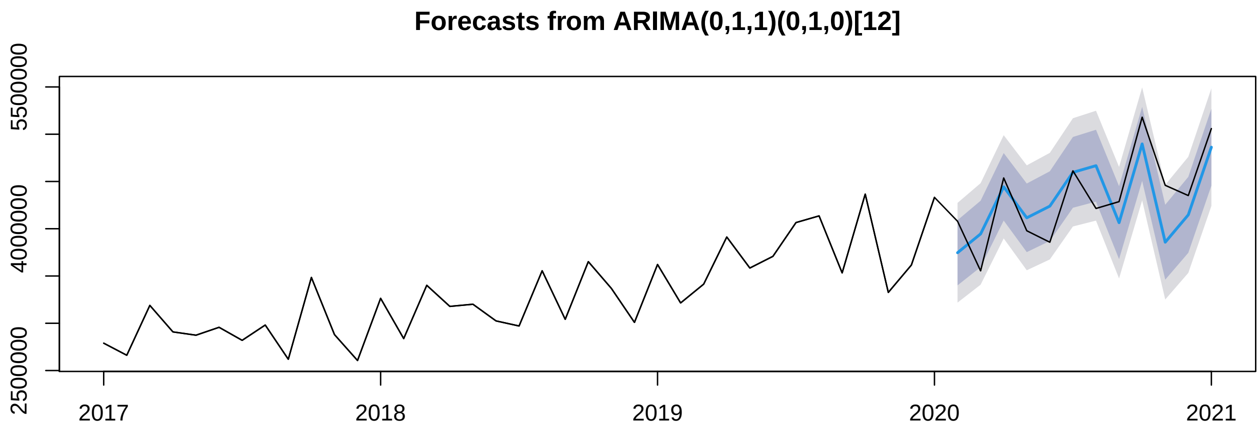
```
ts_f <- ts(dataset$TotalInv[13:49], frequency = 12, start = c(2017,
    1))
ts1_f <- window(ts_f)
summary(arima1_a <- auto.arima(ts1_f, D = 1, seasonal = TRUE))
```

```
## Series: ts1_f
## ARIMA(0,1,1)(0,1,0)[12]
##
## Coefficients:
##           ma1
##       -0.8114
## s.e.   0.1198
##
## sigma^2 estimated as 7.243e+10:  log likelihood=-334.15
## AIC=672.3   AICc=672.87   BIC=674.66
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE
## Training set 26831.17 212183.7 126313.7 0.5024001 3.566076
##                   MASE        ACF1
## Training set 0.3001183 -0.06986415
```

```
fcast2 <- forecast(arima1_a, h = 12)
```

The desired outcome is to have the black line (*observed values*) to stay within the shaded area (range of confidence levels). This indicates on a proper model forecasting technique while all predicted values are withing 80-95% of confidence level.

```
plot(fcast2)
lines((window(ts(dataset$TotalInv[13:61], frequency = 12, start = c(2017,
    1)), D = 1)))
```



7. Prediction Error Rates Comparison Between Two Models

I calculate prediction error rates of the second model with defined seasonality as following:

```
actuals<-dataset[50:61,c(3,5)]

df_2_12mo <- cbind(Date =  as.Date(actuals$Date, origin ="1970-01-01"),

              actuals=actuals$TotalInv,
              predicted=fcast2$mean[1:12],
              error=round(((actuals$TotalInv-fcast2$mean[1:12])/actuals$TotalInv),2)

             )
df_2_12mo<-data.frame(df_2_12mo)
df_2_12mo$Date =  as.Date(df_2_12mo$Date, origin ="1970-01-01")
library(lubridate)
colnames(df_2_12mo)<-c("Date", "Actuals", "Predicted", "Pred.Error Rate")
df_2_12mo
```

```
##          Date Actuals Predicted Pred.Error Rate
## 1  2020-02-01 4079161   3745981            0.08
## 2  2020-03-01 3554428   3945030           -0.11
## 3  2020-04-01 4537376   4443739            0.02
## 4  2020-05-01 3978675   4115515           -0.03
## 5  2020-06-01 3857241   4239019           -0.10
## 6  2020-07-01 4612485   4596395            0.00
## 7  2020-08-01 4214495   4667063           -0.11
## 8  2020-09-01 4286579   4063754            0.05
## 9  2020-10-01 5180160   4897537            0.05
## 10 2020-11-01 4459344   3857453            0.13
## 11 2020-12-01 4351108   4147005            0.05
## 12 2021-01-01 5059811   4863000            0.04
```

Now I recall outputs from the first model and second model to compare error rates. Magnitude of the error rates indicate that first model predicts better. However, I intend to consider two and discuss the output with the domain knowledge expert to have an insight what makes more sense. Variances described by error rates sometimes can be a result of rare business events that analysts are not aware of. So, having those variances would actually mean a correct prediction of the natural course of business.

```
df1 <- as.data.frame(tail(df_1_12mo, 12))
df1
```

```
##          Date Predicted Actuals Pred.Error_Rate
## 1  2020-02-01   3535931 4079161            0.13
## 2  2020-03-01   4070634 3554428            0.15
## 3  2020-04-01   4174973 4537376            0.08
## 4  2020-05-01   3972652 3978675            0.00
## 5  2020-06-01   4037439 3857241            0.05
## 6  2020-07-01   4301916 4612485            0.07
## 7  2020-08-01   4011924 4214495            0.05
## 8  2020-09-01   4199740 4286579            0.02
## 9  2020-10-01   4574009 5180160            0.12
## 10 2020-11-01   4327525 4459344            0.03
## 11 2020-12-01   4680376 4351108            0.08
## 12 2021-01-01   5073783 5059811            0.00
```

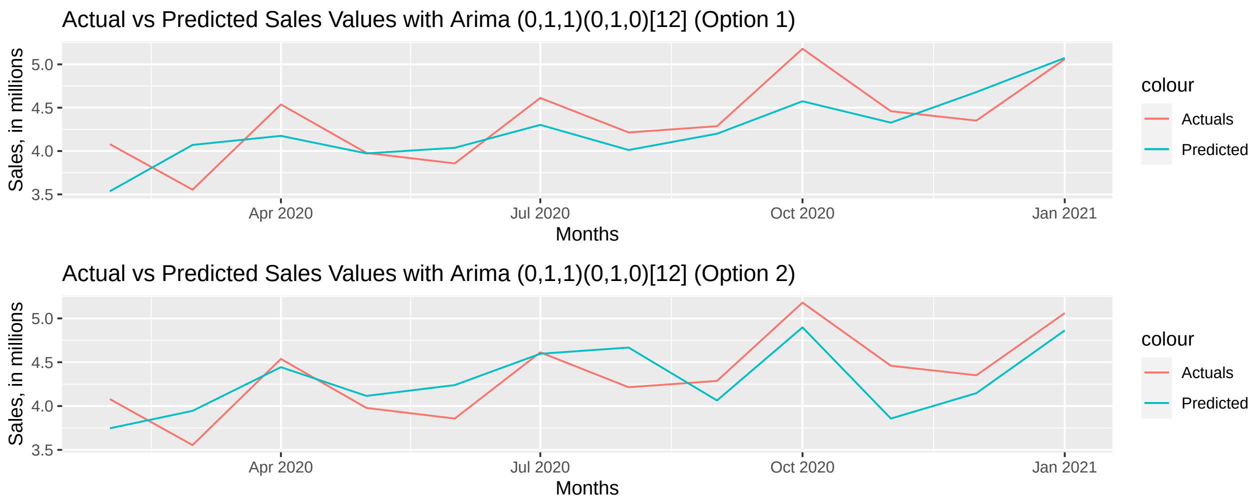
```r
df_2_12mo
```

```
##          Date Actuals Predicted Pred.Error Rate
## 1  2020-02-01 4079161   3745981            0.08
## 2  2020-03-01 3554428   3945030           -0.11
## 3  2020-04-01 4537376   4443739            0.02
## 4  2020-05-01 3978675   4115515           -0.03
## 5  2020-06-01 3857241   4239019           -0.10
## 6  2020-07-01 4612485   4596395            0.00
## 7  2020-08-01 4214495   4667063           -0.11
## 8  2020-09-01 4286579   4063754            0.05
## 9  2020-10-01 5180160   4897537            0.05
## 10 2020-11-01 4459344   3857453            0.13
## 11 2020-12-01 4351108   4147005            0.05
## 12 2021-01-01 5059811   4863000            0.04
```

```r
a <- ggplot(df1, aes(x = Date, group = 1)) + geom_line(aes(y = Actuals/1e+06,
    col = "Actuals")) + geom_line(aes(y = Predicted/1e+06, col = "Predicted")) +
    xlab("Months") + ylab("Sales, in millions") + ggtitle("Actual vs Predicted Sales Values with Arima

b <- ggplot(df_2_12mo, aes(x = Date, group = 1)) + geom_line(aes(y = Actuals/1e+06,
    col = "Actuals")) + geom_line(aes(y = Predicted/1e+06, col = "Predicted")) +
    xlab("Months") + ylab("Sales, in millions") + ggtitle("Actual vs Predicted Sales Values with Arima

grid.arrange(a, b, nrow = 2)
```



8. 2021 Sales Forecast

Now I take all historical data from January 2017 to January 2021 and output the prediction for the next 12 months of 2021.
First, I proceed with utilizing the model by forecasting one month at a time, not counting for seasonality.

```
n_total <- nrow(dataset[13:61, ])
dataset_train1 <- (dataset[13:(train_index), ])  # training 1-61
predicted <- numeric(12)
for (i in 1:(12)) {
    dataset_train <- dataset[13:(60 + i), ]
    arima.model <- auto.arima(ts(dataset_train$TotalInv))
    pred <- forecast(arima.model, 1)
    predicted[i] <- pred$mean
}
```

Next, I utilize ARIMA model with defined seasonality and predict future values based on **forecast** built-in function:

```
ts_f2 <- ts(dataset$TotalInv[13:61], frequency = 12, start = c(2017,
    1))
ts2_f <- window(ts_f2)
summary(arima2 <- auto.arima(ts_f2, D = 1, seasonal = TRUE))
```

```
## Series: ts_f2
## ARIMA(0,1,1)(0,1,0)[12]
##
## Coefficients:
##           ma1
##       -0.8122
## s.e.   0.1219
##
## sigma^2 estimated as 8.416e+10:  log likelihood=-503.92
## AIC=1011.84   AICc=1012.2   BIC=1015.01
##
## Training set error measures:
##                     ME     RMSE    MAE       MPE     MAPE
## Training set 35414.3 245180.5 164521 0.6166287 4.331813
##                    MASE       ACF1
## Training set 0.348146 -0.0186137
```

```
fcast2021 <- forecast(arima2, h = 12)
```

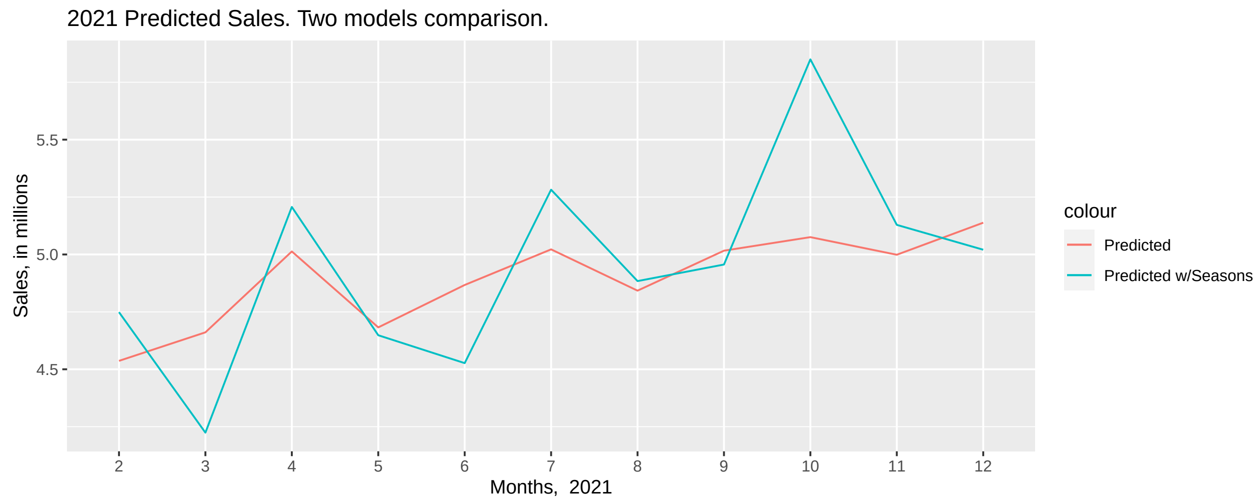
Predicted sales for the rest of 2021 (February to December) from both models look like following. Side-by-side comparison shows that some months are nearly the same, however, there are a few differences that worth discussing with domain knowledge experts in Sales Department.

```
df_2021 <- tibble(Month_2021 = c(2:12), Predicted = c(predicted[1:11]),
    Predicted_wSeas = fcast2021$mean[1:11])
df_2021
```

```
## # A tibble: 11 x 3
##    Month_2021 Predicted Predicted_wSeas
##         <int>     <dbl>           <dbl>
##  1          2  4536655.        4748728.
##  2          3  4660806.        4223995.
##  3          4  5013118.        5206944.
```

```
## 4          5  4682425.        4648243.
## 5          6  4867139.        4526808.
## 6          7  5022229.        5282053.
## 7          8  4842441.        4884063.
## 8          9  5016534.        4956146.
## 9         10  5075512.        5849727.
## 10        11  4998787.        5128911.
## 11        12  5138352.        5020675.
```

```
ggplot(df_2021, aes(x = as.factor(Month_2021), group = 1)) +
    geom_line(aes(y = Predicted/1e+06, col = "Predicted")) +
    geom_line(aes(y = Predicted_wSeas/1e+06, col = "Predicted w/Seasons")) +
    xlab("Months,  2021") + ylab("Sales, in millions") + ggtitle("2021 Predicted Sales. Two models compa
```



Considering the nature of business, I am in favor of the model with the defined seasonality. It is also worth looking at the values that lay within confidence intervals of 95% and 80%.

```
fcast2021
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Feb 2021        4748728 4376948 5120508 4180140 5317317
## Mar 2021        4223995 3845713 4602277 3645463 4802527
## Apr 2021        5206944 4822270 5591617 4618637 5795251
## May 2021        4648243 4257282 5039203 4050320 5246165
## Jun 2021        4526808 4129660 4923957 3919422 5134194
## Jul 2021        5282053 4878811 5685294 4665348 5898757
## Aug 2021        4884063 4474819 5293306 4258179 5509947
## Sep 2021        4956146 4540988 5371305 4321216 5591077
## Oct 2021        5849727 5428737 6270718 5205877 6493578
## Nov 2021        5128911 4702168 5555655 4476264 5781559
## Dec 2021        5020675 4588256 5453095 4359347 5682004
## Jan 2022        5729378 5291356 6167400 5059481 6399275
```