Capestone Project

James Strayer

02/01/2020

Introduction

I was given data from a Restaurant that sales some of their product in retail to find a model to predict the sales, The data start from 2015

Data import and cleaning

```
## Loading required package: tidyverse

## -- Attaching packages ------------------------------------------------------ tidyverse 1.2.1 --

## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.2
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   0.8.3     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0

## -- Conflicts --------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Parsed with column specification:
## cols(
##   Date = col_date(format = ""),
##   Weekday = col_character(),
##   Year = col_double(),
##   Month = col_character(),
##   Day = col_double(),
##   `annee fiscale` = col_character(),
##   Sales = col_number(),
##   Retail = col_number(),
##   TakeOutSales = col_number(),
##   Bar_Sales = col_number(),
##   Sales_Restaurant = col_number()
## )

## Observations: 1,715
## Variables: 11
## $ Date              <date> 2015-03-29, 2015-03-30, 2015-03-31, 2015-04-01, 201...
## $ Weekday           <chr> "SUNDAY", "MONDAY", "TUESDAY", "WEDNESDAY", "THURSDA...
## $ Year              <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015...
## $ Month             <chr> "March'15", "March'15", "March'15", "April'15", "Apr...
## $ Day               <dbl> 29, 30, 31, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1...
## $ `annee fiscale`   <chr> "2014-15", "2014-15", "2014-15", "2014-15", "2014-15...
## $ Sales             <dbl> 1792.00, NA, 1526.30, 2250.26, 2077.57, 2357.48, 150...
## $ Retail            <dbl> 373.00, NA, 380.95, 363.72, 268.80, 394.23, 476.17, ...
## $ TakeOutSales     <dbl> 0.00, 0.00, 99.25, 402.75, 0.00, 32.50, 93.73, 0.00,...
## $ Bar_Sales        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 227....
## $ Sales_Restaurant <dbl> 1419.00, 0.00, 1046.10, 1483.79, 1808.77, 1930.75, 9...
```

In our data set we have 10 variables for 1715 observations corresponding to the number of days the restaurant was open since the 29-03-2015.We can remove the fiscal year .The Month variable is in a format not pratical for analysis (Month, YY) we want to have just the Full moth written without the year. So for simplicity and because we have the full date in the Date column, we are going to remove the month column as it is and create a new one based on the date column, we will do the same for the weekdays column. Furthermore we don't need the day number in our analysis so we will remove it too:

```
#Remove unncessary column for the analysis
DataForAnalysis<-DailySales%>%select(-Weekday,-Month,-Day,-`annee fiscale`)
#Add a column for weekday and month
DataForAnalysis<-DataForAnalysis%>%mutate(Weekday=weekdays(Date),Month=months(Date))
```

If we look again at the data we see that we have a Sales column, representing the Total Sales for the day and then each column after it, is the total sales for each day for each component of the restaurant possible sales revenu, so Retail, Take-out, Bar and Restaurant. We can see that there is NAs in all of those data

```
#looking for NAs in the sales data
sum(is.na(DataForAnalysis$Sales))
```

```
## [1] 175
```

```
sum(is.na(DataForAnalysis$Bar_Sales))
```

```
## [1] 208
```

```
sum(is.na(DataForAnalysis$Retail))
```

```
## [1] 188
```

```
sum(is.na(DataForAnalysis$TakeOutSales))
```

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

```
sum(is.na(DataForAnalysis$Sales_Restaurant))
```

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

we will change those NAs to 0, considering a $0 CAD sales for that day and variable

```
#Chaning NAs to 0 in the sales data
DataForAnalysis[is.na(DataForAnalysis$Sales),]$Sales<-0
DataForAnalysis[is.na(DataForAnalysis$Bar_Sales),]$Bar_Sales<-0
DataForAnalysis[is.na(DataForAnalysis$Retail),]$Retail<-0
DataForAnalysis[is.na(DataForAnalysis$TakeOutSales),]$TakeOutSales<-0
DataForAnalysis[is.na(DataForAnalysis$Sales_Restaurant),]$Sales_Restaurant<-0
```

Data exploration

```
#look at the structure of the data
glimpse(DataForAnalysis)
```

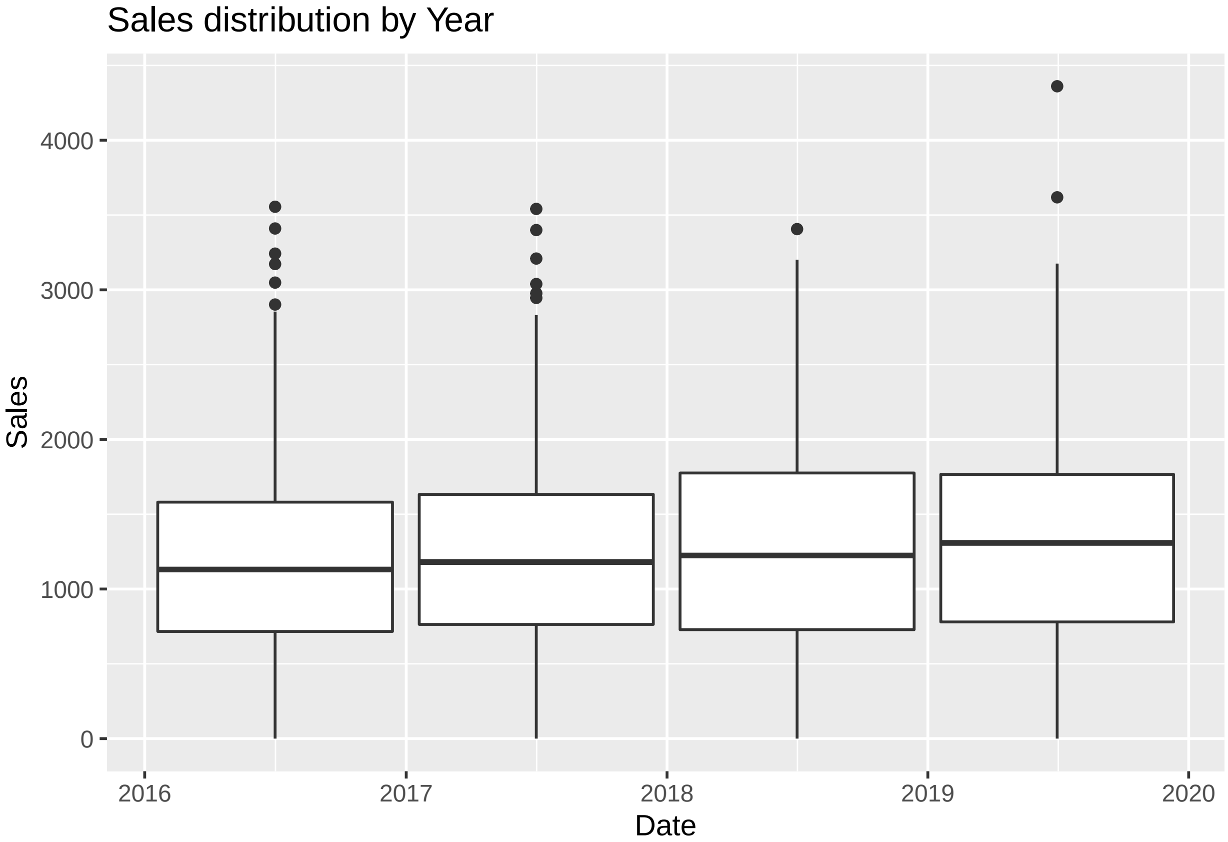
```
## Observations: 1,715
## Variables: 9
## $ Date             <date> 2015-03-29, 2015-03-30, 2015-03-31, 2015-04-01, 201...
## $ Year             <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015...
## $ Sales            <dbl> 1792.00, 0.00, 1526.30, 2250.26, 2077.57, 2357.48, 1...
## $ Retail           <dbl> 373.00, 0.00, 380.95, 363.72, 268.80, 394.23, 476.17...
## $ TakeOutSales     <dbl> 0.00, 0.00, 99.25, 402.75, 0.00, 32.50, 93.73, 0.00,...
## $ Bar_Sales        <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00...
## $ Sales_Restaurant <dbl> 1419.00, 0.00, 1046.10, 1483.79, 1808.77, 1930.75, 9...
## $ Weekday          <chr> "Sunday", "Monday", "Tuesday", "Wednesday", "Thursda...
## $ Month            <chr> "March", "March", "March", "April", "April", "April"...
```

We want to have full years to have the same thing amount of days in the year to predict 2019 weeks. So we filter for 2016 and more

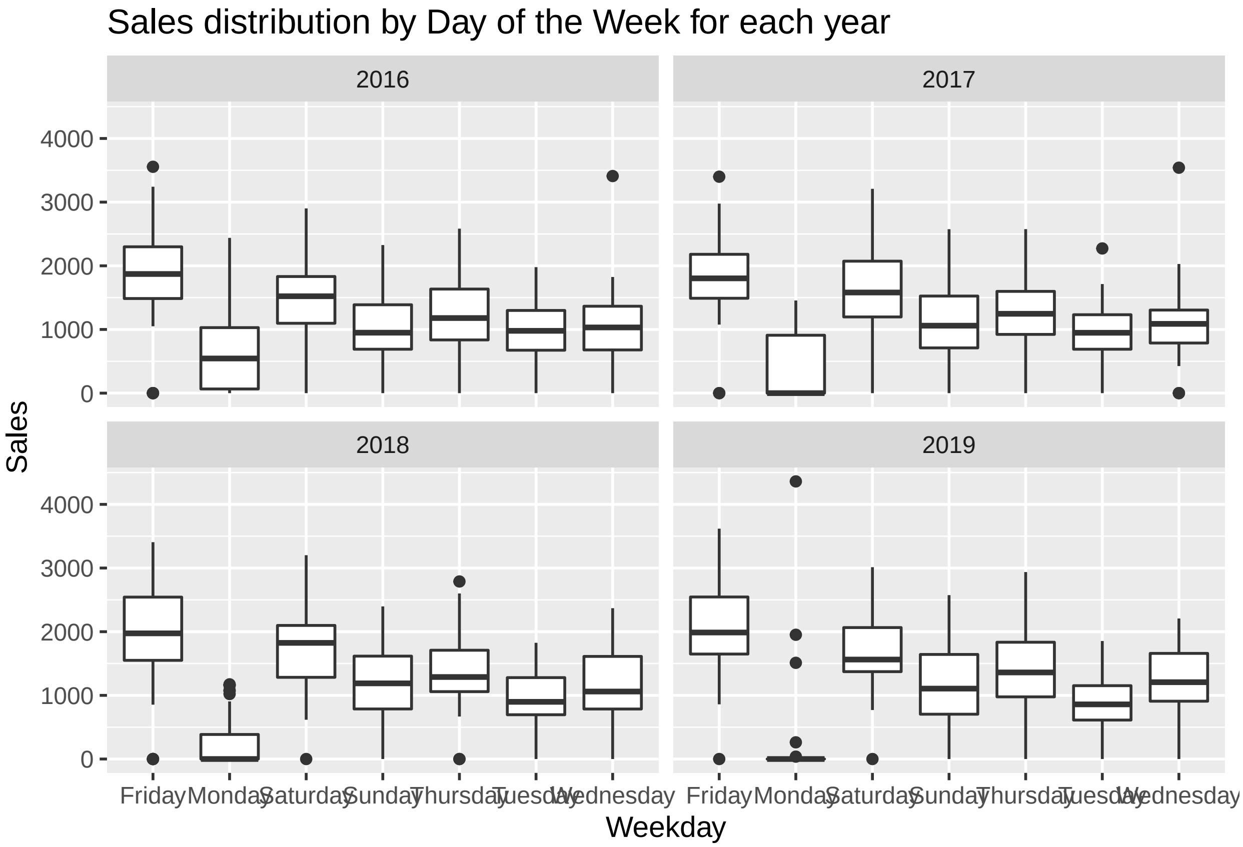
```
#Summary statistics for the variable
summary(DataForAnalysis)
```

```
##       Date                 Year          Sales            Retail
##  Min.   :2016-01-01   Min.   :2016   Min.   :   0.0   Min.   :   0.0
##  1st Qu.:2016-12-25   1st Qu.:2016   1st Qu.: 756.8   1st Qu.: 160.2
##  Median :2017-12-19   Median :2017   Median :1212.2   Median : 257.7
##  Mean   :2017-12-19   Mean   :2017   Mean   :1229.4   Mean   : 266.5
##  3rd Qu.:2018-12-13   3rd Qu.:2018   3rd Qu.:1703.3   3rd Qu.: 367.9
##  Max.   :2019-12-29   Max.   :2019   Max.   :4360.8   Max.   :1436.9
##   TakeOutSales       Bar_Sales      Sales_Restaurant    Weekday
##  Min.   :   0.00   Min.   :   0.0   Min.   :-2299.8   Length:1437
##  1st Qu.:   0.00   1st Qu.:  70.0   1st Qu.:  358.0   Class :character
##  Median :  64.00   Median : 153.8   Median :  665.0   Mode  :character
##  Mean   :  83.85   Mean   : 177.0   Mean   :  702.0
##  3rd Qu.: 120.00   3rd Qu.: 257.5   3rd Qu.:  985.2
##  Max.   :2965.72   Max.   :1045.0   Max.   : 4284.2
##     Month
##  Length:1437
##  Class :character
##  Mode  :character
##
##
##
```

Sales Variable



We see In the box plot above that 2019 was similar to 2018 in term of sales but with 2 outliers



In the box plot above we see that in 2019 they were very few Sales on Mondays and that the biggest, through the years, were Thursday, Friday and Saturday. ##Preparing Data for Modeling I will be considering this data as a time series and such I will use a library made to handle them

```
## Loading required package: timetk
```

```
## Warning: package 'timetk' was built under R version 3.6.2
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Loading required package: forecast
```

```
## Warning: package 'forecast' was built under R version 3.6.2
```

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

```
## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```

```
## Loading required package: sweep
```

```
## Warning: package 'sweep' was built under R version 3.6.2
```

We then transform the data to train and test data. We use 2016 to 2018 to predict 2019 sales data:

Modeling Data

ARIMA Model

We are going to test to 2 models to predict sales at a given week knowing the previous weeks. We will wan to try to predict the first week of 2019. We going to compare Linear Regression and ARIMA (Auto Regressive Integrated Moving Average) model (you can have a brief summary of ARIMA model here:https://machinelearningmastery.com/gentle-introduction-box-jenkins-method-time-series-forecasting/)

We first transform our train data to a time series and plot the sales over the years

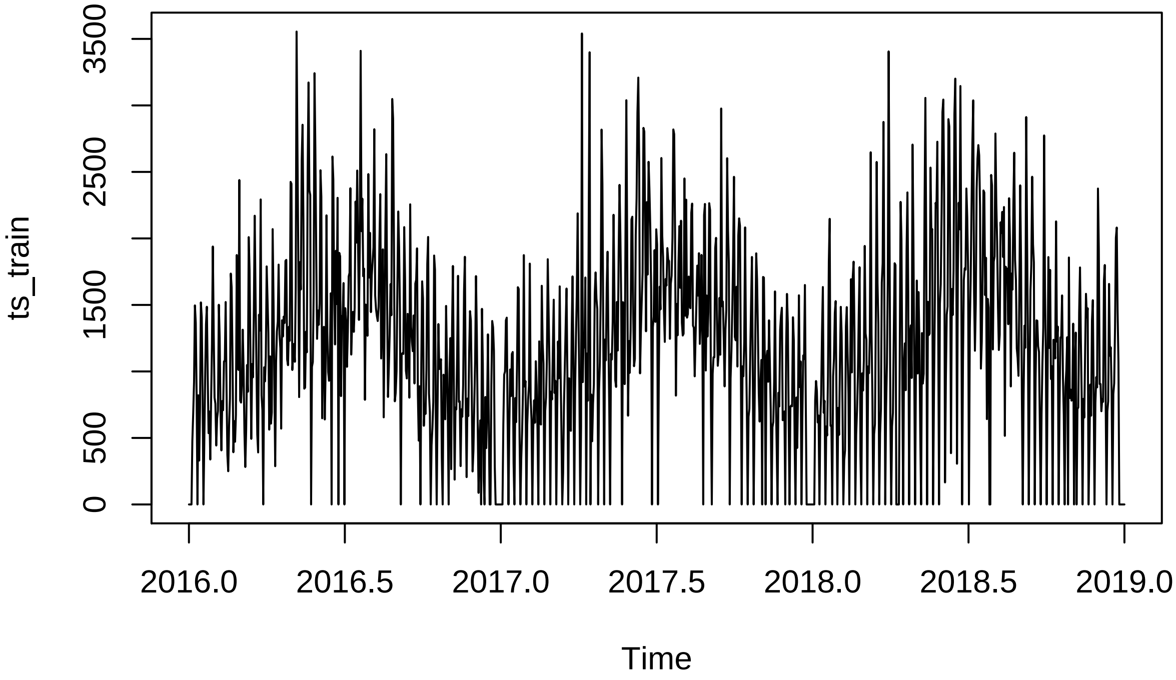
```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.6.2
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Warning in set.seed(412, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```



```
## $title
## [1] "Sales Times Series by year"
##
## attr(,"class")
## [1] "labels"
```

We see that over the 3 years the data looks sationary, so we'll use a 7 days lag with a moving average window of 45 days to we define our model as:

```r
#Build ARIMA model
set.seed(412,sample.kind = "Rounding")
```

```
## Warning in set.seed(412, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

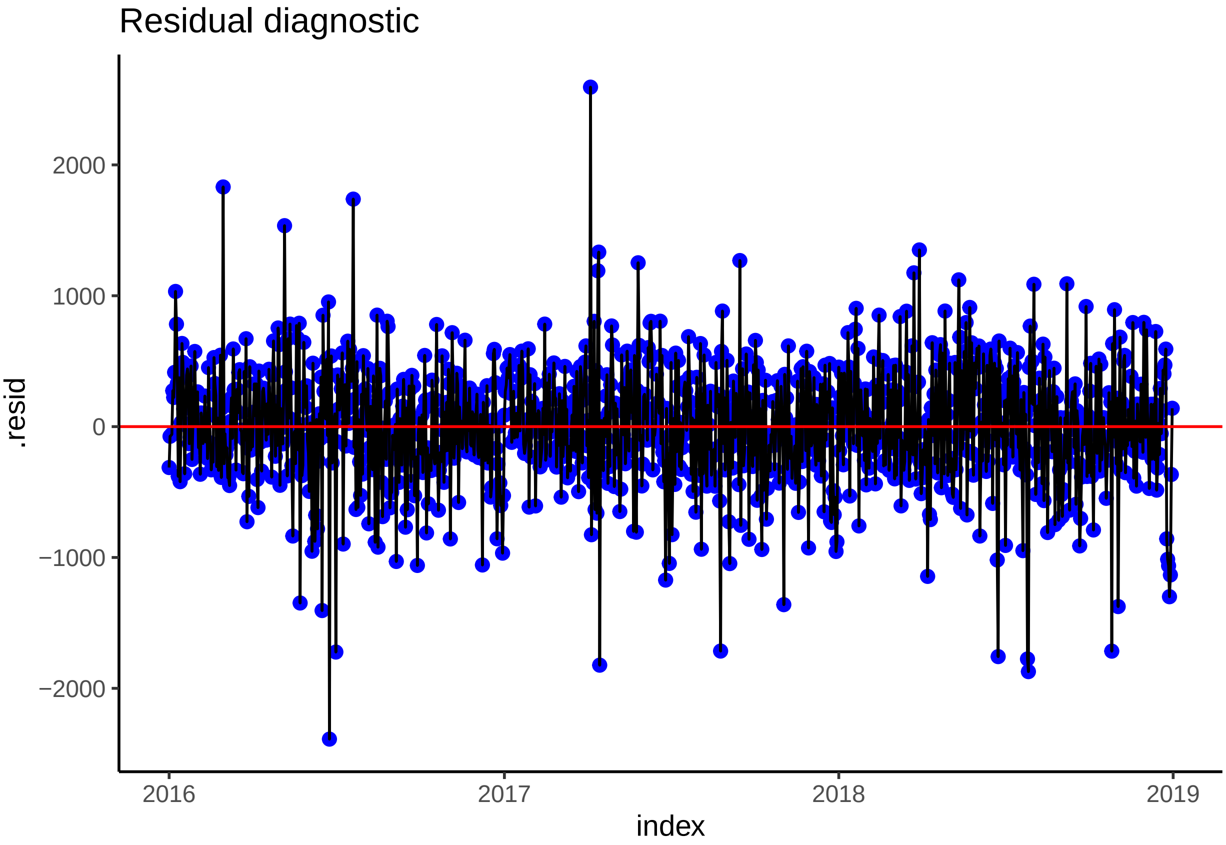
```r
fit_arima<-Arima(ts_train,c(7,0,45))

summary(fit_arima)
```

```
## Series: ts_train
## ARIMA(7,0,45) with non-zero mean
##
## Coefficients:
##           ar1     ar2      ar3     ar4      ar5     ar6     ar7     ma1
##       -0.1495  0.2724  -0.3419  0.3368  -0.2692  0.1494  0.9960  0.3926
## s.e.   0.0020  0.0023   0.0030  0.0032   0.0014  0.0020  0.0027  0.0301
##           ma2     ma3      ma4     ma5      ma6      ma7      ma8      ma9
##       -0.1160  0.3989  -0.2031  0.3088  -0.0179  -0.8268  -0.1032  -0.0133
## s.e.   0.0335  0.0331   0.0353  0.0370   0.0372   0.0375   0.0440   0.0451
##          ma10     ma11    ma12     ma13     ma14    ma15     ma16    ma17
##       -0.0044  -0.0755  0.0451  -0.0414  -0.0315  0.0572  -0.0697  0.0136
## s.e.   0.0459   0.0454  0.0449   0.0450   0.0453  0.0475   0.0477  0.0463
##          ma18    ma19    ma20    ma21     ma22    ma23    ma24     ma25    ma26
##       -0.0088  0.0350  0.0282  0.0583  -0.0049  0.0205  0.1276  -0.0248  0.0171
## s.e.   0.0443  0.0431  0.0449  0.0499   0.0458  0.0455  0.0459   0.0449  0.0476
##          ma27     ma28    ma29     ma30     ma31    ma32    ma33     ma34
##       -0.0290  -0.0755  0.0150  -0.0551  -0.0471  0.0324  0.0988  -0.0220
## s.e.   0.0489   0.0460  0.0452   0.0449   0.0455  0.0438  0.0450   0.0457
##          ma35     ma36     ma37    ma38     ma39     ma40    ma41    ma42
##       -0.0582  -0.0662  -0.0393  0.0696  -0.0118  -0.0278  0.0171  0.0584
## s.e.   0.0452   0.0479   0.0452  0.0451   0.0396   0.0402  0.0402  0.0357
##         ma43    ma44    ma45      mean
##       0.0896  0.0343  0.0018  1083.715
## s.e.  0.0341  0.0343  0.0342  1237.632
##
## sigma^2 estimated as 222582:  log likelihood=-8286.09
## AIC=16680.18   AICc=16685.89   BIC=16950.15
##
## Training set error measures:
##                     ME    RMSE      MAE MPE MAPE      MASE        ACF1
## Training set 2.325613 460.237 344.9813 NaN  Inf 0.5868102 0.002919366
```

```r
#see the acutal value with the prediction
sw_augment(fit_arima,timetk_idx = TRUE)
```

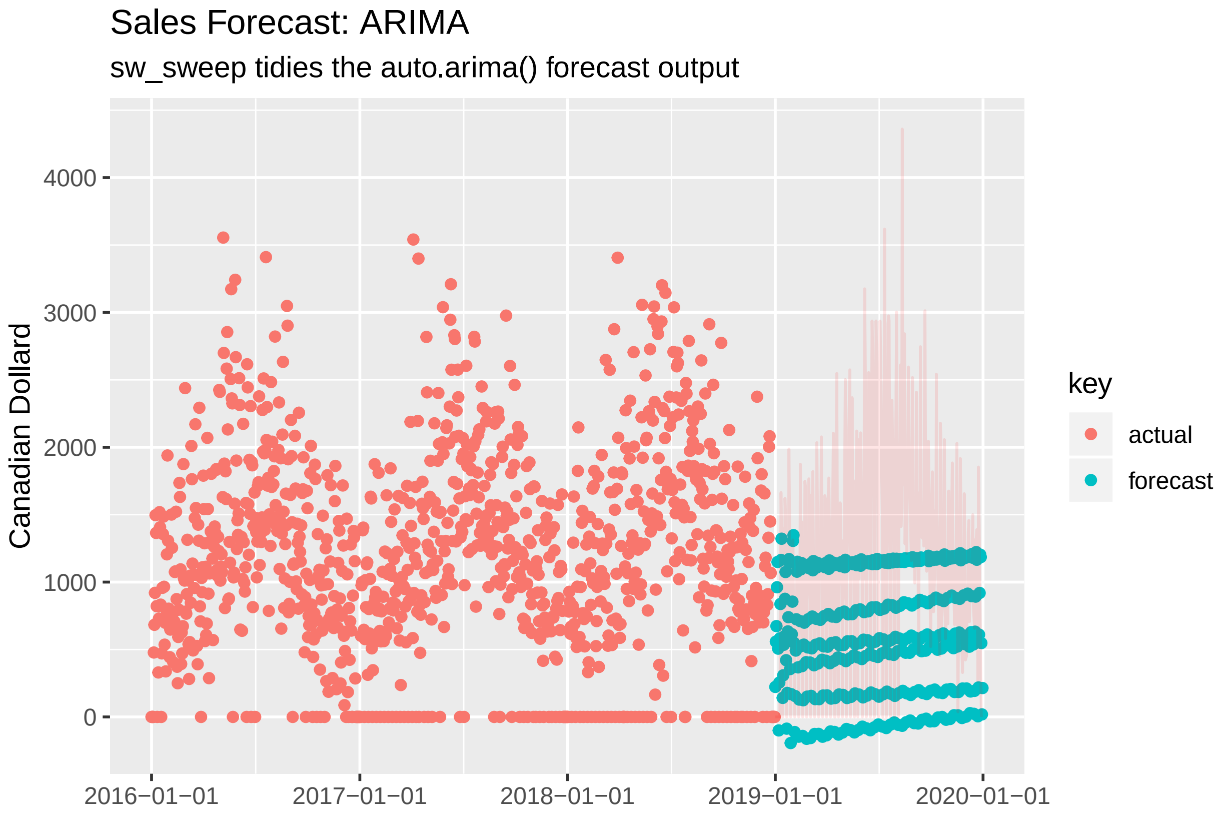
```
## # A tibble: 1,096 x 4
##    index      .actual .fitted .resid
##    <date>       <dbl>   <dbl>  <dbl>
##  1 2016-01-01       0   313.  -313.
##  2 2016-01-02       0    74.2  -74.2
##  3 2016-01-03       0    67.8  -67.8
##  4 2016-01-04       0    58.8  -58.8
##  5 2016-01-05    478.   204.   274.
##  6 2016-01-06    683.   461.   223.
##  7 2016-01-07    920.   506.   414.
##  8 2016-01-08   1496.   464.  1033.
##  9 2016-01-09   1365.   582.   783.
## 10 2016-01-10    822.   492.   330.
## # ... with 1,086 more rows
```

By looking at the residual we see that most of it is between -1000 and 1000 with quite few outliers, the model seems good considering the quantity of data:



Once we have our model we can forecast the next 365 days of our data,so 2019. We see that the ARIMA(7,0,45) model does not predict the high spikes of sales but still distinguish 7 different pattern, that correspond to the distribution of days of the week.

```
## # A tibble: 1,461 x 7
##    index      key    value lo.80 lo.95 hi.80 hi.95
##    <date>     <chr>  <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 2016-01-01 actual    0     NA    NA    NA    NA
##  2 2016-01-02 actual    0     NA    NA    NA    NA
##  3 2016-01-03 actual    0     NA    NA    NA    NA
##  4 2016-01-04 actual    0     NA    NA    NA    NA
##  5 2016-01-05 actual  478.    NA    NA    NA    NA
##  6 2016-01-06 actual  683.    NA    NA    NA    NA
##  7 2016-01-07 actual  920.    NA    NA    NA    NA
##  8 2016-01-08 actual 1496.    NA    NA    NA    NA
##  9 2016-01-09 actual 1365.    NA    NA    NA    NA
## 10 2016-01-10 actual  822.    NA    NA    NA    NA
## # ... with 1,451 more rows
```



```
## # A tibble: 365 x 5
##    date       actual   pred   error error_pct
##    <date>      <dbl>  <dbl>   <dbl>     <dbl>
##  1 2019-01-01      0  223.   -223.   -Inf
##  2 2019-01-02      0  558.   -558.   -Inf
##  3 2019-01-03      0  673.   -673.   -Inf
##  4 2019-01-04      0  962.   -962.   -Inf
##  5 2019-01-05      0 1149.  -1149.   -Inf
##  6 2019-01-06      0  507.   -507.   -Inf
##  7 2019-01-07      0  -99.9    99.9   Inf
##  8 2019-01-08      0  255.   -255.   -Inf
##  9 2019-01-09      0  583.   -583.   -Inf
## 10 2019-01-10   786.  838.    -51.1   -0.0649
## # ... with 355 more rows
```

Linear Regression

To ease the comparison of the models we are going to first transform our full data set to an augmented timeserie, which give us specification on the day, month and years of the Date column. We remove the same variable as the ARIMA model to create our train and test set data for the linear model

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 1096 obs. of  33 variables:
##  $ Date     : Date, format: "2016-01-01" "2016-01-02" ...
##  $ Year     : num  2016 2016 2016 2016 2016 ...
##  $ Sales    : num  0 0 0 0 478 ...
##  $ Weekday  : chr  "Friday" "Saturday" "Sunday" "Monday" ...
##  $ Month    : chr  "January" "January" "January" "January" ...
##  $ index.num: int  1451606400 1451692800 1451779200 1451865600 1451952000 1452038400 1452124800 14522
##  $ diff     : int  NA 86400 86400 86400 86400 86400 86400 86400 86400 86400 ...
##  $ year     : int  2016 2016 2016 2016 2016 2016 2016 2016 2016 2016 ...
##  $ year.iso : int  2015 2015 2015 2016 2016 2016 2016 2016 2016 2016 ...
##  $ half     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ quarter  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ month    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ month.xts: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month.lbl: Ord.factor w/ 12 levels "January"<"February"<..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ day      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ hour     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ minute   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ second   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hour12   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ am.pm    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ wday     : int  6 7 1 2 3 4 5 6 7 1 ...
##  $ wday.xts : int  5 6 0 1 2 3 4 5 6 0 ...
##  $ wday.lbl : Ord.factor w/ 7 levels "Sunday"<"Monday"<..: 6 7 1 2 3 4 5 6 7 1 ...
##  $ mday     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ qday     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ yday     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ mweek    : int  5 1 1 2 2 2 2 2 2 2 ...
##  $ week     : int  1 1 1 1 1 1 1 2 2 2 ...
##  $ week.iso : int  53 53 53 1 1 1 1 1 1 1 ...
##  $ week2    : int  1 1 1 1 1 1 1 0 0 0 ...
##  $ week3    : int  1 1 1 1 1 1 1 2 2 2 ...
##  $ week4    : int  1 1 1 1 1 1 1 2 2 2 ...
##  $ mday7    : int  1 1 1 1 1 1 2 2 2 2 ...
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 341 obs. of  33 variables:
##  $ Date     : Date, format: "2019-01-01" "2019-01-02" ...
##  $ Year     : num  2019 2019 2019 2019 2019 ...
##  $ Sales    : num  0 0 0 0 0 ...
##  $ Weekday  : chr  "Tuesday" "Wednesday" "Thursday" "Friday" ...
##  $ Month    : chr  "January" "January" "January" "January" ...
##  $ index.num: int  1546300800 1546387200 1546473600 1546560000 1546646400 1546732800 1546819200 15469
##  $ diff     : int  86400 86400 86400 86400 86400 86400 86400 86400 86400 86400 ...
##  $ year     : int  2019 2019 2019 2019 2019 2019 2019 2019 2019 2019 ...
##  $ year.iso : int  2019 2019 2019 2019 2019 2019 2019 2019 2019 2019 ...
##  $ half     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ quarter  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ month    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ month.xts: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month.lbl: Ord.factor w/ 12 levels "January"<"February"<..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ day      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ hour     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ minute   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ second   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hour12   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ am.pm    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ wday     : int  3 4 5 6 7 1 2 3 4 5 ...
##  $ wday.xts : int  2 3 4 5 6 0 1 2 3 4 ...
##  $ wday.lbl : Ord.factor w/ 7 levels "Sunday"<"Monday"<..: 3 4 5 6 7 1 2 3 4 5 ...
##  $ mday     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ qday     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ yday     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ mweek    : int  6 1 1 1 1 1 2 2 2 2 ...
##  $ week     : int  1 1 1 1 1 1 1 2 2 2 ...
##  $ week.iso : int  1 1 1 1 1 1 2 2 2 2 ...
##  $ week2    : int  1 1 1 1 1 1 1 0 0 0 ...
##  $ week3    : int  1 1 1 1 1 1 1 2 2 2 ...
##  $ week4    : int  1 1 1 1 1 1 1 2 2 2 ...
##  $ mday7    : int  1 1 1 1 1 1 2 2 2 2 ...
```

We define our linear model such as all the variable defined in our train set is used to predict the Sales variable:

```
set.seed(42,sample.kind = "Rounding")
```

```
## Warning in set.seed(42, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
fit_Lr<-lm(Sales~.,data=select(trainLr_set,-Date))
summary(fit_Lr)
```

```
##
## Call:
## lm(formula = Sales ~ ., data = select(trainLr_set, -Date))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2364.85  -242.80     9.68   253.26  2497.16
##
## Coefficients: (31 not defined because of singularities)
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -2.233e+07  7.660e+07  -0.291 0.770728
## Year              4.853e+04  3.350e+04   1.448 0.147795
## WeekdayMonday    -1.036e+03  2.795e+02  -3.707 0.000221 ***
## WeekdaySaturday  -3.548e+02  8.537e+01  -4.156 3.50e-05 ***
## WeekdaySunday    -9.528e+02  1.464e+02  -6.510 1.15e-10 ***
## WeekdayThursday  -4.997e+02  8.718e+01  -5.731 1.30e-08 ***
## WeekdayTuesday   -6.070e+02  2.114e+02  -2.872 0.004161 **
## WeekdayWednesday -5.791e+02  1.464e+02  -3.954 8.18e-05 ***
## MonthAugust       1.459e+04  1.024e+04   1.424 0.154670
## MonthDecember     2.774e+04  2.046e+04   1.356 0.175544
## MonthFebruary    -7.126e+03  6.318e+03  -1.128 0.259620
## MonthJanuary     -1.084e+04  7.496e+03  -1.446 0.148475
## MonthJuly         1.104e+04  7.553e+03   1.461 0.144237
## MonthJune         7.605e+03  7.786e+03   0.977 0.328863
## MonthMarch       -3.668e+03  7.238e+03  -0.507 0.612461
## MonthMay          3.838e+03  3.829e+03   1.002 0.316472
## MonthNovember     2.448e+04  1.764e+04   1.388 0.165424
## MonthOctober      2.100e+04  1.519e+04   1.383 0.167083
## MonthSeptember    1.787e+04  1.357e+04   1.317 0.188141
## index.num        -3.550e-04  1.234e-03  -0.288 0.773618
## diff                     NA         NA      NA       NA
## year                     NA         NA      NA       NA
## year.iso         -3.719e+04  2.497e+04  -1.490 0.136643
## half                     NA         NA      NA       NA
## quarter                   NA         NA      NA       NA
## month                     NA         NA      NA       NA
## month.xts                 NA         NA      NA       NA
## month.lbl.L               NA         NA      NA       NA
## month.lbl.Q               NA         NA      NA       NA
## month.lbl.C               NA         NA      NA       NA
## month.lbl^4               NA         NA      NA       NA
## month.lbl^5               NA         NA      NA       NA
## month.lbl^6               NA         NA      NA       NA
## month.lbl^7               NA         NA      NA       NA
## month.lbl^8               NA         NA      NA       NA
## month.lbl^9               NA         NA      NA       NA
## month.lbl^10              NA         NA      NA       NA
## month.lbl^11              NA         NA      NA       NA
## day                1.193e+02  1.279e+02   0.933 0.350980
## hour                      NA         NA      NA       NA
## minute                    NA         NA      NA       NA
## second                    NA         NA      NA       NA
## hour12                    NA         NA      NA       NA
## am.pm                     NA         NA      NA       NA
## wday                      NA         NA      NA       NA
## wday.xts                  NA         NA      NA       NA
## wday.lbl.L                NA         NA      NA       NA
## wday.lbl.Q                NA         NA      NA       NA
## wday.lbl.C                NA         NA      NA       NA
## wday.lbl^4                NA         NA      NA       NA
## wday.lbl^5                NA         NA      NA       NA
## wday.lbl^6                NA         NA      NA       NA
## mday                      NA         NA      NA       NA
## qday              -9.500e-01  1.085e+02  -0.009 0.993013
## yday               3.196e+01  6.285e+01   0.508 0.611245
## mweek             -3.149e+00  1.895e+01  -0.166 0.868059
## week              -9.656e+01  5.611e+01  -1.721 0.085575 .
## week.iso          -7.223e+02  4.756e+02  -1.519 0.129119
## week2              1.089e+02  3.168e+01   3.437 0.000611 ***
## week3              1.990e+01  1.776e+01   1.121 0.262743
## week4             -3.182e+01  1.418e+01  -2.245 0.025004 *
## mday7             -1.338e+01  5.157e+01  -0.259 0.795304
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 467 on 1064 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.6066, Adjusted R-squared:  0.5955
## F-statistic: 54.68 on 30 and 1064 DF,  p-value: < 2.2e-16
```

We see that all the variable created by the augmented ts was used to define the model to predict our sales. Now that the model is trained, we can predict the test Data with our model

```
predict_lr<-predict(fit_Lr,newdata=select(testLr_set,-Date))
```

```
## Warning in predict.lm(fit_Lr, newdata = select(testLr_set, -Date)): prediction
## from a rank-deficient fit may be misleading
```

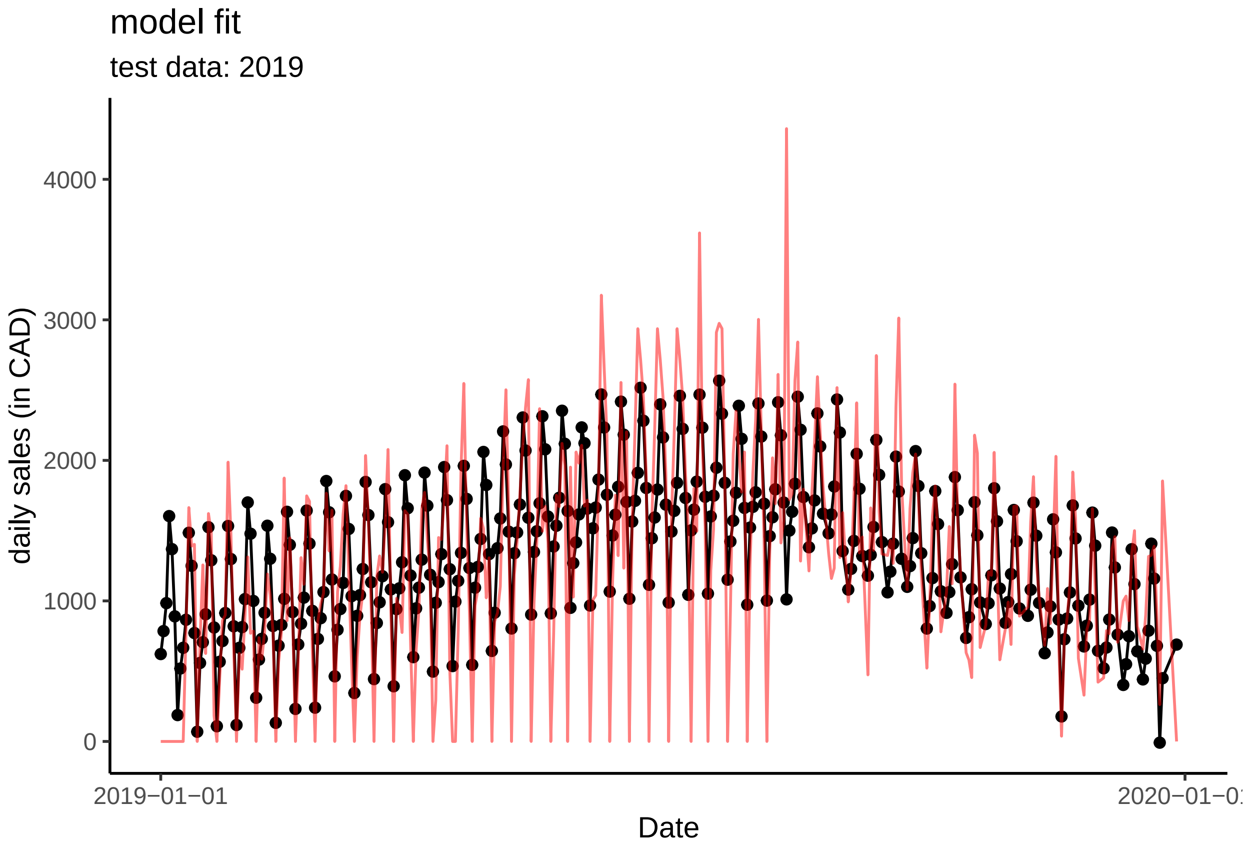
```
error_lr<-testLr_set%>%select(Date,actual=Sales)%>%
  mutate(pred=predict_lr,
         error=actual-predict_lr,
         error_pct=error/actual)
error_lr
```

```
## # A tibble: 341 x 5
##    Date       actual  pred   error error_pct
##    <date>      <dbl> <dbl>   <dbl>     <dbl>
##  1 2019-01-01     0   622.  -622.   -Inf
##  2 2019-01-02     0   785.  -785.   -Inf
##  3 2019-01-03     0   984.  -984.   -Inf
##  4 2019-01-04     0  1604. -1604.   -Inf
##  5 2019-01-05     0  1368. -1368.   -Inf
##  6 2019-01-06     0   890.  -890.   -Inf
##  7 2019-01-07     0   188.  -188.   -Inf
##  8 2019-01-08     0   519.  -519.   -Inf
##  9 2019-01-09     0   666.  -666.   -Inf
## 10 2019-01-10   786.  866.   -79.1    -0.101
## # ... with 331 more rows
```

```
dataToPlot<-testLr_set%>%add_column(pred=predict(fit_Lr,testLr_set)%>%tibble::enframe(name = NULL) %>%
```

```
## Warning in predict.lm(fit_Lr, testLr_set): prediction from a rank-deficient fit
## may be misleading
```

```
dataToPlot %>%
 ggplot(aes(x = Date, y = pred)) +
 geom_line() +
geom_point()+
geom_line(data = testLr_set,aes(x=Date,y=Sales),color="red",alpha=0.5)+
 scale_x_date(date_breaks = "1 year", date_labels = "%F") +
 scale_color_manual(
  values = c(
   "weekly_sales" = "blue",
   "lm_pred" = "#fdc7d7"
  )
 ) +
 theme_classic() +
 labs(title = "model fit",
  subtitle = "test data: 2019",
  x = "Date",
  y = "daily sales (in CAD)"
  )
```



On the plot above we see in red The actual value of sales and in black the prediction made by the linear regression model.

Discussion

We see that the linear regression has a better accuracy than the ARIMA model for predicting the sales for 2019. But The ARIMA model is able to differentiate the trends between the weekdays. One Interesting way we could improve the ARIMA would be to try to find the best ARIMA model for each day in a week for all the years in the data set and combine them to have a better approximation of the sales for any given day and would allows us to compare the sales to the same day last year. We see that both models failed to predict the high spikes of sales corresponding to the middle of the year, April through October, when the sidewalk sitting area is open, which give to the restaurant more sitting places, so potentially more sales. Furthermore it might be interesint to include weather data to analyse the effect of rainy days (or snowy days) on the sales of the Restaurant.