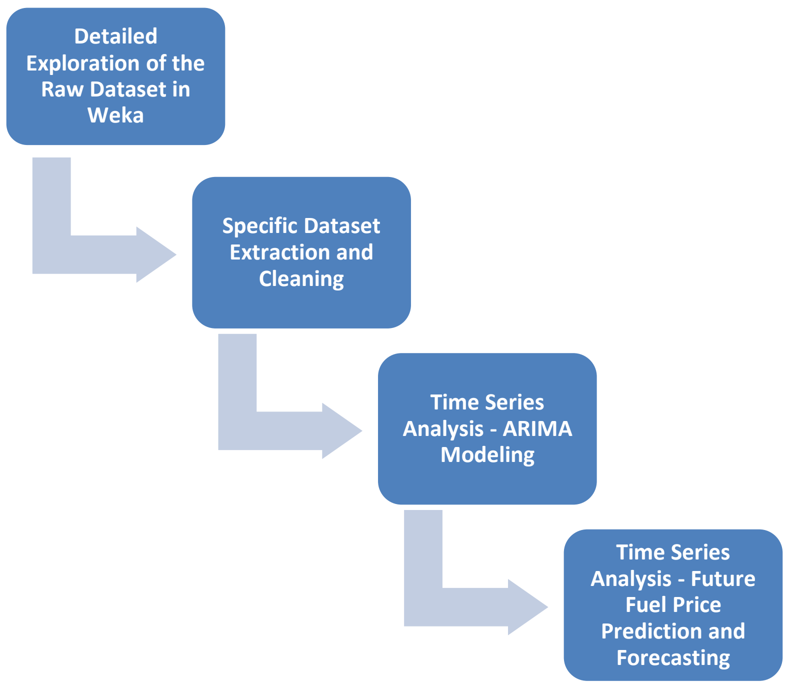
# Fuel Price Analysis and Prediction by Time Series

## Approach

The approach to be taken is shown in the graph below and will be described in the following subsections.




### Step 1: Detailed Exploration of the Raw Dataset in Weka

**1. Clean and format the head (attribute names).**

*ds <- read.csv(file = "D:\\RyersonU\\CKME136 - Data Analytics - Capstone Project\\Possible Datasets\\Fuels price survey information\\fueltypesall.csv", header = T, stringsAsFactors = F, na.strings = c("","NA"))*

*ds_head_clean <- as.data.frame (select(ds, -c(Type.de.carburant)))* *names(ds_head_clean) <- c("DATE", "OTTAWA", "TORONTO_WEST", "TORONTO_EAST", "WINDSOR", "LONDON", "PETERBOROUGH", "ST_CATHARINE", "SUDBURY", "SAULT_SAINT_MARIE","THUNDER_BAY", "NORTH_BAY", "TIMMINS",  "KENORA", "PARRY_SOUND", "ONTARIO_AVG", "SOUTHERN_AVG", "NORTHERN_AVG", "FUEL_TYPE")*

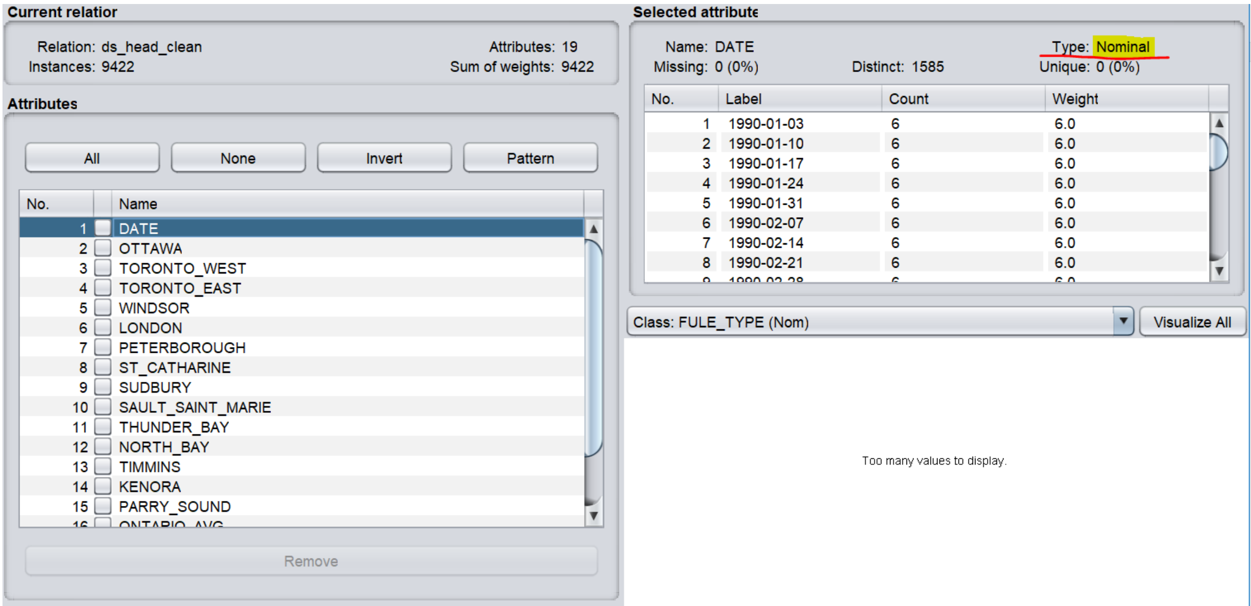
*str(ds_head_clean)*

```
> str(ds_head_clean)
'data.frame':	9422 obs. of  19 variables:
 $ DATE             : chr  "1990-01-03" "1990-01-10" "1990-01-17" "1990-01-24" ...
 $ OTTAWA           : num  55.9 55.9 55.9 55.9 55.9 55.8 55.8 55.8 55.9 56 ...
 $ TORONTO_WEST     : num  49.1 47.7 53.2 53.2 51.9 50.7 49.3 48.2 54.1 53.8 ...
 $ TORONTO_EAST     : num  48.7 46.8 53.2 53.5 52.6 50.7 48.4 47.1 54.4 53.8 ...
 $ WINDSOR          : num  45.2 49.7 49.6 49 48.6 48.5 48.5 48.6 48.5 48.3 ...
 $ LONDON           : num  50.1 47.6 53.7 52.1 49.1 47.8 54.7 53.6 51.7 50.8 ...
 $ PETERBOROUGH     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ST_CATHARINE     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ SUDBURY          : num  56.4 56.4 55.8 55.7 55.6 55.6 55.6 55.6 55.6 55.6 ...
 $ SAULT_SAINT_MARIE: num  54.8 54.9 54.9 54.9 54.8 54.8 54.8 54.8 54.8 54.8 ...
 $ THUNDER_BAY      : num  56.6 56.8 56.8 56.8 56.8 56.8 56.9 57 57 57 ...
 $ NORTH_BAY        : num  55.1 55 54.4 54.3 54.2 54.2 54.1 54 54.1 54 ...
 $ TIMMINS          : num  58.1 58.2 58.2 58.2 58.1 58.1 58.1 58.1 58.1 58.1 ...
 $ KENORA           : num  0 0 0 0 0 0 0 0 0 0 ...
 $ PARRY_SOUND      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ONTARIO_AVG      : num  50.3 49.2 53.6 53.5 52.5 51.4 50.7 49.9 54.1 53.7 ...
 $ SOUTHERN_AVG     : num  49.5 48.3 53.3 53.2 52.1 50.8 50.1 49.1 53.8 53.4 ...
 $ NORTHERN_AVG     : num  56.2 56.2 56 56 55.9 55.9 55.9 55.9 55.9 55.9 ...
 $ FULE_TYPE        : chr  "Regular Unleaded Gasoline" "Regular Unleaded Gasoline" "R
Unleaded Gasoline" ...
```

Now this dataset contains 9422 observations and 19 variables, and the attribute name is English only. Then I export this head cleaned dataset as a csv file, which will be used for the next exploration in Weka.

*write.csv (ds_head_clean, "D:\\RyersonU\\CKME136 - Data Analytics - Capstone Project\\Possible Datasets\\Fuels price survey information\\ds_head_clean.csv", row.names = FALSE)*

**2. Load the head cleaned table in Weka to look at the attribute type for each attribute.**

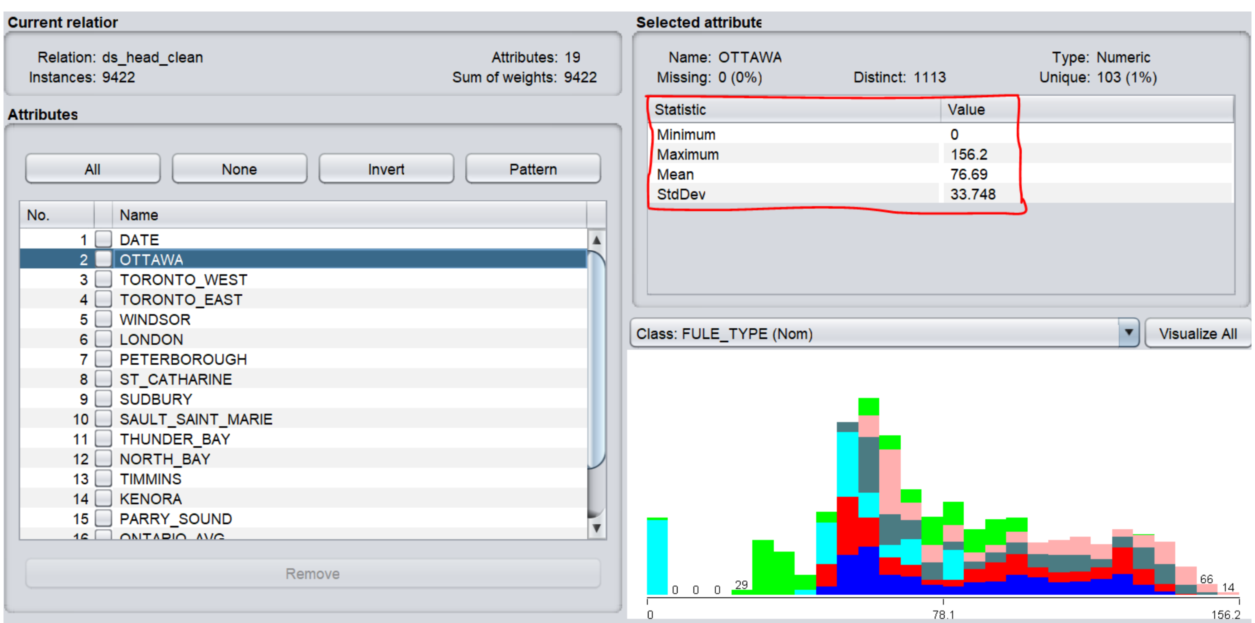



Nominal (Qualitative) Type includes the attributes: DATE and FUEL-TYPE;

Numeric (Quantitative) Type includes the attributes: other 17 attributes.

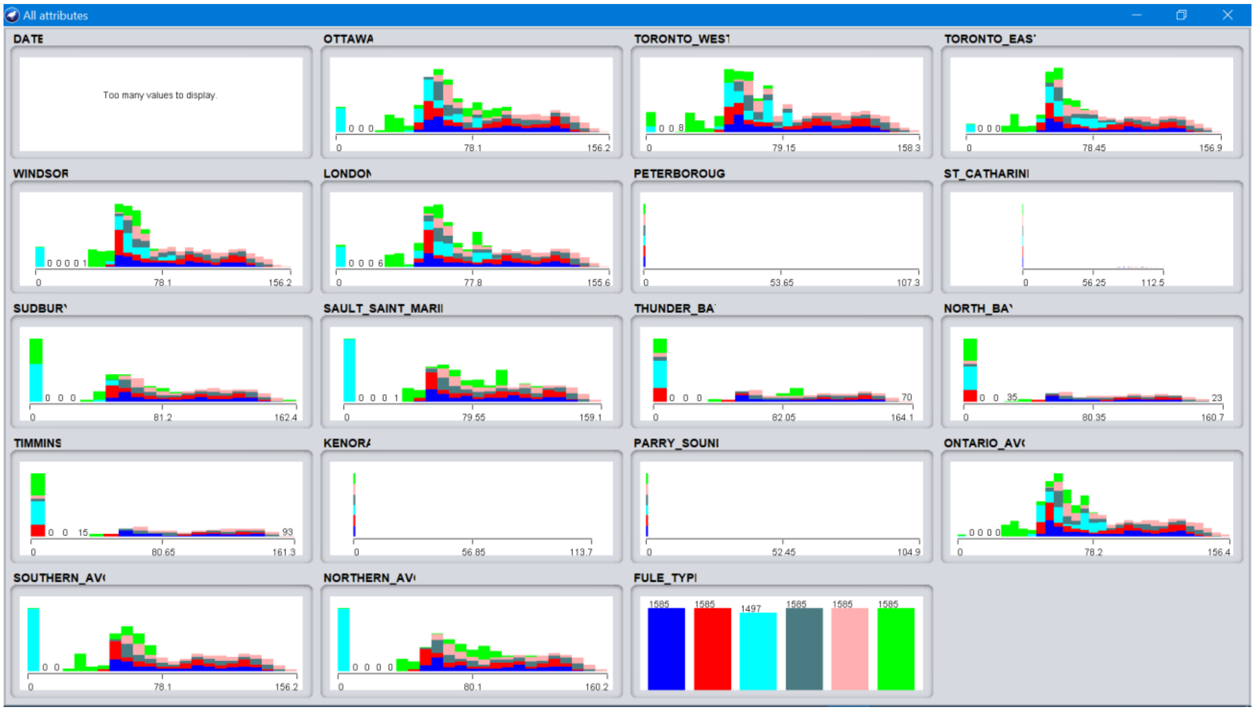
## 3. Find max, min, mean and standard deviation of attributes.

By selecting each attribute name, we can get the statistics for each numeric attribute.

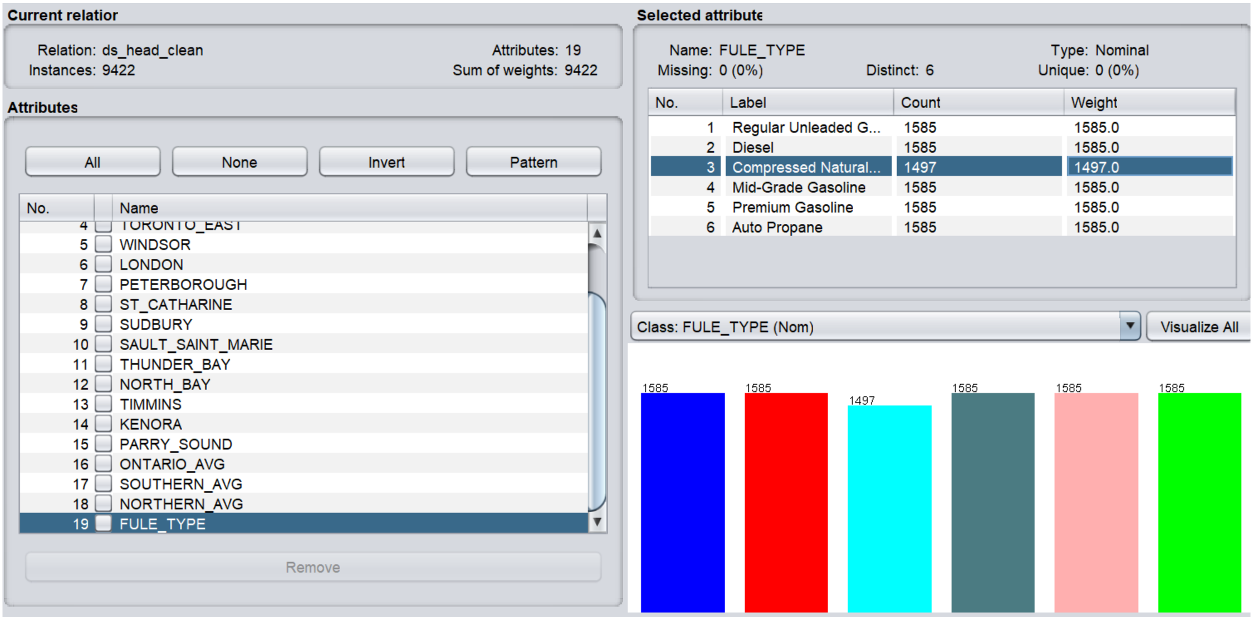


The exploration indicates that all those numeric attributes do not have missing values. The contained value ‘0’ is treated as a data value instead of missing data.

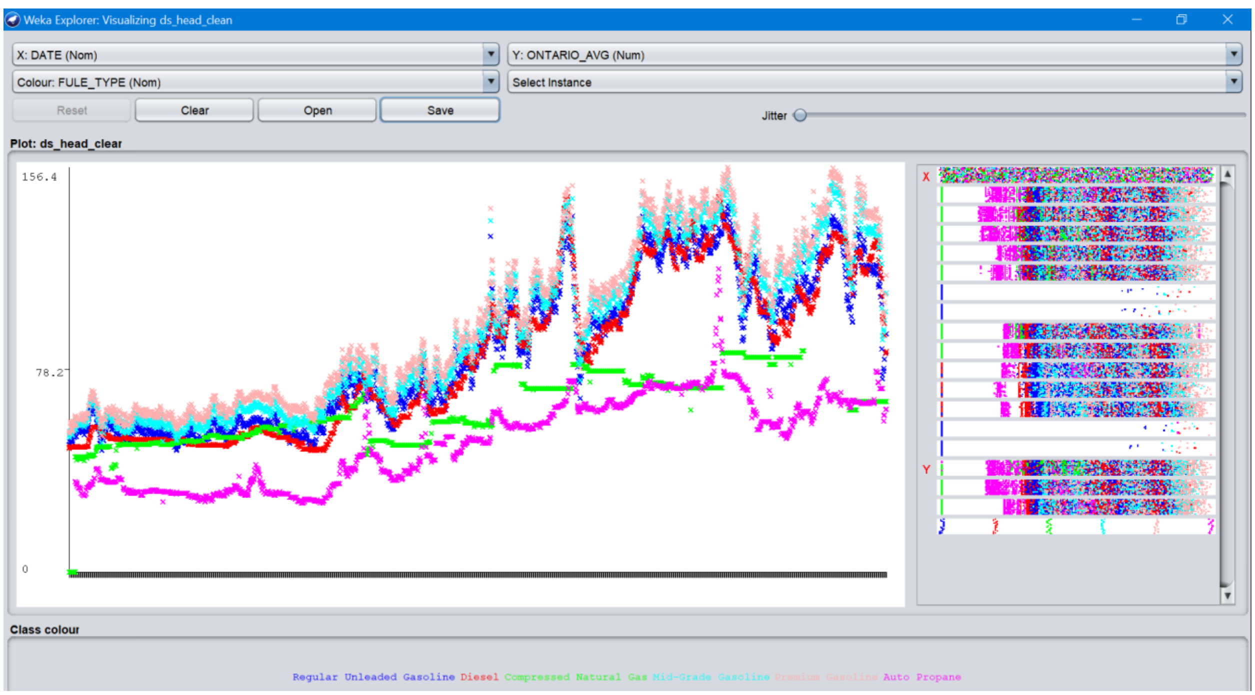
**4. Observe the data distribution for these attributes by visualizing all in Weka.**




By observing this visualization, we can again see some cities (Markets) do not have efficient data collected. And by specifically looking at Fuel Type, we can see the type of ‘Compressed Natural Gas’ (blue color bar) has missing data records. This data missing will be further investigated in R in the step 2 to identify which time periods is the data missing.




## 5. Observe interesting trends in Weka Visualization.




By plotting time (Date) as X axis and Ontario Average fuel price as Y axis, we can observe a general increase trend during the period of 1991 – 2020 for all fuel types. And we can also observe some seasonal fluctuations (similar repeated pattern in a certain season) through the changes of the fuel price.

## Step 2: Specific Dataset Extraction and Cleaning

**1. Load the head-cleaned dataset (from step 1) in R and aggregate into monthly interval.**

*RAW <- read.csv (file = "D:\\RyersonU\\CKME136 - Data Analytics - Capstone Project\\Possible Datasets\\Fuels price survey information\\ds_head_clean.csv", header = T, stringsAsFactors = F, na.strings = c("","NA"))*

*RAW$YEAR_MONTH <- substr(RAW$DATE, 1, 7)*

*Data_Aggregated <- aggregate (x = list (RAW$OTTAWA, RAW$TORONTO_WEST, RAW$TORONTO_EAST, RAW$WINDSOR, RAW$LONDON, RAW$PETERBOROUGH, RAW$ST_CATHARINE, RAW$SUDBURY, RAW$NORTH_BAY, RAW$SAULT_SAINT_MARIE, RAW$THUNDER_BAY, RAW$TIMMINS, RAW$KENORA, RAW$PARRY_SOUND), by = list (RAW$YEAR_MONTH, RAW$FUEL_TYPE), FUN = mean)*

*names (Data_Aggregated) <- c ("YEAR_MONTH", "FUEL_TYPE", "OTTAWA", "TORONTO_WEST", "TORONTO_EAST", "WINDSOR","LONDON", "PETERBOROUGH", "ST_CATHARINE", "SUDBURY", "NORTH_BAY", "SAULT_SAINT_MARIE", "THUNDER_BAY", "TIMMINS", "KENORA", "PARRY_SOUND")*

**2. Further transfer the dataset into a new dataset with only four attributes: "YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY"**

*OTTAWA = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, OTTAWA))*
*OTTAWA$CITY = "OTTAWA"*
*names (OTTAWA) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*TORONTO_WEST = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, TORONTO_WEST))*
*TORONTO_WEST$CITY = "TORONTO_WEST"*

*names (TORONTO_WEST) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*TORONTO_EAST = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, TORONTO_EAST))*
*TORONTO_EAST$CITY = "TORONTO_EAST"*
*names (TORONTO_EAST) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*WINDSOR = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, WINDSOR))*
*WINDSOR$CITY = "WINDSOR"*
*names (WINDSOR) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*LONDON = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, LONDON))*
*LONDON$CITY = "LONDON"*
*names (LONDON) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*PETERBOROUGH = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, PETERBOROUGH))*
*PETERBOROUGH$CITY = "PETERBOROUGH"*
*names (PETERBOROUGH) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*ST_CATHARINE = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, ST_CATHARINE))*
*ST_CATHARINE$CITY = "ST_CATHARINE"*
*names (ST_CATHARINE) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*SUDBURY = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, SUDBURY))*
*SUDBURY$CITY = "SUDBURY"*
*names (SUDBURY) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*NORTH_BAY = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, NORTH_BAY))*
*NORTH_BAY$CITY = "NORTH_BAY"*
*names (NORTH_BAY) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*SAULT_SAINT_MARIE = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, SAULT_SAINT_MARIE))*
*SAULT_SAINT_MARIE$CITY = "SAULT_SAINT_MARIE"*

*names (SAULT_SAINT_MARIE) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*THUNDER_BAY = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, THUNDER_BAY))*
*THUNDER_BAY$CITY = "THUNDER_BAY"*
*names (THUNDER_BAY) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*TIMMINS = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, TIMMINS))*
*TIMMINS$CITY = "TIMMINS"*
*names (TIMMINS) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*KENORA = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, KENORA))*
*KENORA$CITY = "KENORA"*
*names (KENORA) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*PARRY_SOUND = select (Data_Aggregated, c (YEAR_MONTH, FUEL_TYPE, PARRY_SOUND))*
*PARRY_SOUND$CITY = "PARRY_SOUND"*
*names (PARRY_SOUND) = c ("YEAR_MONTH", "FUEL_TYPE", "PRICE", "CITY")*

*City_Combined <- list (OTTAWA, TORONTO_WEST, TORONTO_EAST, WINDSOR, LONDON, PETERBOROUGH, ST_CATHARINE, SUDBURY, NORTH_BAY, SAULT_SAINT_MARIE, THUNDER_BAY, TIMMINS, KENORA, PARRY_SOUND)*
*City_Combined <- do.call ("rbind", City_Combined)*
*City_Combined <- as.data.frame (City_Combined)*

## 3. Have a look at the new dataset: “City_Combined”

*str (City_Combined)*
*summary (City_Combined)*

```
> str(City_Combined)
'data.frame':	30380 obs. of  4 variables:
 $ YEAR_MONTH: chr  "1990-01" "1990-02" "1990-03" "1990-04" ...
 $ FULE_TYPE : chr  "Auto Propane" "Auto Propane" "Auto Propane" "Auto Propane" ...
 $ PRICE     : num  0 0 0 30.8 30.6 ...
 $ CITY      : chr  "OTTAWA" "OTTAWA" "OTTAWA" "OTTAWA" ...
> summary (City_Combined)
  YEAR_MONTH          FULE_TYPE              PRICE            CITY          
 Length:30380       Length:30380       Min.   :  0.00   Length:30380      
 Class :character   Class :character   1st Qu.:  0.00   Class :character  
 Mode  :character   Mode  :character   Median : 51.90   Mode  :character  
                                       Mean   : 49.61                     
                                       3rd Qu.: 86.90                     
                                       Max.   :162.97                     
```

*First_10_Row <- head (City_Combined, n=10)*

| | YEAR_MONTH | FULE_TYPE | PRICE | CITY |
|---|---|---|---|---|
| 1 | 1990-01 | Auto Propane | 0.000 | OTTAWA |
| 2 | 1990-02 | Auto Propane | 0.000 | OTTAWA |
| 3 | 1990-03 | Auto Propane | 0.000 | OTTAWA |
| 4 | 1990-04 | Auto Propane | 30.800 | OTTAWA |
| 5 | 1990-05 | Auto Propane | 30.560 | OTTAWA |
| 6 | 1990-06 | Auto Propane | 30.350 | OTTAWA |
| 7 | 1990-07 | Auto Propane | 28.625 | OTTAWA |
| 8 | 1990-08 | Auto Propane | 31.240 | OTTAWA |
| 9 | 1990-09 | Auto Propane | 33.400 | OTTAWA |
| 10 | 1990-10 | Auto Propane | 37.140 | OTTAWA |

*Last_10_Row <- tail (City_Combined, n=10)*

| | YEAR_MONTH | FULE_TYPE | PRICE | CITY |
|---|---|---|---|---|
| 30371 | 2019-08 | Regular Unleaded Gasoline | 0.000 | PARRY_SOUND |
| 30372 | 2019-09 | Regular Unleaded Gasoline | 0.000 | PARRY_SOUND |
| 30373 | 2019-10 | Regular Unleaded Gasoline | 0.000 | PARRY_SOUND |
| 30374 | 2019-11 | Regular Unleaded Gasoline | 0.000 | PARRY_SOUND |
| 30375 | 2019-12 | Regular Unleaded Gasoline | 0.000 | PARRY_SOUND |
| 30376 | 2020-01 | Regular Unleaded Gasoline | 0.000 | PARRY_SOUND |
| 30377 | 2020-02 | Regular Unleaded Gasoline | 0.000 | PARRY_SOUND |
| 30378 | 2020-03 | Regular Unleaded Gasoline | 0.000 | PARRY_SOUND |
| 30379 | 2020-04 | Regular Unleaded Gasoline | 74.375 | PARRY_SOUND |
| 30380 | 2020-05 | Regular Unleaded Gasoline | 81.550 | PARRY_SOUND |

## 4. Identify and Remove Missing Values

Based on the general exploration in Step 1, I found there are missing values under the fuel type of Compressed Natural Gas. Now I am investigating those missing values in R.

1) **To confirm the missing value is for Compressed Natural Gas**

*table (City_Combined$FUEL_TYPE)*

```
> table (City_Combined$FULE_TYPE)

            Auto Propane    Compressed Natural Gas                    Diesel
                    5110                      4830                      5110
      Mid-Grade Gasoline          Premium Gasoline Regular Unleaded Gasoline
                    5110                      5110                      5110
```

2) **Introduce a column of YEAR to help checking which year(s) contains the missing value**

*City_Combined$YEAR <- substr(City_Combined$YEAR_MONTH, 1, 4)*

*table (City_Combined$YEAR)*

```
> table(City_Combined$YEAR)

1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005
1008 1008 1008 1008 1008 1008 1008 1008 1008 1008 1008 1008 1008 1008 1008 1008
2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020
1008 1008 1008 1008 1008 1008 1008 1008 1008 1008 1008  896  840 1008  420
```

We can see the missing value is in 2017 and 2018. We can ignore 2020 because the original data collected is only up to May 2020.

3) **Final confirms for missing value**

*CNG <- subset (City_Combined, FUEL_TYPE == "Compressed Natural Gas")*

*table (CNG$YEAR)*

```
> CNG <- subset(City_Combined, FULE_TYPE == "Compressed Natural Gas")
> table(CNG$YEAR)

1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005
 168  168  168  168  168  168  168  168  168  168  168  168  168  168  168  168
2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2019 2020
 168  168  168  168  168  168  168  168  168  168  168   56  168   70
```

Therefore, the missing value is caused by that the Compressed Natural Gas (CNG) only has partial data in year of 2017 and does not have any data in year of 2018. So, I decided to remove the CNG fuel type from my dataset.

*City_Combined_NoCNG <- subset (City_Combined, FUEL_TYPE != "Compressed Natural Gas")*

*table (City_Combined_NoCNG$FUEL_TYPE)*

```
> table(City_Combined_NoCNG$FULE_TYPE)

           Auto Propane                   Diesel        Mid-Grade Gasoline
                   5110                     5110                      5110
       Premium Gasoline Regular Unleaded Gasoline
                   5110                     5110
```

## 5. Generate Subsets Based on different types of Fuel and Identify 0 values

### 1) Generate 5 subsets and Plotting them

*AP <- subset (City_Combined_NoCNG, FUEL_TYPE == "Auto Propane")*

*DSL <- subset (City_Combined_NoCNG, FUEL_TYPE == "Diesel")*

*MGG <- subset (City_Combined_NoCNG, FUEL_TYPE == "Mid-Grade Gasoline")*

*RUG <- subset (City_Combined_NoCNG, FUEL_TYPE == "Regular Unleaded Gasoline")*

*PRG <- subset (City_Combined_NoCNG, FUEL_TYPE == "Premium Gasoline")*

*Plot_AP <- ggplot(AP, aes(x = YEAR, y = PRICE, group = CITY)) +*

*geom_line(aes(color=CITY)) + geom_point(aes(color=CITY))*

*Plot_DSL <- ggplot(DSL, aes(x = YEAR, y = PRICE, group = CITY)) +*

*geom_line(aes(color=CITY)) + geom_point(aes(color=CITY))*

*Plot_MGG <- ggplot(MGG, aes(x = YEAR, y = PRICE, group = CITY)) +*

*geom_line(aes(color=CITY)) + geom_point(aes(color=CITY))*

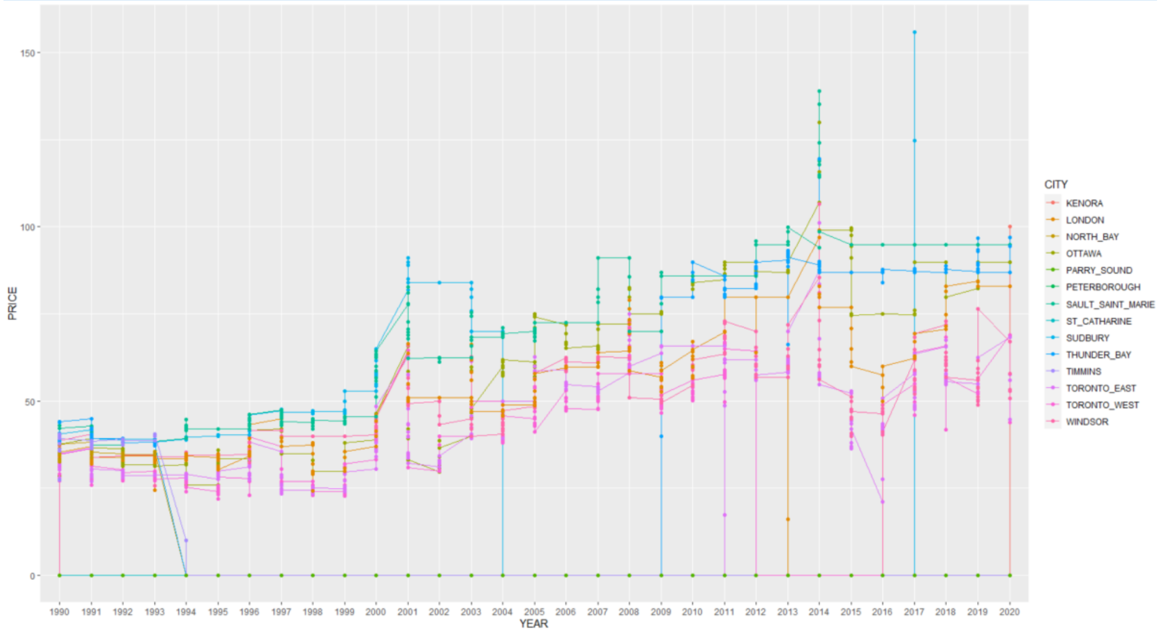
*Plot_RUG <- ggplot(RUG, aes(x = YEAR, y = PRICE, group = CITY)) +*

*geom_line(aes(color=CITY)) + geom_point(aes(color=CITY))*

*Plot_PRG <- ggplot(PRG, aes(x = YEAR, y = PRICE, group = CITY)) +*

*geom_line(aes(color=CITY)) + geom_point(aes(color=CITY))*

The below is the plot of Auto Propane and we can see there are some cities have 0 value.




**2) Identify 0 values for each Fuel Type**

# AP Type (Auto Propane) #

*table (AP$PRICE==0, AP$CITY)*

```
> table(AP$PRICE==0, AP$CITY)

        KENORA  LONDON  NORTH_BAY  OTTAWA  PARRY_SOUND  PETERBOROUGH
  FALSE      2     359         45     362            0             0
  TRUE     363       6        320       3          365           365

        SAULT_SAINT_MARIE  ST_CATHARINE  SUDBURY  THUNDER_BAY  TIMMINS  TORONTO_EAST
  FALSE               362             0      172          181       46           361
  TRUE                  3           365      193          184      319             4

        TORONTO_WEST  WINDSOR
  FALSE          313      361
  TRUE            52        4
```

"TRUE" indicates the count of 0 values contained in each city. We can see that all cities have 0 values for Auto Propane price and some of them contains only 0 rather than other valid values.

# DSL Type (Diesel) #

*table (DSL$PRICE==0, DSL$CITY)*

```
> table(DSL$PRICE==0, DSL$CITY)

        KENORA LONDON NORTH_BAY OTTAWA PARRY_SOUND PETERBOROUGH
  FALSE      2    365       200    365           2            2
  TRUE     363      0       165      0         363          363

        SAULT_SAINT_MARIE ST_CATHARINE SUDBURY THUNDER_BAY TIMMINS TORONTO_EAST
  FALSE               365            2     365         200     200          365
  TRUE                  0          363       0         165     165            0

        TORONTO_WEST WINDSOR
  FALSE          365     365
  TRUE             0       0
```

These cities do not contain 0 values for Diesel price: LONDON, OTTAWA, SAULT_SAINT_MARIE, SUDBURY, TORONTO_EAST, TORONTO_WEST, WINDSOR

# MGG Type (Mid-Grade Gasoline) #

*table (MGG$PRICE==0, MGG$CITY)*

```
> table(MGG$PRICE==0, MGG$CITY)

        KENORA LONDON NORTH_BAY OTTAWA PARRY_SOUND PETERBOROUGH
  FALSE      2    365       321    365           2            2
  TRUE     363      0        44      0         363          363

        SAULT_SAINT_MARIE ST_CATHARINE SUDBURY THUNDER_BAY TIMMINS TORONTO_EAST
  FALSE               365            2     365         321     322          365
  TRUE                  0          363       0          44      43            0

        TORONTO_WEST WINDSOR
  FALSE          365     365
  TRUE             0       0
```

These cities do not contain 0 values for Mid-Grade Gasoline price: LONDON, OTTAWA, SAULT_SAINT_MARIE, SUDBURY, TORONTO_EAST, TORONTO_WEST, WINDSOR.

# RUG Type (Regular Unleaded Gasoline) #

*table (RUG$PRICE==0, RUG$CITY)*

```
> table(RUG$PRICE==0, RUG$CITY)

        KENORA LONDON NORTH_BAY OTTAWA PARRY_SOUND PETERBOROUGH
  FALSE      2    365       364    365           2            2
  TRUE     363      0         1      0         363          363

        SAULT_SAINT_MARIE ST_CATHARINE SUDBURY THUNDER_BAY TIMMINS TORONTO_EAST
  FALSE               365            2     365         364     364          365
  TRUE                  0          363       0           1       1            0

        TORONTO_WEST WINDSOR
  FALSE          365     365
  TRUE             0       0
```

These cities do not contain 0 values for Regular Unleaded Gasoline price: LONDON, OTTAWA, SAULT_SAINT_MARIE, SUDBURY, TORONTO_EAST, TORONTO_WEST, WINDSOR.

# PRG Type (Premium Gasoline) #

*table (PRG$PRICE==0, PRG$CITY)*

```
> table(PRG$PRICE==0, PRG$CITY)

        KENORA LONDON NORTH_BAY OTTAWA PARRY_SOUND PETERBOROUGH
  FALSE      2    365       321    365           2            2
  TRUE     363      0        44      0         363          363

        SAULT_SAINT_MARIE ST_CATHARINE SUDBURY THUNDER_BAY TIMMINS TORONTO_EAST
  FALSE               365            2     365         321     322          365
  TRUE                  0          363       0          44      43            0

        TORONTO_WEST WINDSOR
  FALSE          365     365
  TRUE             0       0
```

These cities do not contain 0 values for Premium Gasoline price: LONDON, OTTAWA, SAULT_SAINT_MARIE, SUDBURY, TORONTO_EAST, TORONTO_WEST, WINDSOR

**3) Cleaning 0 values for each Fuel Type**

Because the 0 value is treated as invalid value, I will clean those 0 values by removing those cities who contain 0 values and keep only those cities who does not contain 0 values for future my analysis. Therefore, I will drop the Auto Propane (AP) fuel type, and keep only Diesel (DSL), Mid-Grade Gasoline (MGG), Regular Unleaded Gasoline (RUG), and Premium Gasoline (PRG) fuel types. I will also keep only the cities of LONDON,

OTTAWA, SAULT_SAINT_MARIE, SUDBURY, TORONTO_EAST, TORONTO_WEST, and WINDSOR for my future analysis.

*DSL_Clean <- subset (DSL, grepl ('LONDON|OTTAWA|SAULT_SAINT_MARIE|SUDBURY|TORONTO|WINDSOR', DSL$CITY))*
*MGG_Clean <- subset (MGG, grepl ('LONDON|OTTAWA|SAULT_SAINT_MARIE|SUDBURY|TORONTO|WINDSOR', MGG$CITY))*
*RUG_Clean <- subset (RUG, grepl ('LONDON|OTTAWA|SAULT_SAINT_MARIE|SUDBURY|TORONTO|WINDSOR', RUG$CITY))*
*PRG_Clean <- subset (PRG, grepl ('LONDON|OTTAWA|SAULT_SAINT_MARIE|SUDBURY|TORONTO|WINDSOR', PRG$CITY))*

**6. Get Final Subsets Ready for Time Series Analysis**

To get final subsets ready, I will drop the previously introduced column of YEAR which is used for missing value investigation. So, in each subset, there will be four attributes: YEAR_MONTH, FUEL_TYPE, PRICE, CITY.

*DSL_Ready <- select (DSL_Clean, -c(YEAR))*
*MGG_Ready <- select (MGG_Clean, -c(YEAR))*
*RUG_Ready <- select (RUG_Clean, -c(YEAR))*
*PRG_Ready <- select (PRG_Clean, -c(YEAR))*

| | YEAR_MONTH | FUEL_TYPE | PRICE | CITY |
|---|---|---|---|---|

Therefore, the time series analysis in my project will be an analysis of the prices for four different fuel types (Diesel, Mid-Grade Gasoline, Regular Unleaded Gasoline, Premium Gasoline) based on seven different Ontario marketplaces (Ottawa, Toronto West, Toronto East, Windsor, London, Sudbury, Sault Saint Marie).

## Step 3: Time Series Analysis - ARIMA Modeling

ARIMA stands for Autoregressive Integrated Moving Average. Auto Regressive (AR) refers to the lags of the differenced series, Moving Average (MA) refers to the lags of errors, and (I) refers the number of differences used to make the time series stationary.

The Assumptions of ARIMA model include:

- Data should be stationary: the series does not depend on the time when it is captured. White noise series and series with cyclic behavior can be considered as stationary series.
- Data should be univariate: ARIMA works on a single variable and Autoregression is regression with the past values.

My ARIMA modeling will include three sub-steps: Time Series Exportation, Model Fitting, and Measures Evaluation.

R package will be used are as following:

*library (fUnitRoots)*

*library (lmtest)*

*library (forecast)*

*library (FitAR)*

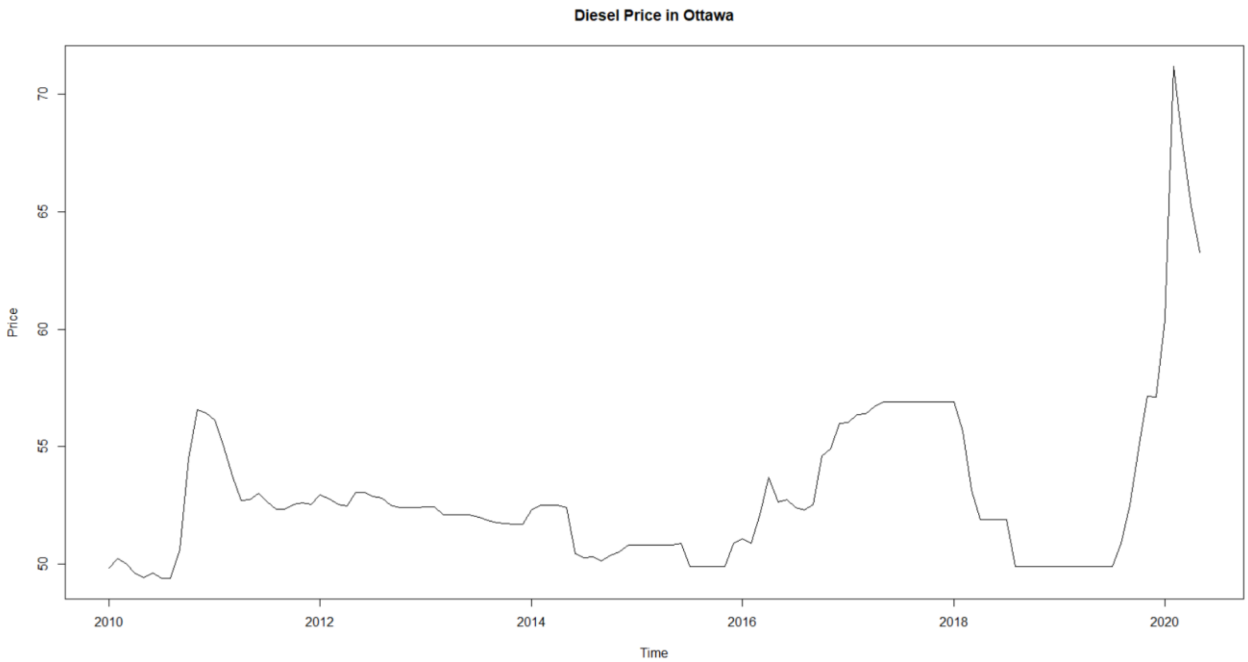
### 1. Time Series Exportation

#### 1) Data Converting to Time Series

To convert data, I did a further subset of the DSL_Ready dataset to focus Diesel price only in Ottawa marketplace first. In ts() function, I select the "PRICE" attribute as the univariate, and I shorten the time period to Jan 2010 – May 2020 in order to achieve a better view for the following graphing. Frequency is set to 12 because of monthly interval data.

*DSL_Ottawa <- subset (DSL_Ready, CITY == "OTTAWA")* *ts_DSL_Ottawa = ts (DSL_Ottawa$PRICE, start = c (2010,1), end = c (2020,5), frequency=12)*

*plot (ts_DSL_Ottawa, main="Diesel Price in Ottawa", ylab="Price")*

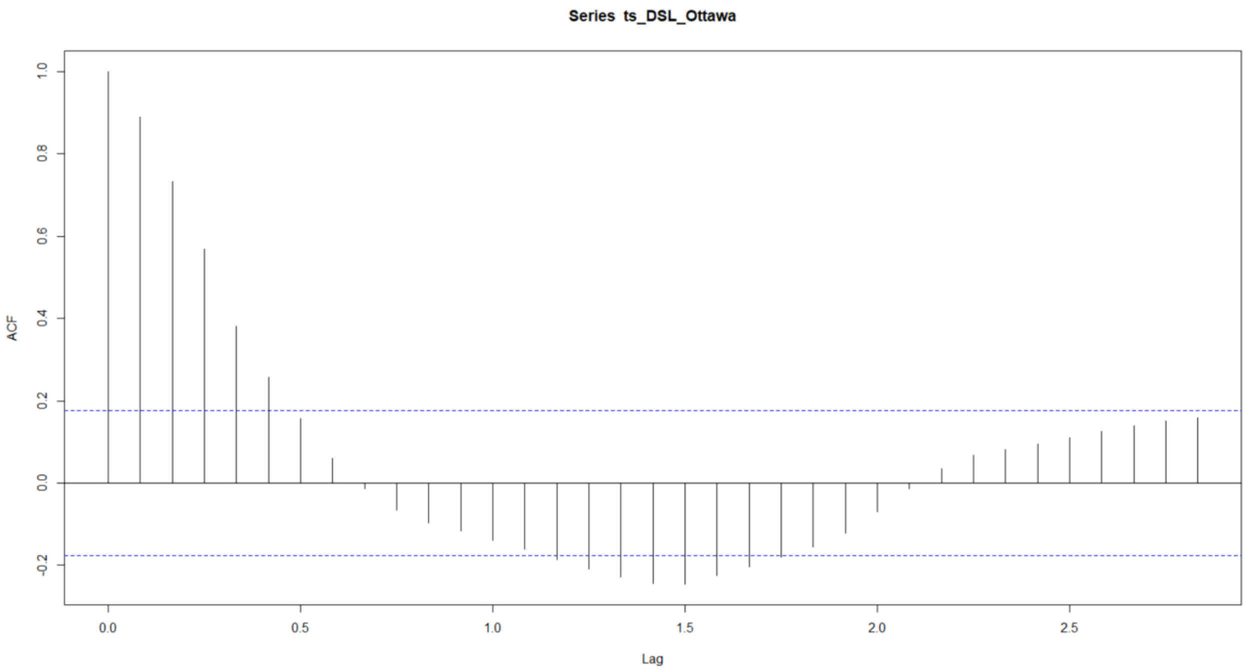



I can see from the graph that the data follows an overall upward trend. Next I will conduct the Autocorrelation analysis to examine serial dependence. And then I will analyze the components of the time series. After that, I will find the non-stationarity and seasonality in the data.

**2) Autocorrelation Analysis – ACF Plotting**

To calculate autocorrelation, I applied the acf() function and achieved the graph below.

*acf (ts_DSL_Ottawa,lag.max=34)*

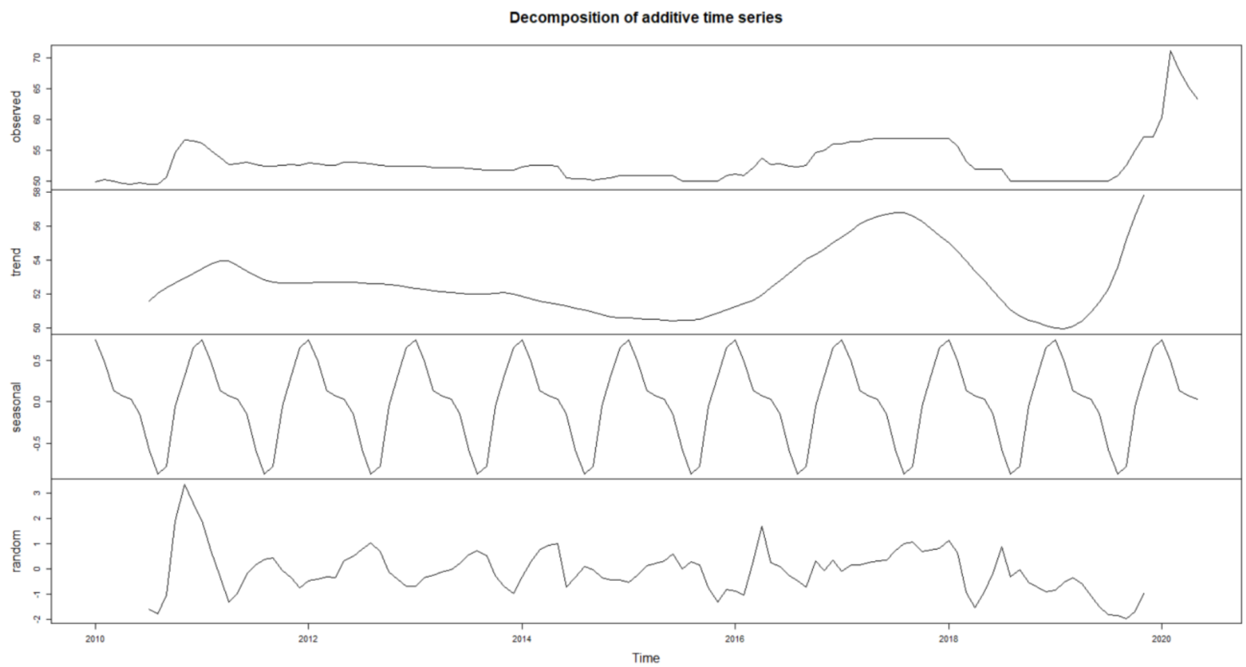



The autocorrelation at lag 0 is included by default which always takes the value 1 because it represents the correlation between the data and themselves. From this above graph, I can see that the autocorrelation (ACF) is decreasing as the Lag increases. This means that no linear association between observations separated by larger lags.

**3) Components Analysis**

*TSC_ts_DSL_Ottawa <- decompose(ts_DSL_Ottawa)*

*plot (TSC_ts_DSL_Ottawa)*

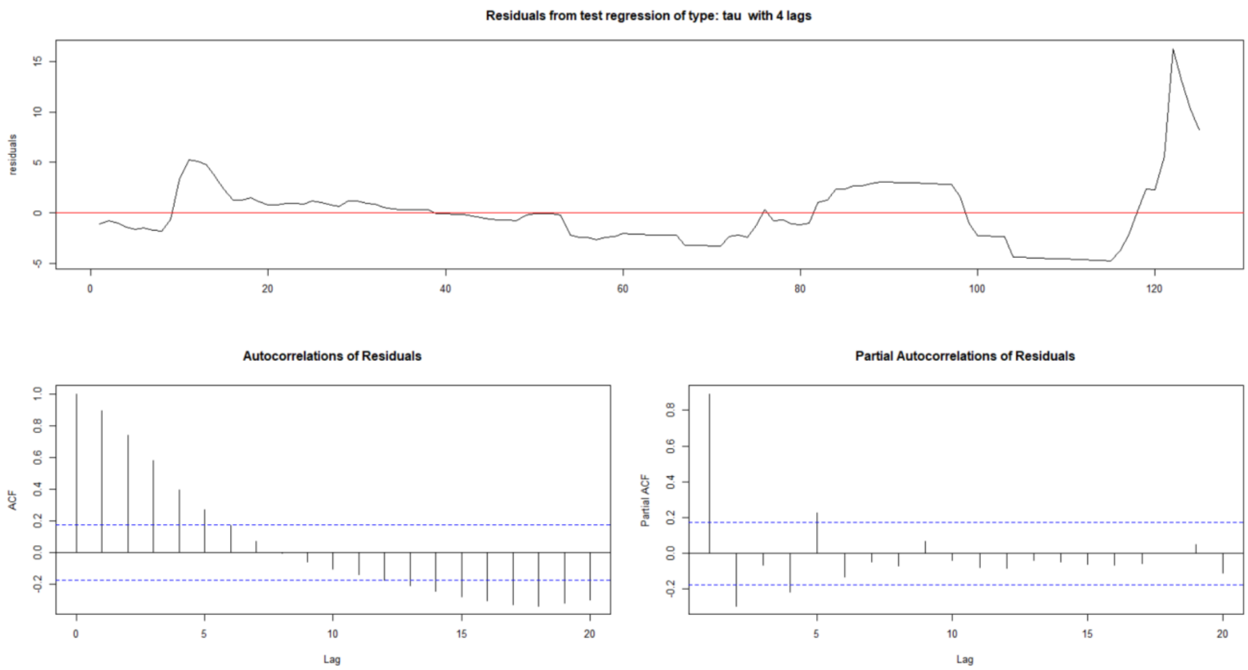



From above ACP plot, I get four components: Observed (the actual data plot), Trend (the overall upward or downward movement), Seasonal (yearly/monthly pattern of the data), Random (unexplainable part of the data).

**4) Determine Stationarity of data and Remove Non-Stationarity**

The process of determing stationarity includes: Difference the data (compute the differences between consecutive observations), Log or Square Root the series data (stabilize non-constant variance), Fit curve to the data and Model residuals from the fit (if the data contains a trend), Unit Root Test (find first difference or regression used on the trending data to make it stationary. In Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test, small p-values suggest that differencing is required.

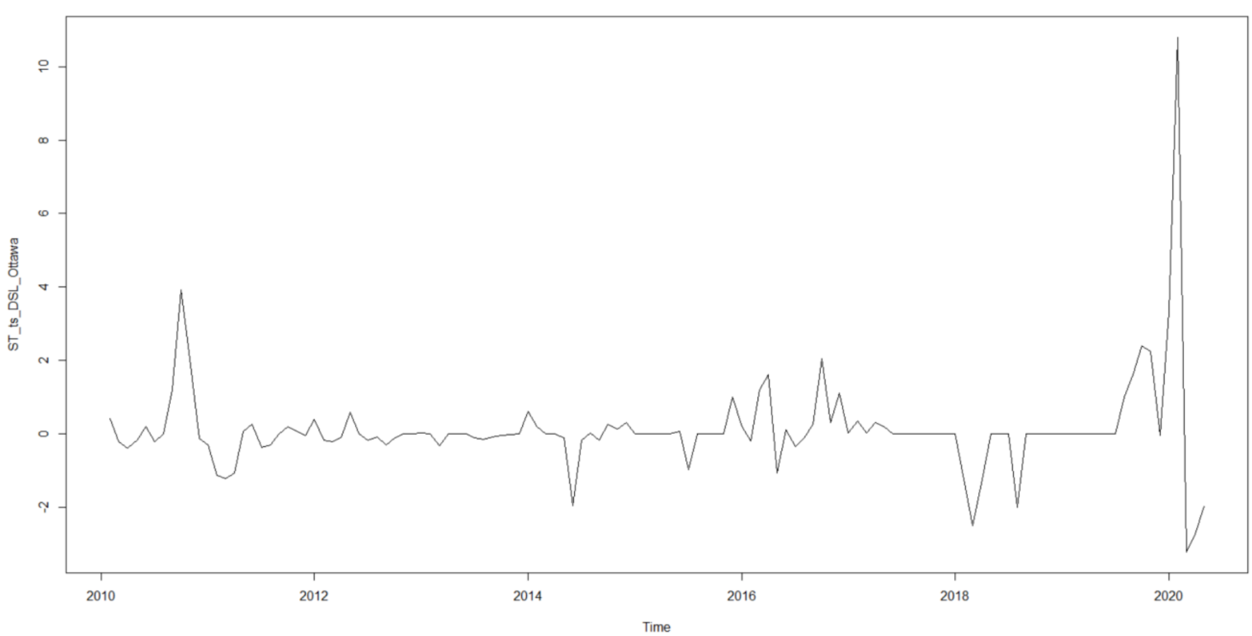
*urkpssTest (ts_DSL_Ottawa, type = c ("tau"), lags = c ("short"), use.lag = NULL, doplot = TRUE)*




To remove non-stationarity, I applied the below code and achieved the graph after non-stationarity removement.

*ST_ts_DSL_Ottawa <- diff (ts_DSL_Ottawa, differences=1)*

*plot (ST_ts_DSL_Ottawa)*

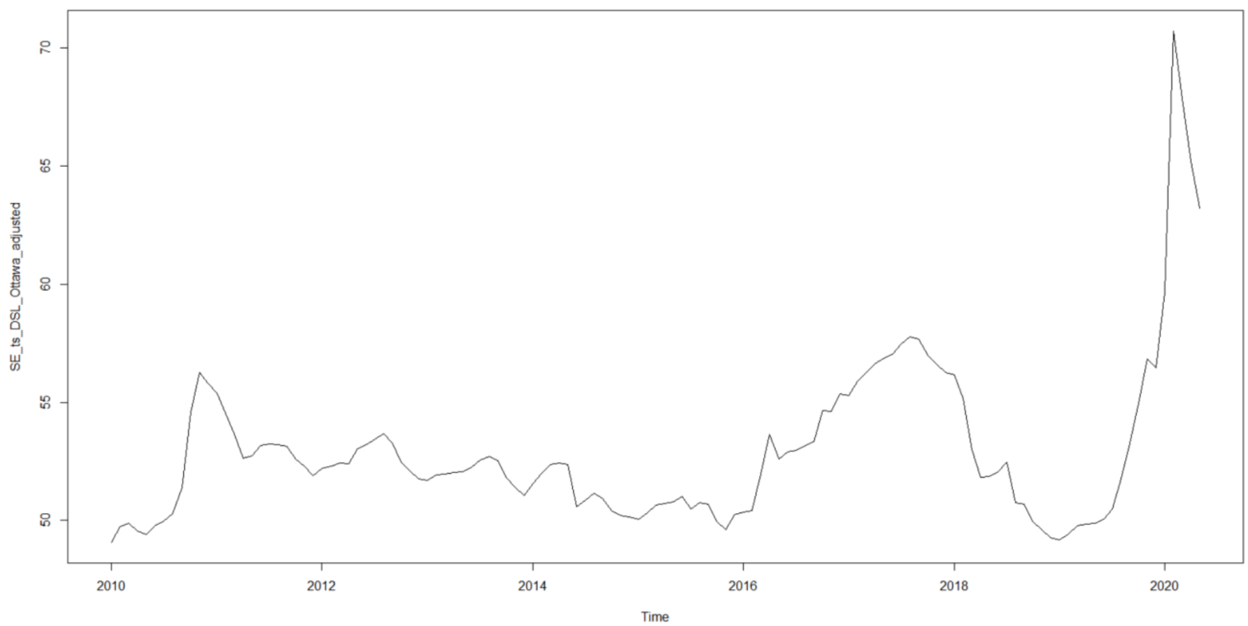


**5) Determine Seasonality of data and Remove Seasonality**

To determine and remove seasonality from the data, I first subtract the seasonal component from the original series. Then I difference it to make it stationary. The below graph is the seasonality after the subtraction.

*SE_ts_DSL_Ottawa_adjusted <- ts_DSL_Ottawa - TSC_ts_DSL_Ottawa$seasonal*

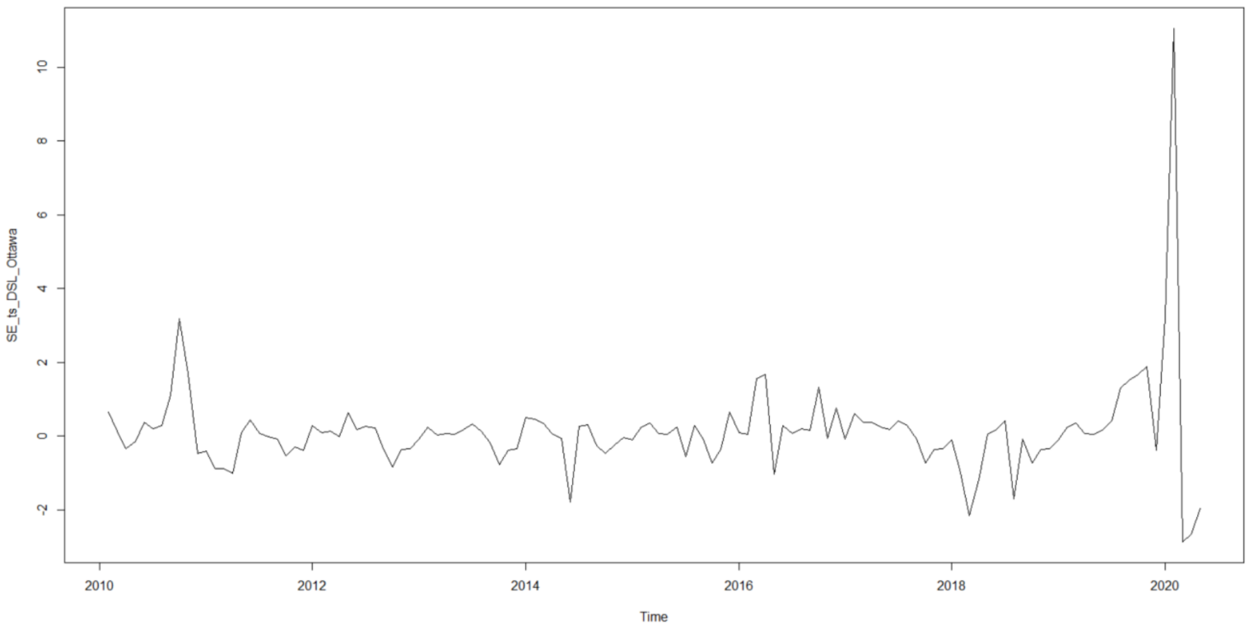
*plot (SE_ts_DSL_Ottawa_adjusted)*




And this below is the graph after removing seasonality and making stationary.

*SE_ts_DSL_Ottawa <- diff (SE_ts_DSL_Ottawa_adjusted, differences=1)*

*plot (SE_ts_DSL_Ottawa)*




## 2. Model Fitting

### 1) Examine p and q values

We will need the three variables: p, d, q to determine the order of the model to be fitted to the data. The three variables are non-negative integers that represent the order of the autoregressive, integrated, and moving average parts of the model respectively.

I applied pacf () function and acf () function to examine appropriate values of p and q. The pacf () is autocorrelation function at lag k. The function describes the correlation between those data points which are exactly k steps apart. It helps to identify the number of AR coefficients (p-value) in an ARIMA model. The acf () function is to define values of p and q by looking at the shape of the graph. Combines with the below table, we can determine which type of the model to select and the values of p, d and q.

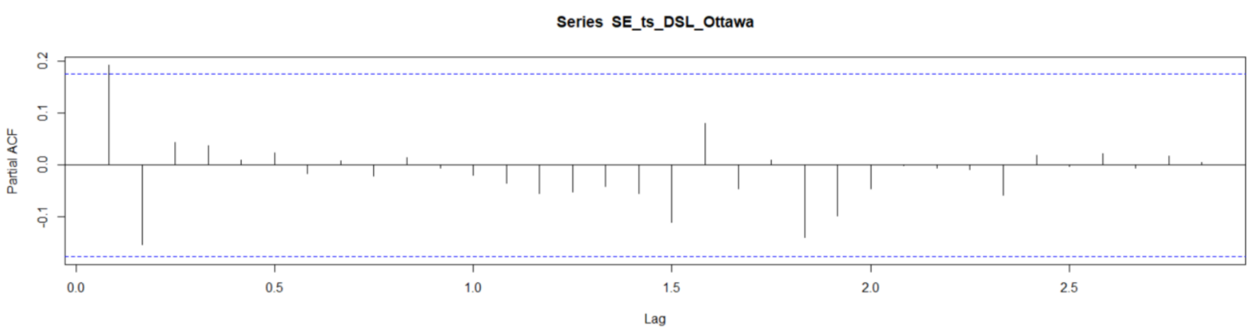
| Shape | Indicated Model |
|---|---|
| Exponential series decaying to 0 | Auto Regressive (AR) model. pacf() function to be used to identify the order of the model |
| Alternative positive and negative spikes, decaying to 0 | Auto Regressive (AR) model. pacf() function to be used to identify the order of the model |
| One or more spikes in series, rest all are 0 | Moving Average(MA) model, identify order where plot becomes 0 |
| After a few lags overall a decaying series | Mixed AR & MA model |
| Total series is 0 or nearly 0 | Data is random |
| Half values at fixed intervals | We need to include seasonal AR term |
| Visible spikes, no decay to 0 | Series is not stationary |

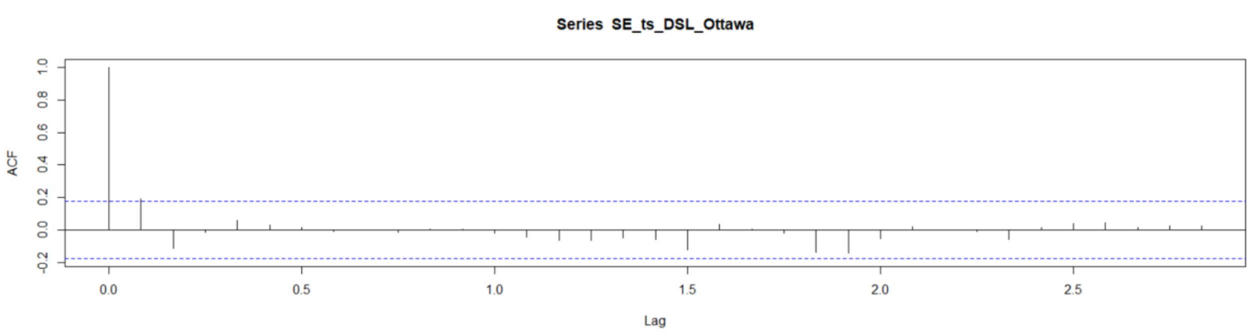
The R code to run pacf () function and acf () function for this dataset is shown below associated with the result graph. The purpose of par () function is to make the two graphs display in one plot view.

*par (mfrow=c (2,1))*

*pacf (SE_ts_DSL_Ottawa, lag.max=34)*

*acf (SE_ts_DSL_Ottawa, lag.max=34)*



**2) Fit the Model**

To fit the model, I applied the arima () function with decided parameter values.

*fitARIMA_ts_DSL_Ottawa <- arima (ts_DSL_Ottawa, order=c (1,1,1), seasonal = list (order = c (1,0,0), period = 12), method="ML")*

The "order" parameter indicates the non-seasonal part of the ARIMA model and the order of (p,d,q) is the AR, the degree of difference, and the MA. The "seasonal" parameter indicates the seasonal part of the ARIMA model. The period is the frequency which is 12 (monthly) in this case. And this "seasonal" parameter requires a list of components for order and period, but it will give a numeric vector of length 3 and then turn into a suitable list with the specification as the "order". The "method" parameter indicates the fitting method, including ML (Maximum Likelihood) and CSS (Minimize Conditional Sum-of-Squares). The default is CSS. And the ML will maximize the log-likelihood for the given (p,d,q) values to maximize the probability of obtaining the observed data.

The output of the ARIMA model can be seen by just tying the model name:

*fitARIMA_ts_DSL_Ottawa*

```
> fitARIMA_ts_DSL_Ottawa

Call:
arima(x = ts_DSL_Ottawa, order = c(1, 1, 1), seasonal = list(order = c(1, 0,
    0), period = 12), method = "ML")

Coefficients:
          ar1     ma1    sar1
      -0.1876  0.4528  0.0215
s.e.   0.2739  0.2498  0.1762

sigma^2 estimated as 1.666:  log likelihood = -207.62,  aic = 423.25
```

From the output, we can check the fitted coefficients with standard error (s.e.) for each coefficient. By checking the coefficients, we can exclude the insignificant ones. We can use a function confint () for this purpose. We can also use coeftest () function for Z test of the coefficients to check p value and significant level.

*confint (fitARIMA_ts_DSL_Ottawa)*

*coeftest (fitARIMA_ts_DSL_Ottawa)*

```
> confint(fitARIMA_ts_DSL_Ottawa)
           2.5 %    97.5 %
ar1  -0.72444068 0.3492247
ma1  -0.03673989 0.9423868
sar1 -0.32394342 0.3669358
> coeftest(fitARIMA_ts_DSL_Ottawa)

z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ar1  -0.187608   0.273899 -0.6850  0.49337
ma1   0.452823   0.249782  1.8129  0.06985 .
sar1  0.021496   0.176248  0.1220  0.90293
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The model fitting can be a repetitive process because we need to adjust different (p,d,q) values to find the most efficient and optimized model. Additionally, we can also apply the auto.arima () function to automate the ARIMA modeling, which will automatically help us decide the best (p,d,q) values. The auto.arima () function in R uses a combination of the Unit Root Test and the minimization of AIC and MLE to obtain the ARIMA model.

For example, I applied the auto.arima () function in R and achieve the following result:

*auto.arima (ts_DSL_Ottawa, trace=TRUE)*

```
> auto.arima(ts_DSL_Ottawa, trace=TRUE)

 ARIMA(2,1,2)(1,0,1)[12] with drift         : Inf
 ARIMA(0,1,0)            with drift         : 427.1068
 ARIMA(1,1,0)(1,0,0)[12] with drift         : 425.1892
 ARIMA(0,1,1)(0,0,1)[12] with drift         : 423.5177
 ARIMA(0,1,0)                               : 425.8586
 ARIMA(0,1,1)            with drift         : 421.4081
 ARIMA(0,1,1)(1,0,0)[12] with drift         : 423.5208
 ARIMA(0,1,1)(1,0,1)[12] with drift         : 425.349
 ARIMA(1,1,1)            with drift         : 423.048
 ARIMA(0,1,2)            with drift         : 422.9383
 ARIMA(1,1,0)            with drift         : 423.0807
 ARIMA(1,1,2)            with drift         : 425.0865
 ARIMA(0,1,1)                               : 419.8122
 ARIMA(0,1,1)(1,0,0)[12]                    : 421.8992
 ARIMA(0,1,1)(0,0,1)[12]                    : 421.8975
 ARIMA(0,1,1)(1,0,1)[12]                    : 423.7008
 ARIMA(1,1,1)                               : 421.464
 ARIMA(0,1,2)                               : 421.3923
 ARIMA(1,1,0)                               : 421.4619
 ARIMA(1,1,2)                               : 423.5219

 Best model: ARIMA(0,1,1)

Series: ts_DSL_Ottawa
ARIMA(0,1,1)

Coefficients:
         ma1
      0.2807
s.e.  0.0926

sigma^2 estimated as 1.686:  log likelihood=-207.86
AIC=419.71   AICc=419.81   BIC=425.35
```

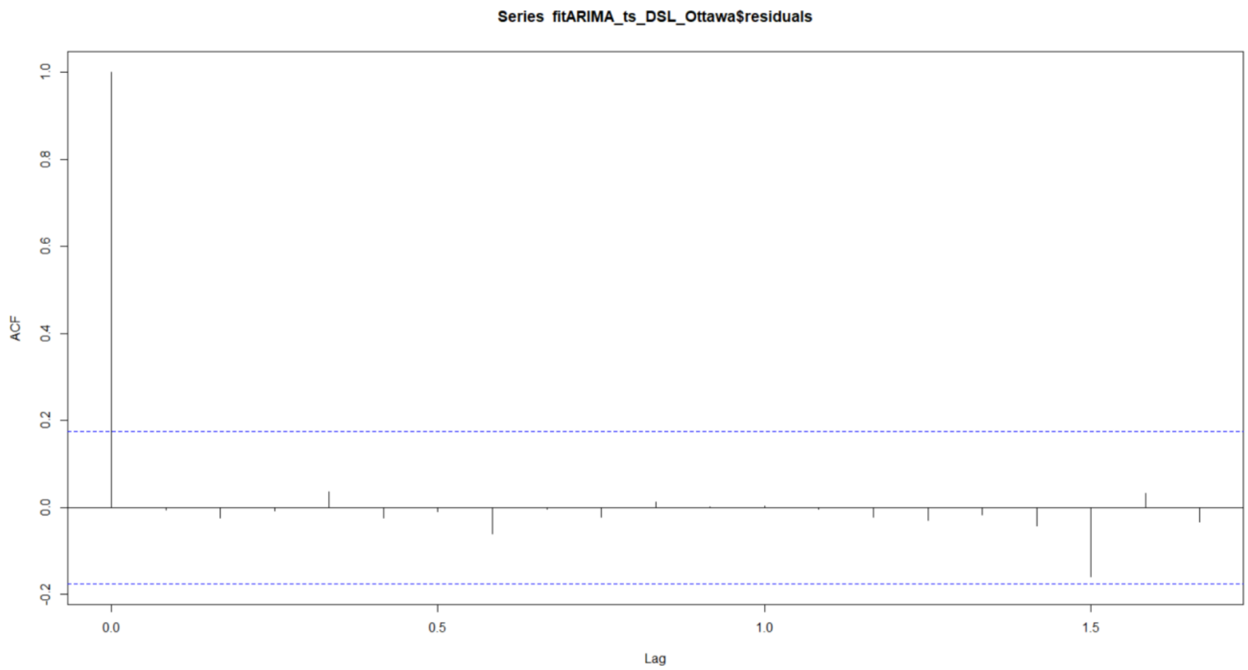
The result indicates that the automation selected ARIMA (0,1,1) as the best model.

### 3. Measures Evaluation

In measures evaluation, I use the acf () function to plot the pattern in the residuals of the model, which is to check if the pattern would be like white noise. Because we will only do forecasting when residuals are like white noise, otherwise, we will need to modify the selected model. I will also do a portmanteau test for the residuals of the model. The portmanteau test is to test the independence at all lags up to the one specified, which is an overall randomness test based on several lags. Because the portmanteau test is on the residuals, its hypothesis tested is also for the residuals of the selected ARIMA model and does not have autocorrelation.

### 1) ACF Plotting

*acf (fitARIMA_ts_DSL_Ottawa$residuals)*




The ACF plotting shows that the residuals do not have significant autocorrelations.

### 2) Portmanteau Testing - Ljung–Box Q test

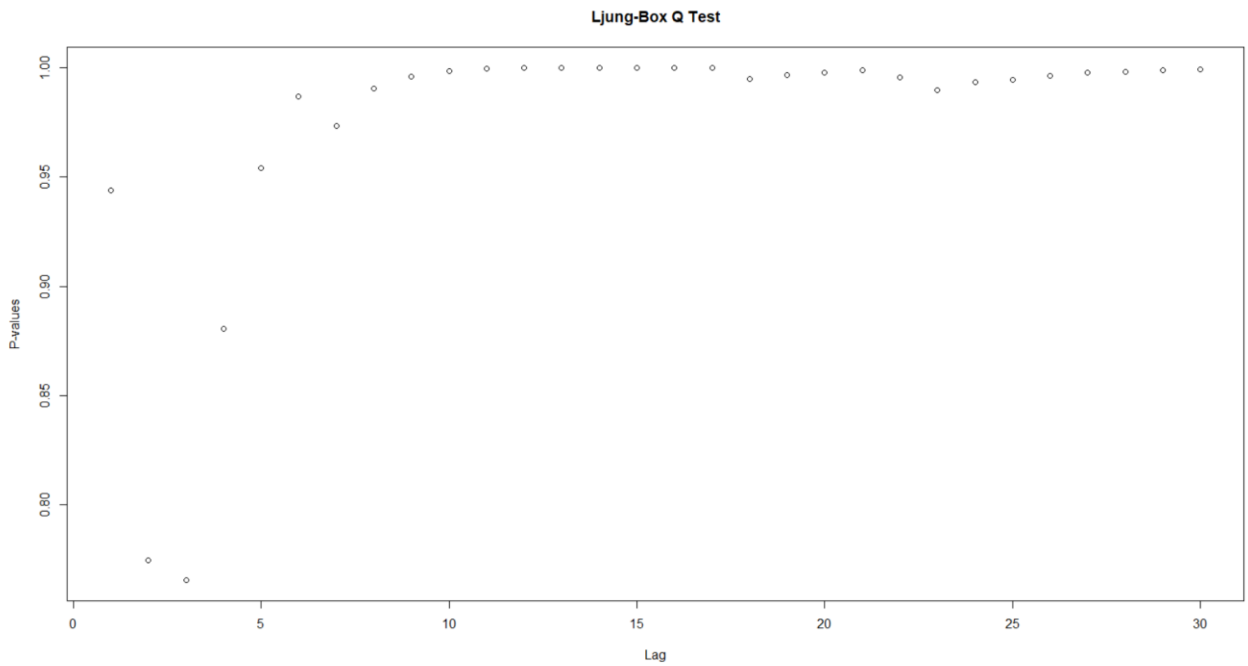
For portmanteau testing, I use Ljung–Box Q test.

*boxresult <- LjungBoxTest (fitARIMA_ts_DSL_Ottawa$residuals,k=2,StartLag=1)*

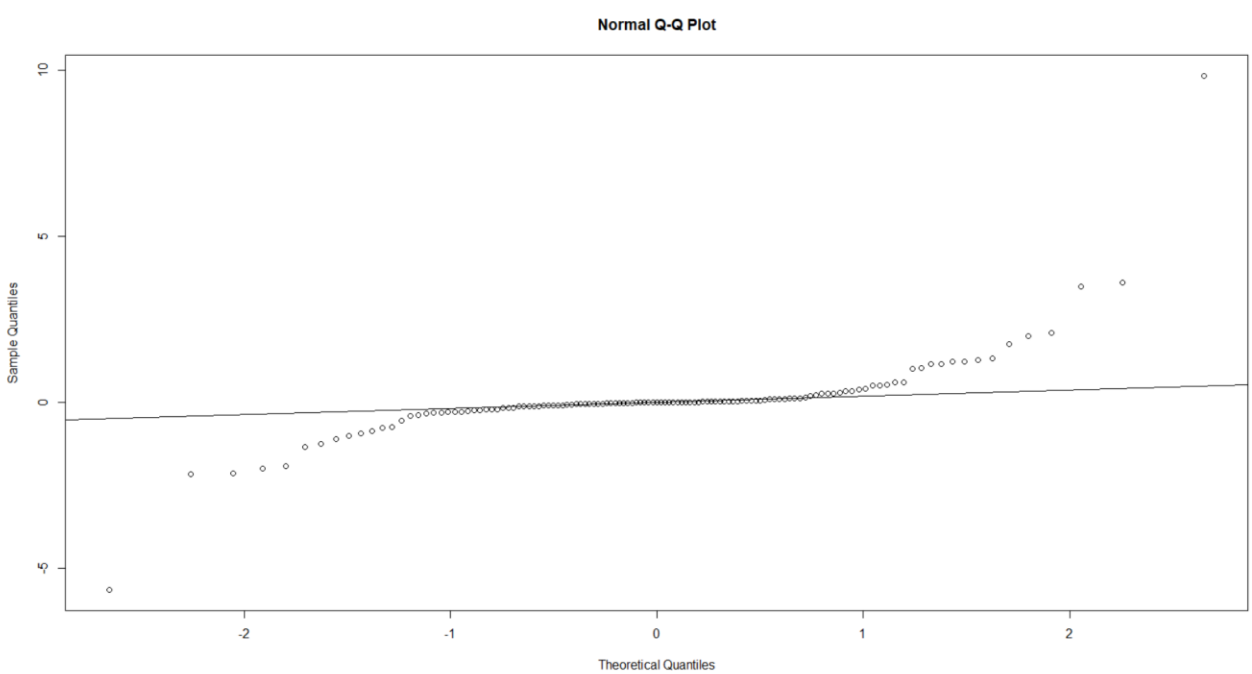
*plot(boxresult[,3],main= "Ljung-Box Q Test", ylab= "P-values", xlab= "Lag")*

*qqnorm (fitARIMA_ts_DSL_Ottawa$residuals)*

*qqline (fitARIMA_ts_DSL_Ottawa$residuals)*




From the result graph, we can see that the p-values for the Ljung–Box Q test are all greater than 0.05, which means non-significance.




And the above Normal graph indicates the values are normal because they are almost aligning on a line.

Therefore, based on all the graphs generated in this measure evaluation process, it is indicating that there is no pattern in the residuals. So, I can proceed to the Step 4 of the Approach section to conduct the forecasting.

## Step 4: Time Series Analysis - Future Fuel Price Prediction and Forecasting

The selected ARIMA model can be used as a predictive model for the forecasting of future values of the time series. To run prediction in R, there are two functions I found, one is the predict () function, another is the forecast.Arima () function.

### 1. The predict () function

In this function, the "n.ahead" argument represents how many time steps ahead to predict. In this case, I used 12 as a one-year ahead prediction. So, I will archive the predicted value from June 2020 to May 2021.

*predict (fitARIMA_ts_DSL_Ottawa, n.ahead = 12)*

```
> predict(fitARIMA_ts_DSL_Ottawa,n.ahead = 12)
$pred
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
2020                                              62.78422 62.78422 62.80507 62.83842 62.88846 62.93536 62.93432
2021 63.00312 63.22827 63.16114 63.10422 63.06274

$se
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
2020                                              1.292936 2.100865 2.674992 3.146035 3.555205 3.921917 4.257157
2021 4.567859 4.858733 5.133150 5.393624 5.642086
```

### 2. The forecast.Arima () function

In this function, the "h" argument represents how many time steps ahead. In this case, I keep the same time ahead which is the 12-month forecasting. The result contains both 80% prediction interval and 95% prediction interval.

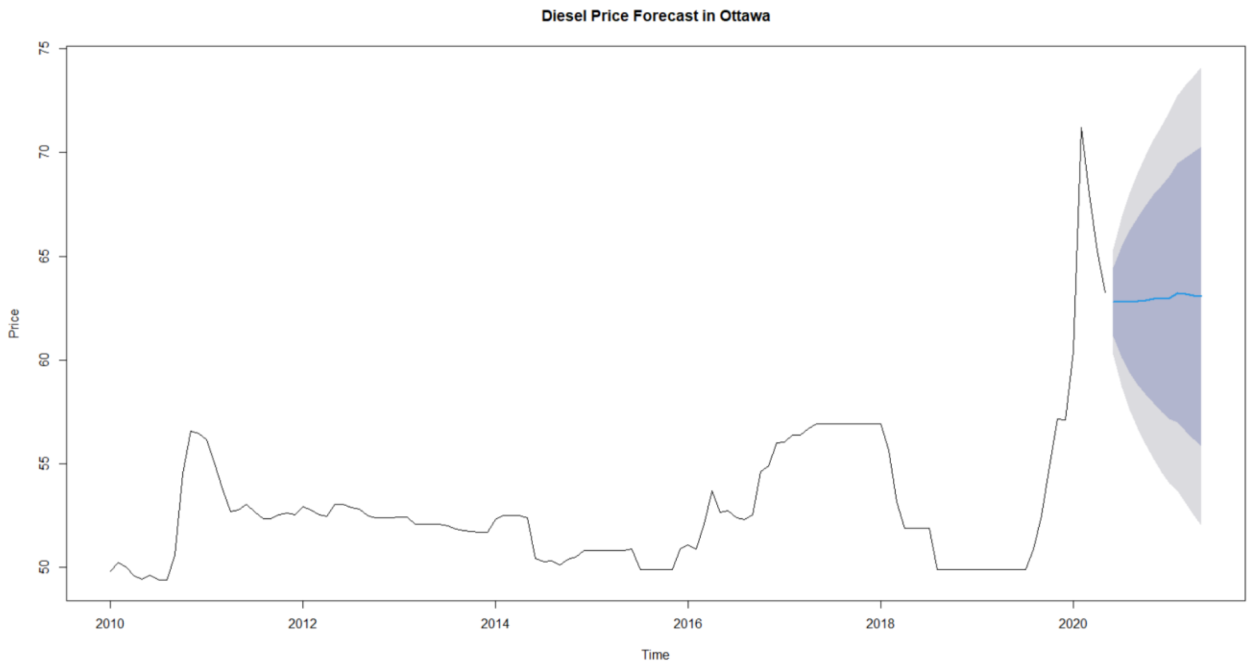
*forecast (fitARIMA_ts_DSL_Ottawa, h=12)*

```
> forecast (fitARIMA_ts_DSL_Ottawa, h=12)
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Jun 2020       62.78422 61.12725 64.44118 60.25011 65.31833
Jul 2020       62.78422 60.09185 65.47659 58.66660 66.90184
Aug 2020       62.80507 59.37693 66.23321 57.56218 68.04795
Sep 2020       62.83842 58.80662 66.87023 56.67231 69.00454
Oct 2020       62.88846 58.33228 67.44463 55.92038 69.85653
Nov 2020       62.93536 57.90922 67.96150 55.24855 70.62218
Dec 2020       62.93432 57.47855 68.39009 54.59045 71.27819
Jan 2021       63.00312 57.14917 68.85706 54.05028 71.95595
Feb 2021       63.22827 57.00155 69.45498 53.70533 72.75121
Mar 2021       63.16114 56.58274 69.73953 53.10035 73.22193
Apr 2021       63.10422 56.19202 70.01643 52.53292 73.67553
May 2021       63.06274 55.83211 70.29336 52.00445 74.12102
```

Additionally, I can also visualize the forecasting by plotting the prediction.

*future_DSL_Ottawa <- forecast (fitARIMA_ts_DSL_Ottawa, h=12)*

*plot (future_DSL_Ottawa, main="Diesel Price Forecast in Ottawa", ylab="Price", xlab="Time")*




In the result graph, the prediction / forecast is illustrated as a blue line. The light shaded area represents the 95% prediction interval and the dark shaded area represents the 80% prediction interval.

## Initial Result

My first prediction of Diesel price for Ottawa marketplace indicates that from the period of June 2020 and May 2021, the price would be stable with a slight increase, and the price would probably stay between 60 and 70 (prices in cents per litre).

I would think that my first try of the forecasting is working. I will apply the same approach to other fuel types and cities. And I will finally generate a comprehensive result to complete my time series analysis project.

All the R codes that I currently used are included here. I will organize my codes for final version and post on GitHub in future weeks.