Walmart Sales Prediction using SARIMA without Features

```
library(readxl)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyr)
options(warn=-1)
traindata <- read.csv("trainWalmart.csv")
test<- read.csv("testWalmart.csv")
features<-read.csv('featuresWalmart.csv')
stores<- read.csv('storesWalmart.csv')
str(traindata)

## 'data.frame':    421570 obs. of  5 variables:
##  $ Store       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Dept        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date        : chr  "2/5/2010" "2/12/2010" "2/19/2010" "2/26/2010" ...
##  $ Weekly_Sales: num  24925 46039 41596 19404 21828 ...
##  $ IsHoliday   : logi  FALSE TRUE FALSE FALSE FALSE FALSE ...

str(test)

## 'data.frame':    115064 obs. of  4 variables:
##  $ Store    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Dept     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date     : chr  "11/2/2012" "11/9/2012" "11/16/2012" "11/23/2012" ...
##  $ IsHoliday: logi  FALSE FALSE FALSE TRUE FALSE FALSE ...

str(features)

## 'data.frame':    8190 obs. of  12 variables:
##  $ Store       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Date        : chr  "2/5/2010" "2/5/2010" "2/5/2010" "2/5/2010" ...
##  $ Temperature : num  42.3 40.2 45.7 43.8 39.7 ...
##  $ Fuel_Price  : num  2.57 2.57 2.57 2.6 2.57 ...
##  $ MarkDown1   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MarkDown2   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MarkDown3   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MarkDown4   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MarkDown5   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ CPI         : num  211 211 214 126 212 ...
```

```
##  $ Unemployment: num  8.11 8.32 7.37 8.62 6.57 ...
##  $ IsHoliday   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
str(stores)
```

```
## 'data.frame':    45 obs. of  3 variables:
##  $ Store: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Type : chr  "A" "A" "B" "A" ...
##  $ Size : int  151315 202307 37392 205863 34875 202505 70713 155078 125833
126512 ...
```

Data Exploration:

Converting data type

```
traindata$Date<-as.Date(traindata$Date, "%m/%d/%Y")
head(traindata)
```

```
##   Store Dept       Date Weekly_Sales IsHoliday
## 1     1    1 2010-02-05     24924.50     FALSE
## 2     1    1 2010-02-12     46039.49      TRUE
## 3     1    1 2010-02-19     41595.55     FALSE
## 4     1    1 2010-02-26     19403.54     FALSE
## 5     1    1 2010-03-05     21827.90     FALSE
## 6     1    1 2010-03-12     21043.39     FALSE
```

Testing for missing values

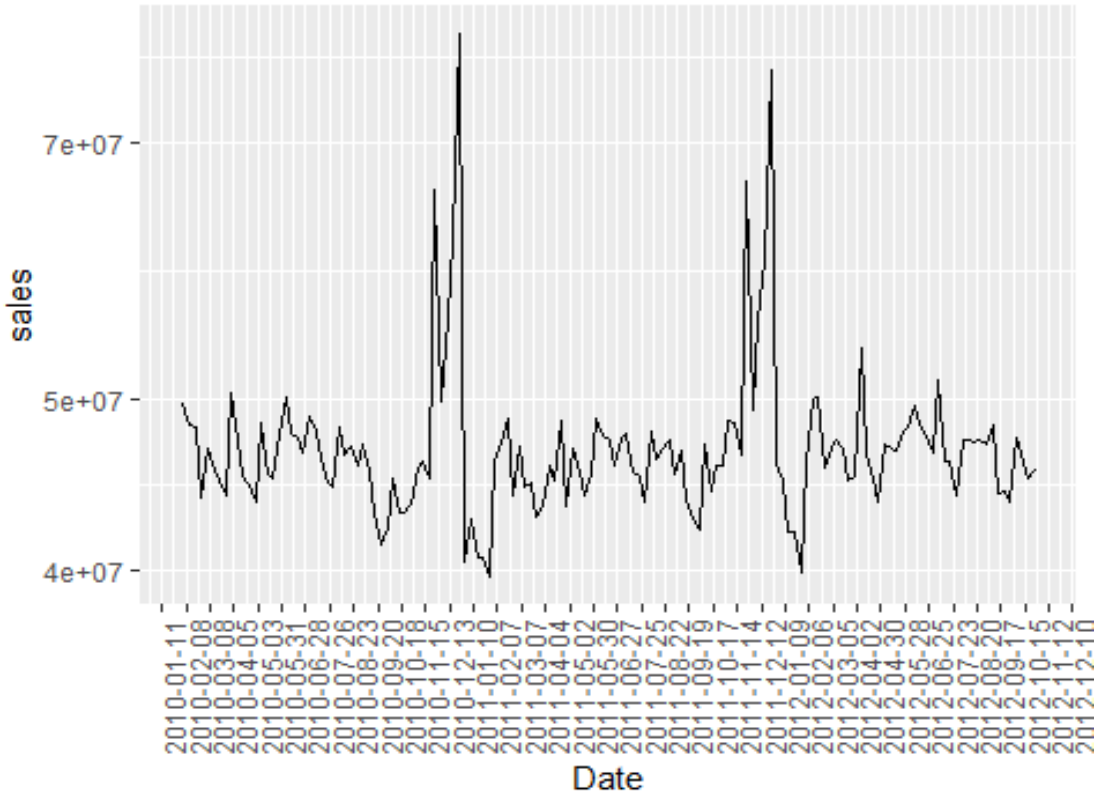
```
null<-sapply(traindata, function(x) sum(is.na(x)))
print(null)
```

```
##        Store         Dept         Date Weekly_Sales    IsHoliday
##            0            0            0            0            0
```

plotting to check data

```
library(dplyr)
library(ggplot2)
traindata %>%
  group_by(Date) %>% summarise(sales = sum(Weekly_Sales)) %>%
  ggplot(aes(x = Date, y = sales)) +
  geom_line() + scale_y_log10() + scale_x_date(date_labels = "%Y-%m-%d",
date_breaks = "4 weeks") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



It can be observed from the plot that sales are highest at the end of every year, which indicates that sales peak during the holiday and festive weeks of Thanks Giving, Christmas and New Year. In addition, there are lesser spike in sales during April-May weeks, that may be due to Easter and Labor day.

Grouping data and Creating a new data frame

```
traindata <- traindata %>% group_by(Date, Store) %>% summarise(weekly_sales =
sum(Weekly_Sales))

## `summarise()` regrouping output by 'Date' (override with `.groups`
argument)

traindata

## # A tibble: 6,435 x 3
## # Groups:   Date [143]
##    Date       Store weekly_sales
##    <date>     <int>        <dbl>
##  1 2010-02-05     1     1643691.
##  2 2010-02-05     2     2136989.
##  3 2010-02-05     3      461622.
##  4 2010-02-05     4     2135144.
##  5 2010-02-05     5      317173.
##  6 2010-02-05     6     1652635.
##  7 2010-02-05     7      496725.
##  8 2010-02-05     8     1004137.
```

```
##  9 2010-02-05     9      549506.
## 10 2010-02-05    10     2193049.
## # ... with 6,425 more rows
```

```
data<-data.frame(Date = traindata$Date, Weekly_sales =
traindata$weekly_sales)
head(data)
```

```
##         Date Weekly_sales
## 1 2010-02-05    1643690.9
## 2 2010-02-05    2136989.5
## 3 2010-02-05     461622.2
## 4 2010-02-05    2135143.9
## 5 2010-02-05     317173.1
## 6 2010-02-05    1652635.1
```

```
newdata<- data %>% group_by(Date) %>% summarise(weekly_sales =
sum(Weekly_sales))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(newdata)
```

```
## # A tibble: 6 x 2
##   Date       weekly_sales
##   <date>            <dbl>
## 1 2010-02-05    49750740.
## 2 2010-02-12    48336678.
## 3 2010-02-19    48276994.
## 4 2010-02-26    43968571.
## 5 2010-03-05    46871470.
## 6 2010-03-12    45925397.
```

Spliting data with 80:20 ratios

```
numrow<- nrow(newdata)
print(numrow)
```

```
## [1] 143
```

```
trainratio<- nrow(newdata)*0.80
print(trainratio)
```

```
## [1] 114.4
```

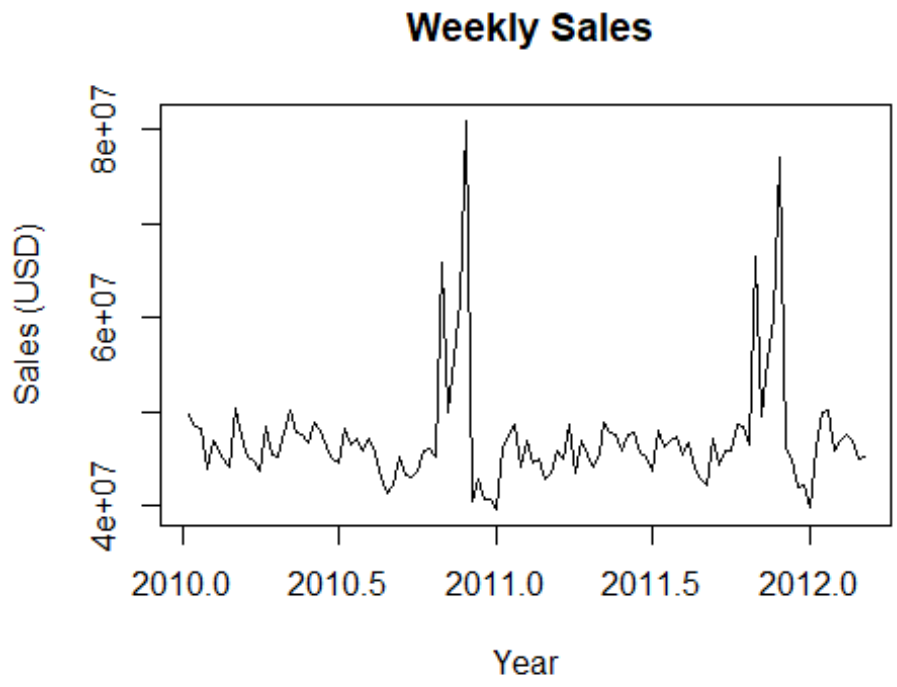
```
train <- newdata %>% slice(1:114)
str(train)
```

```
## tibble [114 x 2] (S3: tbl_df/tbl/data.frame)
##  $ Date        : Date[1:114], format: "2010-02-05" "2010-02-12" ...
##  $ weekly_sales: num [1:114] 49750741 48336678 48276994 43968571 46871470
...
```

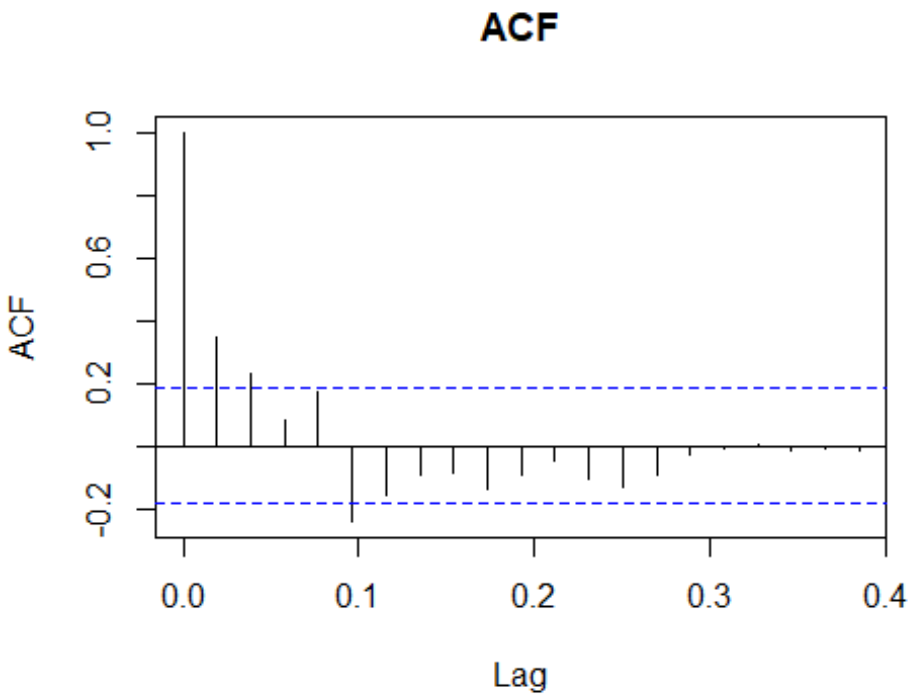
```
test <- newdata %>% slice(115:143)
str(test)

## tibble [29 x 2] (S3: tbl_df/tbl/data.frame)
##  $ Date        : Date[1:29], format: "2012-04-13" "2012-04-20" ...
##  $ weekly_sales: num [1:29] 46629261 45072530 43716799 47124198 46925879
...
```

Checking data

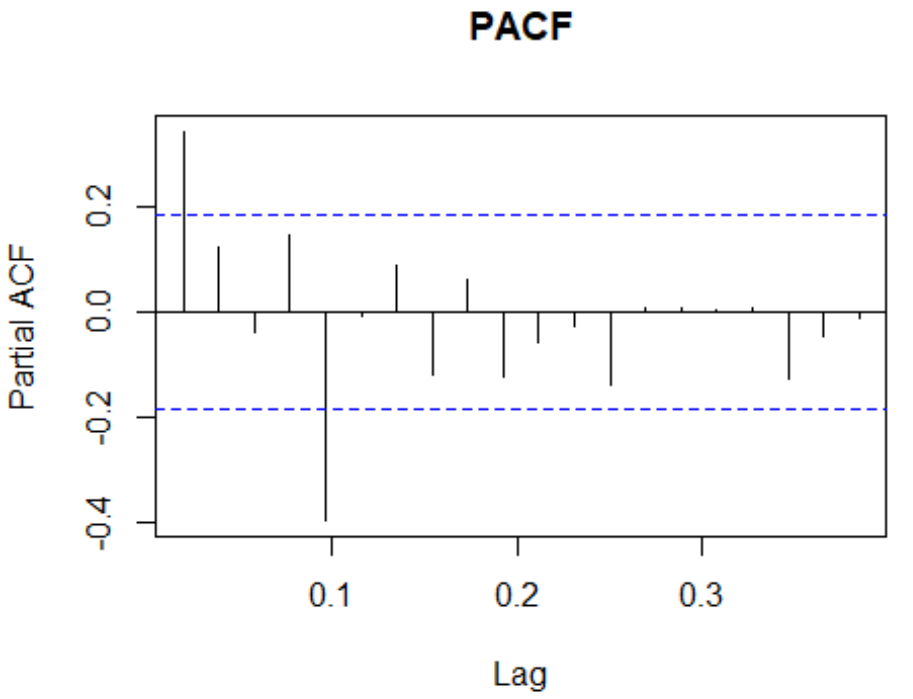
```
library(fpp2)

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

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

## v forecast  8.13     v expsmooth 2.3
## v fma       2.4

##

weekly_sales<-ts(train[-1], frequency = 52,start=c(2010,02),end=c(2012,10))
plot(weekly_sales,xlab="Year",ylab="Sales (USD)",main=" Weekly Sales")
```



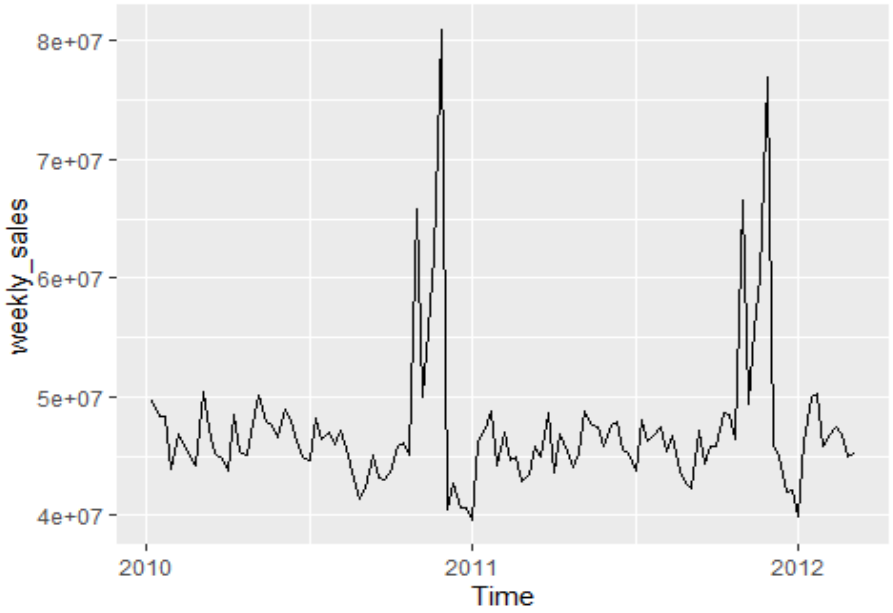

```
acf(weekly_sales,main = 'ACF')
```



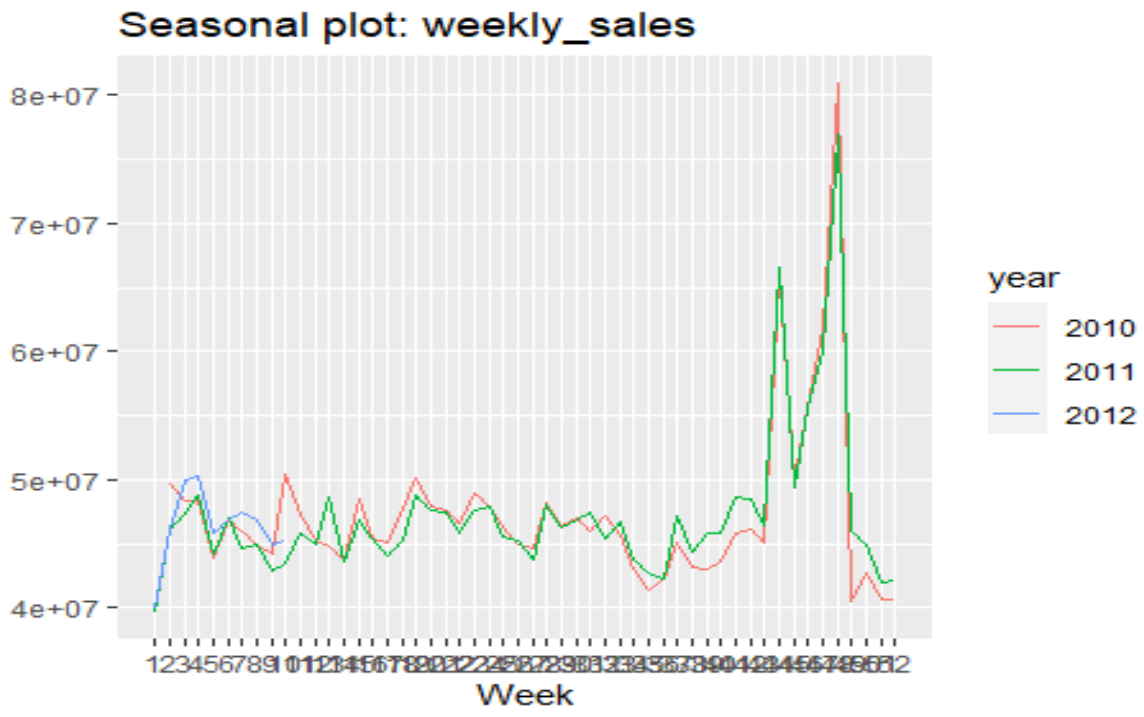

```
pacf(weekly_sales,main = 'PACF')
```




```
autoplot(weekly_sales, space.skip = 1)
```




```
ggseasonplot(weekly_sales, labelgap = 0.1)
```

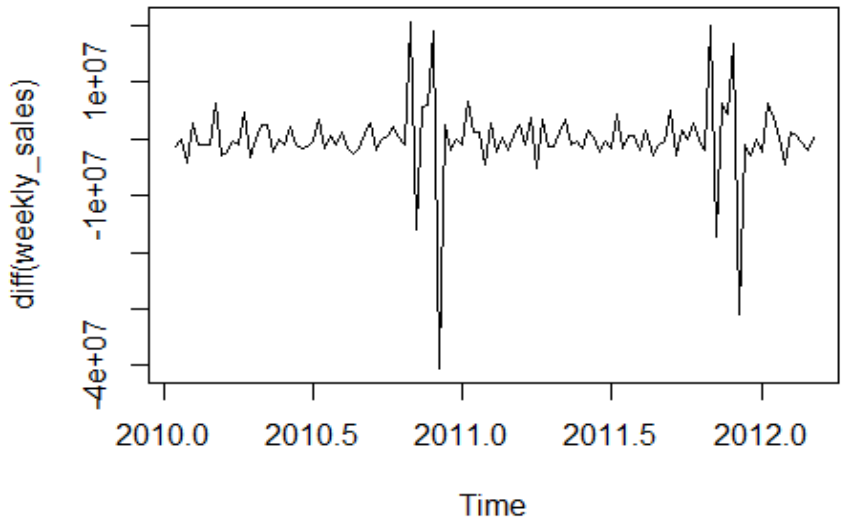


From the plots, we can observe that the weekly sales are not constant with seasonality component since sales are high at the end of the year. As we know that a stationary time series is one whose properties do not depend on time, time series with trends, or with seasonality, are not stationary. As a result, it can be observed that the series are not stationary.

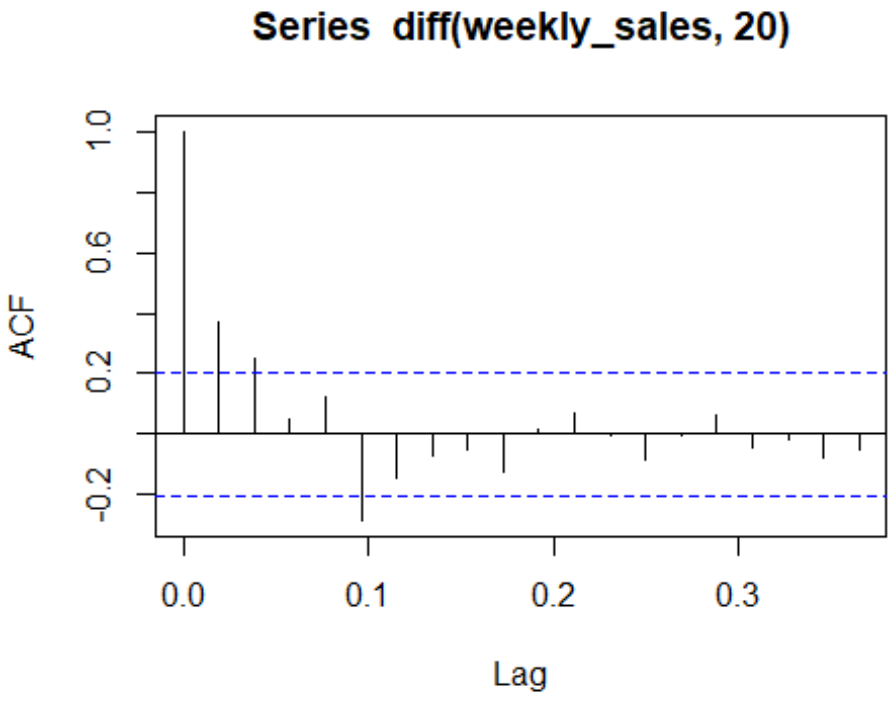
Data transformation:

Model selection by checking ACF and PACF plots of transformed data

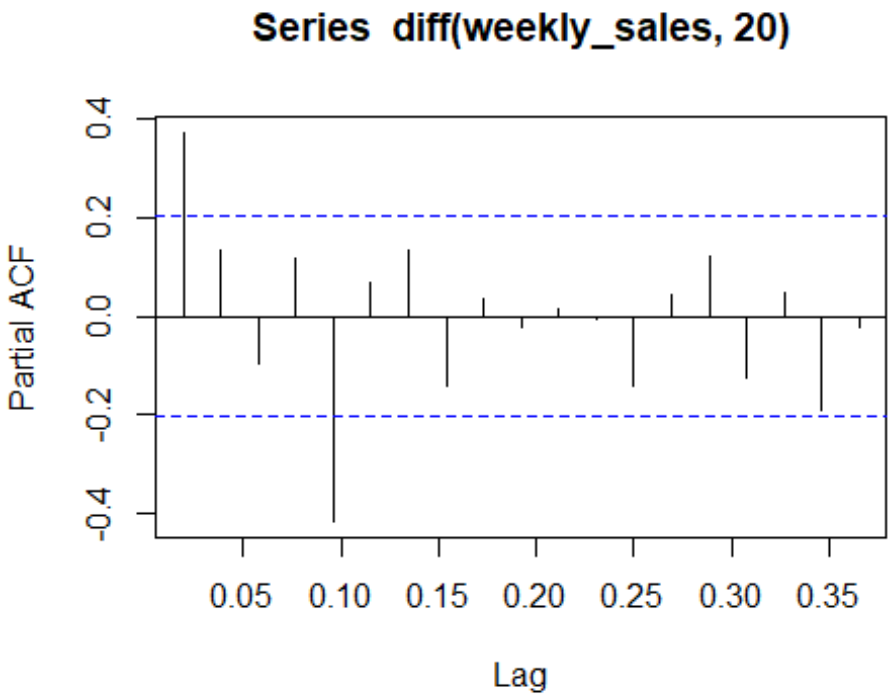
```
plot(diff(weekly_sales))
```




```
acf(diff(weekly_sales,20))
```



```
pacf(diff(weekly_sales,20))
```



By looking at the plot of the differenced monthly sales, it appears more stationarized. After diffencing data, we inspect the ACF and PACF plots, and we feel that the ACF is cutting off at the lag 2 and the PACF is tailing off. This would suggest that the weekly sales data follow an MA(2) process, or it follows an ARIMA(0,1,2) model. Rather than focus on this process, the plots also suggest that the data follow an AR(1) model since the ACF is tailing off at lag 1 and PACF is cutting off. Thus, AR(1)and MA(2) processes should be included with seasonal MA process.

Therefore, the model that we choose to fit to this data is (1,1,2) (0,1,1).

Model fitting

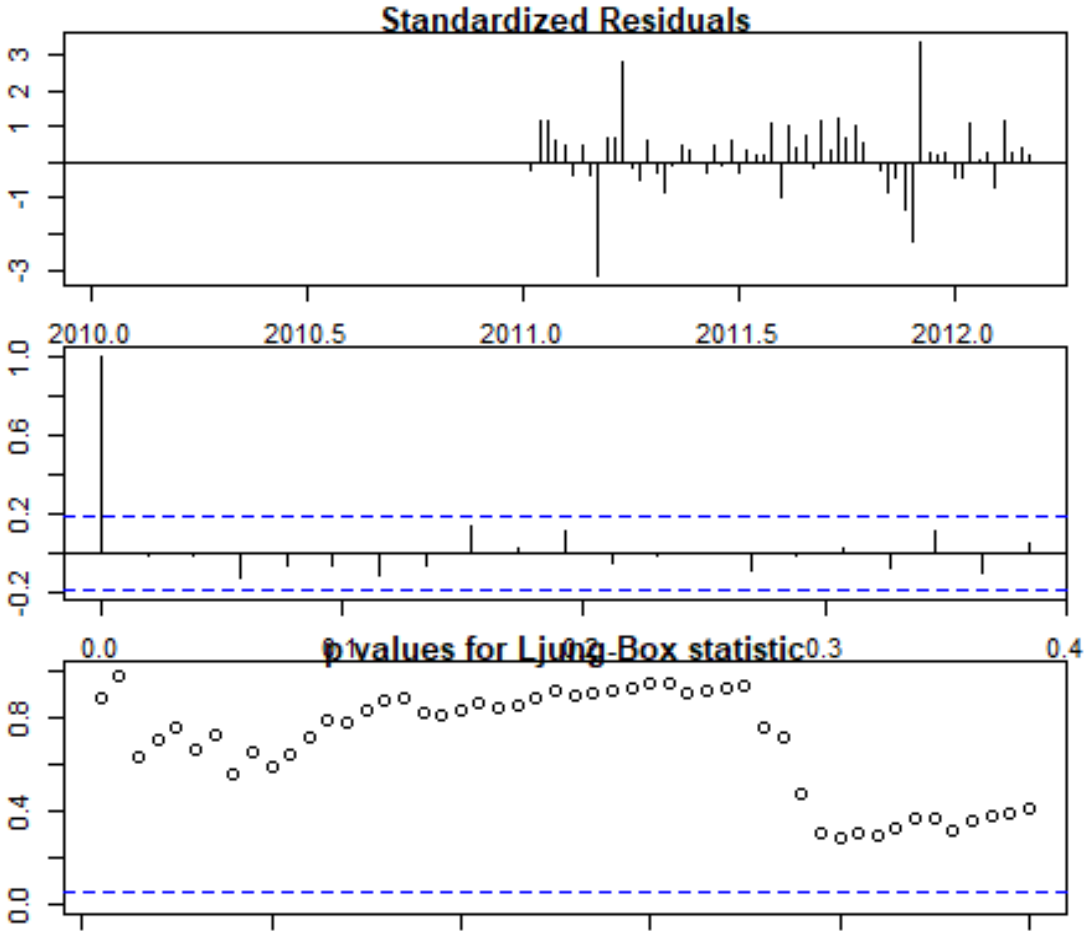
```
library(forecast)
fit.model = Arima(weekly_sales, order = c(1,1,2), seasonal = list(order =
c(0,1,1)))
summary(fit.model)
```

```
## Series: weekly_sales
## ARIMA(1,1,2)(0,1,1)[52]
##
## Coefficients:
##          ar1      ma1     ma2    sma1
##       0.0495  -0.7725  -0.101  0.0147
## s.e.  0.5792   0.5671   0.489  0.3129
##
```

```
## sigma^2 estimated as 3.494e+12:  log likelihood=-950.16
## AIC=1910.31   AICc=1911.42   BIC=1920.78
##
## Training set error measures:
##                     ME    RMSE      MAE       MPE     MAPE      MASE
ACF1
## Training set 208657.7 1315884 680110.6 0.4772069 1.425833 0.4789824 -
0.01371401
```

Model diagnosis

```
par(mfrow=c(1,1),mar=c(1,2,1,4))
tsdiag(fit.model, gof.lag = 50)
```



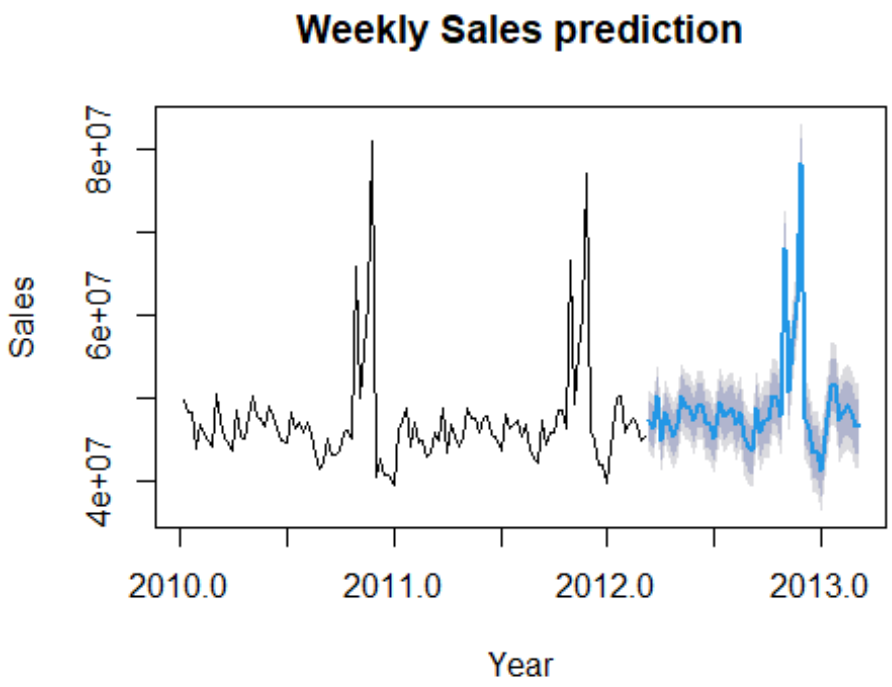
From the standardized residuals, the plot pattern is not clear, but it seems to be white noise process. The ACF of residuals plot indicates that all the lags are significant. Most of the p-values are above 0, indicating model fits

Model prediction

```
mod.pred<- fitted(fit.model)
print(mod.pred)
```

```
## Time Series:
## Start = c(2010, 2)
## End = c(2012, 10)
## Frequency = 52
##   [1] 49722017 48324927 48269352 43966675 46867332 45922823 44987643
44133600
##   [9] 50417560 47362558 45183268 44734518 43706177 48499589 45329739
45119992
##  [17] 47754832 50183657 47824218 47620034 46608109 48914343 47897520
46243596
##  [25] 44889886 44631614 48202694 46463905 47059871 45909827 47193077
45634790
##  [33] 43083622 41363022 42243386 45103563 43151974 43069186 43604753
45781704
##  [41] 46124191 45125976 65800930 49905143 55657278 61805426 80897461
40439351
##  [49] 42780161 40680045 40660909 39641463 46518345 45246246 46525192
42977241
##  [57] 46074351 45248958 44015951 43485745 49384434 44674732 43657627
43501262
##  [65] 43759892 47777229 44346309 44567894 46912022 48954894 46767354
46850743
##  [73] 45882124 48103132 46935506 45607636 44104829 44176894 47455192
45929587
##  [81] 46541158 45447128 47223160 44879916 43087200 41315058 42476074
45064446
##  [89] 43734268 43536700 44602908 46811860 47478189 46493218 67024311
50992917
##  [97] 56365784 62518927 81076394 39787843 44438091 41651762 41623557
40633159
## [105] 46868323 47978830 50054848 45227495 48199278 45378508 46402518
44249062
## [113] 44872020
```

```
plot(forecast(fit.model,52), main = 'Weekly Sales prediction', ylab =
'Sales', xlab = 'Year')
```



From the prediction, it appears that the weekly sales are not stable with seasonality component since the sales are generally high at the end of every year.

Model evaluation using test data

```
library(forecast)
library(MLmetrics)

##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall

weekly_sales_test<-ts(test[-1], frequency =
52,start=c(2010,02),end=c(2012,10))
fit.test<- Arima(weekly_sales_test, order = c(1,1,2), seasonal = list(order =
c(0,1,1)))
summary(fit.test)

## Series: weekly_sales_test
## ARIMA(1,1,2)(0,1,1)[52]
##
## Coefficients:
##          ar1      ma1      ma2    sma1
##       0.2269  -0.9249  -0.0749  -0.998
## s.e.  0.3051   0.3009   0.2959   1.090
```

```
##
## sigma^2 estimated as 3.6e+12:  log likelihood=-971.76
## AIC=1953.52   AICc=1954.63   BIC=1963.99
##
## Training set error measures:
##                     ME    RMSE      MAE       MPE     MAPE      MASE
## Training set 100898.3 1335763 822764.7 0.1574946 1.757173 0.3660331
##                      ACF1
## Training set -0.008375278
```

```
onestep <- fitted(fit.test)
mspe<-mean((as.vector(weekly_sales_test)-as.vector(onestep))^2)
sprintf('%s = %3.2f', 'MSPE of this model', mspe)
```

```
## [1] "MSPE of this model = 1784263646531.73"
```

```
mape<- MAPE(as.vector(onestep), as.vector(weekly_sales_test))
sprintf('%s = %3.4f', 'MAPE of this model', mape)
```

```
## [1] "MAPE of this model = 0.0176"
```

From the result, the MSPE is a very large value since the data are large; however, we can observe that the MAPE of this model is equal to 0.0176 which means the prediction is off by 1.76% on average.

Walmart Sales Prediction using SARIMA with additional features

Data Exploration:

Merging train and features data

```
features$Date <-as.Date(features$Date, "%m/%d/%Y")
train.features<- merge(traindata,features, by = c('Date', 'Store'))
head(train.features)
```

```
##         Date Store weekly_sales Temperature Fuel_Price MarkDown1 MarkDown2
## 1 2010-02-05     1      1643691       42.31      2.572        NA        NA
## 2 2010-02-05    10      2193049       54.34      2.962        NA        NA
## 3 2010-02-05    11      1528009       46.04      2.572        NA        NA
## 4 2010-02-05    12      1100046       49.47      2.962        NA        NA
## 5 2010-02-05    13      1967221       31.53      2.666        NA        NA
## 6 2010-02-05    14      2623470       27.31      2.784        NA        NA
##   MarkDown3 MarkDown4 MarkDown5      CPI Unemployment IsHoliday
## 1        NA        NA        NA 211.0964        8.106     FALSE
## 2        NA        NA        NA 126.4421        9.765     FALSE
## 3        NA        NA        NA 214.4249        7.368     FALSE
## 4        NA        NA        NA 126.4421       13.975     FALSE
## 5        NA        NA        NA 126.4421        8.316     FALSE
## 6        NA        NA        NA 181.8712        8.992     FALSE
```

Cleaning data

```
Missingvalue = function (x) {sum(is.na(x)) }
apply(train.features, 2, Missingvalue)
```

```
##         Date        Store weekly_sales  Temperature   Fuel_Price
MarkDown1
##            0            0            0            0            0
4155
##    MarkDown2    MarkDown3    MarkDown4    MarkDown5          CPI
Unemployment
##         4798         4389         4470         4140            0
0
##    IsHoliday
##            0
```

```
## Fill the null values of markdown with zero
train.features$MarkDown1[is.na(train.features$MarkDown1)] = 0
train.features$MarkDown2[is.na(train.features$MarkDown2)] = 0
train.features$MarkDown3[is.na(train.features$MarkDown3)] = 0
train.features$MarkDown4[is.na(train.features$MarkDown4)] = 0
train.features$MarkDown5[is.na(train.features$MarkDown5)] = 0
apply(train.features, 2, Missingvalue)
```

```
##         Date        Store weekly_sales  Temperature   Fuel_Price
MarkDown1
##            0            0            0            0            0
0
##    MarkDown2    MarkDown3    MarkDown4    MarkDown5          CPI
Unemployment
##            0            0            0            0            0
0
##    IsHoliday
##            0
```

```
head(train.features)
```

```
##         Date Store weekly_sales Temperature Fuel_Price MarkDown1 MarkDown2
## 1 2010-02-05     1      1643691       42.31      2.572         0         0
## 2 2010-02-05    10      2193049       54.34      2.962         0         0
## 3 2010-02-05    11      1528009       46.04      2.572         0         0
## 4 2010-02-05    12      1100046       49.47      2.962         0         0
## 5 2010-02-05    13      1967221       31.53      2.666         0         0
## 6 2010-02-05    14      2623470       27.31      2.784         0         0
##   MarkDown3 MarkDown4 MarkDown5      CPI Unemployment IsHoliday
## 1         0         0         0 211.0964        8.106     FALSE
## 2         0         0         0 126.4421        9.765     FALSE
## 3         0         0         0 214.4249        7.368     FALSE
## 4         0         0         0 126.4421       13.975     FALSE
## 5         0         0         0 126.4421        8.316     FALSE
## 6         0         0         0 181.8712        8.992     FALSE
```

Data visualization:

Variables correlation:

```
cor(train.features[,-1])
```

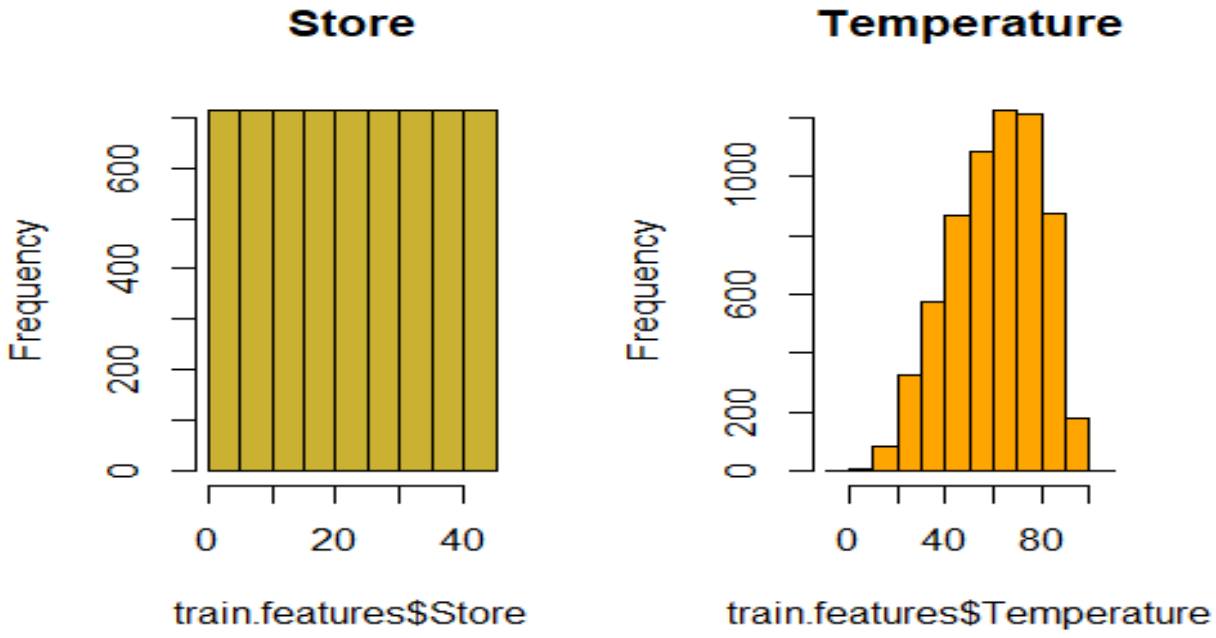
```
##                   Store weekly_sales Temperature   Fuel_Price
MarkDown1
## Store         1.00000000 -0.335332015 -0.02265908  0.060022955 -
0.079661274
## weekly_sales -0.33533201  1.000000000 -0.06381001  0.009463786
0.179107167
## Temperature  -0.02265908 -0.063810013  1.00000000  0.144981806 -
0.036237609
## Fuel_Price    0.06002295  0.009463786  0.14498181  1.000000000
0.289324610
## MarkDown1    -0.07966127  0.179107167 -0.03623761  0.289324610
1.000000000
## MarkDown2    -0.04079185  0.080156586 -0.17886063  0.029477342
0.177761780
## MarkDown3    -0.02383884  0.120288743 -0.05631762  0.018229516 -
0.011984381
## MarkDown4    -0.05610674  0.139194928 -0.05552536  0.162676685
0.839298757
## MarkDown5    -0.02704243  0.173272609 -0.02077404  0.212811108
0.416222048
## CPI          -0.20949193 -0.072634162  0.17688768 -0.170641795
0.008191866
## Unemployment  0.22353127 -0.106176090  0.10115786 -0.034683745 -
0.108214593
## IsHoliday     0.00000000  0.036890968 -0.15509133 -0.078346518 -
0.002902258
##                 MarkDown2   MarkDown3    MarkDown4   MarkDown5
CPI
## Store        -0.040791845 -0.02383884 -0.056106745 -0.02704243 -
0.209491930
## weekly_sales  0.080156586  0.12028874  0.139194928  0.17327261 -
0.072634162
## Temperature  -0.178860630 -0.05631762 -0.055525363 -0.02077404
0.176887676
## Fuel_Price    0.029477342  0.01822952  0.162676685  0.21281111 -
0.170641795
## MarkDown1     0.177761780 -0.01198438  0.839298757  0.41622205
0.008191866
## MarkDown2     1.000000000 -0.00509847  0.115666963  0.13365166 -
0.004449103
## MarkDown3    -0.005098470  1.00000000 -0.010370860  0.04363920 -
0.005895830
## MarkDown4     0.115666963 -0.01037086  1.000000000  0.30449086 -
0.003575157
## MarkDown5     0.133651663  0.04363920  0.304490859  1.00000000
```

```
0.065375674
## CPI          -0.004449103 -0.00589583 -0.003575157  0.06537567
1.000000000
## Unemployment -0.042417067 -0.01857460 -0.078118568 -0.12289513 -
0.302020064
## IsHoliday     0.200394297  0.25661480  0.011059340 -0.01486512 -
0.002162091
##              Unemployment    IsHoliday
## Store           0.22353127  0.000000000
## weekly_sales   -0.10617609  0.036890968
## Temperature     0.10115786 -0.155091329
## Fuel_Price     -0.03468374 -0.078346518
## MarkDown1      -0.10821459 -0.002902258
## MarkDown2      -0.04241707  0.200394297
## MarkDown3      -0.01857460  0.256614795
## MarkDown4      -0.07811857  0.011059340
## MarkDown5      -0.12289513 -0.014865124
## CPI            -0.30202006 -0.002162091
## Unemployment    1.00000000  0.010960284
## IsHoliday       0.01096028  1.000000000
```

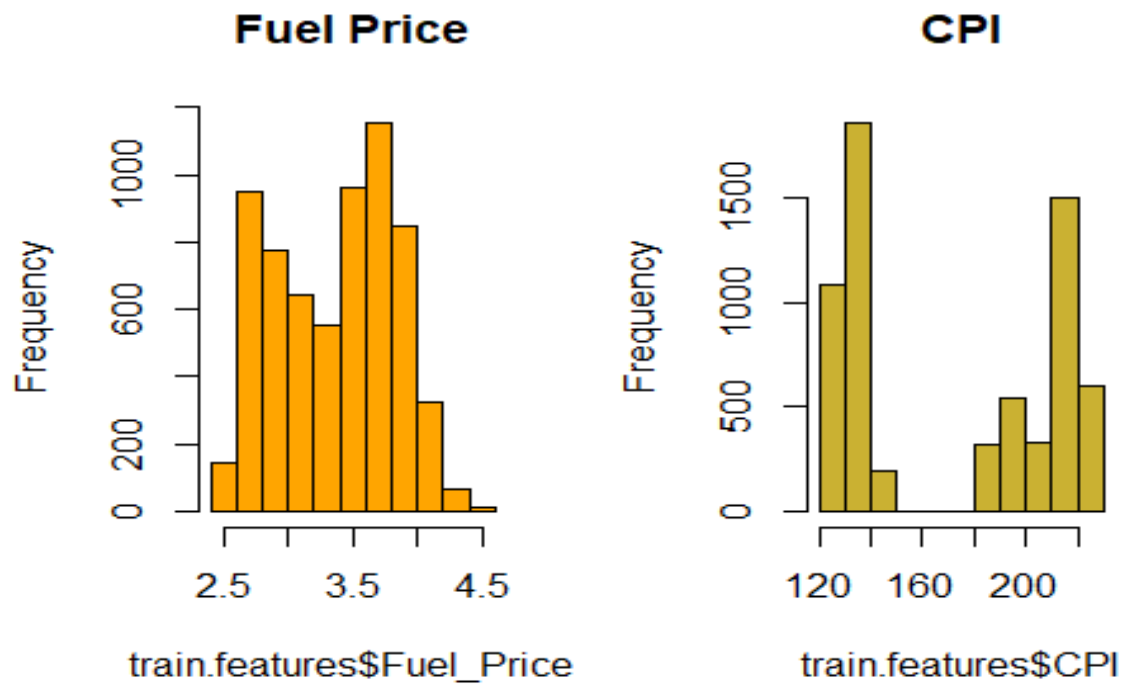
From the results, the relationship between weekly sales and features seems very weak.

Features distribution:

```
par(mfrow=c(1,2))
hist(train.features$Store, col = '#CAB132', main = "Store")
hist(train.features$Temperature, col = 'orange', main = "Temperature")
```



```
hist(train.features$Fuel_Price, col = 'orange', main = "Fuel Price")
hist(train.features$CPI, col = '#CAB132', main = "CPI")
```



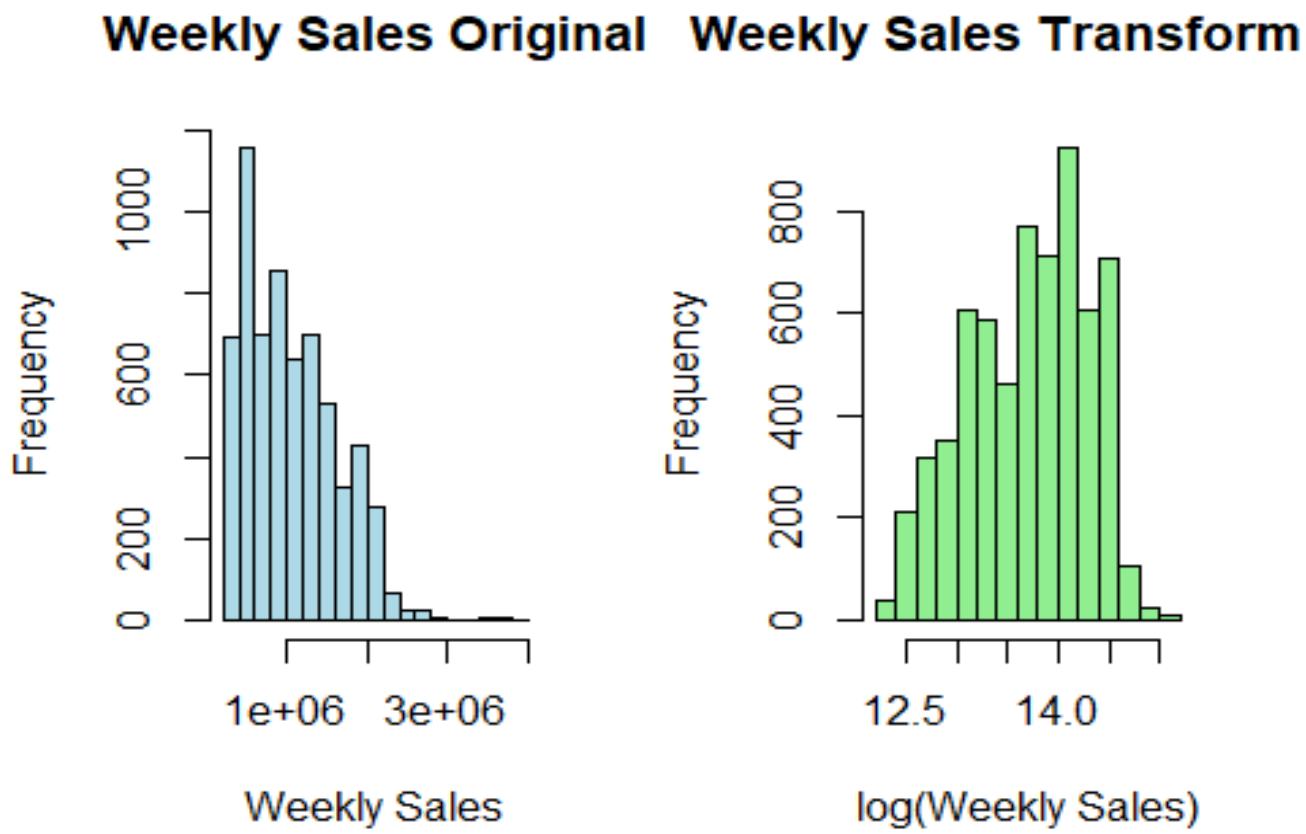

```
hist(train.features$Unemployment, col = 'orange', main = "Unemployment")
```

By looking to the histogram plots, we can observed that Temperature, Fuel Price, and Unemployment features seem normally distributed.

Weekly sales distribution:

```
par(mfrow = c(1,2))
hist(train.features$weekly_sales, col = 'light blue', main = "Weekly Sales
Original", xlab = "Weekly Sales")
hist(log(train.features$weekly_sales), col = 'light green', main = "Weekly
Sales Transformed", xlab ='log(Weekly Sales)')
```

sa

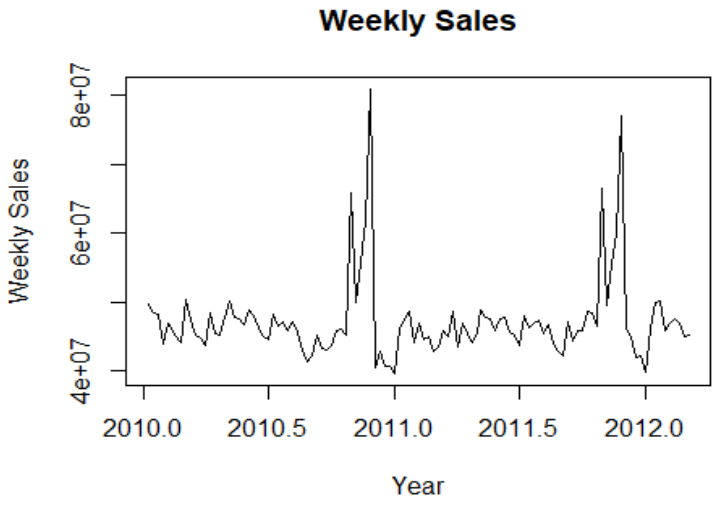


Weekly Sales are positively skewed indicating few large values which could be caused by the holiday Weeks like Christmas and Thanksgiving which are few but Sales value for these weeks would be very high compared to other weeks. However, once we transform data using log, weekly sales seem normally distributed.

Since most of the features have no or little affect on sales except week of the month and holidays, we would first model the sales using Time Series approach as it will capture trend and seasonality of the weekly sales perfectly.

First, cleaning data and converting it to time series. We decide to use the CPI and Unemployment features to fit this model.

```
data <- train.features %>% group_by(Date, Store, Unemployment, Temperature,
CPI) %>% summarise(weekly_sales = sum(weekly_sales))

## `summarise()` regrouping output by 'Date', 'Store', 'Unemployment',
'Temperature' (override with `.groups` argument)

head(data)

## # A tibble: 6 x 6
## # Groups:   Date, Store, Unemployment, Temperature [6]
##   Date       Store Unemployment Temperature   CPI weekly_sales
##   <date>     <int>        <dbl>       <dbl> <dbl>        <dbl>
## 1 2010-02-05     1         8.11        42.3  211.     1643691.
## 2 2010-02-05     2         8.32        40.2  211.     2136989.
## 3 2010-02-05     3         7.37        45.7  214.      461622.
## 4 2010-02-05     4         8.62        43.8  126.     2135144.
## 5 2010-02-05     5         6.57        39.7  212.      317173.
## 6 2010-02-05     6         7.26        40.4  213.     1652635.

sales<- data.frame(Date = data$Date, weekly_sales = data$weekly_sales)
sales <- sales %>% group_by(Date) %>% summarise(weekly_sales =
sum(weekly_sales))

## `summarise()` ungrouping output (override with `.groups` argument)

weekly_sales<-ts(sales[-1], frequency =
52,start=c(2010,02,05),end=c(2012,10,26))
plot(weekly_sales, xlab = 'Year', ylab = 'Weekly Sales', main = 'Weekly
Sales')
```



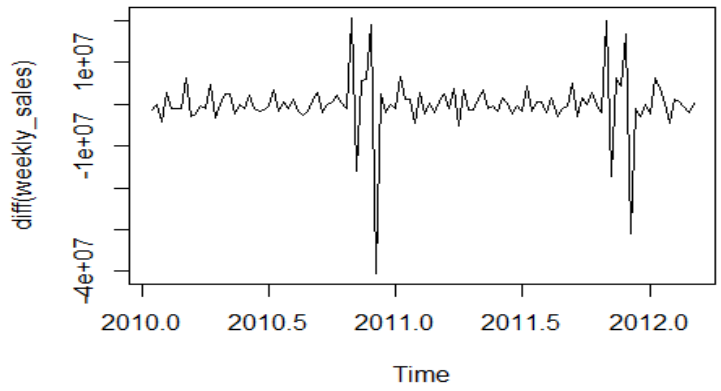

```
cpi.df<- data.frame(Date = data$Date, CPI = data$CPI)
cpi.df <- cpi.df %>% group_by(Date) %>% summarise(CPI = mean(CPI))

## `summarise()` ungrouping output (override with `.groups` argument)
```

```
cpi<- ts(cpi.df[-1], frequency = 52, start = c(2010,02), end = c(2012,10))


unemp.df<- data.frame(Date = data$Date, UnempRate = data$Unemployment)
unemp.df <- unemp.df %>% group_by(Date) %>% summarise(UnempRate =
mean(UnempRate))

## `summarise()` ungrouping output (override with `.groups` argument)

unemp<- ts(unemp.df[-1], frequency = 52, start = c(2010,02), end =
c(2012,10))
```

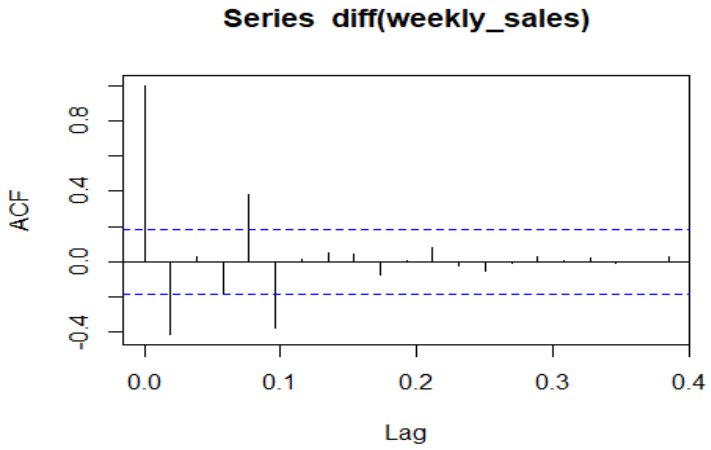
From time series plot, we can see that the weekly sales are not stationary.

Spliting data with 80:20 ratios

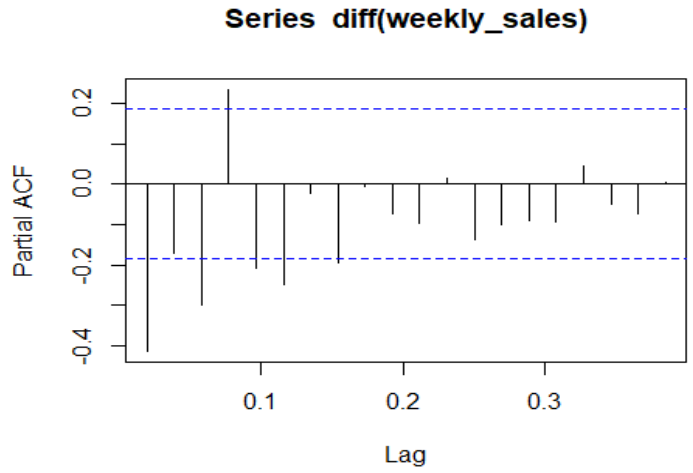
```
numrow<- nrow(sales)
print(numrow)

## [1] 143

trainratio<- nrow(sales)*0.80
print(trainratio)

## [1] 114.4

train.id<- 1:114
test.id<- 115:143
train<- sales[train.id,]
test<- sales[test.id,]

train.cpi<- cpi.df[train.id,]
test.cpi<- cpi.df[test.id,]

train.unemp<- unemp.df[train.id,]
test.unemp<- unemp.df[test.id,]

library(fpp2)
plot(diff(weekly_sales))
```



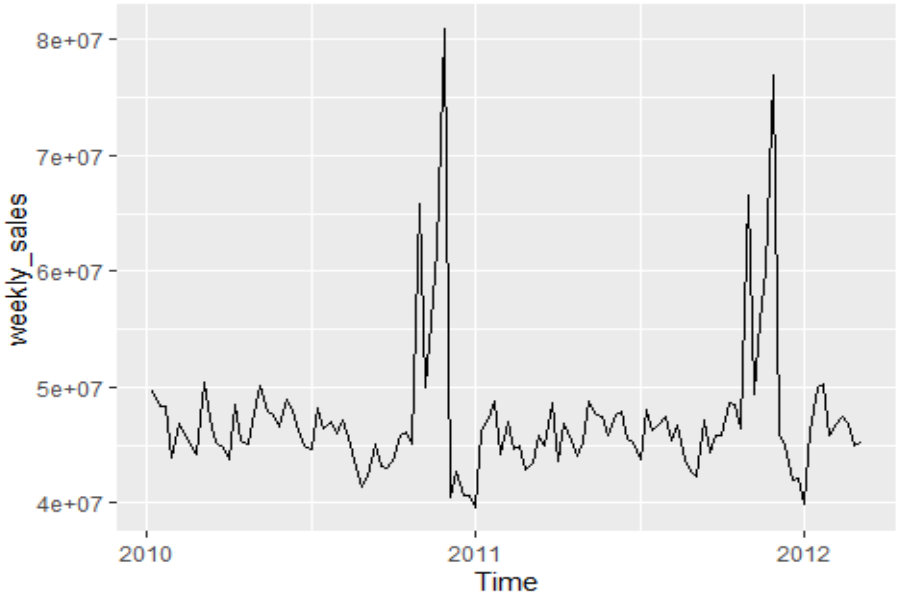

```
acf(diff(weekly_sales))
```

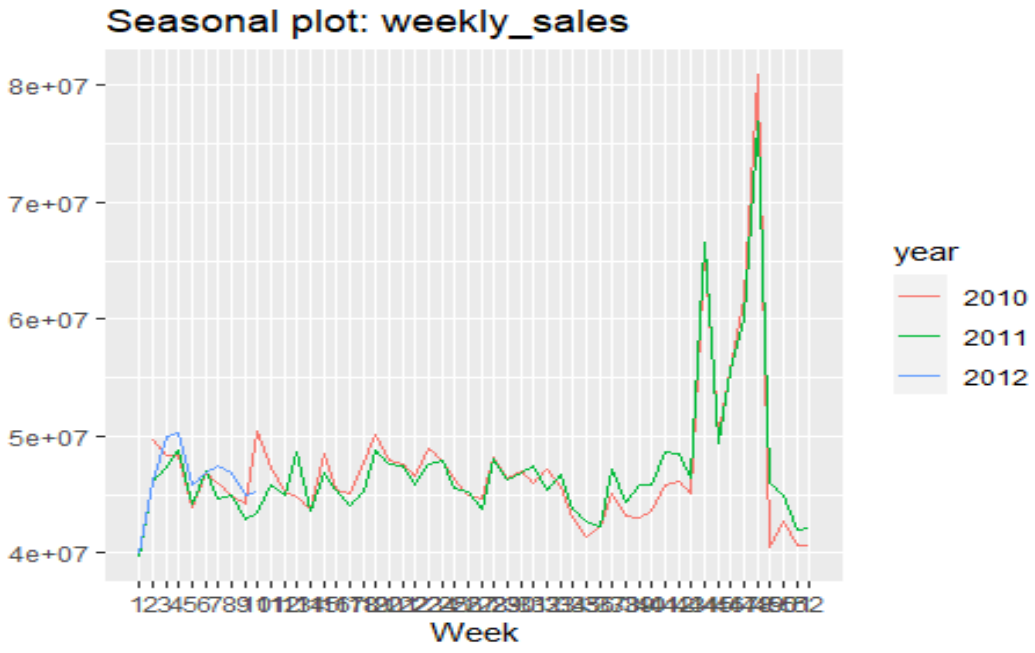



```
pacf(diff(weekly_sales))
```



```
autoplot(weekly_sales);ggseasonplot(weekly_sales)
```





.

Looking at the ACF and PACF plots of the differenced series we see our first significant value at lag 1 for ACF and at the same lag 1 for the PACF which suggest to use p = 1 and q = 1.Since this is a differenced series for SARIMA we set d = 1, and since the seasonal pattern is stable over time we set D = 1. All together this gives us a SARIMA(1,1,1)(0,1,0)[52] model. Next, we run SARIMA with these values to fit a model on our training data.

Model fitting

```
library(forecast)
train.ts<- ts(train[-1], frequency =52, start = c(2010,02), end = c(2012,10))
cpi.ts<- ts(train.cpi[-1], frequency =52, start = c(2010,02), end =
c(2012,10))
unemp.ts<- ts(train.unemp[-1], frequency =52, start = c(2010,02), end =
c(2012,10))
cpi<- lag(as.vector(cpi.ts))
unemp<- lag(as.vector(unemp.ts))
fit.model =
arima(train.ts,c(1,1,1),seasonal=list(order=c(0,1,0),period=52),xreg=cbind(cp
i.ts, unemp.ts))
summary(fit.model)
```

```
##
## Call:
## arima(x = train.ts, order = c(1, 1, 1), seasonal = list(order = c(0, 1,
0),
##     period = 52), xreg = cbind(cpi.ts, unemp.ts))
##
## Coefficients:
##          ar1      ma1    cpi.ts  unemp.ts
##       0.1521  -1.0000  318245.3  -3393329
## s.e.  0.1316   0.0526  438909.0   2330047
##
## sigma^2 estimated as 2.903e+12:  log likelihood = -947.95,  aic = 1905.89
##
## Training set error measures:
##                     ME    RMSE      MAE       MPE     MAPE     MASE
ACF1
## Training set 88219.06 1241728 637860.1 0.2226126 1.329048 0.182953
0.01442809
```

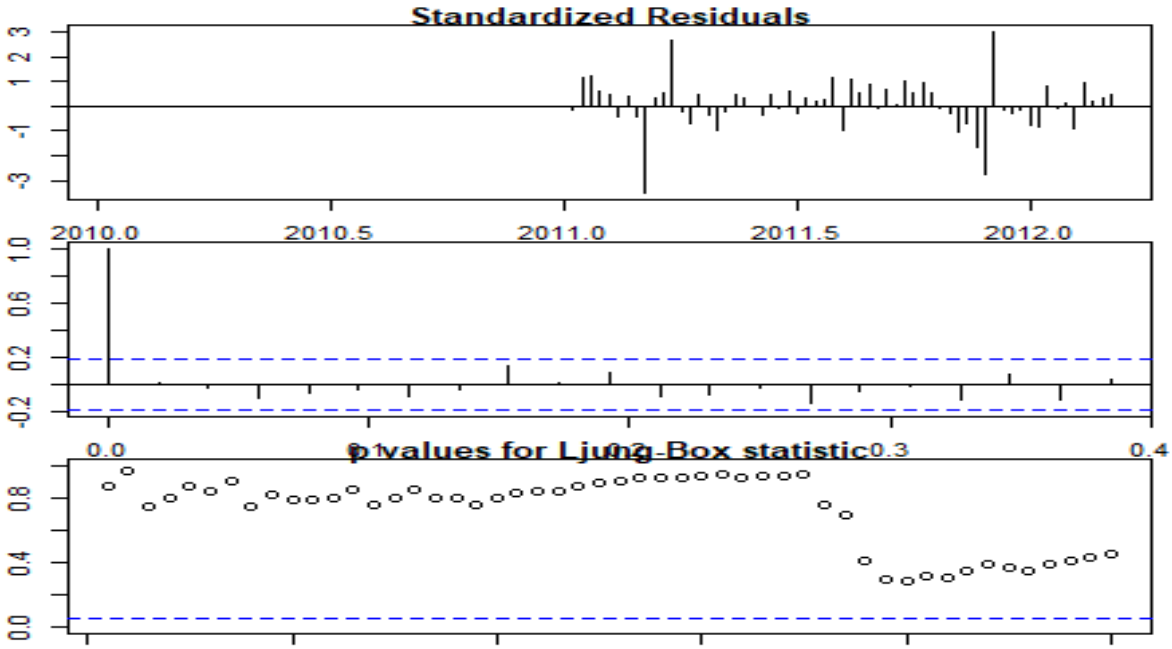
Model diagnosis

```
library(astsa)
```

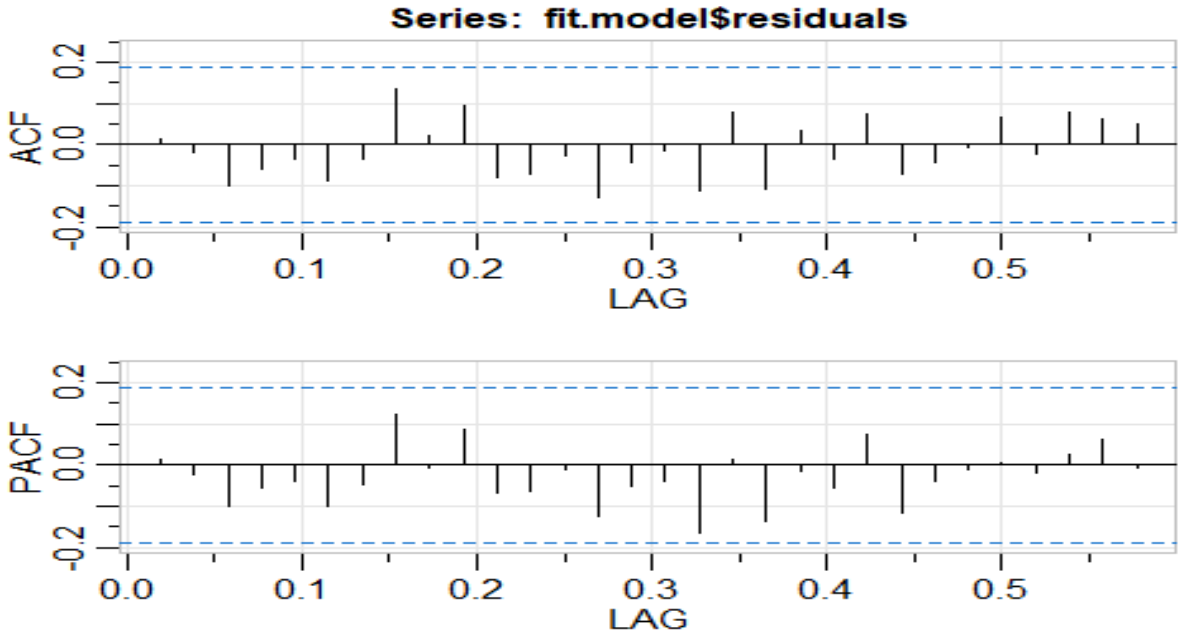
```
##
## Attaching package: 'astsa'

## The following objects are masked _by_ '.GlobalEnv':
##
##     sales, unemp

## The following objects are masked from 'package:fma':
##
##     chicken, sales

## The following object is masked from 'package:forecast':
##
##     gas
```

```
## The following object is masked from 'package:fpp2':
##
##     oil
```

```
par(mfrow=c(1,1),mar=c(1,2,1,4))
tsdiag(fit.model, gof.lat = 50)
```




```
acf2(fit.model$residuals, 30)
```




```
##      [,1]  [,2] [,3]  [,4]  [,5]  [,6]  [,7] [,8]  [,9] [,10] [,11] [,12]
[,13]
## ACF  0.01 -0.02 -0.1 -0.06 -0.04 -0.09 -0.03 0.13  0.02  0.10 -0.08 -0.07
```

```
-0.03
## PACF 0.01 -0.02 -0.1 -0.06 -0.04 -0.10 -0.05 0.12 -0.01  0.09 -0.07 -0.06
-0.01
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
[,25]
## ACF  -0.13 -0.04 -0.01 -0.11  0.08 -0.11  0.03 -0.04  0.08 -0.07 -0.04 -
0.01
## PACF -0.12 -0.05 -0.04 -0.16  0.01 -0.14 -0.02 -0.06  0.08 -0.12 -0.04 -
0.01
##      [,26] [,27] [,28] [,29] [,30]
## ACF   0.07 -0.02  0.08  0.06  0.05
## PACF  0.01 -0.02  0.03  0.06 -0.01
```

From the standardized residuals, the plot pattern is not clear, but it seems to be white noise process. The ACF of residuals plot indicates that all the lags are significant.The p-values are above 0, indicating model fits.

Model prediction

```
mod.pred<- fitted(fit.model)
print(mod.pred)

## Time Series:
## Start = c(2010, 2)
## End = c(2012, 10)
## Frequency = 52
##   [1] 49735949 48331181 48273459 43969746 46869793 45924882 44989390
44135106
##   [9] 50419268 47364073 45184628 44735753 43707307 48500628 45330704
45120935
##  [17] 47755757 50184569 47825120 47620894 46608918 48915334 47898458
46244497
##  [25] 44890772 44632487 48203556 46464758 47060694 45910623 47193847
45635536
##  [33] 43084354 41363748 42243950 45104127 43152538 43069751 43605320
45782272
##  [41] 46124761 45126518 65801439 49905620 55657724 61805863 80897905
40439803
##  [49] 42781239 40681125 40662035 39626460 46351559 45394284 46640056
43073205
##  [57] 46187809 45292527 44201055 43587776 49439709 45267711 44047207
44067182
##  [65] 43884665 48038251 44658761 44654928 46988867 49097022 46854788
46891641
##  [73] 45930264 48121256 47059699 45635901 44192758 44117594 47491836
45883641
##  [81] 46508317 45423631 46981306 44933835 42893307 41180886 42373313
46042036
##  [89] 44311358 44120660 44930623 47072838 47548426 46525501 67104348
51131052
##  [97] 56706632 62907318 81706023 40895909 45141203 42465040 42268281
```

```
41174086
## [105] 47483665 48584526 50372570 45587770 48440637 45804864 46505337
44371621
## [113] 44481029
```

From the prediction, it appears that the weekly sales are not stable with seasonality component since the sales are generally high at the end of every year.

Model evaluation

```
library(forecast)
library(MLmetrics)
test.ts<- ts(test[-1], frequency =52, start = c(2010,02), end = c(2012,10))
cpi.ts<- ts(test.cpi[-1], frequency =52, start = c(2010,02), end =
c(2012,10))
unemp.ts<- ts(test.unemp[-1], frequency =52, start = c(2010,02), end =
c(2012,10))
cpi<- lag(as.vector(cpi.ts))
unemp<- lag(as.vector(unemp.ts))
fit = arima(test.ts/1000,c(1,1,1),seasonal=list(order=c(0,1,0), period =
52),xreg=cbind(cpi, unemp))
summary(fit)

##
## Call:
## arima(x = test.ts/1000, order = c(1, 1, 1), seasonal = list(order = c(0,
1,
##     0), period = 52), xreg = cbind(cpi, unemp))
##
## Coefficients:
##          ar1    ma1       cpi     unemp
##       0.3170  -1.00  2848.410  8831.368
## s.e.  0.1492   0.06  1731.181  3951.984
##
## sigma^2 estimated as 6248311:  log likelihood = -547.05,  aic = 1104.11
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
ACF1
## Training set 156.3933 1848.713 1116.892 0.2685824 2.393392 0.7621685
0.07037405

onestep <- fitted(fit)
print(onestep)

## Time Series:
## Start = c(2010, 2)
## End = c(2012, 10)
## Frequency = 52
##   [1]       NA 45372.94 43852.27 47210.09 46990.16 46875.39 47934.29
48317.08
```

```
##   [9] 49680.83 48439.61 47693.72 46621.31 51270.79 46119.60 46078.08
44116.37
##  [17] 47500.64 47417.42 47367.71 47459.94 47172.15 48341.09 44240.83
44368.83
##  [25] 43749.57 47577.33 46138.11 45133.01 45553.97 46637.70 45083.02
43728.38
##  [33] 47132.15 46933.88 46831.90 47899.15 48287.73 49655.70 48417.71
47674.47
##  [41] 46604.26 51255.45 46105.66 46065.43 44104.86 47490.10 47407.66
47358.63
##  [49] 47451.48 47164.25 48333.67 43889.13 43893.12 40036.64 43565.91
46930.84
##  [57] 45969.52 45396.74 46646.96 47711.25 47896.18 45717.54 46598.38
45768.47
##  [65] 50599.29 45829.79 47152.09 45787.55 49427.18 47990.35 47714.23
47634.63
##  [73] 45986.93 46990.82 41789.91 44304.42 43486.18 47469.77 47300.73
45828.55
##  [81] 47181.66 46951.80 44502.99 43909.59 46696.23 45925.44 45419.11
46659.14
##  [89] 47869.69 48381.50 46315.86 47115.90 46228.43 51093.67 46322.72
47514.90
##  [97] 46030.14 49645.20 48262.01 47945.17 47827.60 46197.95 47235.74
41995.83
## [105] 44399.78 43500.83 47442.43 47275.52 45781.60 47138.98 46957.54
44536.26
## [113] 43920.32
```

From the summary, we can see that the MAPE of this model is equal to 2.39 which means the prediction is off by 2.39%.

Walmart Sales Prediction using Prophet

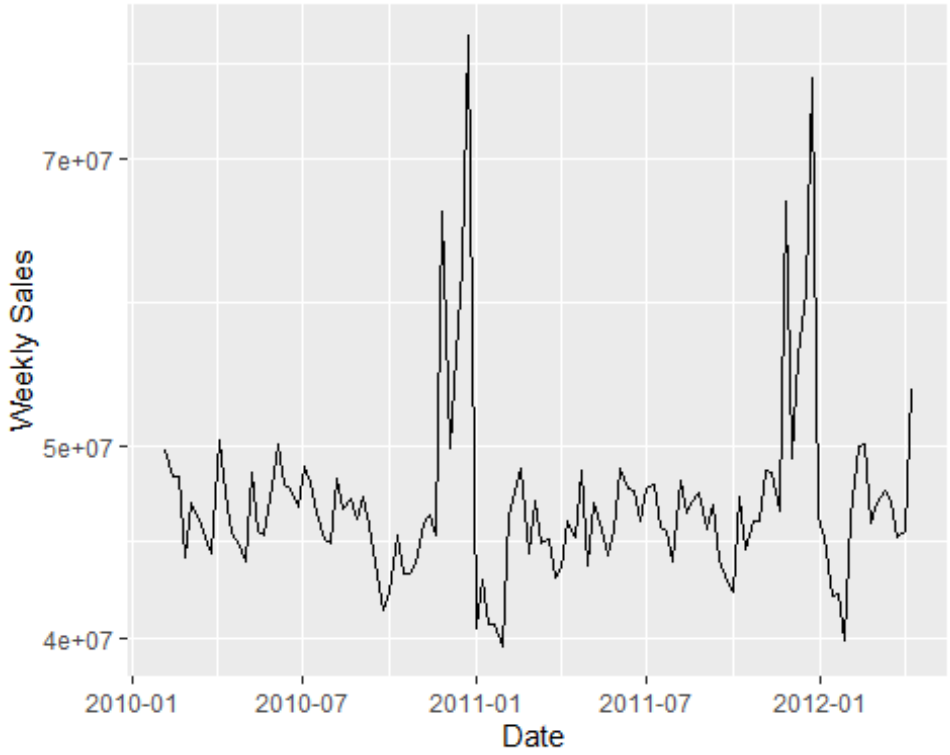
Creating a new dataframe including ds and y variables

```
Walmart_sales<- data.frame(ds = train$Date, y = train$weekly_sales)
head(Walmart_sales)
```

```
##           ds        y
## 1 2010-02-05 49750741
## 2 2010-02-12 48336678
## 3 2010-02-19 48276994
## 4 2010-02-26 43968571
## 5 2010-03-05 46871470
## 6 2010-03-12 45925397
```

Checking data

```
library(ggplot2)
ggplot(data = Walmart_sales, aes(x = ds, y = y)) + geom_line()+
scale_y_log10()  +xlab('Date ') + ylab('Weekly Sales')
```



From the plot, we can observe that the weekly sales are not constant with seasonality component since sales are high at the end of the year.As a result, it can be observed that the series are not stationary.

Model fitting

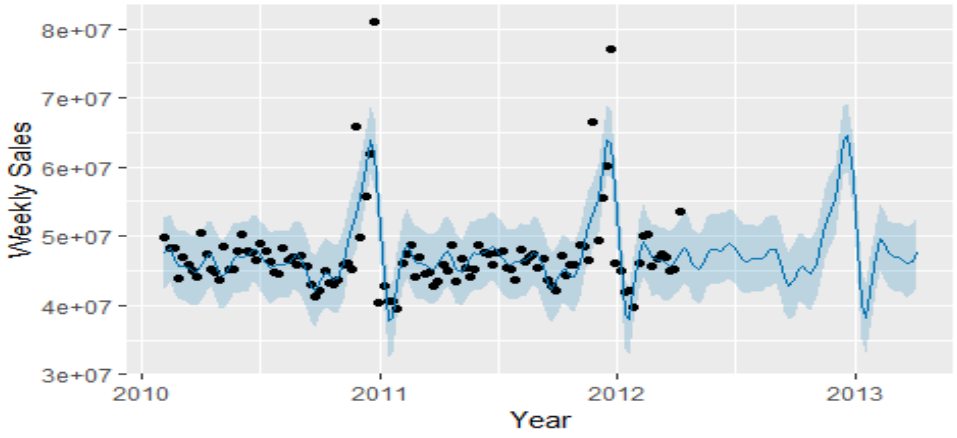
Using Prophet model to fit this data

```
library(prophet)

## Loading required package: Rcpp

## Loading required package: rlang

model <- prophet(Walmart_sales, changepoint.prior.scale=0.01,
weekly.seasonality = TRUE, daily.seasonality = FALSE)
```

Model Forecast

```
future <- make_future_dataframe(model, periods = 52, freq = 'week')
fcst <- predict(model, future)
tail(fcst[c('ds','yhat','yhat_lower', 'yhat_upper')])

##              ds     yhat yhat_lower yhat_upper
## 161 2013-03-01 46937253   41980598   51744064
## 162 2013-03-08 46912730   41963742   51850129
## 163 2013-03-15 46569109   42007053   51763139
## 164 2013-03-22 46081012   40945518   51295808
```

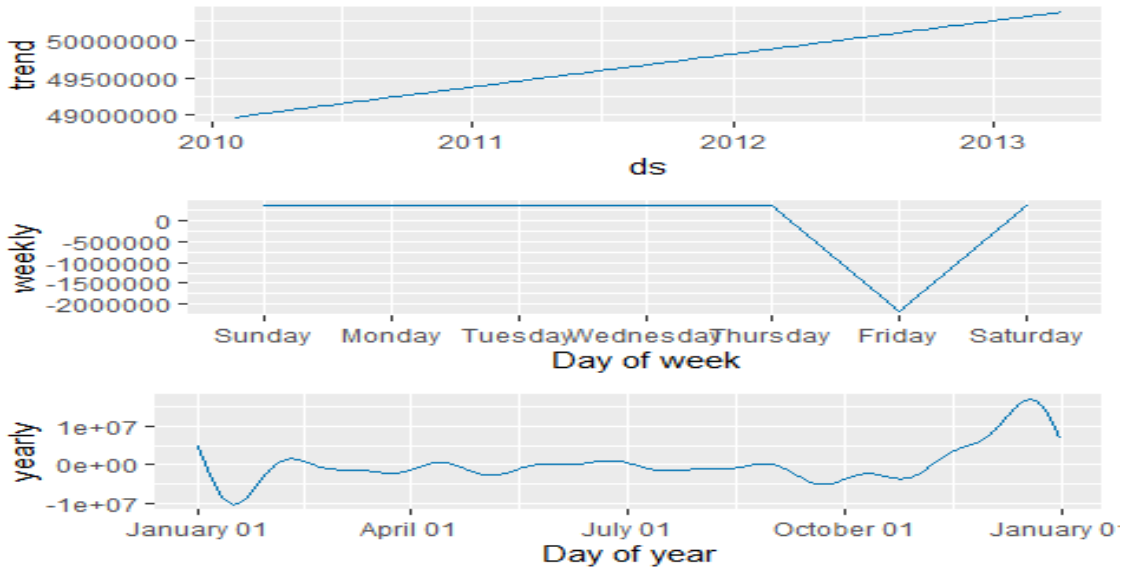
```
## 165 2013-03-29 46423722   41716780   51827812
## 166 2013-04-05 47736301   42750608   52601763

plot(model, fcst, xlab = 'Year', ylab = 'Weekly Sales')
```



The prediction plot indicates seasonality and suggests that the Walmart sales seem to be the same pattern as the history data, which are high at the end of the year.

Model Diagnosis:

```
prophet_plot_components(model, fcst)
```



As you can see from the trend graph, Prophet did a good job by fitting the accelerated growth of new sales. The graph of weekly seasonality leads to the conclusion that usually there are less sales on Friday than on the other days of the week. In the yearly seasonality graph there is a prominent dip on Christmas Day.

Model Evaluation:

```
test_sales<- data.frame(ds = test$Date, y = test$weekly_sales)
test_sales.ts<- ts(test_sales$y, frequency =52, start = c(2010,02), end =
c(2012,10))
y_true<- as.vector(test_sales.ts)[1:81]
test.mod <- prophet(test_sales, changepoint.prior.scale=0.01,
weekly.seasonality = TRUE)

## Disabling yearly seasonality. Run prophet with yearly.seasonality=TRUE to
override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to
override this.

## n.changepoints greater than number of observations. Using 22

future <- make_future_dataframe(test.mod, periods = 52, freq = 'week')
forecast <- predict(test.mod, future)
onestep <- forecast$yhat
mspe<- mean((y_true - onestep)^2)
mape <- MAPE(y_true, onestep)
sprintf('%s = %3.2f', 'MSPE of this model', mspe)

## [1] "MSPE of this model = 5780126914621.07"

sprintf('%s = %3.4f', 'MAPE of this model', mape)

## [1] "MAPE of this model = 0.0429"
```

From the result, the MSPE is a very large value since the data are large; however, we can observe that the MAPE of this model is equal to 0.0429 which means the prediction is off by 4.29% on average.

Summary

After we predict weekly sales of Walmart using three different methods, it can be concluded that the most accurate model is the model that we fit by using SARIMA without any additional features. This would be concluded that the Walmart weekly sales depend on seasonality component since most of the features have no or little affect on sales except week of the month and holidays. Also, the prophet model is based on an additive model and works best with daily periodicity; therefore, this method may not work well for these data.