Rossmann Store Sales Prediction Problem

Shan Chen

Nov 27, 2015

Introduction

This is a pretty interesting problem cause I always wonder waht can be the relatively vital effects to a store daily sales? My first tuitive on this I think might be weekends to be #1 and then possibily distance to competitors and then promotions (as far as I'm considered I was always "couraged" to buy those "buy one, get one" stuff), well, things would never be that easy.. So here we go, let's focus on the data and example to have an overall view on this prediction problem.

I'll basically cover the following procedures and those are kinda my thought process.

1. Feature Engineering and EDA

- 1.1 Data Preparation
- 1.2 Customers Vs. Sales
- 1.3 Open, StateHoliday Vs. Sales
- 1.4 DayOfWeek Vs. Sales
- 1.5 Date Vs. Sales
- 1.6 Competition Vs. Sales
- 1.7 Promotion Vs. Sales

2. Modeling and Prediction

- 2.1 Random Forest
- 2.2 Linear Model
- 2.3 Prediction

1. Feature Engineering and EDA

1.1 Data Preparation

Loading packages and data set

```
library('party')
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

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

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

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

```
## Loading required package: sandwich
```

```
library('rpart')
library('randomForest') # classification algorithm
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library('ggplot2') # Visualization
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
library('sqldf')
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
## Loading required package: RSQLite
```

```
## Loading required package: DBI
```

```
library('dplyr') # Data munipulation
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

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

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

```
library('lubridate') # Month variable extraction
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library('zoo')
library('mice') # imputation
```

```
## Loading required package: Rcpp
```

```
## mice 2.25 2015-11-09
```

```
train<-read.csv("C:\\Users\\Shan\\Desktop\\Rossmann Store Sales Prediction\\train.csv")
store<-read.csv("C:\\Users\\Shan\\Desktop\\Rossmann Store Sales Prediction\\store.csv")
test<-read.csv("C:\\Users\\Shan\\Desktop\\Rossmann Store Sales Prediction\\test.csv")
```

train(test) and store are two related data sets with the same store id column, here let’s use `sqldf()` to horizontally join them to get a complete set.

```
#first vertically bind train and test sets to get a complete set
complete<- bind_rows(train,test)
```

```
## Warning in rbind_all(x, .id): Unequal factor levels: coercing to character

## Warning in rbind_all(x, .id): Unequal factor levels: coercing to character
```

```
complete<- sqldf("select complete.*,store.* from complete
              inner join store on complete.store=store.store")
```

```
## Loading required package: tcltk
```

```
dim(train)
```

```
## [1] 1017209       9
```

```
dim(test)
```

```
## [1] 41088     8
```

```
dim(store)
```

```
## [1] 1115   10
```

```
complete<-complete[,-11]
train<-complete[1:1017209,]
test<-complete[1017210:1058279,]
dim(complete)
```

```
## [1] 1058297      19
```

```
head(complete)
```

```
##   Store DayOfWeek      Date Sales Customers Open Promo StateHoliday
## 1     1         5 7/31/2015  5263       555    1     1            0
## 2     2         5 7/31/2015  6064       625    1     1            0
## 3     3         5 7/31/2015  8314       821    1     1            0
## 4     4         5 7/31/2015 13995      1498    1     1            0
## 5     5         5 7/31/2015  4822       559    1     1            0
## 6     6         5 7/31/2015  5651       589    1     1            0
##   SchoolHoliday Id StoreType Assortment CompetitionDistance
## 1             1 NA         c          a                1270
## 2             1 NA         a          a                 570
## 3             1 NA         a          a               14130
## 4             1 NA         c          c                 620
## 5             1 NA         a          a               29910
## 6             1 NA         a          a                 310
##   CompetitionOpenSinceMonth CompetitionOpenSinceYear Promo2
## 1                         9                     2008      0
## 2                        11                     2007      1
## 3                        12                     2006      1
## 4                         9                     2009      0
## 5                         4                     2015      0
## 6                        12                     2013      0
##   Promo2SinceWeek Promo2SinceYear   PromoInterval
## 1              NA              NA
## 2              13            2010 Jan,Apr,Jul,Oct
## 3              14            2011 Jan,Apr,Jul,Oct
## 4              NA              NA
## 5              NA              NA
## 6              NA              NA
```

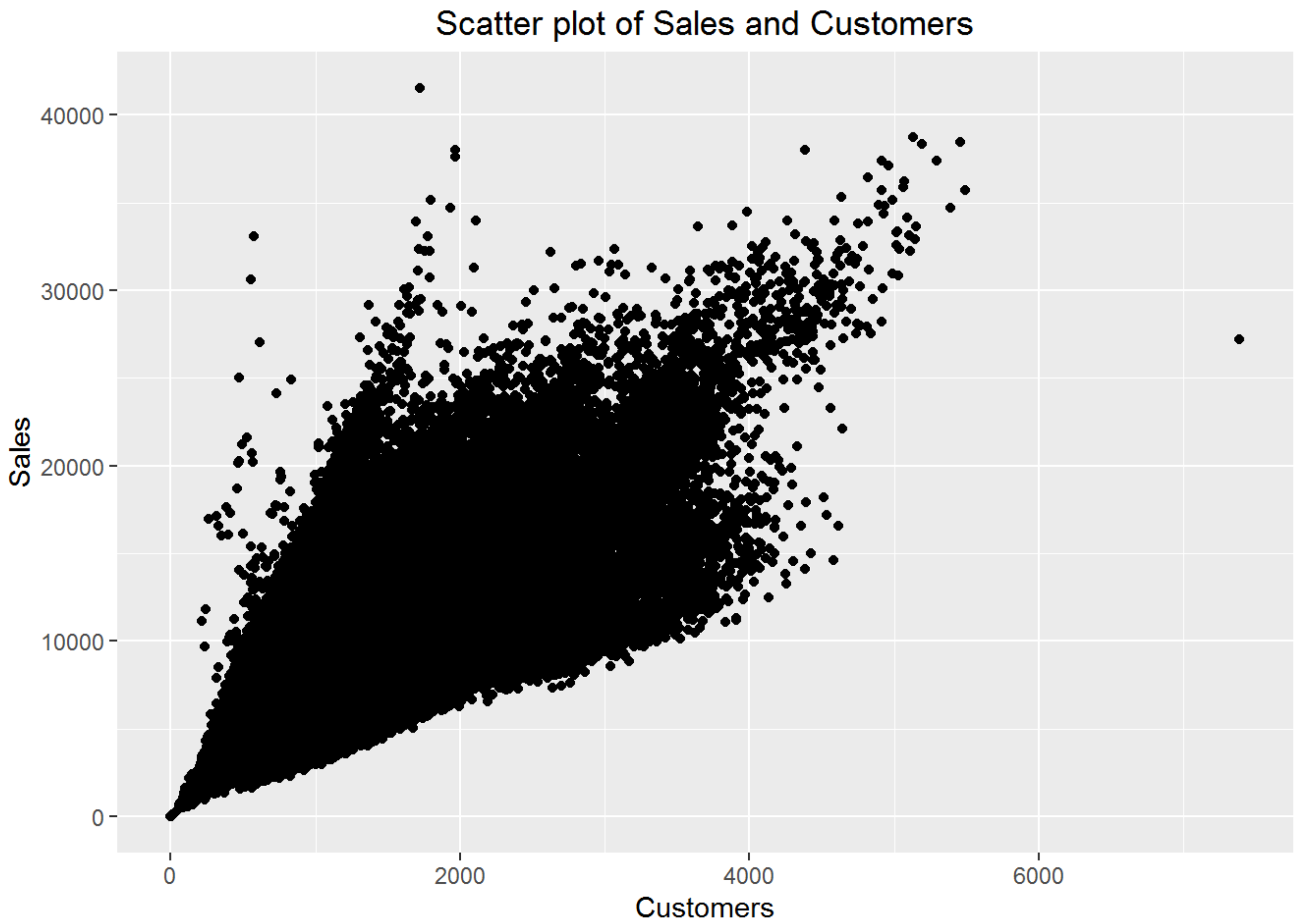
```
str(complete)
```

```
## 'data.frame':	1058297 obs. of  19 variables:
##  $ Store                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ DayOfWeek                : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Date                     : chr  "7/31/2015" "7/31/2015" "7/31/2015" "7/31/2015"
...
##  $ Sales                    : int  5263 6064 8314 13995 4822 5651 15344 8492 8565 71
85 ...
##  $ Customers                : int  555 625 821 1498 559 589 1414 833 687 681 ...
##  $ Open                     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Promo                    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ StateHoliday             : chr  "0" "0" "0" "0" ...
##  $ SchoolHoliday            : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Id                       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ StoreType                : Factor w/ 4 levels "a","b","c","d": 3 1 1 3 1 1 1 1 1
1 ...
##  $ Assortment               : Factor w/ 3 levels "a","b","c": 1 1 1 3 1 1 3 1 3 1
...
##  $ CompetitionDistance      : int  1270 570 14130 620 29910 310 24000 7520 2030 3160
...
##  $ CompetitionOpenSinceMonth: int  9 11 12 9 4 12 4 10 8 9 ...
##  $ CompetitionOpenSinceYear : int  2008 2007 2006 2009 2015 2013 2013 2014 2000 2009
...
##  $ Promo2                   : int  0 1 1 0 0 0 0 0 0 0 ...
##  $ Promo2SinceWeek          : int  NA 13 14 NA NA NA NA NA NA NA ...
##  $ Promo2SinceYear          : int  NA 2010 2011 NA NA NA NA NA NA NA ...
##  $ PromoInterval            : Factor w/ 4 levels "","Feb,May,Aug,Nov",..: 1 3 3 1 1
1 1 1 1 1 ...
```

Dimensionality for complete passed through our checking, next let’s consider those predictors.

1.2 Customers Vs. Sales

```
ggplot(train, aes(x=Customers, y=Sales)) + geom_point()+ggtitle("Scatter plot of Sales
and Customers")
```



No doubt that Customers is one perfect predictor for sales, however since we can not know the future Customers, I'm not using Customers in the prediction.

1.3 Open, StateHoliday Vs. Sales

```
a=table(complete$Open, complete$StateHoliday)
addmargins(a)
```

```
##
##             0       a       b       c     Sum
##   0    148507   19720    6545    4029  178801
##   1    878549     720     145      71  879485
##   Sum 1027056   20440    6690    4100 1058286
```

There are 178801 out of 1058286 obs are closed so have no sales, it may due to holiday or for other private reasons. Hereby I will not include those obs with closed days. Though mostly all of the stores would be closed on the holiday (what a life!), there are still some rare stores keep openning, let's do a boxplot of Sales on StateHoliday.

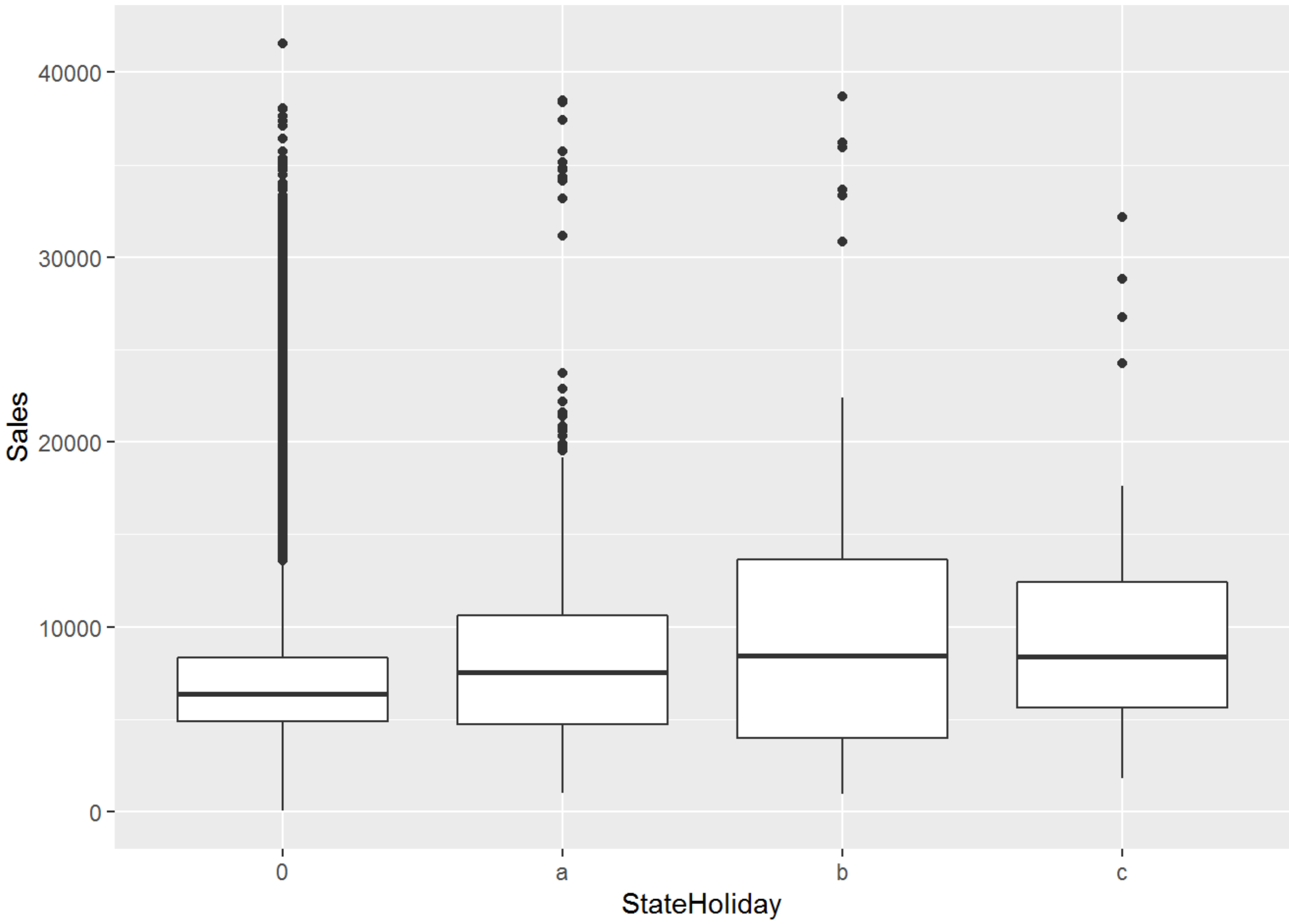
```
#Delete rows of Sales="0"
train<-sqldf("select * from train where Sales!=0 ")
test<-sqldf("select * from test where Open=1 ")
complete<- bind_rows(train,test)
dim(train)
```

```
## [1] 844338     19
```

```
dim(test)
```

```
## [1] 35075    19
```

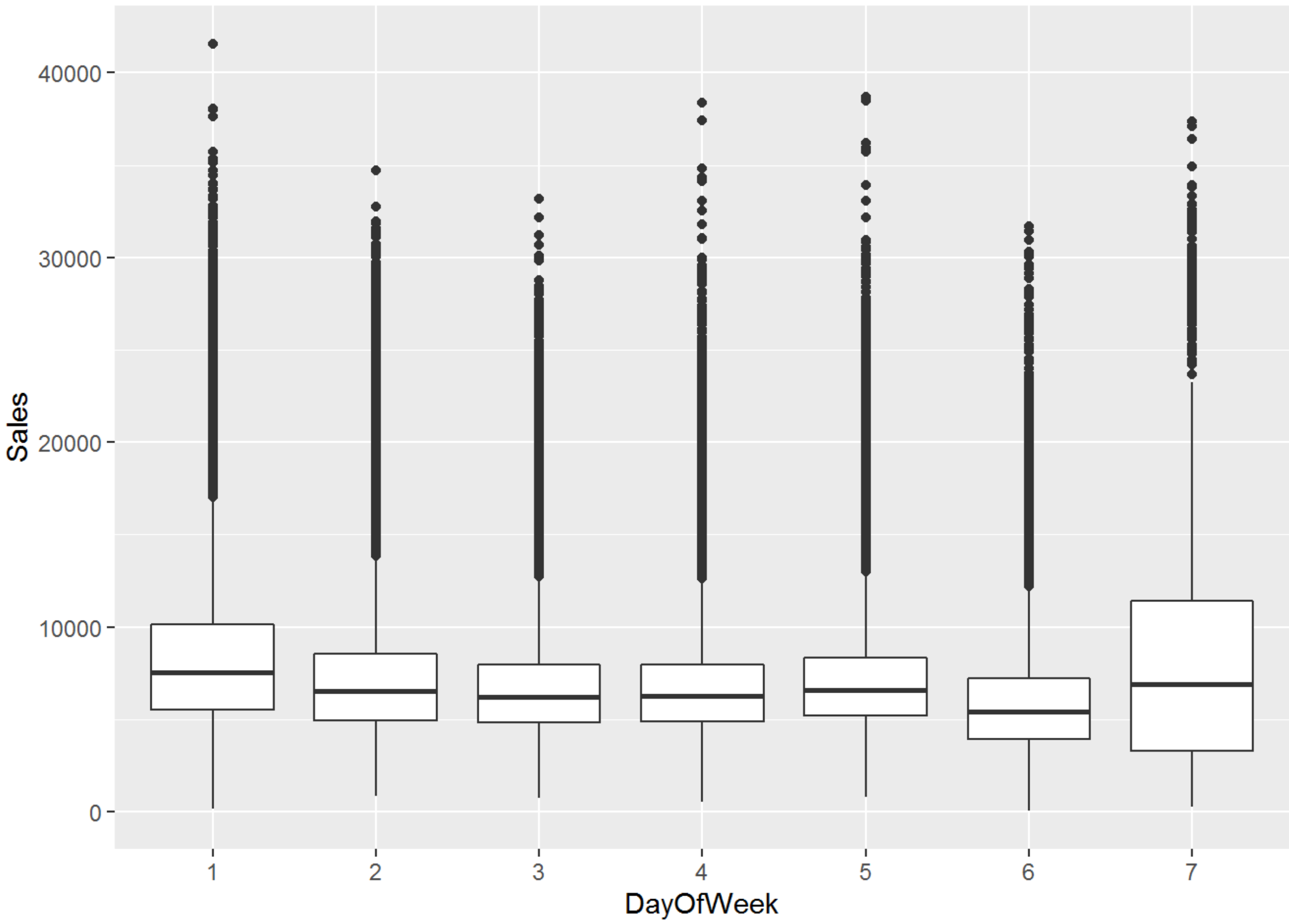
```
#Subset complete to new train and test sets
train<-complete[1:844338,]
test<-complete[844339:879413,]
#Factorize StateHoliday Variable
complete$SchoolHoliday <- as.factor(complete$SchoolHoliday)
complete$StateHoliday<-as.factor(complete$StateHoliday)
#Boxplots
Sales_vs_StateHoliday <- ggplot(train, aes(x=StateHoliday, y=Sales)) + geom_boxplot()
Sales_vs_StateHoliday
```



It seems that Sales are higher on holidays.

1.4 DayOfWeek Vs. Sales

```
#Factorize DayOfWeek Variable
complete$DayOfWeek<-as.factor(complete$DayOfWeek)
#Boxplots
Sales_vs_DayOfWeek <- ggplot(complete[1:844338,], aes(x=DayOfWeek, y=Sales)) + geom_box
plot()
Sales_vs_DayOfWeek
```




```
#total_Counts for DayOfWeek
total_Counts<-sqldf("select DayOfWeek, count(*) from train group by DayOfWeek order by
DayOfWeek")
total_Counts
```

```
##   DayOfWeek count(*)
## 1         1   137557
## 2         2   143955
## 3         3   141922
## 4         4   134626
## 5         5   138633
## 6         6   144052
## 7         7     3593
```

From `total_Counts` lists, the number of obs for Sunday is much lower than the other week days possibily due to Germany tend to close all the business on Sunday, but the sales tend to be higher, same with Moday which is quite interesting.

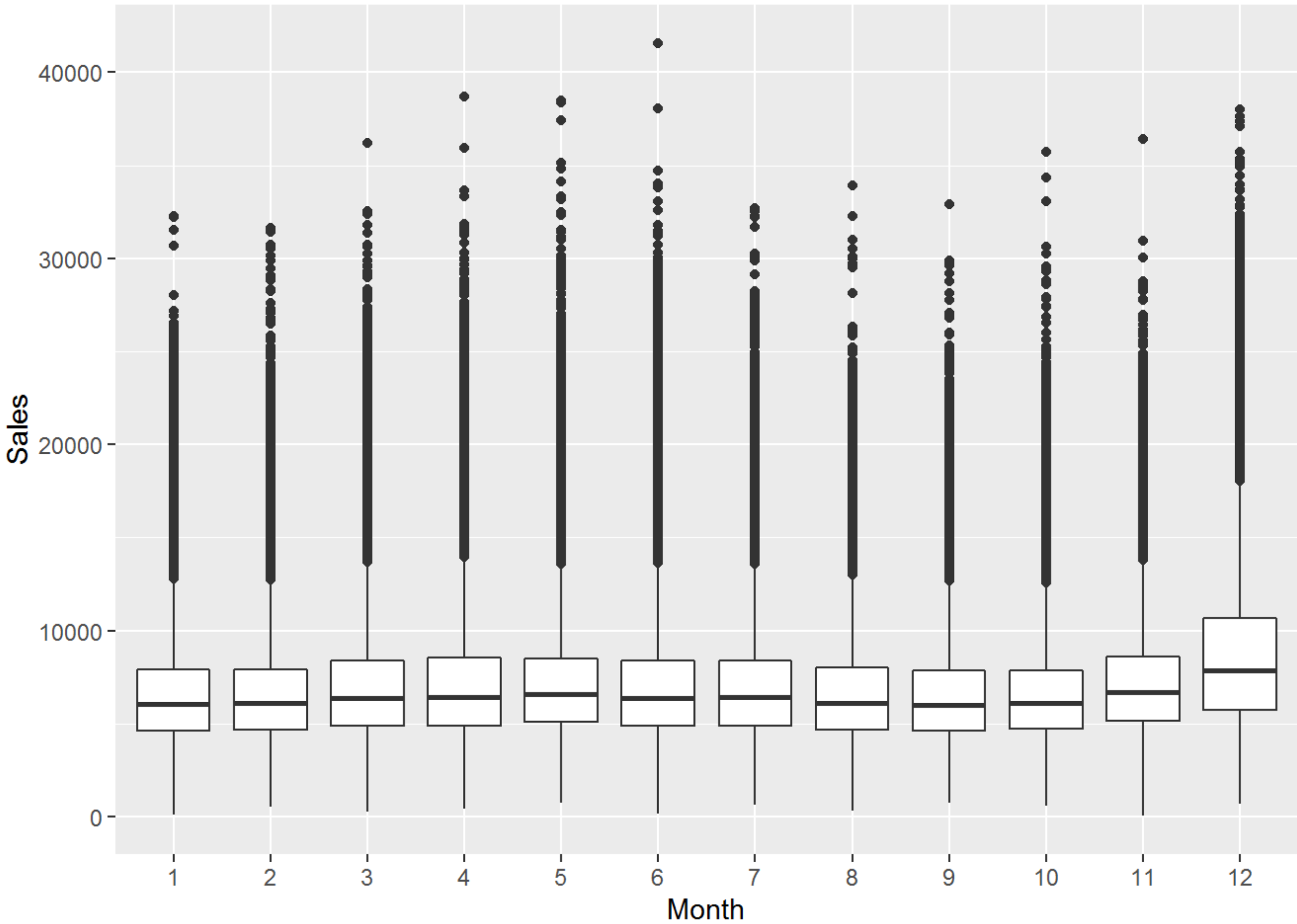
1.5 Date Vs. Sales

For variable of Date, I would like to extract the month imformation out (since DayOfWeek is preferred than the day information from Date and year is too overall for our daily sales prediction).

```
#as.Date Date variable
complete$Date=as.Date(complete$Date,format = "%m/%d/%Y")
complete=complete %>%
  filter%>%
  mutate(Month = lubridate::month(Date))
```

Boxplots to show effects of new added variable Month on Sales

```
#Factorize Month Variable
complete$Month<-as.factor(complete$Month)
#Boxplots
Sales_vs_Month <- ggplot(complete[1:844338,], aes(x=Month, y=Sales)) + geom_boxplot()
Sales_vs_Month
```



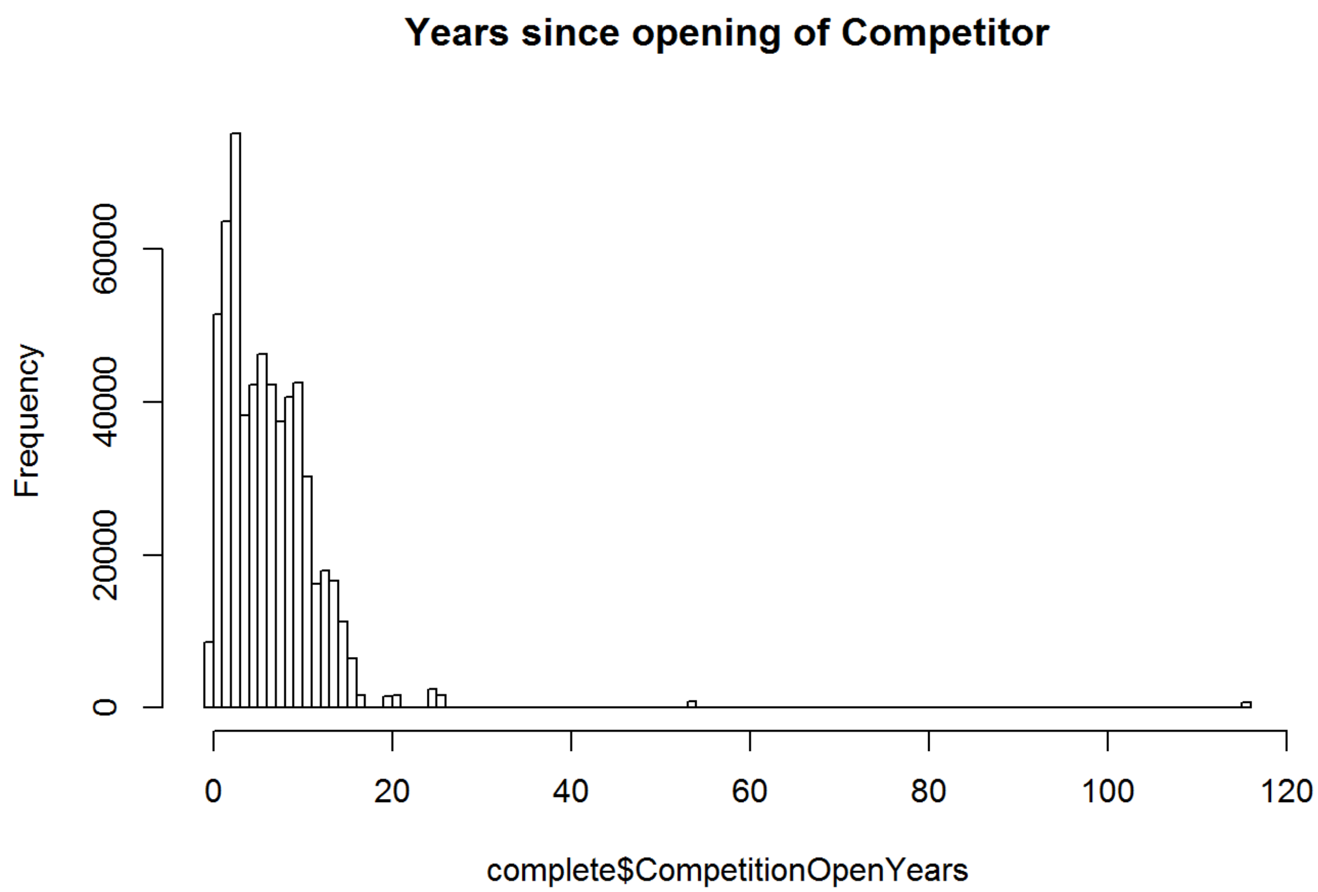
It seems Dec has the highest Sales, it's obvious that people all love to shopping at the end of the year.

1.6 Competition Vs. Sales

Competition relationship at my first sight should be one important factor to our prediction, stores with rare(too far away) competitors may have a relatively higher sales than the others, but the sales possibily would not be that high due to also lack of population of residence around…It could be a complicated point to analyze thoughly, maybe we should combine the number of customers and competiton information to show a kinda significant factor variable.

First let's calculate years since opening of the competitor store.

```
complete$CompetitionOpenYears <- as.yearmon("2015-07-31") - as.yearmon(paste(complete$C
ompetitionOpenSinceYear,complete$CompetitionOpenSinceMonth, sep = "-"))
#Histogram
hist(complete$CompetitionOpenYears,breaks=100,main = "Years since opening of Competito
r")
```




```
#many missing values
summary(complete$CompetitionOpenYears)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   -0.08    2.42    5.42    6.29    9.17  115.50  281598
```

```
#Impute N/A's with mean
complete[is.na(complete$CompetitionOpenYears), c("CompetitionOpenYears")] = mean(comple
te$CompetitionOpenYears,na.rm = TRUE)
```

Then go to Competition$Distance part

```
summary(complete$CompetitionDistance)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       20     710    2330    5446    6880   75860    2251
```

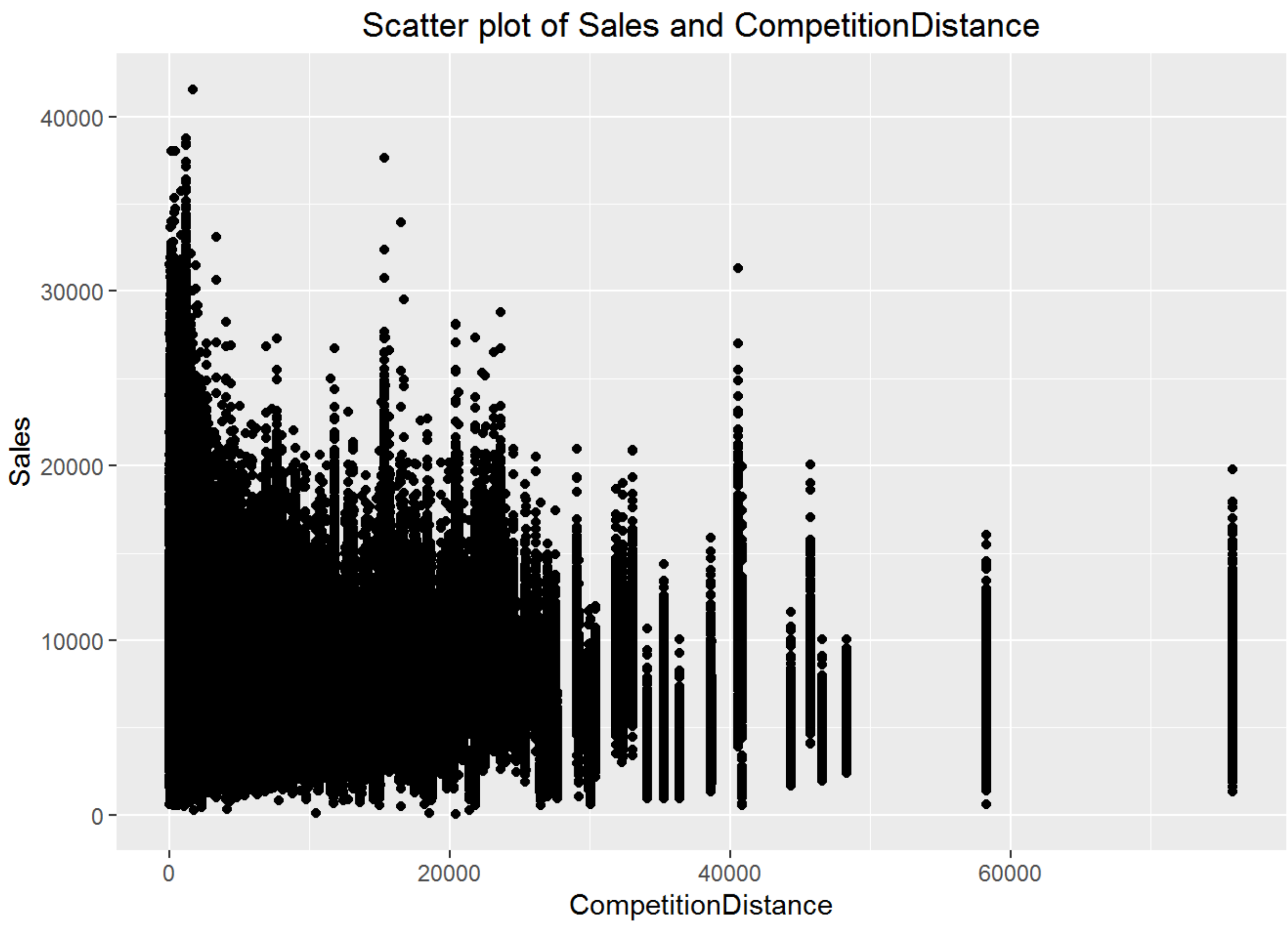
```
#Three stores have missing CompetitionDisrance value
```

Impute the N/A CompetitionDistance observations with the mean of CompetitionDistance observations.

```
complete[is.na(complete$CompetitionDistance), c("CompetitionDistance")] = mean(complete
$CompetitionDistance,na.rm = TRUE)
```

Scatterplot:

```
ggplot(complete[1:844338,], aes(x=CompetitionDistance, y=Sales)) + geom_point()+ggtitle
("Scatter plot of Sales and CompetitionDistance")
```



It’s hard to describe the pattern of the two variables, but we can still tell that **those highest Daily Sales did happen on the stores with the nearest competitor**, which is not what I thought anout the competiton effects(nearer the competitor is, lower the Sales would be), so such situation may happen due to the corresponding crowded population around.

one way I’ll use to try to split the effects is to split the population crowd to four levels through **discreting Customers variable**.

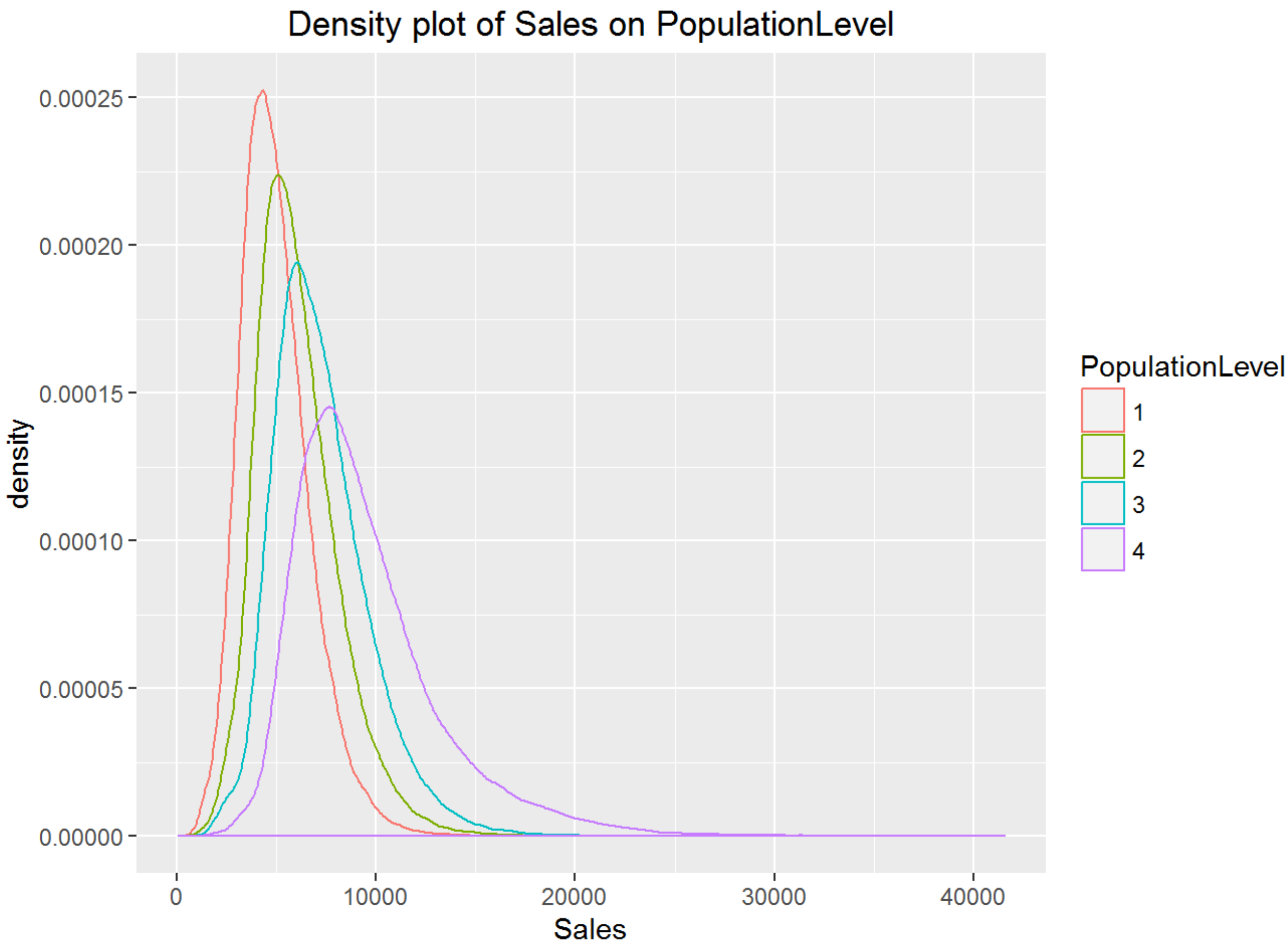
```
#build a new data set with store id and mean number of customers visited that store bef
ore
mean_customers_store <- sqldf("select Store, avg(Customers) as MeanCustomers from train
group by Store order by Store")
head(mean_customers_store)
```

```
##   Store MeanCustomers
## 1     1      564.0499
## 2     2      583.9987
## 3     3      750.0770
## 4     4     1321.7526
## 5     5      537.3402
## 6     6      635.2346
```

```
#get quatiles of MeanCustomers as the splition criterion
summary(mean_customers_store$MeanCustomers)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   240.2   541.5   678.7   754.6   866.2  3403.0
```

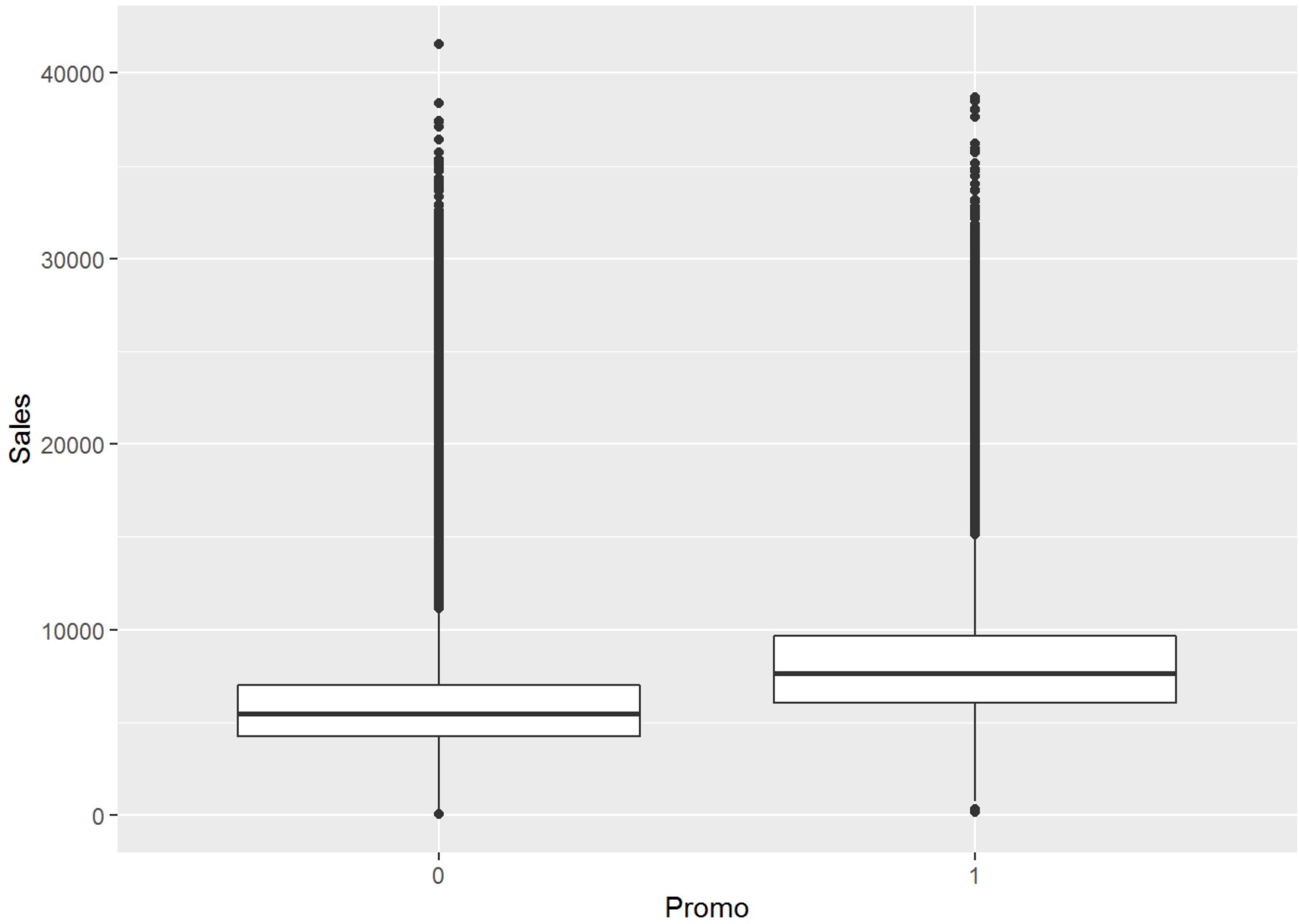
```
#Creat a new Customers dependent variable called PopulationLevel
mean_customers_store$PopulationLevel <- 1
mean_customers_store$PopulationLevel[mean_customers_store$MeanCustomers >541.5 & mean_c
ustomers_store$MeanCustomers <=678.7 ] <- 2
mean_customers_store$PopulationLevel[mean_customers_store$MeanCustomers >678.7 & mean_c
ustomers_store$MeanCustomers <=866.2 ] <- 3
mean_customers_store$PopulationLevel[mean_customers_store$MeanCustomers >866.2 ] <- 4
#Join PopulationLevel into our complete data
complete<- sqldf("select complete.*,b.PopulationLevel from complete
                  inner join mean_customers_store b on complete.Store=b.Store")
#Density plot of Sales on PopulationLevel
complete$PopulationLevel <- as.factor(complete$PopulationLevel)
ggplot(complete[1:844338,], aes(x=Sales, color=PopulationLevel)) + geom_density() + ggt
itle("Density plot of Sales on PopulationLevel")
```



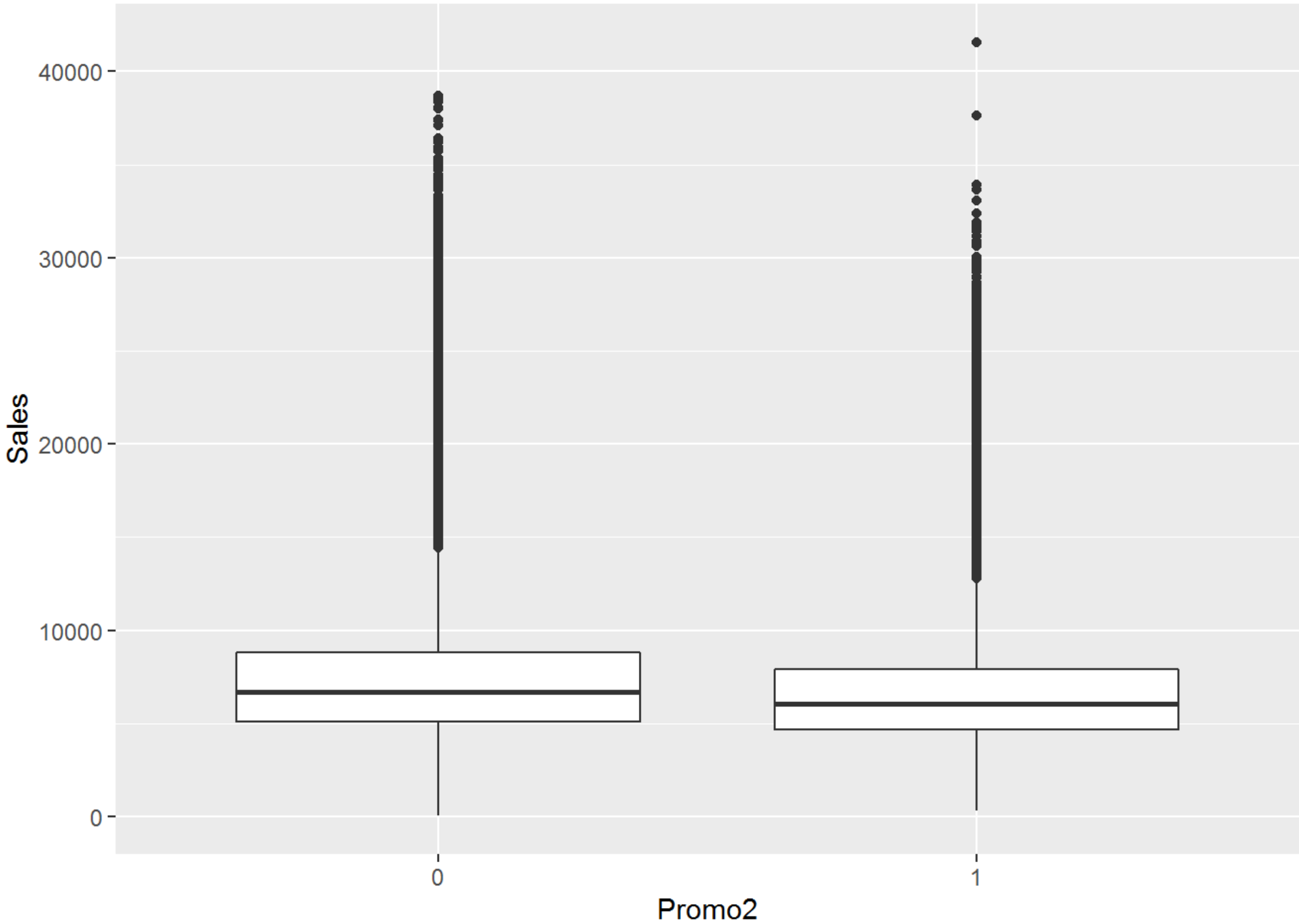
The multiple density plots show Sales is more right skewed on the higher population level. Here, we again take in the important predictor Customers' information, it's kinda addressed the issue of population size nearby.

1.7 Promotion Vs. Sales

```
#Factorize Promo Variable
complete$Promo<-as.factor(complete$Promo)
complete$Promo2<-as.factor(complete$Promo2)
#Boxplots
ggplot(complete[1:844338,], aes(x=Promo, y=Sales)) + geom_boxplot()
```



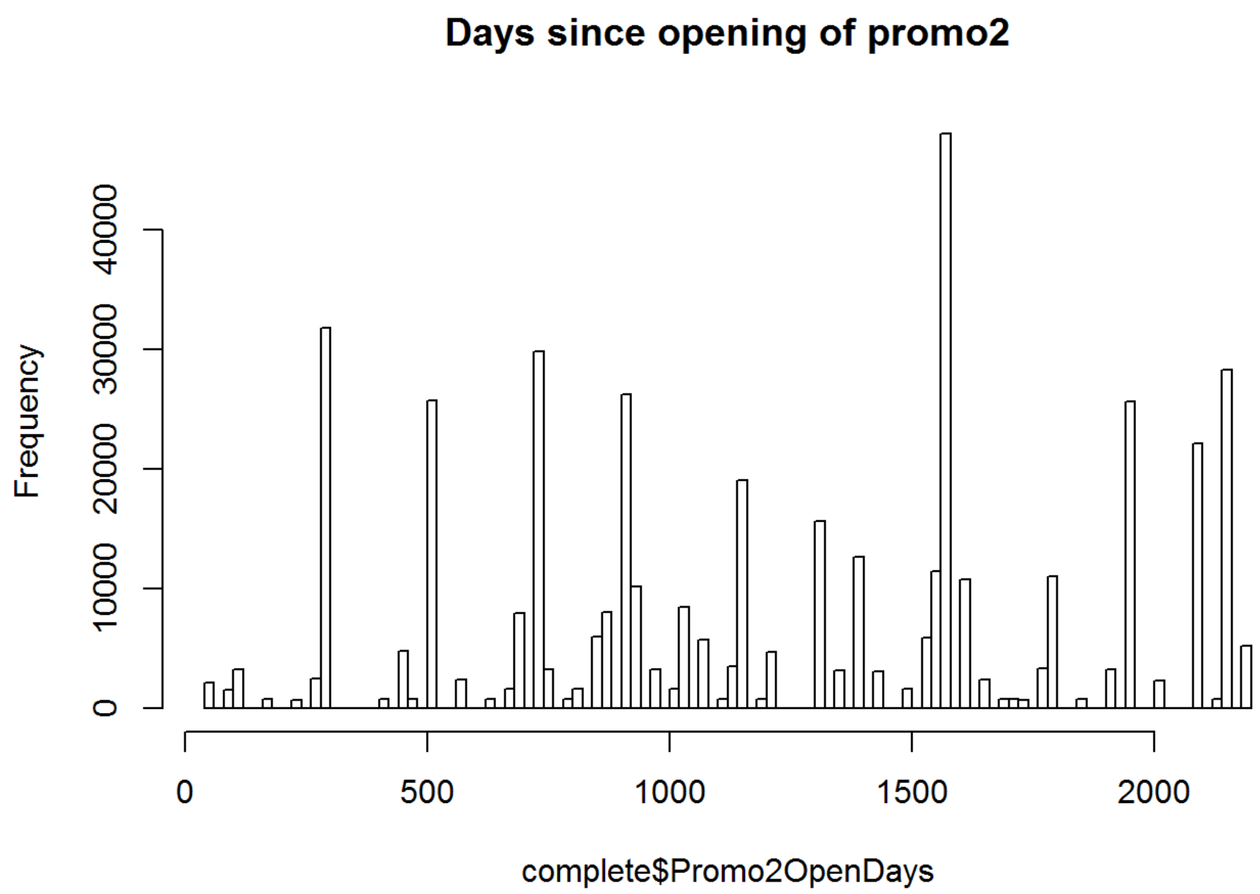

```
ggplot(complete[1:844338,], aes(x=Promo2, y=Sales)) + geom_boxplot()
```



For Promotion running on that day, the sales would increase a bit, but not the same for the consecutive promotion activity. Lets guess that it’s possible only poor sales store need a knida long-term promotion to be more attractive to customers.. it makes sense!

Next, let’s consider the duration of Promo2.

```
#Creat new variable complete$Promo2OpenDays
complete$Promo2OpenDays <- as.numeric(as.POSIXct("2015-07-31", format = "%Y-%m-%d") - a
s.POSIXct(paste(complete$Promo2SinceYear,complete$Promo2SinceWeek, 1, sep = "-"),format
= "%Y-%U-%u"))
#histogram
hist(complete$Promo2OpenDays, breaks=100, main = "Days since opening of promo2")
```

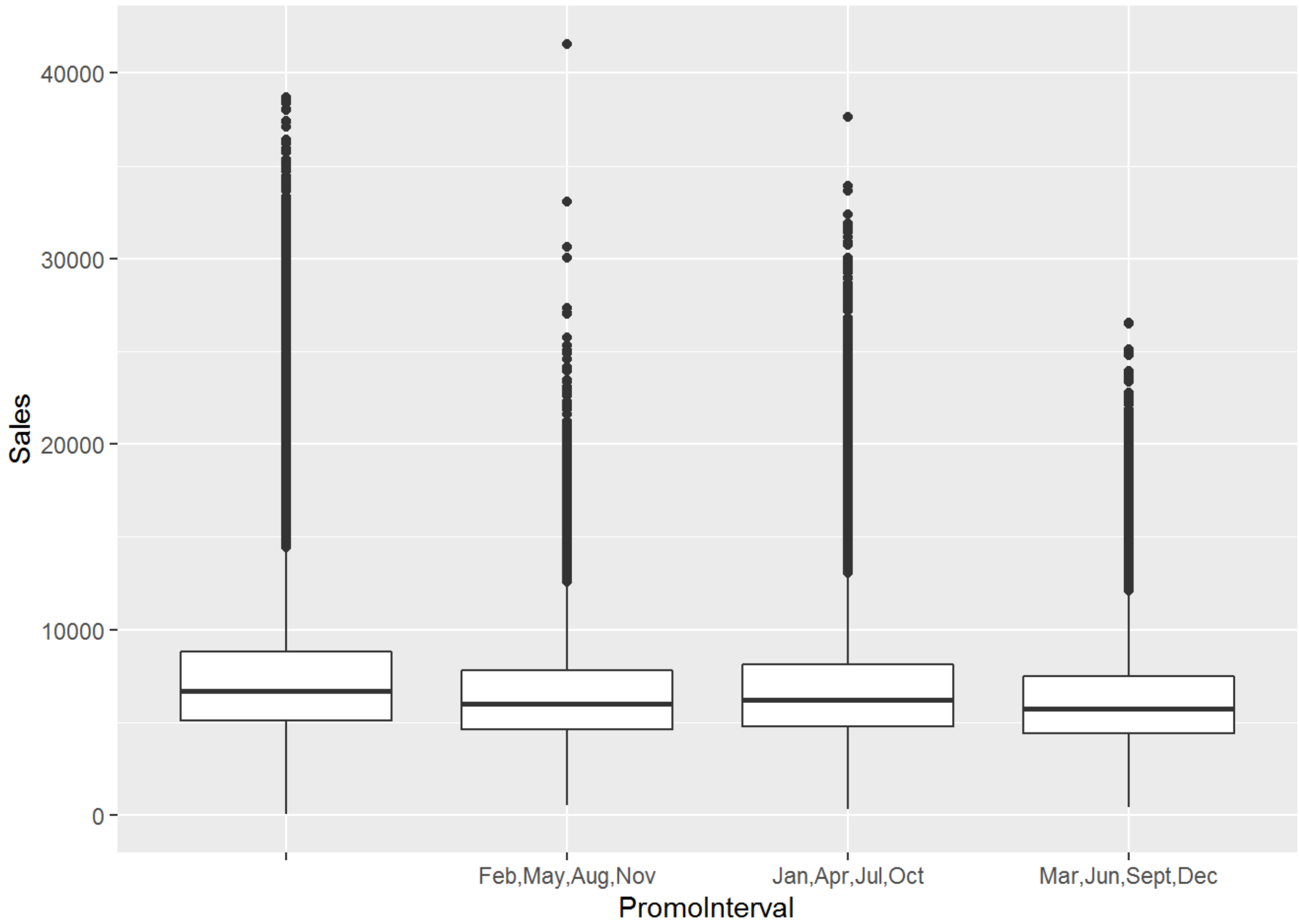



```
#impute NA's with 0 days
complete[is.na(complete$Promo2OpenDays), c("Promo2OpenDays")] = 0
summary(complete$Promo2OpenDays)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0      53     620    1215    2188
```

Finally for promo2interval, there are four levels `None`, `Feb,May,Aug,Nov`, `Jan,Apr,Jul,Oct` and `Mar,Jun,Sept,Dec`. From the boxplots below, it seems sales differ cross different promo-renew-interval.

```
ggplot(complete[1:844338,], aes(x=PromoInterval, y=Sales)) + geom_boxplot()
```



2. Modeling and Prediction

```
#check data type and missing value
complete<-sqldf("select Id,Sales,DayOfWeek,Month,StateHoliday,SchoolHoliday,StoreType,A
ssortment,CompetitionDistance,CompetitionOpenYears,PopulationLevel,Promo,Promo2,Promo2O
penDays,PromoInterval from complete")
str(complete)
```

```
## 'data.frame':    879413 obs. of  15 variables:
##  $ Id                  : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Sales               : int  5263 6064 8314 13995 4822 5651 15344 8492 8565 7185
...
##  $ DayOfWeek           : Factor w/ 7 levels "1","2","3","4",..: 5 5 5 5 5 5 5 5 5 5
...
##  $ Month               : Factor w/ 12 levels "1","2","3","4",..: 7 7 7 7 7 7 7 7 7 7
...
##  $ StateHoliday        : Factor w/ 4 levels "0","a","b","c": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SchoolHoliday       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ StoreType           : Factor w/ 4 levels "a","b","c","d": 3 1 1 3 1 1 1 1 1 1 ...
##  $ Assortment          : Factor w/ 3 levels "a","b","c": 1 1 1 3 1 1 3 1 3 1 ...
##  $ CompetitionDistance : num  1270 570 14130 620 29910 ...
##  $ CompetitionOpenYears: num  6.83 7.67 8.58 5.83 0.25 ...
##  $ PopulationLevel     : Factor w/ 4 levels "1","2","3","4": 2 2 3 4 1 2 4 2 2 2 ...
##  $ Promo               : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Promo2              : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 1 1 1 ...
##  $ Promo2OpenDays      : num  0 1950 1579 0 0 ...
##  $ PromoInterval       : Factor w/ 4 levels "","Feb,May,Aug,Nov",..: 1 3 3 1 1 1 1 1
1 1 ...
```

```
#Give "No Interval" to missing value in PromoInterval
sqldf() # start a sequence of SQL statements
```

```
## <SQLiteConnection>
```

```
fn$sqldf("update complete set PromoInterval ='No Interval' where PromoInterval = '' ")
```

```
## NULL
```

```
complete<- sqldf("select * from main.complete")
sqldf() # SQL statements finished
```

```
## NULL
```

```
#Factorization all cha columns
complete<- as.data.frame(unclass(complete))
summary(complete$Sales)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##      46    4859    6369    6956    8360   41550   35075
```

```
summary(complete$CompetitionDistance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      20     710    2330    5446    6880   75860
```

```r
summary(complete$CompetitionOpenYears)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
##  -0.08333   3.66700   6.28600   6.28600   7.25000 115.50000
```

```r
complete$CompetitionOpenYears[complete$CompetitionOpenYears<0]<-0
summary(complete$Promo2OpenDays)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##       0       0      53     620    1215    2188
```

2.1 Random Forest

In this session, we are ready to predict Sales for Rossmann stores based on variables that we carefully treated and extracted. First let's use `randomForest` classification algorithm to deal with such a bunch of categorical variables.

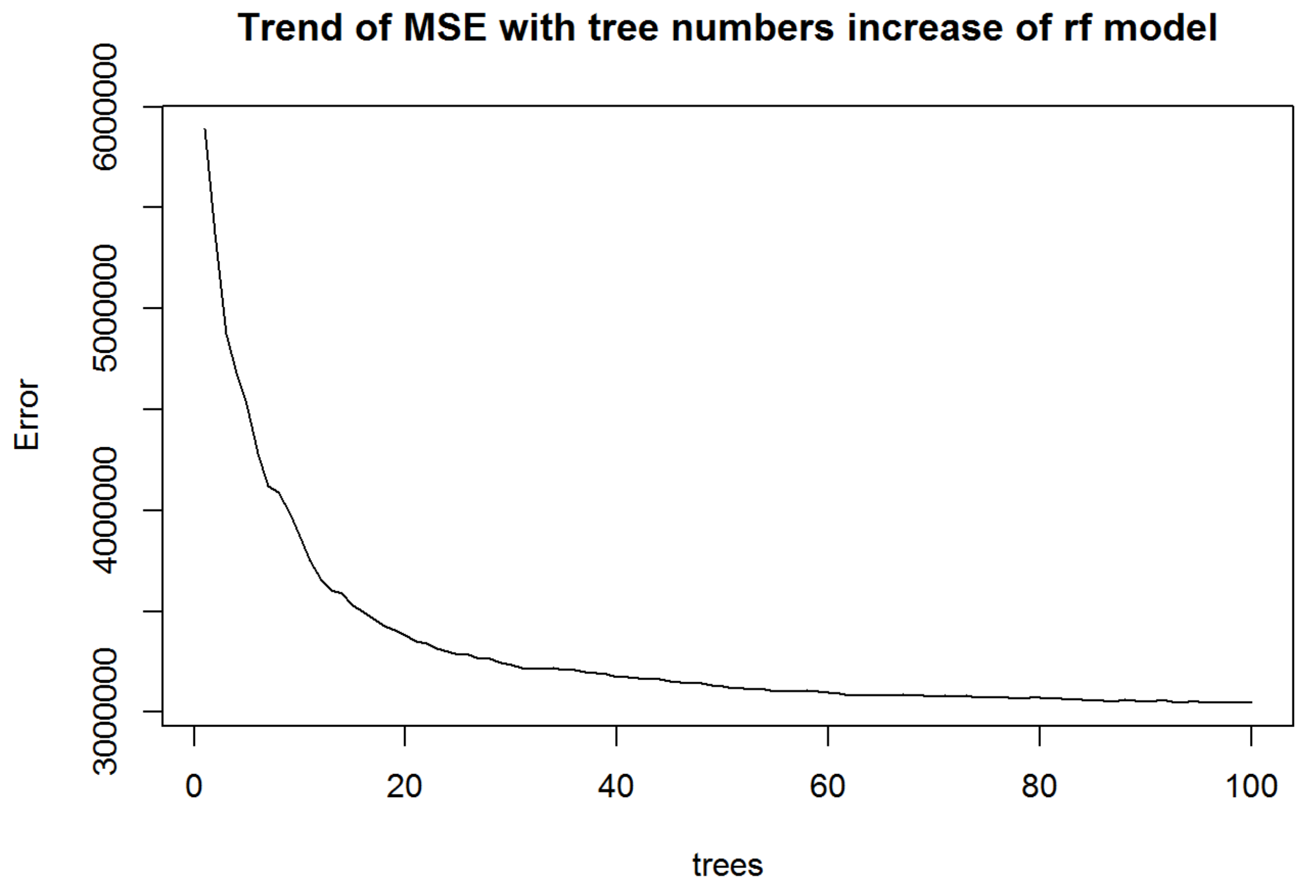
```r
#split complete to train and test data set
train<-complete[1:844338,-1]
test<-complete[844339:879413,]

# Using a random sample from the train set due to the PC memory issue
#I first run a sample of 10,000, and set the ntree option to 100
sample<-train[sample(nrow(train), 10000), ]

#Build rf model
rf.model1 <- randomForest(Sales~DayOfWeek+Month+StateHoliday+SchoolHoliday+StoreType+Assortment+CompetitionDistance+CompetitionOpenYears+PopulationLevel+Promo+Promo2+Promo2OpenDays+PromoInterval,data=sample,importance=TRUE, proximity=TRUE,ntree=100)
rf.model1
```

```
## 
## Call:
##  randomForest(formula = Sales ~ DayOfWeek + Month + StateHoliday +      SchoolHoliday + StoreType + Assortment + CompetitionDistance +      CompetitionOpenYears + PopulationLevel + Promo + Promo2 +      Promo2OpenDays + PromoInterval, data = sample, importance = TRUE,      proximity = TRUE, ntree = 100) 
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 4
## 
##           Mean of squared residuals: 3002073
##                     % Var explained: 67.9
```

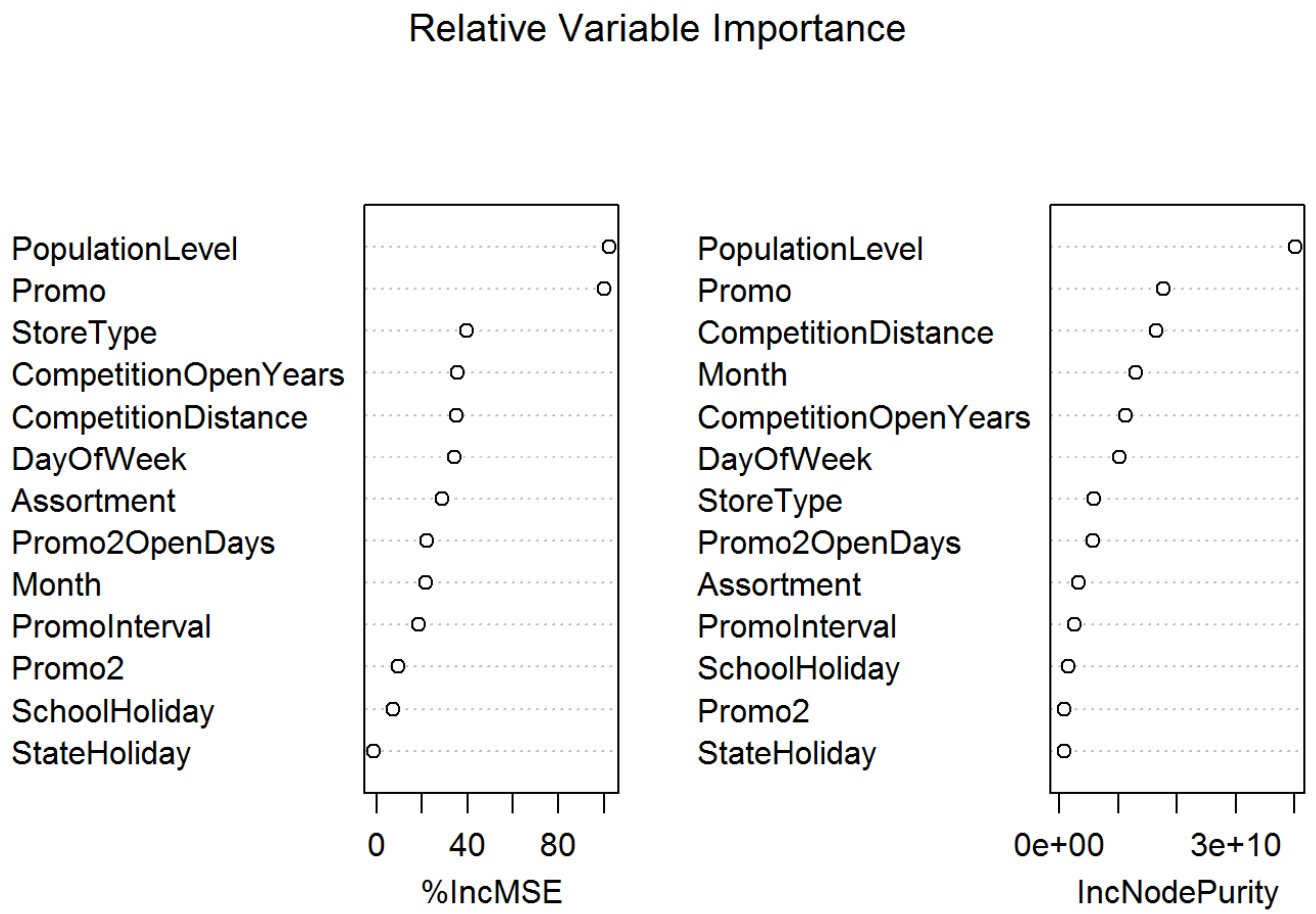
```r
#Then I tried sample size of 15,000
sample<-train[sample(nrow(train), 15000), ]

rf.model2 <- randomForest(Sales~DayOfWeek+Month+StateHoliday+SchoolHoliday+StoreType+Assortment+CompetitionDistance+CompetitionOpenYears+PopulationLevel+Promo+Promo2+Promo2OpenDays+PromoInterval,data=sample,importance=TRUE, proximity=TRUE,ntree=100)
rf.model2
```

```
##
## Call:
##  randomForest(formula = Sales ~ DayOfWeek + Month + StateHoliday +      SchoolHoliday + StoreType + Assortment + CompetitionDistance +      CompetitionOpenYears + PopulationLevel + Promo + Promo2 +      Promo2OpenDays + PromoInterval, data = sample, importance = TRUE,      proximity = TRUE, ntree = 100)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 3047404
##                     % Var explained: 68.81
```

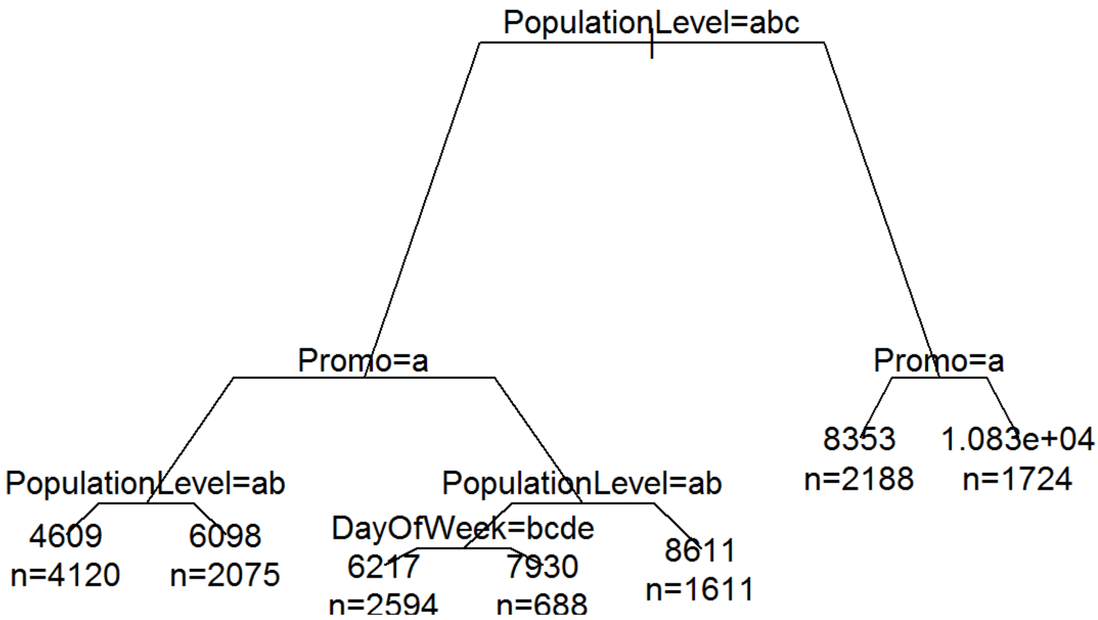
```r
#Won't work anymore for the size of 20,000, I need to buy a bigger computer :(

# Show model error, since my response variable Sales is one continueous variable, the randomForest is running a regression type and the error rate plot would only show one black solid line representing the OOB MSE
plot(rf.model2,main="Trend of MSE with tree numbers increase of rf model")
```




```
#Show the relative variable importance
varImpPlot(rf.model2, main="Relative Variable Importance")
```




```
#Build rpart the model
rpart.model <- rpart(Sales~DayOfWeek+Month+StateHoliday+SchoolHoliday+StoreType+Assortm
ent+CompetitionDistance+CompetitionOpenYears+PopulationLevel+Promo+Promo2+Promo2OpenDay
s+PromoInterval,data=sample)
plot(rpart.model, branch=0.6, margin=0.05)
text(rpart.model, use.n=TRUE)
```



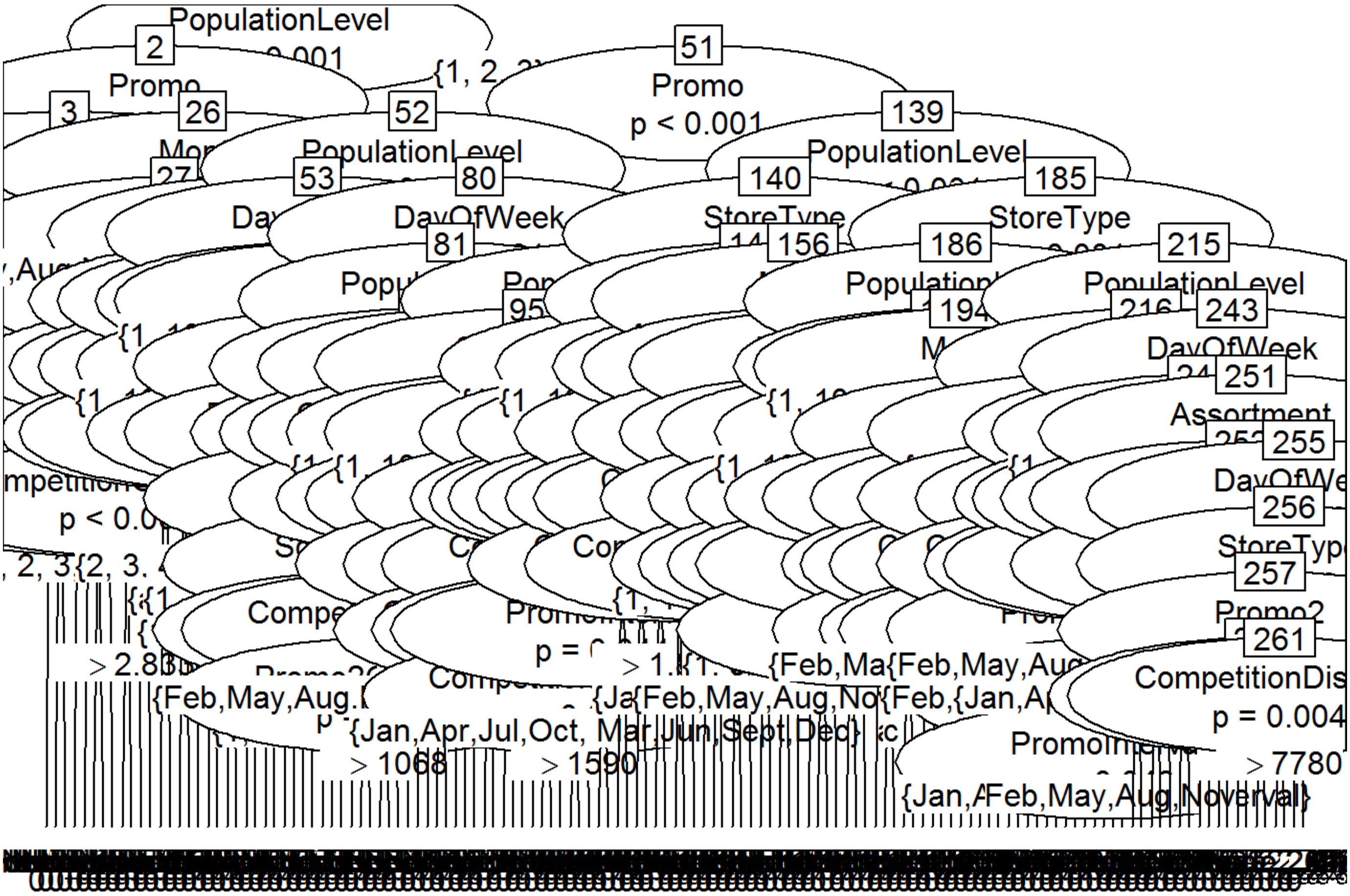

```
summary(rpart.model)
```

```
## Call:
## rpart(formula = Sales ~ DayOfWeek + Month + StateHoliday + SchoolHoliday +
##     StoreType + Assortment + CompetitionDistance + CompetitionOpenYears +
##     PopulationLevel + Promo + Promo2 + Promo2OpenDays + PromoInterval,
##     data = sample)
##   n= 15000
##
##           CP nsplit rel error    xerror       xstd
## 1 0.22735527      0 1.0000000 1.0000310 0.02238110
## 2 0.08532602      1 0.7726447 0.7727562 0.01719501
## 3 0.04045673      2 0.6873187 0.6874735 0.01696451
## 4 0.03053721      3 0.6468620 0.6472056 0.01668164
## 5 0.02088304      4 0.6163248 0.6167061 0.01656095
## 6 0.01088748      5 0.5954417 0.5958872 0.01639407
## 7 0.01000000      6 0.5845543 0.5850116 0.01632627
##
## Variable importance
##     PopulationLevel               Promo           DayOfWeek
##                  58                  26                   6
## CompetitionDistance           StoreType          Assortment
##                   4                   3                   2
##
## Node number 1: 15000 observations,    complexity param=0.2273553
##   mean=6936.563, MSE=9771023
##   left son=2 (11088 obs) right son=3 (3912 obs)
##   Primary splits:
##       PopulationLevel splits as  LLLR, improve=0.22735530, (0 missing)
##       Promo           splits as  LR, improve=0.12504050, (0 missing)
##       DayOfWeek       splits as  RLLLLLR, improve=0.02908482, (0 missing)
##       Promo2OpenDays  < 598.9583   to the right, improve=0.01981068, (0 missing)
##       Month           splits as  LLLRLLLLLLLL, improve=0.01944527, (0 missing)
##   Surrogate splits:
##       CompetitionDistance < 275        to the right, agree=0.758, adj=0.073, (0 spli
t)
##       StoreType           splits as  LRLL, agree=0.758, adj=0.071, (0 split)
##       Assortment          splits as  LRL, agree=0.749, adj=0.037, (0 split)
##       DayOfWeek           splits as  LLLLLLR, agree=0.742, adj=0.011, (0 split)
##       StateHoliday        splits as  LRRR, agree=0.740, adj=0.002, (0 split)
##
## Node number 2: 11088 observations,    complexity param=0.08532602
##   mean=6051.253, MSE=5314217
##   left son=4 (6195 obs) right son=5 (4893 obs)
##   Primary splits:
##       Promo           splits as  LR,           improve=0.21223660, (0 missing)
##       PopulationLevel splits as  LLR-,         improve=0.12288250, (0 missing)
##       DayOfWeek       splits as  RLLLLLL,      improve=0.05019538, (0 missing)
##       StoreType       splits as  L-LR,         improve=0.02829672, (0 missing)
##       Month           splits as  LLLRLLLLLLLL, improve=0.02396391, (0 missing)
##   Surrogate splits:
##       DayOfWeek            splits as  RRRRRLL, agree=0.614, adj=0.126, (0 split)
##       StateHoliday         splits as  LR--, agree=0.559, adj=0.001, (0 split)
##       CompetitionOpenYears < 0.04166667 to the right, agree=0.559, adj=0.000, (0 spl
it)
```

```
##
## Node number 3: 3912 observations,    complexity param=0.04045673
##   mean=9445.846, MSE=1.38852e+07
##   left son=6 (2188 obs) right son=7 (1724 obs)
##   Primary splits:
##       Promo                splits as  LR, improve=0.10916190, (0 missing)
##       DayOfWeek            splits as  RLLLLLR, improve=0.02957379, (0 missing)
##       Month                splits as  LLLRLLLLLLLL, improve=0.02639026, (0 missing)
##       Assortment           splits as  LLR, improve=0.01510439, (0 missing)
##       CompetitionOpenYears < 2.208333   to the right, improve=0.01213442, (0 missin
g)
##   Surrogate splits:
##       DayOfWeek            splits as  RRRRRLL, agree=0.626, adj=0.151, (0 split)
##       CompetitionOpenYears < 22.83333   to the left,  agree=0.561, adj=0.003, (0 spl
it)
##       CompetitionDistance  < 58200      to the left,  agree=0.560, adj=0.001, (0 spl
it)
##       StateHoliday         splits as  LLRL, agree=0.560, adj=0.001, (0 split)
##
## Node number 4: 6195 observations,    complexity param=0.02088304
##   mean=5107.417, MSE=3417209
##   left son=8 (4120 obs) right son=9 (2075 obs)
##   Primary splits:
##       PopulationLevel     splits as  LLR-, improve=0.14458140, (0 missing)
##       StoreType           splits as  L-LR, improve=0.05324117, (0 missing)
##       Assortment          splits as  L-R, improve=0.04207632, (0 missing)
##       CompetitionDistance < 2080       to the left,  improve=0.03820283, (0 missing)
##       Month               splits as  LLLRLLLLLLLL, improve=0.03301390, (0 missing)
##   Surrogate splits:
##       CompetitionDistance  < 205        to the right, agree=0.693, adj=0.082, (0 spl
it)
##       CompetitionOpenYears < 16.04167   to the left,  agree=0.669, adj=0.011, (0 spl
it)
##       DayOfWeek            splits as  LLLLLLR, agree=0.666, adj=0.001, (0 split)
##       StateHoliday         splits as  LR--, agree=0.665, adj=0.000, (0 split)
##
## Node number 5: 4893 observations,    complexity param=0.03053721
##   mean=7246.239, MSE=5160145
##   left son=10 (3282 obs) right son=11 (1611 obs)
##   Primary splits:
##       PopulationLevel     splits as  LLR-, improve=0.17726520, (0 missing)
##       DayOfWeek           splits as  RLLLL--, improve=0.11007010, (0 missing)
##       Month               splits as  LLLRLLLLLLLL, improve=0.03867596, (0 missing)
##       StoreType           splits as  L-LR, improve=0.02173866, (0 missing)
##       CompetitionDistance < 3855       to the left,  improve=0.02150701, (0 missing)
##   Surrogate splits:
##       CompetitionDistance  < 145        to the right, agree=0.692, adj=0.065, (0 spl
it)
##       CompetitionOpenYears < 16.04167   to the left,  agree=0.674, adj=0.010, (0 spl
it)
##
## Node number 6: 2188 observations
##   mean=8353.006, MSE=1.193559e+07
##
```

```
## Node number 7: 1724 observations
##   mean=10832.81, MSE=1.292011e+07
##
## Node number 8: 4120 observations
##   mean=4608.587, MSE=2520577
##
## Node number 9: 2075 observations
##   mean=6097.864, MSE=3722459
##
## Node number 10: 3282 observations,    complexity param=0.01088748
##   mean=6576.168, MSE=3803335
##   left son=20 (2594 obs) right son=21 (688 obs)
##   Primary splits:
##       DayOfWeek       splits as  RLLLL--, improve=0.12783670, (0 missing)
##       PopulationLevel splits as  LR--, improve=0.09620262, (0 missing)
##       StoreType       splits as  L-LR, improve=0.05847065, (0 missing)
##       Month           splits as  LLLRLLLLLLLL, improve=0.05188765, (0 missing)
##       Promo2OpenDays  < 2117.979   to the left,  improve=0.02733961, (0 missing)
##
## Node number 11: 1611 observations
##   mean=8611.338, MSE=5146088
##
## Node number 20: 2594 observations
##   mean=6217.064, MSE=2938934
##
## Node number 21: 688 observations
##   mean=7930.112, MSE=4743060
```

```
#Finally I tried Conditional Inference Tree
ctree.model <- ctree(Sales~DayOfWeek+Month+StateHoliday+SchoolHoliday+StoreType+Assortm
ent+CompetitionDistance+CompetitionOpenYears+PopulationLevel+Promo+Promo2+Promo2OpenDay
s+PromoInterval,data=sample)
plot(ctree.model)
```



See? the OOB error rate from rf though is not so pretty, around `0.3` , as long as I increased the sample size from `10,000` to `15,000` and keep ntree the same, the error rate would decrease as well. So I would like to conclude that our model is still reasonable once there's a way to include that really large train data set(844,392 obs).

Let's have a look at the Relative Variable Importance table above. Whoa, NICE we made our PopulationLevel Varible(extracted information from our vital predictor Customers), and as we guess earlier that Promo and Competition gave some importance to the store Sales. Also some other new created variable such as CompetitionOpenYears, Month and Promo2OpenDays are all ranked well, from this case we can tell that feature engineering work is so worthing!

For regression tree model, I added another two algorithms: rpart and ctree for comparison

2.2 Linear Model

Just out of curiousity, I still wanna run a simple linear model including the same features, maybe just as a comparison of my rf model.

```
#Linear Model
lm.fit <- lm(Sales~.,data=train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Sales ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11478.2  -1254.2   -185.0    948.2  31619.6
##
## Coefficients: (1 not defined because of singularities)
##                                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                    3.785e+03  1.200e+01  315.434  < 2e-16 ***
## DayOfWeek2                    -1.077e+03  8.124e+00 -132.517  < 2e-16 ***
## DayOfWeek3                    -1.404e+03  8.157e+00 -172.155  < 2e-16 ***
## DayOfWeek4                    -1.371e+03  8.270e+00 -165.799  < 2e-16 ***
## DayOfWeek5                    -1.013e+03  8.207e+00 -123.416  < 2e-16 ***
## DayOfWeek6                    -9.672e+02  8.778e+00 -110.179  < 2e-16 ***
## DayOfWeek7                    -9.850e+02  3.828e+01  -25.731  < 2e-16 ***
## Month10                       -1.904e+01  1.194e+01   -1.595    0.111
## Month11                        4.752e+02  1.202e+01   39.517  < 2e-16 ***
## Month12                        1.965e+03  1.211e+01  162.279  < 2e-16 ***
## Month2                         8.597e+01  1.056e+01    8.140 3.96e-16 ***
## Month3                         3.019e+02  1.038e+01   29.086  < 2e-16 ***
## Month4                         4.259e+02  1.055e+01   40.377  < 2e-16 ***
## Month5                         5.249e+02  1.057e+01   49.636  < 2e-16 ***
## Month6                         4.461e+02  1.049e+01   42.524  < 2e-16 ***
## Month7                         1.792e+02  1.061e+01   16.894  < 2e-16 ***
## Month8                        -9.885e+01  1.227e+01   -8.058 7.77e-16 ***
## Month9                        -8.032e+01  1.194e+01   -6.728 1.73e-11 ***
## StateHolidaya                 -3.458e+01  8.224e+01   -0.420    0.674
## StateHolidayb                 -2.665e+02  1.794e+02   -1.485    0.138
## StateHolidayc                 -1.072e+03  2.563e+02   -4.183 2.87e-05 ***
## SchoolHoliday1                 2.838e+02  6.729e+00   42.168  < 2e-16 ***
## StoreTypeb                     2.941e+03  2.634e+01  111.652  < 2e-16 ***
## StoreTypec                    -4.080e+02  7.244e+00  -56.319  < 2e-16 ***
## StoreTyped                     8.570e+02  5.711e+00  150.076  < 2e-16 ***
## Assortmentb                   -3.304e+03  3.465e+01  -95.373  < 2e-16 ***
## Assortmentc                    5.421e+02  4.960e+00  109.291  < 2e-16 ***
## CompetitionDistance            2.862e-03  3.155e-04    9.072  < 2e-16 ***
## CompetitionOpenYears           2.300e+00  4.782e-01    4.809 1.51e-06 ***
## PopulationLevel2               1.154e+03  6.745e+00  171.034  < 2e-16 ***
## PopulationLevel3               2.427e+03  6.898e+00  351.867  < 2e-16 ***
## PopulationLevel4               4.898e+03  7.376e+00  664.044  < 2e-16 ***
## Promo1                         2.304e+03  5.199e+00  443.125  < 2e-16 ***
## Promo21                       -3.488e+02  1.076e+01  -32.416  < 2e-16 ***
## Promo2OpenDays                 2.821e-01  5.718e-03   49.339  < 2e-16 ***
## PromoIntervalJan,Apr,Jul,Oct   2.420e+02  8.164e+00   29.649  < 2e-16 ***
## PromoIntervalMar,Jun,Sept,Dec -1.251e+00  1.040e+01   -0.120    0.904
## PromoIntervalNo Interval              NA         NA       NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2153 on 844301 degrees of freedom
## Multiple R-squared:  0.519,  Adjusted R-squared:  0.519
```

```
## F-statistic: 2.53e+04 on 36 and 844301 DF,  p-value: < 2.2e-16
```

Check some benchmarks `R-square:0.519` meaning only about 52% variance of Sales can be explained by predictors included in the model, `P-value` shows that the linear model is meaningful anyway, but it seems rf yields a better result, hence prediction on test set will use rf model.

2.3 Prediction

Once we got our model, we are almost there! Just a very quick thing left–prediction.

```
# Predict using the test set
test$prediction <- predict(rf.model2, test[, 3:15])

# Save the submission with Id and predicted Sales(Only for Open=1)
submission<- test[,c(1,16)]

#view the format
head(submission)
```

```
##        Id prediction
## 844339  1   5064.980
## 844340  2   7861.872
## 844341  3  11022.824
## 844342  4   5772.017
## 844343  5   6150.727
## 844344  6   5844.600
```

```
#Write solution to CSV file
write.csv(submission, file = 'submission.csv', row.names = F)
```

Thank you for taking your time to read this analysis, any recommendation or cureness is so welcomed!