Data Analytics Preprocessing Plots and Result Plot

Sai Rakesh Ghanta (sag163@pitt.edu)

2016/11/17

Import data

```
df <- read.csv("~/desktop/Final Analysis Dataset.csv", header=FALSE, st
ringsAsFactors=FALSE)
colnames(df) <- c("Sdate","Syear","Smonth","Sday","week","Shour","Edate
","Smin","Eyear","Emonth","Eday","Ehour","Emin","Fromstation","Tostatio
n")
df[1:3,]
```

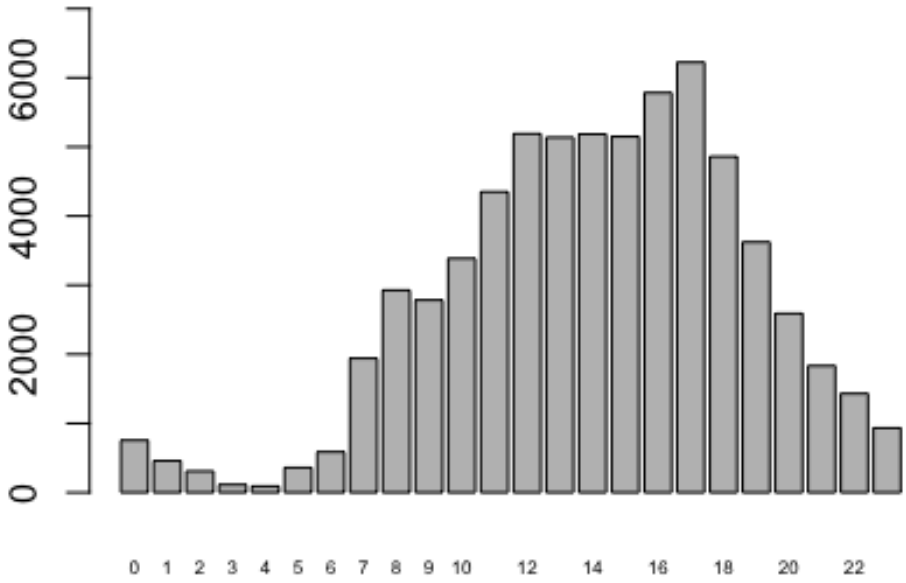
```
##       Sdate Syear Smonth Sday     week Shour Edate     Smin Eyear Em
onth
## 1 Starttime Syear Smonth Sday Sweekday Shour  Smin  EndTime Eyear Em
onth
## 2 6/30/2015  2015      6   30        3    23    59 7/1/2015  2015
   7
## 3  7/1/2015  2015      7    1        4     0    44 7/1/2015  2015
   7
##   Eday Ehour Emin Fromstation Tostation
## 1 Eday Ehour Emin FromStation ToStation
## 2    1     0   09        1001      1007
## 3    1     0   58        1006      1000
```

Plot 1

```
############### Plot 1 ###############

A <- c(1:24) # 24 hours
for (k in 0:23) {
  A[k+1] <- length(which(df$Shour==k))
}
A
```

```
##  [1]  755  459  308  114   91  360  587 1941 2927 2784 3386 4350 519
0 5137
## [15] 5184 5149 5784 6224 4860 3623 2590 1833 1431  933
```

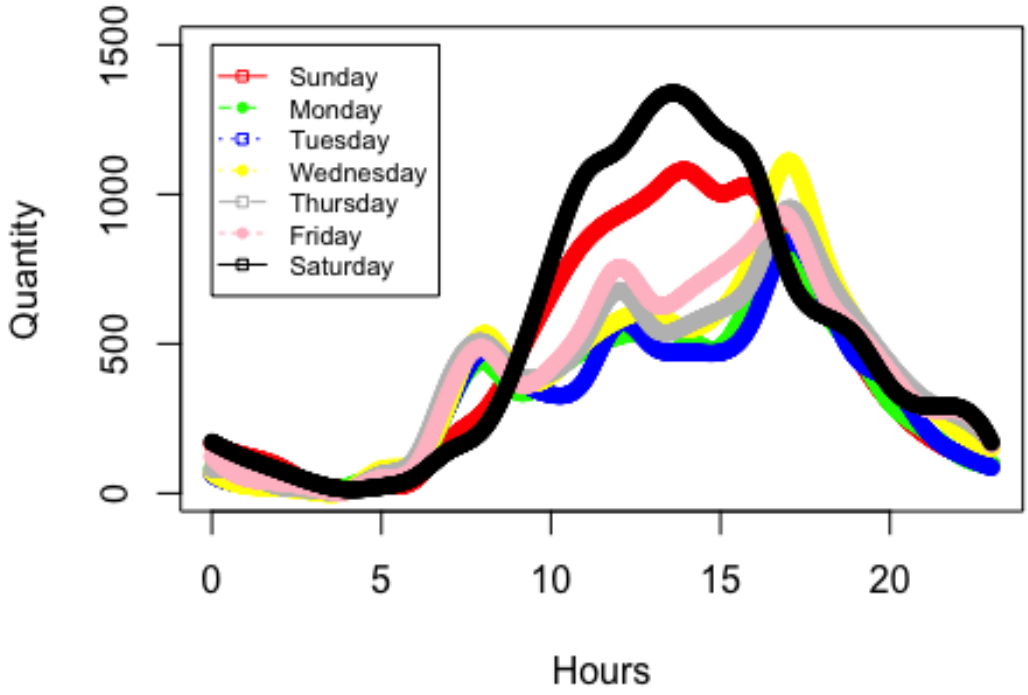
```
name <- c("0","1","2","3","4","5","6","7","8","9","10","11","12","13","
14","15","16","17","18","19","20","21","22","23") # X
day <- c(1:31)
barplot(A,names.arg=name,ylim = c(0,7000), cex.names=0.5) # y = A, x= n
ame
```



Plot2

```
############### Plot 2 ###############

B <- matrix(1, nrow = 7, ncol = 24) #empty

for (j in 1:7) {
  for (i in 0:23) {
    B[j,(i+1)] <- length(which(df[df$week==j,]$Shour==i))
  }
}

sp=spline(name,B[1,],n=1000)  #let line become smooth
sp1=spline(name,B[2,],n=1000)
sp2=spline(name,B[3,],n=1000)
sp3=spline(name,B[4,],n=1000)
sp4=spline(name,B[5,],n=1000)
sp5=spline(name,B[6,],n=1000)
sp6=spline(name,B[7,],n=1000)
plot(sp,col="red",ylim = c(0,1500),pch=16, xlab="Hours", ylab="Quantity
",cex=1,type="b")   # backgroud + first line
# Q1 0~270,  Q4 0~350
lines(sp1,col="green",pch=16,cex=1,type="b")     # add line
```

```
lines(sp2,col="blue",pch=16,lwd=2,cex=1,type="b")
lines(sp3,col="yellow",pch=16,lwd=2,cex=1,type="b")
lines(sp4,col="gray",pch=16,lwd=2,cex=1,type="b")
lines(sp5,col="pink",pch=16,lwd=2,cex=1,type="b")
lines(sp6,col="black",pch=16,lwd=2,cex=1,type="b")

legend(0,1500,legend=c("Sunday","Monday","Tuesday","Wednesday","Thursda
y","Friday","Saturday"),
       pch=c(22,16),col=c("red","green","blue","yellow","gray","pink","
black"),lty=c(1:7),cex=0.7)
```

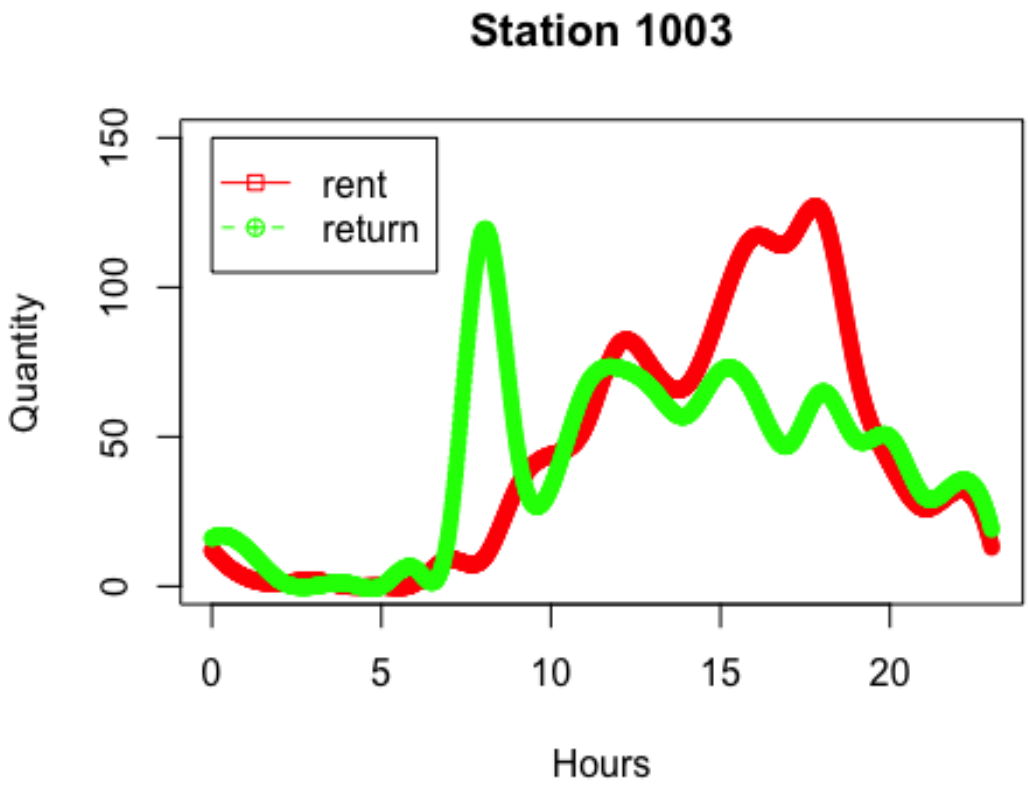



```
# Q1(0,270), Q4(0,350) is the legend location, cex is the size of the l
egend
# Sunday is first day!
```

Plot4 Use one station as an example, so we do it manually.

```
############### Plot 4 ###############

C <- matrix(1, nrow = 24, ncol = 3)
C[1:24,1] = c(1:24)
C[1:24,2] = c("12","3","1","2","0","0","1","9","9","33","44","53","81",
"73","67","93","117","115","125",
```

```
              "72","41","26","32","13")
C[1:24,3] = c("16","13","2","0","1","0","6","19","119","47","33","66","
73","66","57","72","64","47","65",
              "49","50","30","35","19")

colnames(C) <- c("hour","rent","return")
C = t(C) #Transation
#C
name <- c("0","1","2","3","4","5","6","7","8","9","10","11","12","13","
14","15","16","17","18","19","20","21","22","23") # X

sp11=spline(name,C[2,],n=1000) #let line become smooth
sp12=spline(name,C[3,],n=1000)

plot(sp11,col="red",ylim = c(0,150),pch=16, xlab="Hours", ylab="Quantit
y", main="Station 1003",cex=1,type="b") #backgroud + first line
lines(sp12,col="green",pch=16,cex=1,type="b") # second line
legend(0,150,legend=c("rent","return"),pch=c(22,10),col=c("red","green"
),lty=c(1:2),cex=1) #explain label
```



Result Plot (Prediction base on the best model-SVM)

```
############### Plot 3 ###############

library(ggmap)

## Loading required package: ggplot2

library(car)  # for recode
library(mapproj)

## Loading required package: maps

map <- get_map(location = 'Pittsburgh', zoom = 13, maptype = "roadmap")
 #set map backgroud

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=
Pittsburgh&zoom=13&size=640x640&scale=2&maptype=roadmap&language=en-EN&
sensor=false

## Information from URL : http://maps.googleapis.com/maps/api/geocode/j
son?address=Pittsburgh&sensor=false

ggmap(map)
```


```
mapdot <- read.csv("~/desktop/L2.csv", header=FALSE, stringsAsFactors=F
ALSE)
```

```r
colnames(mapdot) <- c("StationNum","Latitude","Longitude","size1","colo
r1")
mapdot = mapdot[-1,]
mapdot[1:3,]
```

```
##   StationNum  Latitude  Longitude size1 color1
## 2       1000 40.441326 -80.004679 -2103  green
## 3       1001 40.440877  -80.00308  -352  green
## 4       1002  40.43903  -80.00186    73    red
```

```r
mapdot
```

```
##    StationNum    Latitude   Longitude size1 color1
## 2        1000   40.441326  -80.004679 -2103  green
## 3        1001   40.440877   -80.00308  -352  green
## 4        1002    40.43903   -80.00186    73    red
## 5        1003     40.4372  -80.000375    -2  green
## 6        1004  40.4358002 -79.9968767   164    red
## 7        1005  40.4387769 -79.9974405    67    red
## 8        1006  40.4400542 -79.9951376    68    red
## 9        1007   40.437643  -79.986695    47    red
## 10       1008   40.440368  -79.988636   181    red
## 11       1009   40.445844   -79.99238   191    red
## 12       1010   40.444614 -79.9958114  -130  green
## 13       1011  40.4448352 -80.0007479   -29    red
## 14       1012   40.445834  -80.008882  -240  green
## 15       1013   40.447571  -80.003964   -22  green
## 16       1014   40.450595  -80.013204    54    red
## 17       1015 40.45509087 -80.0063467   -48  green
## 18       1016   40.449631  -79.985893  -137  green
## 19       1017   40.452124  -79.983543  -365  green
## 20       1018   40.466103  -79.964628  -182  green
## 21       1019   40.470188 -79.9603066  -388  green
## 22       1020   40.465893  -79.954417   500    red
## 23       1021   40.462769  -79.950867   373    red
## 24       1022  40.4599487 -79.9456124   303    red
## 25       1023   40.456505  -79.939362    34    red
## 26       1024   40.458714  -79.933483    70    red
## 27       1025   40.464443  -79.933188   339    red
## 28       1026   40.460982  -79.926302   -72  green
## 29       1027   40.458972  -79.922023    96    red
## 30       1028   40.455821  -79.915248  -130  green
## 31       1029  40.4573211 -79.9248275    67    red
## 32       1030  40.4589116 -79.9290211   -43  green
## 33       1031    40.45628  -79.930962    19    red
## 34       1032   40.452621  -79.928637   340    red
## 35       1033    40.45177  -79.932324    96    red
## 36       1034   40.448419  -79.947401   181    red
## 37       1035   40.446744  -79.950881   227    red
## 38       1036   40.442398  -79.951479   477    red
```

```
## 39       1037   40.441032  -79.948042   -67  green
## 40       1038   40.434338  -79.951877   344    red
## 41       1039   40.437987   -79.95367   179    red
## 42       1040  40.4446284 -79.9550156   146    red
## 43       1041   40.442325  -79.957604   236    red
## 44       1042   40.445019  -79.977194    64    red
## 45       1043   40.438876  -79.960179   101    red
## 46       1044   40.435986  -79.956942    20    red
## 47       1045   40.428658  -79.965228  -507  green
## 48       1046    40.42802  -79.969799    15    red
## 49       1047   40.428576  -79.974559    56    red
## 50       1048   40.429338  -79.980684  -120  green
## 51       1049  40.4285528 -79.9863687  -129  green
```

```
testresult <- read.csv("~/desktop/Final Analysis Testing result 1105.csv", header=T, stringsAsFactors=FALSE)
testresult[1:3,]
```

```
##   StationNum hour weekday color
## 1       1000    0       1     1
## 2       1000    1       1     1
## 3       1000    2       1     2
```

```
testresult$color=recode(testresult$color,"'1'='red';else='green'")
# 1 = red = need bike(rent>return), 2 = green = not need bike (return>rent)

colorre <- c(1:50) # set empty vector for testing results
hr = 14 #set hour
week = 1 # set weekday

for (zz in 1000:1049){
  colorre[zz-999] = testresult[testresult$StationNum==zz & testresult$hour==hr & testresult$weekday==week,]$color
}
mapdot$color1 = colorre



Lat <- c(1:50)
Lon <- c(1:50)
#size1 <- c(1:50)

for (x in 1:50) {
  Lat[x] = mapdot$Latitude[x]
  Lon[x] = mapdot$Longitude[x]
  #size1[x] = mapdot$size1[x]
}

Lat = as.numeric(Lat) #change to numeric
Lon = as.numeric(Lon)
```

```
#size1 = as.numeric(size1)
Lat
```

```
##  [1] 40.44133 40.44088 40.43903 40.43720 40.43580 40.43878 40.44005
##  [8] 40.43764 40.44037 40.44584 40.44461 40.44484 40.44583 40.44757
## [15] 40.45059 40.45509 40.44963 40.45212 40.46610 40.47019 40.46589
## [22] 40.46277 40.45995 40.45650 40.45871 40.46444 40.46098 40.45897
## [29] 40.45582 40.45732 40.45891 40.45628 40.45262 40.45177 40.44842
## [36] 40.44674 40.44240 40.44103 40.43434 40.43799 40.44463 40.44232
## [43] 40.44502 40.43888 40.43599 40.42866 40.42802 40.42858 40.42934
## [50] 40.42855
```

```
Lon
```

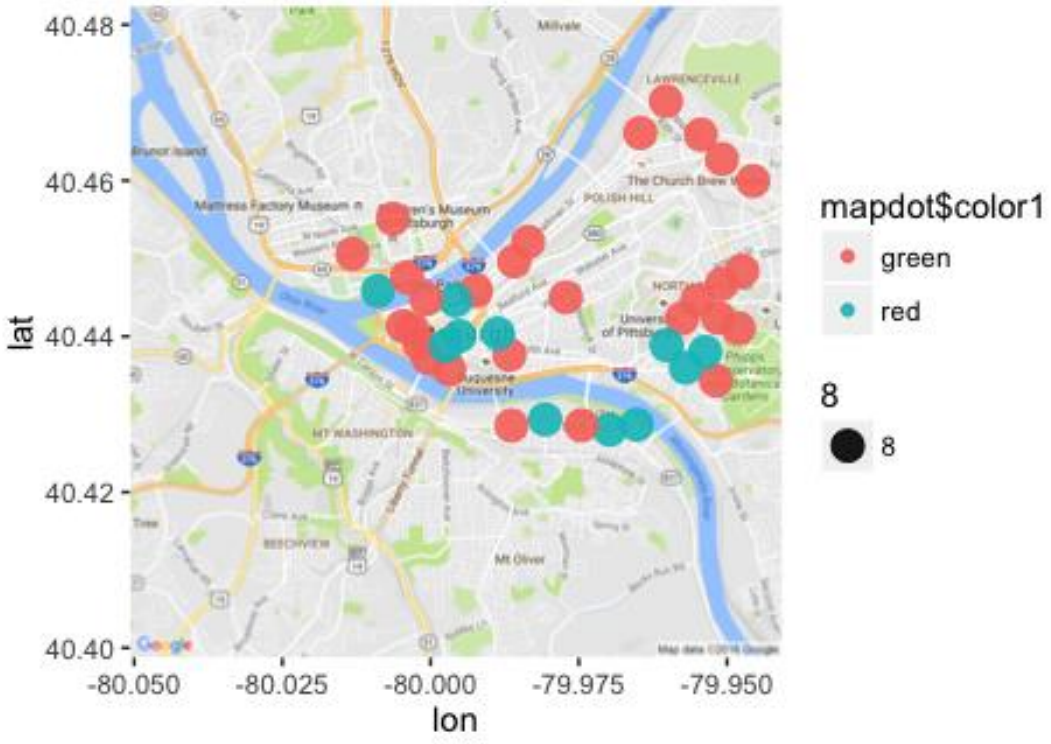
```
##  [1] -80.00468 -80.00308 -80.00186 -80.00038 -79.99688 -79.99744 -79
.99514
##  [8] -79.98669 -79.98864 -79.99238 -79.99581 -80.00075 -80.00888 -80
.00396
## [15] -80.01320 -80.00635 -79.98589 -79.98354 -79.96463 -79.96031 -79
.95442
## [22] -79.95087 -79.94561 -79.93936 -79.93348 -79.93319 -79.92630 -79
.92202
## [29] -79.91525 -79.92483 -79.92902 -79.93096 -79.92864 -79.93232 -79
.94740
## [36] -79.95088 -79.95148 -79.94804 -79.95188 -79.95367 -79.95502 -79
.95760
## [43] -79.97719 -79.96018 -79.95694 -79.96523 -79.96980 -79.97456 -79
.98068
## [50] -79.98637
```

```
mapdot$color1
```

```
##  [1] "green" "green" "green" "green" "green" "red"   "red"   "green"
##  [9] "red"   "green" "red"   "green" "red"   "green" "green" "green"
## [17] "green" "green" "green" "green" "green" "green" "green" "green"
## [25] "green" "green" "green" "green" "red"   "green" "red"   "red"
## [33] "green" "green" "green" "green" "green" "green" "green" "red"
## [41] "green" "green" "green" "red"   "red"   "red"   "red"   "green"
## [49] "red"   "green"
```

```
#size1

#ggmap(map) + geom_point(data=mapdot, aes(x=Lon, y=Lat, size=size1, col
or= mapdot$color1), alpha=0.9)
ggmap(map) + geom_point(data=mapdot, aes(x=Lon, y=Lat, size = 8,color=
mapdot$color1), alpha=0.9)
```

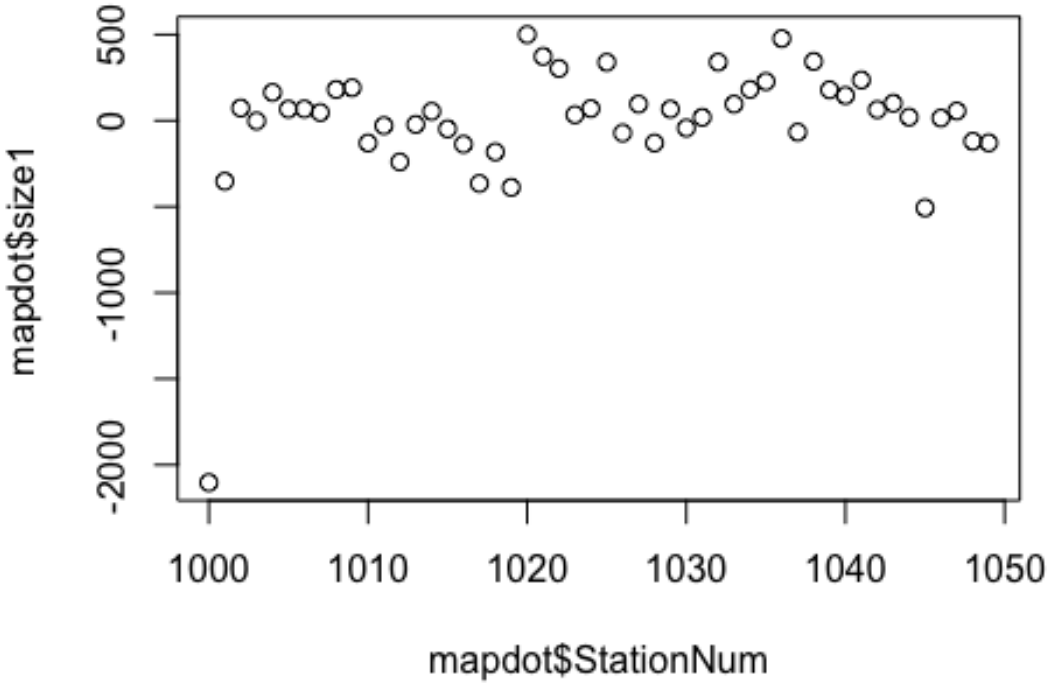
```
## Warning: Removed 11 rows containing missing values (geom_point).
```



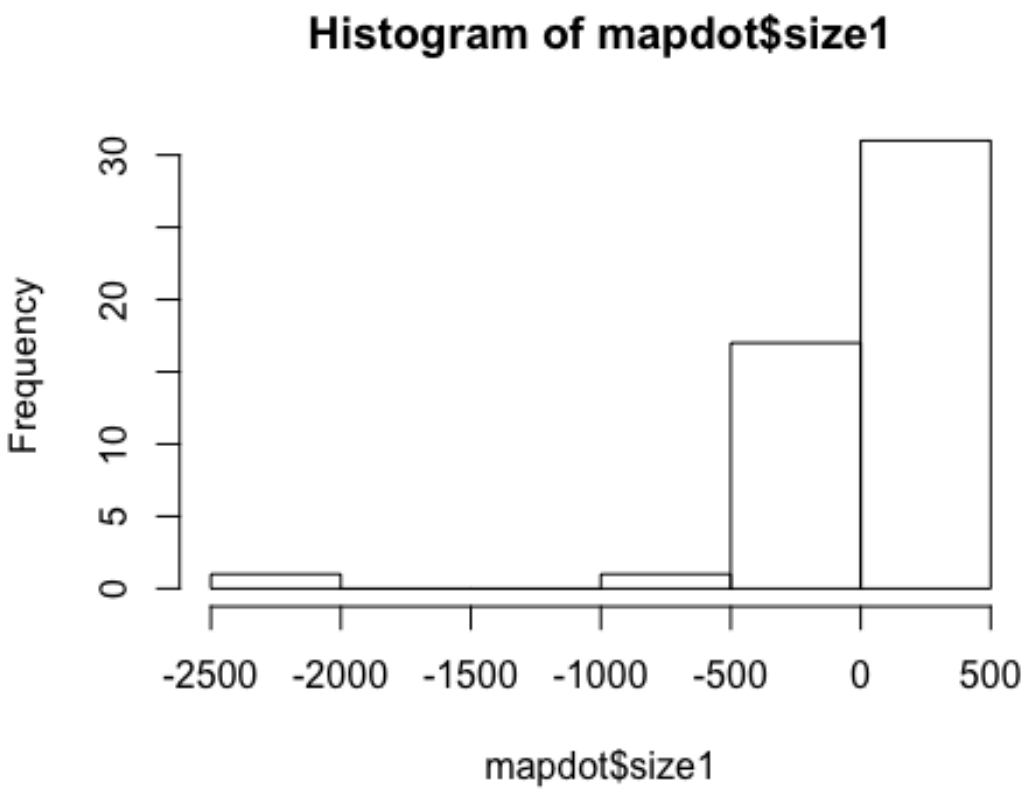

```
# 1 = red = need bike(rent>return), 2 = green = not need bike (return>r
ent)
# There are some code problem, but the plot is no problem.
```

Plot 3

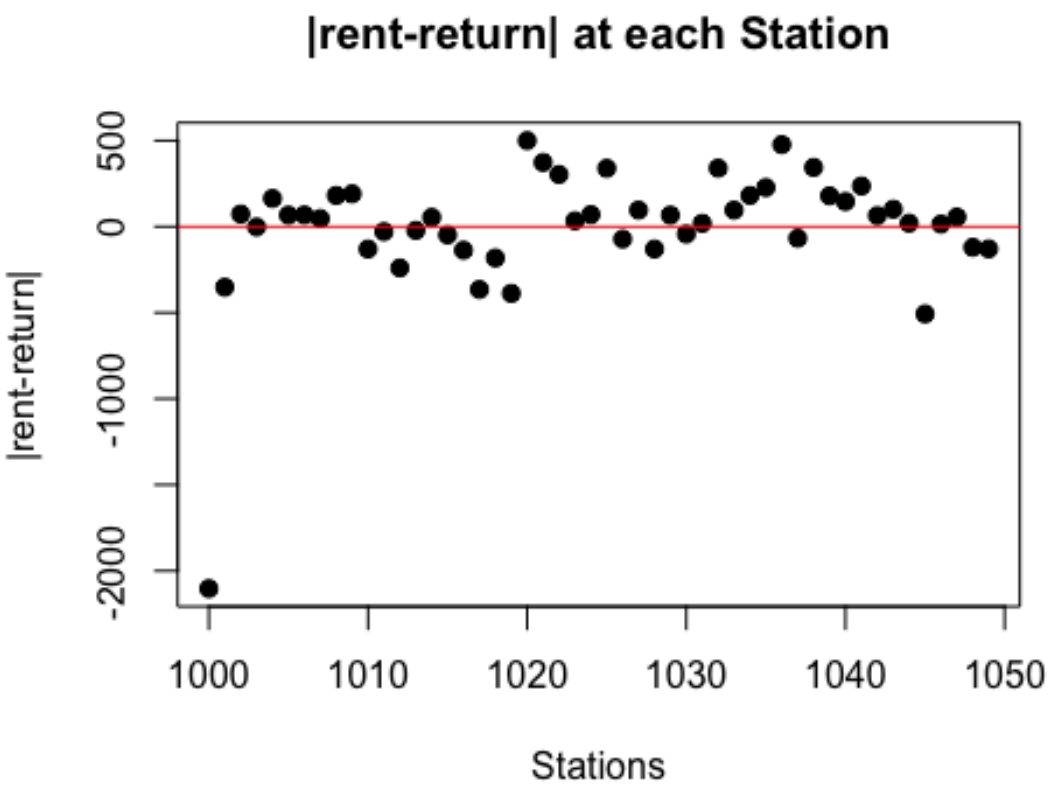
```
####### plot 3 pre modeling
mapdot$size1 = as.integer(mapdot$size1)
mapdot$color1 = as.factor(mapdot$color1)
str(mapdot)

## 'data.frame':    50 obs. of  5 variables:
##  $ StationNum: chr  "1000" "1001" "1002" "1003" ...
##  $ Latitude  : chr  "40.441326" "40.440877" "40.43903" "40.4372" ...
##  $ Longitude : chr  "-80.004679" "-80.00308" "-80.00186" "-80.000375
" ...
##  $ size1     : int  -2103 -352 73 -2 164 67 68 47 181 191 ...
##  $ color1    : Factor w/ 2 levels "green","red": 1 1 1 1 1 2 2 1 2 1
 ...

plot(mapdot$StationNum,mapdot$size1)
```




```
hist(mapdot$size1)
```




```
plot(mapdot$StationNum,mapdot$size1, main="|rent-return| at each Statio
n",
     xlab="Stations", ylab="|rent-return|", pch=19)
abline(0,0, col="red")
```


|rent-return| at each Station
|rent-return|
500
0
-1000
-2000
1000
1010
1020
1030
1040
1050
Stations