```r
library("devtools")
library("RPostgreSQL")
library("dplyr")
library("chron")
library("geosphere")
library("ggmap")
library("ggplot2")
library("sqldf")
library("maps")
library("readr")
library("zoo")
library("stringr")
library("emoGG")
library("knitr")
library("gridExtra")
library("grid")
library("EBImage")
library("gplots")
library("RCurl")
options(digits=8)

#load data, details of which are hidden
```

Clean raw data

```r
#clean the variable 'update date'
newd <- c() #store cleaned date of "update date"
newh <- c() #store cleaned hour of "update date"
for(i in 1:length(rawdata$guid)){
  timestring <- strsplit(rawdata$guid[i], "-")[[1]][3]
  timestring1 <- strsplit(timestring, "")[[1]]
  y <- paste(timestring1[1:4], sep = "", collapse = "") #year
  m <- paste(timestring1[5:6], sep = "", collapse = "") #month
  d <- paste(timestring1[7:8], sep = "", collapse = "") #date
  t <- paste(timestring1[9:length(timestring1)], sep = "", collapse = "")
#time of the day
  ymd <- paste(y,"-", m,"-", d, collapse = "", sep = "") #paste year, month,
day together to get the date
  newd <- c(newd, ymd)
  newh <- c(newh, t)
}

#Clean newh
for(i in 1:length(newh)){
  if(length(grep("m", newh[i])) == 0){
  #if the time data is not in "a.m." or "p.m." format, run this loop
    hms <- newh[i]
    s_hms <- strsplit(hms, split = "")[[1]]
    ht <- paste(s_hms[1:2], sep = "", collapse = "") #hour
    mt <- paste(s_hms[3:4], sep = "", collapse = "") #minute
    st <- paste(s_hms[5:6], sep = "", collapse = "") #second
    newh[i] <- paste(ht, mt, st, sep = ":", collapse = "")
  #paste them together to get the time
  }
  else if(length(grep("m", newh[i])) == 1){
  # if the time data is in "a.m." or "p.m." format, run this loop
    hms <- newh[i]
    s_hms <- strsplit(hms, split = "")[[1]]
    l <- length(s_hms)
    ht <- paste(s_hms[1:(l-4)], sep = "", collapse = "") #hour
    mt <- paste(s_hms[(l-3):l], sep = "", collapse = "") #minute
    hhmm <- strptime(paste(ht, mt, sep = ":", collapse = ""), "%I:%M %p")
    newh[i] <- strsplit(as.character(hhmm), split = " ")[[1]][2]
  }
}

#Convert to time format
cleaned_time <- c()
for(i in 1:length(newh)){
  cleaned_time<-c(cleaned_time,paste(newd[i],newh[i],sep = " ",collapse= ""))
}
cleaned_time <- strptime(cleaned_time, '%Y-%m-%d %H:%M')

#remove unnecessary variables from the environment
rm(y, m, d, ymd, i, timestring, timestring1)
rm(hms, ht, l, mt, newd, newh, s_hms, st, t, hhmm)

#update raw data set with the cleaned "update date"
#and pull out available_date
rawdata[, 6] <- NULL
rawdata$update_date <- cleaned_time
rawdata <- rawdata[, c(1:5, 74, 6:73)] # reorder the columns.
update_date <- data.frame(rawdata[, 6])
colnames(update_date) <-"update_date"
rm(cleaned_time)

#clean variable "available date"
date_close <- factor(rawdata$Available.Date)
rawdata$Available.Date <- as.Date(date_close, "%m/%d/%Y")
rm(date_close)
available_date <- data.frame(rawdata$Available.Date)
colnames(available_date) <-"available_date"

#creat a dateframe called "basic information" and clean it
basic <- data.frame(rawdata[,c(3, 4, 9, 10, 11, 12, 14, 15)])
basic$rent <- as.numeric(basic$rent)
basic$longitude[which(basic$longitude %in% c("NaN", "None"))] <- NA
basic$latitude[which(basic$latitude %in% c("NaN", "None"))] <- NA
basic$longitude <- as.numeric(basic$longitude)
```

```r
basic$latitude <- as.numeric(basic$latitude)

#take source_url, description, picture, title, T information out
#put them into separate tables
source_url <- data.frame(rawdata$Source.URL)
colnames(source_url) <- "source_url"
description <-data.frame(rawdata$description)
colnames(description) <- "description"
picture <-rawdata[, 45:dim(rawdata)[2]]
title <-data.frame(rawdata$title)
colnames(title) <- "title"
```

Find competitors

```r
#Client information
#this part of code is the only part we need to alter for different clients
loca <- "15 Highland Avenue, Cambridge"
loc <- geocode(loca)
bednum <- 3
bathnum <- 1
date_lag <- 300 #Use the data back to how many days
available_date_range <- 300 #Care about the price trend over what period
expected_price <- 1200 #What is the price per bedroom you expect
ava <- "2017-09-01" # What is the available date of this apartment

#First layer stations
client <-c(loc$lon, loc$lat)
distance <- array(distm(client, Tstation[,c(2,3)], fun = distHaversine))
station <- Tstation[which(distance <= 400),]
station <-cbind(station, distance[which(distance <= 400)])
colnames(station)[5] <-"distance"
station <-station[row.names(unique(station[,c(2,3)])),]

#First layer competitors
distance_c_r <- array(distm(client, location[,c(1,2)], fun =
distHaversine))#distance from client to all other rentals.
#row names of competitors around client
competitor_c <- row.names(location[which(distance_c_r <= 400),])

#If there's no first layer station, first layer competitors would be all the
competitors
if(dim(station)[1] == 0){
  competitor <- basic[competitor_c,]
}

#If there is one or more than one stations, find the second layer of stations
and second layer of competitors.
if(dim(station)[1] != 0){
  n <- dim(station)[1]
  station_ad <- NULL
  for(i in 1:n){#one station further from nearby stations, find the rownames station_ad
    distance_t_t <- array(distm(station[,c(2, 3)][i,], Tstation[,c(2,3)], fun = distHaversine))
    station_ad <- unique(c(station_ad, row.names(Tstation[which(distance_t_t <= 400),])))
  }

#longest distance from client to stations
  distance_max <- max(array(distm(client, Tstation[unique(c(rownames(station), station_ad)),][,2:3], fun = distHaversine)))
#use longest distance to find maybe new stations,let's have 10% flexibility
  station <- Tstation[which(distance <= distance_max),]
  station <- station[row.names(unique(station[,c(2,3)])),]
  competitor_ad <- NULL
  n <- dim(station)[1]

#Second layer of competitors #competitor rownames around updated stations
  for(i in 1:n){
    distance_t <- array(distm(station[,c(2, 3)][i,], location[,c(1,2)], fun = distHaversine))
    competitor_ad <- unique(c(competitor_ad, row.names(location[which(distance_t <= 400),])))
  }
  competitor_i <- unique(c(competitor_c, competitor_ad))
  competitor <- basic[competitor_i,]
}

rm(competitor_ad, competitor_c, competitor_i, distance, distance_c_r, distance_max, distance_t, distance_t_t, i, n, station_ad)
```

Clean the competitors data

```r
#clean the "competitor" dataframe
#attach the available date
competitor$available_date <- available_date[rownames(competitor),]

#delete those with rental lower than 1500 and higher than 6000
if(length(which(competitor$rent < 1500 | competitor$rent > 6000))==0){
  competitor <- competitor}
if(length(which(competitor$rent < 1500 | competitor$rent > 6000))!=0){
  competitor <- competitor[-which(competitor$rent < 1500 | competitor$rent > 6000),]}

#for now set a bigger than 350 ft threshould
#convert sf to numeric variable
competitor$Square.Footage <- as.numeric(competitor$Square.Footage)
if(length(which(competitor$Square.Footage < 350))==0){
  competitor <- competitor}
```

```
if(length(which(competitor$Square.Footage < 350))!=0){
  competitor <- competitor[-which(competitor$Square.Footage < 350),]}

#Delete those with 0 bathroom, fake post
if(length(which(competitor$bathrooms == "0"))==0){
  competitor <- competitor}
if(length(which(competitor$bathrooms == "0"))!=0){
  competitor <- competitor[-which(competitor$bathrooms == "0"),]}

#convert "studio" to "1"
competitor$bedrooms[which(competitor$bedrooms == "Studio")] <- "0"

#join more information from other tables to the "competitor" table
#attach the updated date
competitor$update_date <- update_date[rownames(competitor),]
# create a new column for unit price
competitor$price <- round(ifelse(competitor$bedrooms == 0, competitor$rent,
competitor$rent/as.numeric(competitor$bedrooms)))
competitor$ratio <-
round(as.numeric(competitor$bedrooms)/as.numeric(competitor$bathrooms),2)
#bed to bath ratio

#delete those with price per bed higher than 4000, probably fake post
if(length(which(competitor$price > 4000))==0){
  competitor <- competitor}
if(length(which(competitor$price > 4000))!=0){
  competitor <- competitor[-which(competitor$price > 4000),]}
```

Identify and delete dupllications in the competitors data

```
#Identify the dupications
#the same rental information might get extracted multiple times
#Same bed, bath, location, available date: suspecious duplications
#d1 is the table of duplications without the one being duplicated
d1 <-competitor[which(duplicated(competitor[4:9])), 4:9]
uq <- unique(d1) #Unique of the duplications
n <-dim(uq[1])[1] #Number of "groups" is n

g <- data.frame() #create an empty matrix to store the row numbers

for(i in 1:n){
  for(j in 1:dim(competitor)[1]){
      if(sum(paste(competitor[j, 4:9], sep = "") == paste(uq[i,], sep = ""))
== 6){
        g[j,i] <- j
      }
    }
}

group_i <- NULL #group index
```

```r
rname_g <- NULL #rowname index
for(i in 1:n){
  g_i <-rep(i, length(na.omit(g[,i])))
  c_i <-as.numeric(na.omit(g[,i]))
  group_i <- c(group_i, g_i)
  rname_g <- c(rname_g, c_i)
}

g_1 <- data.frame(matrix(nrow = length(group_i), ncol = 1))
colnames(g_1) <- c("group")
g_1$group <- group_i
row.names(g_1) <- rownames(competitor)[rname_g]

#paste the other information.
g_1 <- cbind(g_1, competitor[row.names(g_1),], source_url[row.names(g_1),],
picture[row.names(g_1),])

dele_c <-NULL
for(i in 15:dim(g_1)[2]){
  if(sum(is.na(g_1[,i]))==dim(g_1)[1]){
    dele_c <- c(dele_c, i)
  }
}

g_1[, dele_c] <- NULL
colnames(g_1)[14] <- "source_url"
rm(d1, g, uq, i, j, rname_g, group_i, g_i, dele_c, c_i)

#Delete the duplications
#now dlt does not deal with "realtime"
#deal with "realtime"
dlt <- c()
for(i in 1:n){
  a <- g_1$update_date[g_1$group == i]
  b <- row.names(g_1[which(g_1$group == i),])
  dlt <- c(dlt, b[a != max(a)])
}
rm(a, b, i)

a <- rownames(g_1)[is.na(g_1$update_date)]
dlt1 <- array(na.omit(c(dlt, a[1:length(a)-1])))

#and delete them from competitor
competitor <-competitor[!rownames(competitor) %in% dlt1,]

#take the update date within date_lag days for now
competitor <- competitor[difftime(as.Date("08-10-2017"),
competitor$update_date, units = "days") <= date_lag,]
#take the reasonalble available date
```

```r
start_time <- as.Date("08-10-2017") - available_date_range
competitor <- competitor[competitor$available_date >= start_time,]

#to kinds of competitors, indirect competitors and direct competitors
competitor_d <- competitor[which(competitor$bedrooms == bednum &
(competitor$available_date - as.Date(ava)) <= 14),]
competitor_i <- competitor[which(competitor$bedrooms != bednum |
(competitor$available_date - as.Date(ava)) > 14),]

com_new <- competitor[,c(10, 11)]
a <-as.character(competitor[,9])
com_new[,1] <- as.yearmon(a)
colnames(com_new)[1] <- "available_date"

rm(a, dlt, dlt1, n, g_1)
```

Table to show basic information of client

```r
#a data frame called df1 to store information about client
df1 <- data.frame()
df1[1,1] <- paste(strsplit(loca, split = ",")[[1]][1:2],sep = "", collapse=
",")
df1[2,1] <- ava
df1[3,1] <- paste(bednum, "bd", ", ", bathnum, "ba", ", " ,"listed price: $",
expected_price*bednum,",",sep = "")
colnames(df1) <- "Rental Information"
```

Locate 3 typical direct competitors, and complete the code for map

```r
#take 3 direct competitors out
#one with highest price, one with lowest, one with most similar price
eq1 <- row.names(competitor_d)[which(abs(expected_price- competitor_d$price)
== min(abs(expected_price- competitor_d$price)))][1]
ma1 <- row.names(competitor_d)[which(competitor_d$price ==
max(competitor_d$price))][1]
mi1 <- row.names(competitor_d)[which(competitor_d$price ==
min(competitor_d$price))][1]

eq_d <- competitor_d[eq1,]
ma_d <- competitor_d[ma1,]
mi_d <- competitor_d[mi1,]

#calculate proper zoom of the map according to the size of local market
z <- 15
if(max(array(distm(client, competitor[,c(7,8)], fun = distHaversine)), na.rm
= T) < 780){
  z <- 16
}

#Awesome map with emoji icons
map <- get_map(location = loc, zoom = z)
```

```
map1 <- ggmap(map)+
  geom_emoji(aes(x = loc$lon, y = loc$lat), emoji="1f3e1")+
  geom_emoji(aes(x=lon, y=lat), data = station, emoji = "1f687")+
  geom_emoji(aes(x =longitude, y =
latitude),data=competitor_i,emoji="1f6a9")+
  geom_emoji(aes(x = longitude, y = latitude), data=competitor_d,
emoji="1f6a9")+
  geom_emoji(aes(x = longitude, y = latitude), data=mi_d, emoji="2b07")+
  geom_emoji(aes(x = longitude, y = latitude), data=eq_d, emoji="2194")+
  geom_emoji(aes(x = longitude, y = latitude), data=ma_d, emoji="2b06")+
  theme(axis.line=element_blank(),axis.text.x=element_blank(),
axis.text.y=element_blank(),axis.ticks=element_blank(),
axis.title.x=element_blank(), axis.title.y=element_blank())
```

Plot price distribution of local market

```
#Create a table for visualization purpose
client_inf <- data.frame(matrix(nrow=1, ncol = 13))
colnames(client_inf) = c(colnames(competitor), "group")
client_inf[1,] <- c(NA, NA, bednum*expected_price, bednum, bathnum, NA,
loc$lon, loc$lat, ava, NA, expected_price, bednum/bathnum, "client")
competitor_plot <- cbind(competitor, rep("competitor",
length(competitor$Type)))
colnames(competitor_plot) <- colnames(client_inf)
competitor_plot <- rbind(competitor_plot, client_inf)
competitor_plot$price <- as.numeric(competitor_plot$price)
competitor_plot$yearmon <- as.yearmon(competitor_plot$available_date)

#calculate the proper ranges and breaks of the axis of the plot
a <- ifelse(sort(unique(c(seq(200*(floor(min(competitor_plot$price, na.rm =
T)/200)+1), 200*(floor(max(competitor_plot$price, na.rm = T)/200)+1), by =
200), expected_price, min(competitor_plot$price, na.rm = T)))) ==
expected_price, "red", "black")

#ggplot
p2 <- ggplot(competitor_plot[!is.na(competitor_plot$price), ],aes(x =
price))+
  geom_histogram(fill = "dark blue") +
  stat_bin(aes(y=..count.., label=..count..), geom="text", vjust=-.5, size =
2) +
  scale_x_continuous(breaks =
sort(unique(c(seq(200*(floor(min(competitor_plot$price, na.rm = T)/200)+1),
200*(floor(max(competitor_plot$price, na.rm = T)/200)+1), by = 200),
expected_price, min(competitor_plot$price, na.rm = T)))))+
  labs(title="Distribution of All Competitors by Price/Bedroom", x ="Price
per Bedroom", y = "Count")+
  geom_vline(xintercept = expected_price, color = "red", size=0.6)+
  theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),
text=element_text(size=10),axis.text.x=element_text(hjust=1,colour=a),
        legend.position="none", plot.title = element_text(hjust = 0.5))

#calculate the proper ranges and breaks of the axis of the plot
b <- ifelse(sort(unique(unique(competitor_plot$yearmon),
as.yearmon(as.Date(ava)))) == as.yearmon(as.Date(ava)), "red", "black")

#ggplot
p3 <- ggplot(competitor_plot[!is.na(competitor_plot$price), ], aes(x =
factor(yearmon), y = price)) +
  geom_boxplot(color = "dark blue")+
  geom_point(aes(colour = group, size = group, shape = group))+
  scale_colour_manual(values=c("dark blue", "red"))+
  scale_size_manual(values = c(1.5, 2.5)) +
  scale_shape_manual(values = c(16, 17))+
  theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),
        text = element_text(size=10), axis.text.y = element_text(hjust = 1,
colour=a),
        legend.position="none", axis.text.x = element_text(hjust = 1,
colour=b),
        plot.title = element_text(hjust = 0.5))+
  labs(title="Price/Bedroom by Available Date (All)", x="Available
Date",y="Price per Bedroom")+
  scale_y_continuous(breaks =
sort(unique(c(seq(200*(floor(min(competitor_plot$price, na.rm = T)/200)+1),

200*(floor(max(competitor_plot$price, na.rm = T)/200)+1), by = 200),
expected_price,
                                            min(competitor_plot$price, na.rm
= T))))) +
  geom_hline(yintercept = expected_price, color = "red", size=0.6)+
  geom_vline(xintercept = which(b == "red"), color = "red", size=0.6)
```

Show competitors' pictures

```
#Find the available urls
pic_com_d <- picture[row.names(competitor_d),]

#if the picture url is not working, replace it with a blank picture url
for(i in 1:dim(pic_com_d)[1]){
  for(j in 1:dim(pic_com_d)[2]){
    if(is.na(pic_com_d[i, j]))
      {pic_com_d[i, j] <-
"https://forums.autodesk.com/autodesk/attachments/autodesk/706/691062/1/WPX%2
0Logo%203D%20v2-B%26W.png"}
    else if(url.exists(pic_com_d[i, j]) == T)
    {pic_com_d[i, j] <- pic_com_d[i, j]}
    else{pic_com_d[i, j] <-
"https://forums.autodesk.com/autodesk/attachments/autodesk/706/691062/1/WPX%2
0Logo%203D%20v2-B%26W.png"}
    }
}
```