# FinalProject

*LingzhouAo*

*5/7/2018*

## Overview

As we all know, convenience is one of the most important factors that influences the house renting price, and obviously, in most cases, if a district is more convenient, the house renting price of this district will be higher. The price and number of restaurants play very important roles in determining whether a district is convenient, and also people cannot live without food, which means when people want to rent a so I decide to explore the relationship between house renting price and restaurant price range. Because house renting and catering industry are very booming in Boston, I choose to use Boston as our study area.

The price and number of restaurants play a very important role in determining whether a district is convenient, and also people cannot live without food, so I decide to explore the relationship between house renting price and restaurant average customer spending.

## Get Data

1. For house information, because the Zillow API doesn't support getting data in a wide range, I implement a web crawler to scrap data from Zillow.
2. For restaurant information, I use Yelp API to get the data of restaurants.
3. In order to know the exact location of houses and restaurants, I choose to use Google Map API to get the latitude and longitude of each house or restaurant. And also, because the size of data is larger than 2,500, I have to pay for the api. So I will stop this api after submitting this project to github.

## Deal with Data

The files houses.R and yelp.R are used to tidy those data I get from website. For yelp data,I turned the original symbol $, $$, $$$, $$$ on Yelp into 1, 2, 3, 4 these four levels. Level one and two means relatively inexpensive$ restaurants and $$ restaurants, level three and four represents for the other two. For houses data, I calculate the average price for each house and based on this price and its coordinate information, I use K-means to cluster the house data

## K-means Results

## Decide K value

```r
library(ggmap)
```

```
## Loading required package: ggplot2
## Google Maps API Terms of Service: http://developers.google.com/maps/terms.
## Please cite ggmap if you use it: see citation("ggmap") for details.
```

```r
suppressMessages(library("tidyverse"))
library(magick)
```

```
## Warning: package 'magick' was built under R version 3.4.4
```

```
## Linking to ImageMagick 6.9.9.39
## Enabled features: cairo, fontconfig, freetype, lcms, pango, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

```r
library(ggplot2)

final <- read_csv("houses.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   address = col_character(),
##   price = col_integer(),
##   rooms = col_integer(),
##   lat = col_double(),
##   lng = col_double()
## )
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 2)
```

```
## Warning: 1 parsing failure.
## row # A tibble: 1 x 5 col     row col   expected   actual file         expected   <int> <chr> <chr>
```

```r
df_final <- data_frame(price = final$price,
                       lat = final$lat,
                       lng = final$lng
)
yelp <- read_csv("YelpData.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

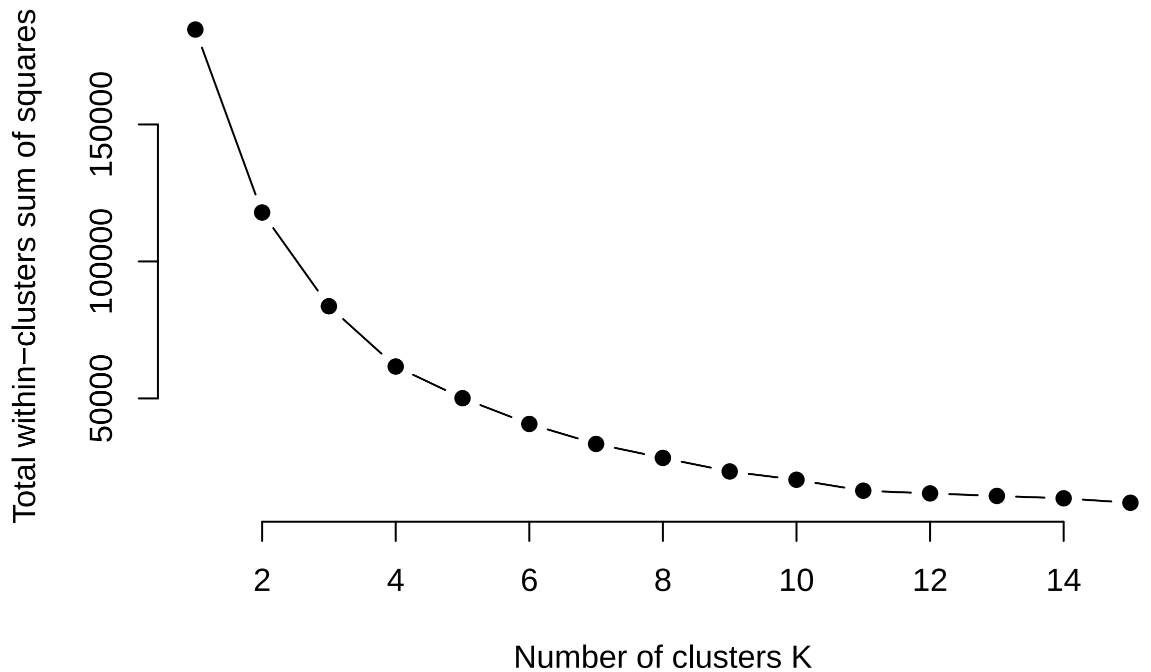
```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   name = col_character(),
##   price = col_integer(),
##   latitude = col_double(),
##   longitude = col_double()
## )
```

```r
df_yelp <- data_frame(price = yelp$price,
                      lat = yelp$latitude,
                      lng = yelp$longitude
)

final.scale <- scale(df_final)

final.scale[,2] <- final.scale[,2] *5
final.scale[,3] <- final.scale[,3] *5

#Elbow Method for finding the optimal number of clusters
set.seed(123)
# Compute and plot wss for k = 2 to k = 15.
k.max <- 15
data <- final.scale
```

```r
wss <- sapply(1:k.max,
              function(k){kmeans(data, k, nstart=50,iter.max = 15 )$tot.withinss})
wss
```

```
##  [1] 184620.00 117860.52  83651.55  61657.66  50092.68  40696.43  33409.14
##  [8]  28308.51  23359.66  20339.82  16338.18  15353.95  14437.97  13562.61
## [15]  11936.33
```

```r
plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```




```r
# Choose 10 as the number of cluters

k.m <- kmeans(data, 10, nstart = 50, iter.max = 15)

df_cluster <- data_frame(cluster = k.m$cluster)
new_final <- cbind(df_final, df_cluster)


price_label <- c(1:10)
lat_label <- c(1:10)
lng_label <- c(1:10)
for (i in c(1:10)){
  new_final$label[new_final$cluster == i] <- mean(new_final$price[new_final$cluster == i])
  price_label[i] <- mean(new_final$price[new_final$cluster == i])
  lat_label[i] <- mean(new_final$lat[new_final$cluster == i])
  lng_label[i] <- mean(new_final$lng[new_final$cluster == i])
}
price_label
```

```
##  [1] 2209.506 2898.991 2664.842 3394.867 3765.917 2912.173 3471.389
```

```
##  [8] 2572.376 2253.943 2703.771
```

```r
lat_label
```

```
##  [1] 42.33744 42.32172 42.30929 42.33673 42.40662 42.34251 42.18162
##  [8] 42.37656 42.27747 42.34552
```

```r
lng_label
```

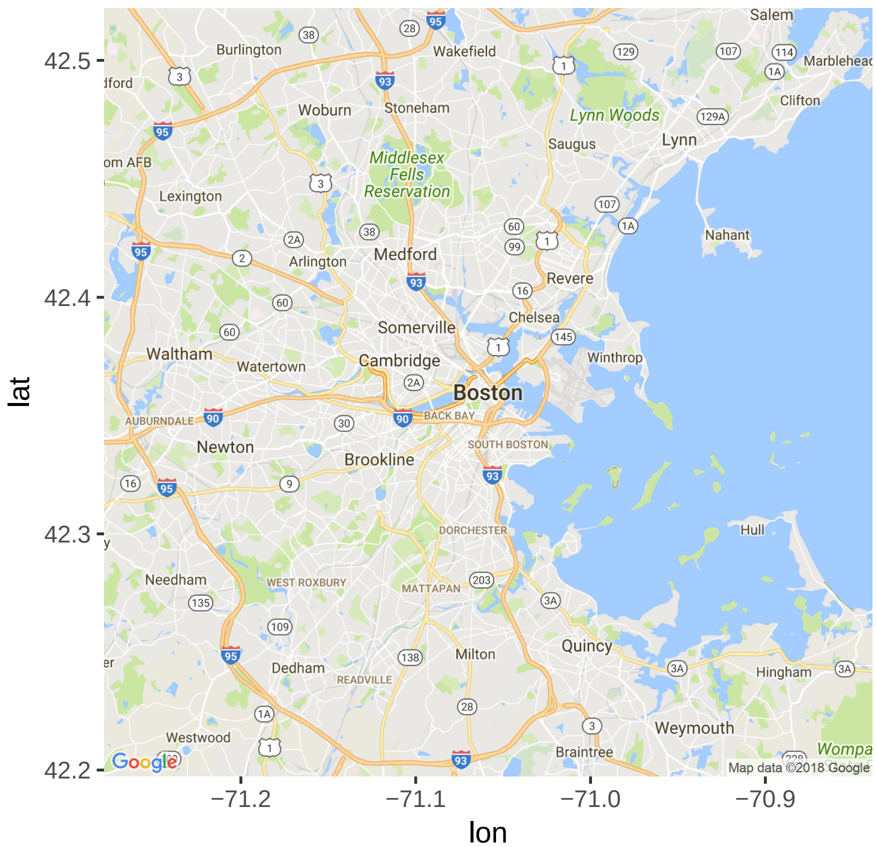
```
##  [1] -71.22865 -71.10901 -71.06423 -71.04471 -71.14288 -71.08542 -71.20334
##  [8] -71.03682 -71.13287 -71.14223
```

```r
new_final$new_label <- factor(new_final$label)
```

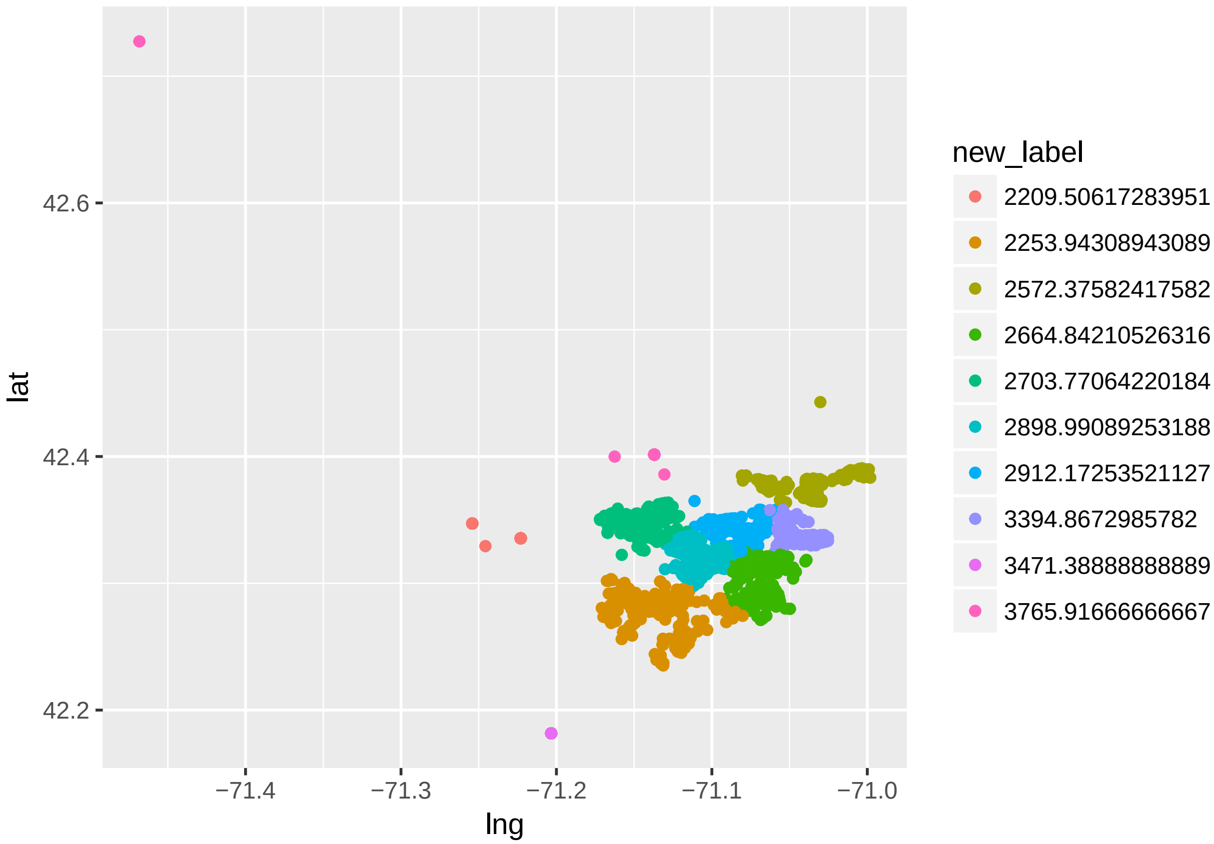
```r
register_google(key = 'AIzaSyAmOQseQL27vmpzc4Lpddmocrzv1jDkXwg')
map <- get_googlemap(center =c(-71.0589,42.3601), zoom = 11, maptype = "roadmap")
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=42.3601,-71.0589&zoom=11&size=640x640
```

```r
ggmap(map)
```

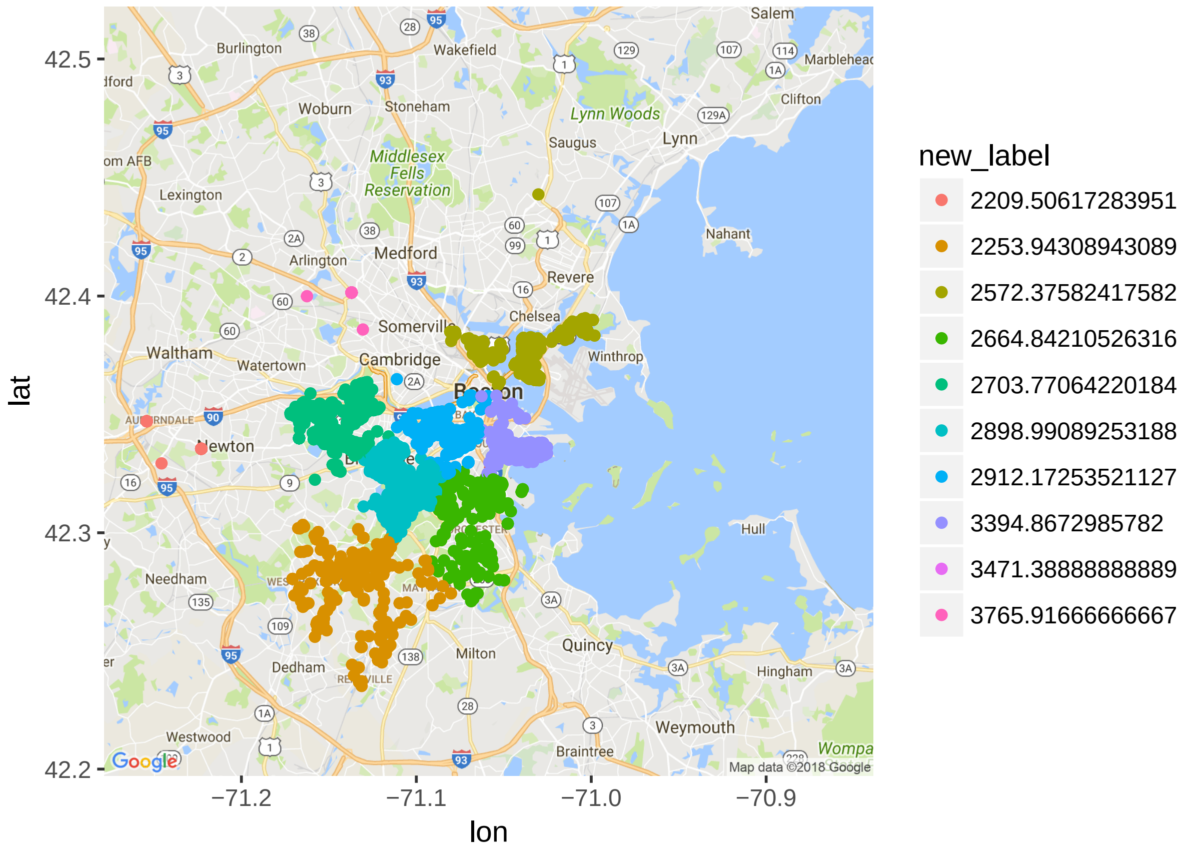



```r
ggplot(data = new_final, aes(x = lng, y = lat , color = new_label), size = 5, shape = 21)+geom_point()
```




```r
ggmap(map)+
  geom_point(data = new_final,aes(x = lng, y = lat, color = new_label))
```

```
## Warning: Removed 37 rows containing missing values (geom_point).
```




Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot. ##Merge Data Then I want to related restaurants data with house data, so I put each restaurant into its nearest cluster. After that, I can analyze the relationship between them.

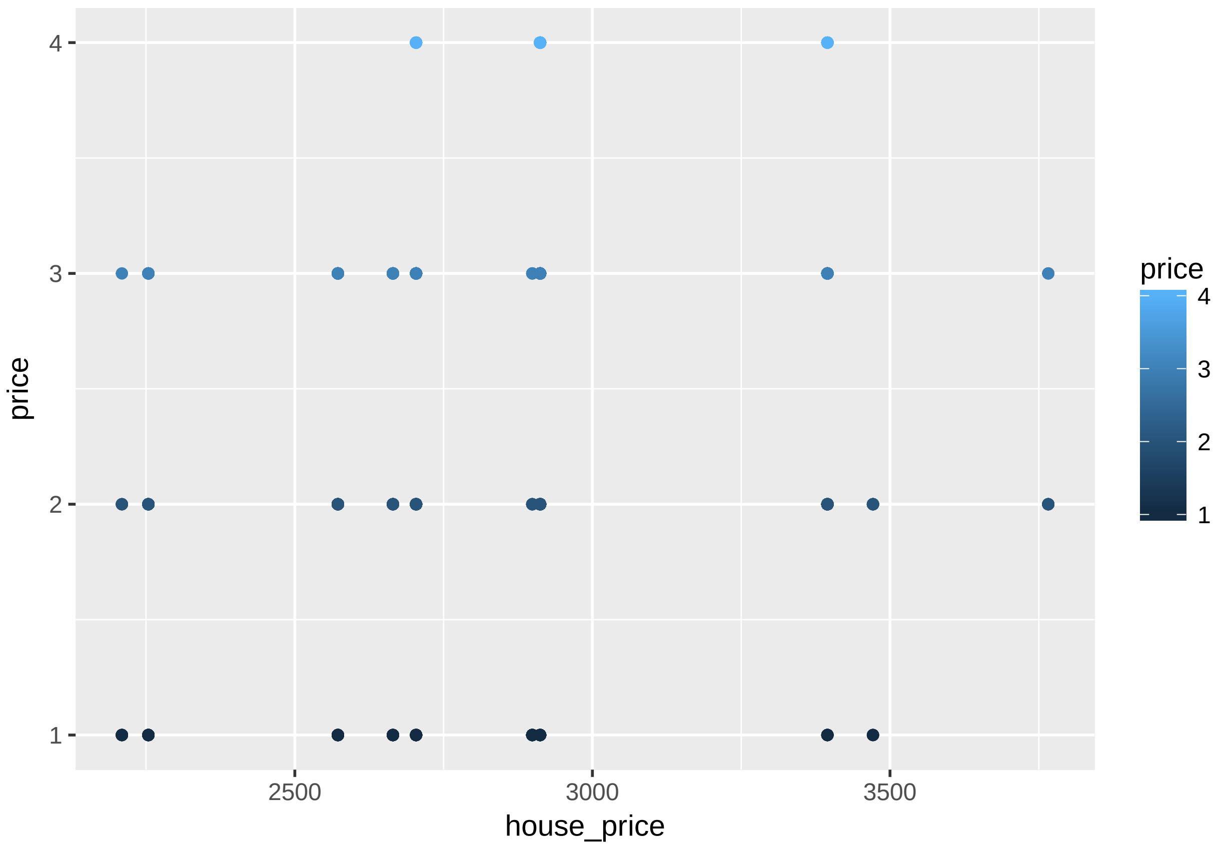
## Results

```
row.has.na <- apply(df_yelp, 1, function(x){any(is.na(x))})
sum(row.has.na)
```

```
## [1] 7
```

```
df_yelp <- df_yelp[!row.has.na,]

label_lat <- k.m$centers[,2]
label_lng <- k.m$centers[,3]

yelp_scale <- scale(df_yelp)
yelp_lat <- yelp_scale[,2]*5
yelp_lng <- yelp_scale[,3]*5

r_label <- c(1:943)

for (i in c(1:943)){
  r_l_x <- yelp_lat[i]
  r_l_y <- yelp_lng[i]
  l <- 1
```

```r
  d <- (r_l_x - label_lat[1])**2 + (r_l_y - label_lng[1])**2
  for (j in c(2:10)){
    new_d <- (r_l_x - label_lat[j])**2 + (r_l_y - label_lng[j])**2
    if (d > new_d){
      l <- j
      d <- new_d
    }
  }
  r_label[i] <- price_label[l]
}

new_df_yelp <- cbind(df_yelp,r_label)
names(new_df_yelp)[4] <- paste("house_price")


ggplot(data = new_df_yelp, aes(x = house_price, y = price, color = price))+geom_point()
```

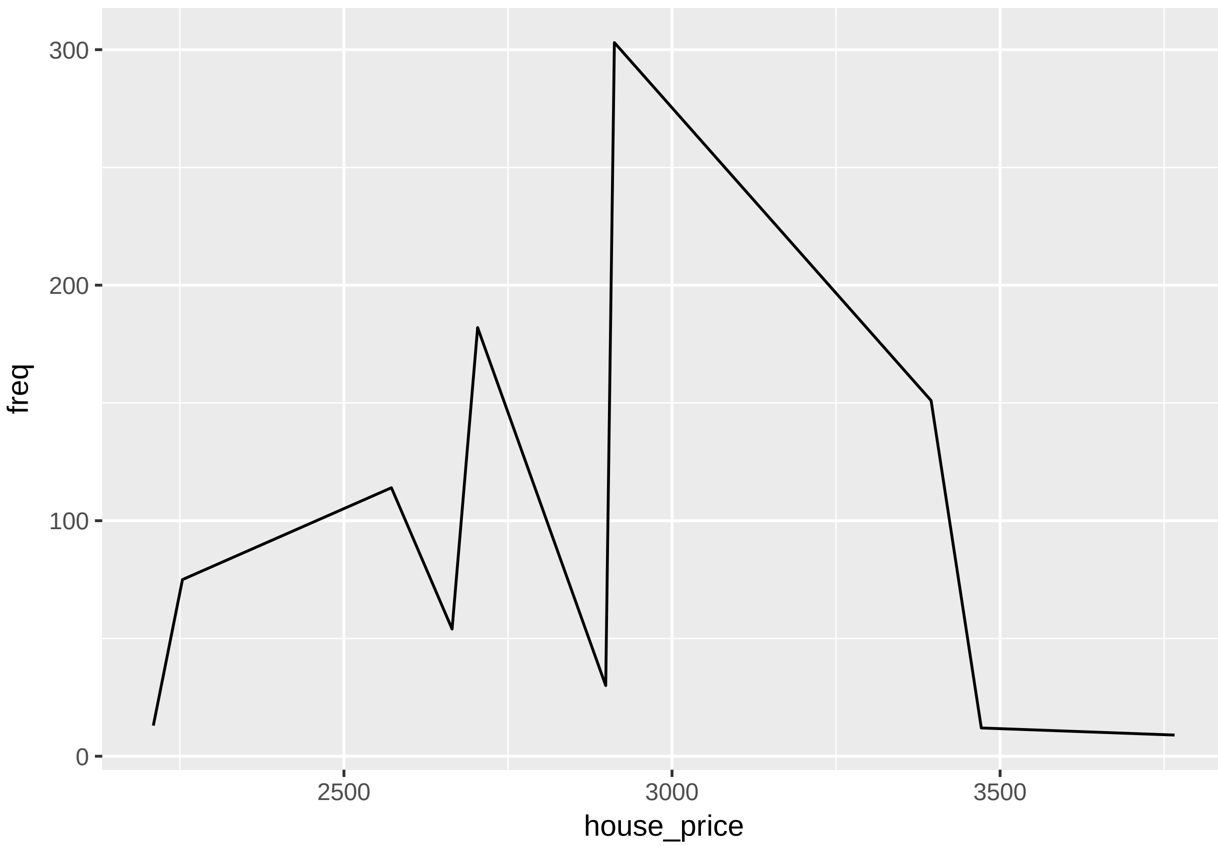



```r
library(plyr)
```

```
## ------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------

##
```

```
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

```r
fre_1 <- count(new_df_yelp,"house_price")

ggplot(data = fre_1,aes(x = house_price, y = freq))+geom_line()
```




### Summary

First, by observing the clustering result on google map, I think the result is quite realiable based on the information we know about the house price of Boston and I put each restaurant into the nearest cluster. The point figure shows for each cluster which price level's restaurant it contains and the line figure shows for each cluster how many restaurants it contains. And based on the results, I figure out for those higest price level restaurants, they will choose the high house price area but not the highest beacause they want to attract as many target customers as they can but also they don't want to pay too much renting fee. For those low price level restaruants, they are everywhere, so I guess some of them are fast food restaurants. Everyone needs fast food. Therefore, based on the map that shows the distribution of house price, you can choose the best place for dinning to live."